



PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE SANTO DOMINGO

Escuela de Hábitat, Infraestructura y Creatividad

APLICACIÓN WEB Y MÓVIL PARA LA GESTIÓN DE EMPRENDIMIENTO EN LA
EMPRESA 100 MUJERES S.A.S. B.I.C. DEL CANTÓN SANTO DOMINGO

PLAN DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Previo a la obtención del título de Ingeniero en Tecnologías de la Información

Línea de investigación: Tecnologías de la información y la comunicación

Autoría:

Silva Avila Marco Javier

Díaz Barreiro Leonardo Andrés

Dirección:

Ocampo Pazos Willian Javier, Mg.

Santo Domingo – Ecuador
mayo, 2026



PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE SANTO DOMINGO

Escuela de Hábitat, Infraestructura y Creatividad

HOJA DE APROBACIÓN

APLICACIÓN WEB Y MÓVIL PARA LA GESTIÓN DE EMPRENDIMIENTO EN LA
EMPRESA 100 MUJERES S.A.S. B.I.C. DEL CANTÓN SANTO DOMINGO

Línea de investigación: Tecnologías de la información y la comunicación

Autoría:

Silva Avila Marco Javier

Díaz Barreiro Leonardo Andrés

Revisado por:

Ocampo Pazos Willian Javier, Mg.
DIRECTOR DEL TRABAJO DE
INTEGRACIÓN CURRICULAR

Ulloa Meneses Luis Javier, Mg.
CALIFICADOR

Córdova Gálvez Rodolfo Sirilo, Mg.
CALIFICADOR

Ulloa Meneses Luis Javier, Mg.
COORDINADOR DE LA CARRERA DE
SISTEMAS DE LA INFORMACIÓN

Santo Domingo – Ecuador
mayo, 2026

DECLARACIÓN DE AUTENTICIDAD Y RESPONSABILIDAD

Yo, Silva Avila Marco Javier, portador de la cédula de ciudadanía 1722846688, Díaz Barreiro Leonardo Andrés, portador de la cédula de ciudadanía 0803589175, declaramos que los resultados obtenidos en la investigación que presento como informe final, previo a la obtención del Título de Ingeniero en Tecnologías de la Información son absolutamente originales, auténticos y personales. En tal virtud, declaramos que el contenido, las conclusiones y los efectos legales y académicos que se desprenden del trabajo propuesto de investigación y luego de la redacción de este documento son y serán de mi sola y exclusiva responsabilidad legal y académica.

Igualmente, declaramos que todo resultado académico que se desprenda de esta investigación y que se difunda tendrá como filiación la Pontificia Universidad Católica del Ecuador Sede Santo Domingo, reconociendo en las autorías al director del Trabajo de Integración Curricular y demás profesores que amerita. Además, declaro que el presente trabajo, producto de las actividades académicas y de investigación, forma parte del capital intelectual de la Pontificia Universidad Católica del Ecuador, Sede Santo Domingo, de acuerdo con lo establecido en el artículo 16, literal j), de la Ley Orgánica de Educación Superior.

En tal razón, autorizo a la Pontificia Universidad Católica del Ecuador, Sede Santo Domingo, para que pueda hacer uso, con fines netamente académicos, del Trabajo de Integración Curricular, ya sea de forma impresa, digital y/o electrónica o por cualquier medio conocido o por conocerse, siendo el presente documento la constancia del consentimiento autorizado; y, para que sea ingresado al Sistema Nacional de Información de la Educación Superior del Ecuador para su conocimiento público, en cumplimiento del artículo 103 de la Ley Orgánica de Educación Superior.



Silva Avila Marco Javier
C.C. 1722846688



Díaz Barreiro Leonardo Andrés
C.C. 0803589175

INFORME DE TRABAJO DE INTEGRACIÓN CURRICULAR ESCRITO

Mikel Ugando Peñate, PhD.

Responsable de Investigación Formativa

Pontificia Universidad Católica del Ecuador Sede Santo Domingo

De mi consideración,

Por medio del presente informe en calidad de director del Trabajo de Titulación del Grado de Ingeniería en Tecnologías de la Información titulado: APLICACIÓN WEB Y MÓVIL PARA LA GESTIÓN DE EMPRENDIMIENTO EN LA EMPRESA 100 MUJERES S.A.S. B.I.C. DEL CANTÓN SANTO DOMINGO, realizado por los estudiantes: Silva Avila Marco Javier, portador de la cédula de ciudadanía 1722846688, Díaz Barreiro Leonardo Andrés, portador de la cédula de ciudadanía 0803589175, previo a la obtención del título de Ingeniero en Tecnologías de la Información, informo que el presente Trabajo de Integración Curricular escrito, se encuentra finalizado conforme a la guía y al formato de la Sede vigente.

Además, certifico haber verificado la originalidad y autenticidad del trabajo de titulación por medio del programa anti-plagio Turnitin, en respuesta a la normativa institucional vigente.

Santo Domingo, 01/05/2026.

Atentamente,


Mg. Willian Javier Ocampo Pazos

Profesor Titular Auxiliar II

RESUMEN

La presente investigación tuvo como objetivo implementar una aplicación *web* y móvil para el fortalecimiento de la gestión de emprendimientos de la empresa 100 Mujeres S.A.S. B.I.C., del cantón Santo Domingo, ante limitaciones detectadas en la actualización de productos y servicios, el seguimiento manual de pedidos, la baja visibilidad de la oferta y las restricciones en la inscripción de nuevas emprendedoras. Se desarrolló bajo un enfoque cuantitativo, con diseño preexperimental, tipo aplicada y de campo, mediante un estudio censal con 109 mujeres emprendedoras; para la recolección de datos se empleó la encuesta, aplicada en fase de *pretest* y *posttest*, y los resultados se procesaron con análisis estadístico en *IBM SPSS* y regresión logística binaria. A partir del diagnóstico, se seleccionaron *React* con *Vite*, *Node.js*, *Firebase* y *Flutter*, junto con *Clean Architecture* y *BLoC*, para desarrollar una solución con módulos de registro, catálogo, pedidos, carga de comprobantes, historial, panel administrativo, cambio de estados en tiempo real, solicitud de nuevas emprendedoras y visualización de perfiles y contenido comunitario. Los resultados evidenciaron alta aceptación de la propuesta y un impacto positivo en la gestión de pedidos y en la organización operativa; además, se obtuvo significancia estadística menor a 0.05, lo que respaldó la hipótesis alternativa. Se concluyó que la implementación fortaleció el control, seguimiento, coordinación y autogestión de los emprendimientos vinculados a la organización.

Palabras clave: Aplicación web, aplicación móvil, Gestión de emprendimientos, mujeres emprendedoras, transformación digital.

ABSTRACT

This research aimed to implement a web and mobile application to strengthen entrepreneurship management at 100 Mujeres S.A.S. B.I.C., in Santo Domingo, in response to limitations identified in product and service updates, manual order tracking, low visibility of offerings, and restrictions on the enrollment of new women entrepreneurs. The study followed a quantitative approach with a pre-experimental design and applied field research, using a census study of 109 women entrepreneurs. Data was collected through surveys administered in pre-test and post-test phases, and the results were processed through statistical analysis in IBM SPSS and binary logistic regression. Based on the diagnosis, React with Vite, Node.js, Firebase, and Flutter were selected, together with Clean Architecture and BLoC, to develop a solution that included modules for registration, catalog management, orders, payment proof upload, order history, an administrative dashboard, real-time status updates, new entrepreneur requests, and the visualization of profiles and community content. The findings showed high acceptance of the proposal and a positive impact on order management and operational organization; in addition, statistical significance below 0.05 supported the alternative hypothesis. It was concluded that the implementation strengthened control, monitoring, coordination, and self-management of the entrepreneurial activities linked to the organization.

Keywords: Web application, mobile application, entrepreneurship management, women entrepreneurs, digital transformation.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN	1
1.1. Antecedentes	1
1.2. Planteamiento y delimitación del problema.....	2
1.3. Preguntas de investigación	4
1.3.1. Pregunta General:	4
1.3.2. Preguntas Específicas.....	4
1.4. Justificación.....	4
1.5. Objetivos de investigación	6
1.5.1. Objetivo general	6
1.5.2. Objetivos específicos	6
2. REVISIÓN DE LA LITERATURA	7
2.1. Fundamentos teóricos.....	7
2.1.1. Aplicación móvil.....	9
2.1.2. Aplicación web	18
2.1.3. Gestión de Emprendimientos	25
2.2. Predicción Científica	32
3. METODOLOGÍA	33
3.1. Enfoque, alcance, diseño y tipo de investigación	33
3.2. Unidades de análisis	34
3.3. Técnicas e instrumentos de investigación	35
3.4. Técnicas de análisis de datos	36
3.5. Operacionalización de las variables.....	37
4. RESULTADOS	41
4.1. Resultado del primer objetivo específico.....	41
4.1.1. Aplicación de las encuestas a las Emprendedoras.....	41

4.1.2.	Identificación de los procesos en la gestión de emprendimientos.	44
4.1.3.	Diagrama BPD del proceso para la gestión de emprendimientos.....	44
4.2.	Resultado del segundo objetivo específico	46
4.2.1.	Comparación de almacenamiento para web y móvil.....	47
4.2.2.	Comparación de patrones arquitectónicos.....	48
4.2.3.	Tecnologías de <i>frontend</i>	49
4.2.4.	Tecnologías de <i>backend</i> y servicios en la nube.....	50
4.2.5.	Gestión de estado en la aplicación móvil	52
4.3.	Resultado del tercer objetivo específico: Aplicación móvil y Web.....	52
4.3.1.	Nomenclatura y Logotipo	52
4.3.2.	Marco de Trabajo Scrum.....	53
4.3.3.	Sprint 1	53
4.3.4.	Sprint 2	108
4.3.5.	Sprint 3	152
4.3.6.	Sprint 4	160
4.4.	Resultado general	177
4.4.1.	Implementación del sistema Web y Móvil	177
4.4.2.	Validación de la propuesta de intervención.....	182
4.4.3.	Validación de la Hipótesis	186
5.	DISCUSIÓN	189
6.	CONCLUSIONES Y RECOMENDACIONES	191
6.1.	Conclusiones.....	191
6.2.	Recomendaciones	192
7.	REFERENCIAS	194
8.	ANEXOS	207

1. INTRODUCCIÓN

En distintos contextos, el emprendimiento femenino se desenvuelve en escenarios cada vez más digitalizados, sin embargo, todavía persisten limitaciones que reducen su presencia y proyección. Mashapure et al. (2025) advierten que, la formación digital limitada, falta de infraestructura y barreras financieras restringen el uso tecnológico de las emprendedoras (p. 2). De ese modo, la evolución de los procesos comerciales vuelve pertinente el uso de soluciones tecnológicas que aporten visibilidad, articulen la oferta y la demanda, y favorezcan la autogestión en entornos donde aún predominan prácticas manuales y escasa presencia digital.

En relación con ello, Flores-Bañuelos et al. (2023) sostienen que el *marketing* digital y habilidades de *internet* se relacionan, apoyadas en *web*, correo y medios móviles (pp. 5, 11). Desde esta perspectiva, las aplicaciones *web* y móviles resultan relevantes para el emprendimiento, al fortalecer sus actividades, ampliar el contacto con clientes y propiciar un entorno comercial más ordenado, accesible y competitivo. Así, su implementación cobra importancia en contextos locales de apoyo al emprendimiento femenino, al fortalecer la visibilidad de las emprendedoras, organizar sus productos, dar seguimiento comercial e incorporar nuevas participantes en entornos digitales accesibles.

1.1. Antecedentes

Según Slinger et al. (2024), identificó que muchos micro emprendedores en Sudáfrica, tras la pandemia enfrentaban barreras para integrar tecnologías digitales, sumado al aumento de desempleo, provocó una brecha en la adopción de aplicaciones móviles por parte de microempresas, especialmente en sectores vulnerables. En este trabajo, se empleó una metodología cuantitativa a 221 encuestados, aplicando el modelo de Teoría Unificada de Aceptación y Uso de Tecnología. Se analizaron variables como expectativa de rendimiento, esfuerzo percibido y condiciones facilitadoras. Encontrando que la intención de adopción está fuertemente influida por la expectativa de rendimiento y la facilidad de uso para que las aplicaciones móviles sean efectivas (p. 1).

A nivel Latinoamérica, específicamente en Panamá, se identificó el uso de las aplicaciones móviles como una tendencia emergente. Tal como Díaz y Perez (2024), quienes abordaron la falta de soluciones digitales adaptadas localmente a emprendedores del sector cultural en la región del Azuero, donde se requería de una herramienta de gestión que permitiera mejorar la operatividad y visibilidad de los servicios. Se diseñó un prototipo de aplicación móvil, centrado en principios heurísticos de Nielsen y experiencia de usuario (UX), y facilidades de pago. Al brindar herramientas para aumentar la visibilidad en línea, se permitió mejorar la interacción con los usuarios, obtener *feedback* para los emprendedores, crecer la reputación y ampliar los contactos (p. 1).

En el contexto de Ecuador, Godoy-Rosero et al. (2022) evidenciaron que, pese al uso creciente de aplicaciones móviles, su integración en los procesos todavía es parcial; muchos micro emprendedores de Esmeraldas presentan dificultades para incorporarlas, sobre todo en contextos comerciales informales o emergentes. El estudio, realizado en 5763 microemprendimientos, halló que más de la mitad, alcanzando el 58%, promociona y ofrece productos y servicios por medio del uso de *Apps* de entrega y redes sociales. Además, el uso de estas herramientas se asocia positivamente con el flujo de dinero y con modelos de negocio con propuesta de valor, lo que refleja nuevas ideas y su potencial para fortalecer la gestión comercial y digital a nivel local (pp. 51-53).

1.2. Planteamiento y delimitación del problema

A partir del análisis de los antecedentes recopilados, se identifica que, pese al creciente uso de aplicaciones móviles, son limitados los estudios que abordan directamente la gestión de emprendimientos desde una perspectiva tecnológica y aplicada. La literatura se centra en adopción tecnológica, diseño de soluciones digitales y uso de herramientas móviles en contextos comerciales. Por ello, esta revisión sugiere investigar cómo fortalecer la gestión de estas iniciativas desde un enfoque local, orientado a una Sociedad Comercial de beneficio e interés colectivo que apoye a mujeres que buscan emprender.

El agrave de situaciones de carencia y dificultades de vida, han provocado el aumento del deseo por emprender y mejorar su situación. Como se indica en Chen et al.

(2025), un estudio sociodemográfico hecho a mujeres emprendedoras de 21 a 30 años, realizado en zonas rurales de Pakistán, encontró que mujeres inmersas económicamente en actividades productivas, tenían una alta experiencia laboral de 4 a más de 6 años. Siendo que, el 41% eran emprendedoras y el resto se desempeñaba en agricultura u otras profesiones. Evidenciando una problemática compartida con otros contextos en desarrollo, donde las mujeres participan en actividades emprendedoras o productivas bajo condiciones de vulnerabilidad, con pocas oportunidades, baja tasa de escolaridad, escaso apoyo institucional y dificultades para mantener sus negocios (pp. 2, 7).

Rivera-Barberán et al. (2024) menciona que, América Latina es la región con la mayoría de los emprendimientos femeninos (p. 105). Con esto en mente, en un estudio realizado en Colombia, por Arbulú-Ballesteros et al. (2024), se identificaron ciertos factores psicosociales como la motivación extrínseca, el entorno familiar, e incluso con mayor fuerza la innovación, influyen directamente en la actitud emprendedora de las mujeres universitarias, lo que a su vez potencia su intención de emprender. Sin embargo, elementos como la autoeficacia o el entorno sociocultural no muestran un efecto significativo, lo que sugiere limitaciones en la gestión de emprendimientos cuando no se consideran enfoques integrales que fortalezcan tanto habilidades como actitudes (pp. 1, 10).

Godoy-Rosero et al. (2022) mencionan que, a nivel ecuatoriano, la mayoría de los emprendedores no han logrado articular herramientas en modelos de negocio innovadores, limitándose a una presencia digital básica. Y, varios que podrían potenciar su crecimiento, no aprovechan métricas o automatizaciones, reflejando una brecha entre uso tecnológico superficial y gestión estratégica eficiente (p. 51). En este contexto, Sánchez-Riofrío et al. (2024) señalaron que, en el 2022 la mayoría de las mujeres no alcanzaban un empleo adecuado, siendo que sólo el 28.8% estaba en esta condición, frente al 41.1% de los hombres. Además, muchas emprendedoras carecen de capacitación, apoyo y herramientas para la sostenibilidad y crecimiento de sus negocios (pp. 4, 24).

En consecuencia, en la empresa 100 Mujeres S.A.S. B.I.C., del cantón Santo Domingo, se realizó una visita que permitió identificar una limitada gestión de los

emprendimientos. Principalmente, se constató una actualización limitada de productos y servicios, así como una gestión y seguimiento manual de pedidos y limitaciones en la inscripción de emprendedoras. Esta situación provoca baja visibilidad de productos y servicios, dificultades en la gestión y seguimiento de pedidos y menor agilidad en la inscripción de emprendedoras.

1.3. Preguntas de investigación

1.3.1. Pregunta General:

¿Cómo fortalecer la gestión de los emprendimientos vinculados a la empresa “100 Mujeres” del cantón Santo Domingo?

1.3.2. Preguntas Específicas

- ¿Cuáles son los procesos actuales en relación a la gestión de emprendimientos dentro de la empresa?
- ¿Qué herramientas y tecnologías son necesarias para el desarrollo e implementación de la propuesta realizada a la empresa?
- ¿Qué se puede desarrollar para el fortalecimiento de la gestión de los emprendimientos de la empresa 100 Mujeres del cantón Santo Domingo?

1.4. Justificación

En este sentido, el presente Trabajo de Titulación de grado, se justifica bajo el respaldo legal establecido por la Asamblea Constituyente (2008) de la Constitución de la Republica del Ecuador, en el numeral 2, del artículo 16, siendo aquel destinado a garantizar la disponibilidad universal de infraestructura de TI a todos los individuos o colectivos (p. 14). Asimismo, el numeral 3, del artículo 385, promueve el fomentar la ciencia, tecnología e innovación como herramientas que aportan al Buen Vivir (p. 185). Siendo, pertinente impulsar soluciones tecnológicas orientadas al fortalecimiento de los procesos, mediante la contribución sustancial hacia un desarrollo y bienestar de la sociedad óptimos.

Por lo tanto, en la Agenda 2030, de la Organización de las Naciones Unidas (2023), señala en el quinto objetivo de desarrollo sostenible, que la igualdad de género es un

objetivo transversal y debe ser el centro de las instituciones, los presupuestos y las normativas nacionales. A la par, el *ODS 8*, indica que las duras condiciones económicas llevan a más trabajadores hacia el empleo informal (pp. 22, 28). Por su parte, la Comisión Económica para América Latina y el Caribe (2022) señala que, las brechas en acceso a recursos productivos, financiamiento, tecnología y redes de información limitan la competitividad y expansión de los emprendimientos, requiriendo el diversificar hacia sectores más intensivos en conocimiento, tecnología y empleo de calidad (pp. 129, 131). En este contexto, la gestión del emprendimiento depende de factores que influyen al desarrollo y sostenibilidad, siendo relevante fortalecer la participación económica en emprendimientos liderados por mujeres.

Además, en la “Ley Orgánica de Emprendimiento e Innovación”, la Asamblea Nacional del Ecuador (2020) en su Art. 1 establece como objeto el fomentar el emprendimiento, la innovación y el desarrollo tecnológico; además, Igualmente, la Disposición General Segunda toma en cuenta a las plataformas digitales que promueven los emprendimientos, priorizando productos o servicios de compañías dirigidas por mujeres o grupos minoritarios (pp. 4, 43, 47). Este marco, resulta pertinente para implementar instrumentos tecnológicos que apoyen, viabilicen y fortalezcan la gestión del emprendimiento acorde con sus necesidades operativas, con fin de lograr más eficiente en procesos específicos de la organización.

En este contexto, el Reglamento General a la Ley Orgánica de Emprendimiento e Innovación por la Presidencia de la República del Ecuador (2020) en su Art. 2, establece que, el ámbito de aplicación comprende todas las actividades de carácter público o privado vinculadas con el desarrollo del emprendimiento y la innovación, en el marco de las diversas formas de economía, entre ellas la pública, privada, mixta, popular y solidaria, cooperativista, asociativa, comunitaria y artesanal (p. 2). A partir de este régimen normativo, resulta pertinente facilitar el acceso a los recursos tecnológicos y plataformas digitales, con fin de fortalecer el desarrollo de actividades vinculadas al emprendimiento. Este ecosistema

emprendedor, incluye a grupos relevantes como el femenino, lo cual, promueve un desarrollo económico más equitativo y sostenible.

Por último, en la Ordenanza Municipal “N.º M-014-WEA”, el Gobierno Autónomo Descentralizado Municipal de Santo Domingo (2021), manifiesta la importancia de impulsar y promover nuevas alternativas y acciones solidarias por parte del GAD Municipal de Santo Domingo. Este argumento se presenta, en concordancia con las políticas nacionales, que propendan a mejorar el nivel y calidad de vida de los ciudadanos a través de la generación de actividades comerciales, turísticas, constructivas, productivas, industriales, entre otras (p. 2). Esto demuestra el interés por fortalecer iniciativas orientadas al desarrollo económico y social, cuyo propósito es elevar las condiciones de vida de la población mediante el impulso en diversos sectores estratégicos.

1.5. Objetivos de investigación

1.5.1. Objetivo general

Implementar una aplicación *web* y móvil para el fortalecimiento de la gestión de emprendimiento en la empresa “100 Mujeres” del cantón Santo Domingo durante el año 2025.

1.5.2. Objetivos específicos

- Identificar los procesos actuales en la empresa “100 Mujeres”, con respecto a la gestión de los emprendimientos.
- Determinar las herramientas y tecnologías necesarias para el desarrollo e implementación del proyecto propuesto a la empresa.
- Desarrollar la aplicación *web* y móvil para la gestión de emprendimientos en la empresa “100 mujeres” del cantón Santo Domingo.

2. REVISIÓN DE LA LITERATURA

2.1. Fundamentos teóricos

Para fundamentar teóricamente, se prescindió del método vertebrado de índices, propuesto por Hernández-Sampieri y Mendoza-Torres (2018), el cual sugiere vertebrar los temas, al elaborar un esquema jerárquico, el cual se bifurcará progresivamente desde un índice general, hasta alcanzar un nivel específico, incorporando las referencias según los subtemas. Con este proceso, se obtiene una estructura clara y ordenada del contenido (p. 89), en la figura 1 y 2, se indican la estructura seleccionada para las variables independientes que son aplicación móvil y *web*, y, por último, la figura 3 comprende la variable dependiente gestión de emprendimientos.

Figura 1. Panorama bibliográfico de la variable independiente aplicación móvil

Fases	Apartado	Sub apartado
Aplicación móvil	Plataforma de ejecución	Android iOS Multiplataforma
	Entorno de desarrollo móvil	Flutter Dart Librerías y paquetes externos (Pub.dev)
	Arquitectura del software	Clean Architecture Estructura modular del código Navegación entre pantallas Gestión de estado de la aplicación
	Diseño funcional del sistema móvil	Diseño de interfaz de usuario (UI) Diseño de experiencia de usuario (UX) Capacidades del dispositivo móvil
	Almacenamiento y servicios backend	Bases de datos en la nube: Firebase Bases de datos local: SQLite Comunicación con servicios externos: APIs Seguridad de datos y autenticación

Figura 2. Panorama bibliográfico de la variable independiente aplicación web

Fases	Apartado	Sub apartado
Aplicación Web	Lenguaje de Programación	PHP JavaScript CSS
	Herramientas de Desarrollo	React Angular Node.js
	Software de Programación	Visual Studio Code Sublime Text Android Studio
	Base de Datos	Relacional MySQL Firebase
	Metodología de Desarrollo	Metodologías Tradicionales Modelo En Cascada

Figura 3. Panorama bibliográfico de la variable dependiente gestión de emprendimientos.

Fases	Apartado	Sub apartado
Gestión de Emprendimientos	Gestión de Operaciones y Logística	Procesos productivos Control de calidad Optimización de inventarios
	Gestión de Talento Humano	Reclutamiento y selección Capacitación y evaluación de desempeño Cultura organizacional
	Gestión Legal y Aspectos Regulatorios	Constitución legal de la empresa Licencias Permisos
	Gestión de Redes y Ecosistemas Emprendedores	Alianzas estratégicas Incubadoras y aceleradoras Emprendimiento social colaborativo

2.1.1. Aplicación móvil

Es un *software*, que permite al usuario, interactuar con el dispositivo móvil para cumplir un propósito. Randhawa (2022), las aplicaciones móviles del mundo real proporcionan funcionalidades enriquecidas, se componen de conocimientos de sistemas operativos, bases de datos, protocolos de red, diseño *UI/UX*. En sí, forman parte de un ecosistema en aumento, que requiere cumplir con tiempos de lanzamiento. Además, son hechas para varias plataformas de *hardware*, con diferentes configuraciones de pantalla táctil, duración de la batería, sensores y otros accesorios. Diferentes bandas de red inalámbrica, y sistemas operativos, de diferentes versiones de este, con actualizaciones posibles; y en diferentes ubicaciones y momentos (pp. vi, 507).

2.1.1.1. Plataforma de ejecución

Según Biørn-Hansen et al. (2020) la plataforma objetiva, hace énfasis a los diversos sistemas operativos, proveídos por el *software* y *hardware* de los fabricantes. Siendo, usualmente *Android* y *iOS*, entre otros. En este sentido, existen varios tipos de enfoques de desarrollo que determinan la plataforma, comúnmente de forma específica para cada sistema operativo con un enfoque nativo, utilizando lenguajes y herramientas propias de cada plataforma. Existe heterogeneidad entre las principales plataformas móviles, debido a sus interfaces de usuario, experiencia de usuario, lenguaje de programación y ecosistemas, popularizando el uso de marcos de desarrollo multiplataforma (pp. 2997-2998, 3001).

2.1.1.1.1. Android

En el último informe de seguridad de *Android*, realizado por Google (2024), se indica que es un sistema operativo de código abierto, diseñado para varios tipos de dispositivos, el cual, incorpora características de seguridad en un ecosistema seguro, proporcionando una plataforma para desarrolladores y fabricantes. En parte, se caracteriza por incorporar funciones de seguridad y mantener una arquitectura modular basada en capas: aplicaciones, marco de trabajo, bibliotecas nativas, entorno de ejecución (*ART*), capa de abstracción de hardware (*HAL*), el núcleo *Linux*, y elementos de seguridad. La versión más

reciente al momento es *Android 15*, fortalece las capacidades de protección de los dispositivos e introducir funciones de protección para los dispositivos (pp. 9-11).

2.1.1.1.2. iOS

Por lo mencionado en Sathiya-Priya y Vijay-Ananth (2025) *iOS* es un sistema operativo propiedad de *Apple Inc.*, el cual fue lanzado en el año 2007, junto con el primer *iPhone*, dando una mejor seguridad, optimización y experiencia de usuario para la época. Es utilizado en dispositivos como *iPhone*, *iPad*, y dispositivos táctiles *iPod*. Está basado en el *Kernel Darwin*, sistema operativo similar a *Unix*. *iOS* forma parte de un ecosistema cerrado, con control exclusivo sobre su desarrollo y distribución. Ofrece un sistema operativo poderoso e intuitivo que ofrece una experiencia de usuario fluida (pp. b69, b72).

2.1.1.1.3. Multiplataforma

Biørn-Hansen et al. (2020) expresan que los desarrollos multiplataforma, utilizan generalmente, un único código base, desde el cual puede ejecutarse en múltiples plataformas. Del mismo modo, en la primera figura, adaptada de Majchrzak et al. (2015), detallan que existen cinco enfoques principales de desarrollo multiplataforma: Aplicaciones *web* progresivas (*PWA*); híbridas; basadas en tiempo de ejecución (*interpreted/runtime-based*); desarrollo dirigido por modelos (*MDSD*); y, por último, los enfoques de compilación. A diferencia, del uso de soluciones *web* que prescinden del uso *HTML* y *JavaScript*. El verdadero desarrollo multiplataforma, se realiza mediante *frameworks*, tales como *Flutter*, *React Native*, *Xamarin* (pp. 2998, 3001-3003).

2.1.1.2. Entorno de desarrollo móvil

Como se explica en Alessandria y Kayfitz (2021), la creación del entorno de desarrollo es el primer paso esencial para iniciar un proyecto en una nueva plataforma, incluyendo la instalación del *kit* de desarrollo de *software SDK*, la configuración de herramientas del sistema operativo móvil, *Android/iOS* y la elección del entorno de desarrollo integrado (*IDE*) adecuado. Además, utilizando paquetes se puede favorecer una creación viable y rápida del *software* móvil (pp. 7, 381). En este sentido, los desarrolladores como se indica en Sarab et al. (2021) , tienen la posibilidad de configurar y personalizar

sus aplicaciones móviles, mediante el entorno de trabajo, la plataforma de diseño y desarrollo (p. 74). En otras palabras, se podría definir como el conjunto comprendido entre tecnologías, lenguajes y librerías, que permiten el desarrollo móvil.

2.1.1.2.1. Flutter

Según Randhawa (2022), *Flutter* es un entorno desarrollo multiplataforma que utiliza *Dart*, y con el cual, se crean aplicaciones para varias plataformas móviles (p. 222). A la par, Payne (2019) lo definió como un conjunto de herramientas que funciona en *iOS*, *Android*, en la *Web* y para escritorio. Es Gratis, de código abierto, es respaldado por colaboradores de *Google* y el mundo, es ampliamente usado, y lo describe rápido porque compila a aplicaciones verdaderamente nativas (p. 4). Y para su funcionamiento, como indica Napoli (2019), se requiere descargar e instalar el *SDK* de *Flutter* desde su sitio web. Se puede instalar en *Windows*, *Linux*, o *macOS*. Sin embargo, para desarrollar en *iOS* se requiere una computadora *Mac* y *Xcode* (p. 13).

2.1.1.2.2. Dart

Alessandria y Kayfitz (2021) indica que *Dart* fue creado en 2011 por *Google*, siendo un lenguaje de programación conservador en sus orígenes, que buscaba crear un entorno de programación predecible y estable (p. 41). En este sentido, Napoli (2019) detalla que *Dart* es el lenguaje de programación utilizado para desarrollar aplicaciones con *Flutter*. Mismas que son rápidas, debido al compilador por anticipado a código nativo (*AOT*). Esto, se complementa con la compilación en tiempo de ejecución (*JIT*), la cual, permite mostrar los cambios del código con la función de recarga en caliente con estado de *Flutter* (p. 4). A la par, Payne (2019) destaca que *Dart* es fuertemente tipado, orientado a objetos y procedural. Añade características interesantes que embellecen el uso del lenguaje (p. 287).

2.1.1.2.3. Librerías y paquetes externos (Pub.dev)

Como expresa Payne (2019), existen bibliotecas o herramientas desarrolladas previamente que pueden ser utilizadas en la creación de Aplicaciones con *Flutter*. Los pasos se resumen a buscar, agregar al archivo *pubspec.yaml*, importarlas desde los archivos *Dart*, y hacer uso en el código (p. 208). Del mismo modo, Napoli (2019) menciona

que *Flutter* admite paquetes de terceros, mismos que contienen su propia lógica y son de fácil acceso. Existiendo dos tipos: los *Dart*, están escritos en *Dart* y contienen dependencias específicas de *Flutter*, y los *Plugin* escritos en *Dart* que expone la *API*, combinando código de plataformas *Android* y *iOS*, con el objetivo de tener compatibilidad (p. 38).

En este sentido, Alessandria y Kayfitz (2021) destacan que el ecosistema de *Flutter* mantiene paquetes a diario, compatibles a ciertas plataformas y versiones, son publicados en el sitio web: <https://pub.dev>. Todos los paquetes son de acceso abierto, no pueden ser removidos tras publicarse, y todos los paquetes dependen de los ya publicados (p. 384).

2.1.1.3. Arquitectura del software

Randhawa (2022) menciona que el diseño de alto nivel o arquitectura de *software* implica diagramas de paquetes, componentes y de despliegue (p. 217). En detalle, Richards y Ford (2020) indica que suele definirse como el plano del sistema, o también como el plan de desarrollo de un sistema. Denotando se debe comprender el contenido de un plano o hoja de ruta. Comprende las características y estructura del sistema, mismo que debe soportar decisiones de arquitectura, y sus principios de diseño. La estructura hace referencia a la forma de implementación del sistema. Como características se definen criterios de funcionalidad ortogonal. Un principio de diseño es una guía general orientativa, a diferencia de una decisión arquitectónica que comprende una regla concreta a cumplir estrictamente (pp. 3-7).

2.1.1.3.1. Clean Architecture

Alessandria y Kayfitz (2021) mantener una arquitectura limpia, implica no prescindir de código adicional, simplificando la estructura (p. 230). En este sentido, Moore (2025) en la sección *Clean architecture* del capítulo 3, indica que, si bien hay muchas arquitecturas, se debe contar con aquellas que permiten crear un código fácil de usar, probar y actualizar. Siendo la *Clean architecture*, un patrón considerado como filosofía de diseño de *software*, la cual separa responsabilidades y prioriza una lógica central del negocio independiente a elementos externos como *Frameworks*, bases de datos, e interfaces de usuario. Entre sus principios comprende la división en capas independientes; reglas de dependencia, donde

las capas internas, no deben depender de las externas; la abstracción, uso de interfaces para facilitar cambios; y la testeo, mediante el uso de componentes con una sola responsabilidad son más fáciles de probar (p. 85).

2.1.1.3.2. Estructura modular del código

Según Richards y Ford (2020) el modularidad diseña los sistemas, bajo un principio arquitectónico organizativo. El cual, describe agrupaciones lógicas de código relacionado, tales como clases de un lenguaje orientado a objetos, funciones de uno estructurado o funcional. Se ve reflejado en el uso de módulos, contienen relación con su propósito. En este sentido, el código según Arquitectura referirá al agrupamiento relacionado (pp. 37-39). Esto hace referencia a la estructura de carpetas y del código del proyecto, considerando la posibilidad de separar funcionalidades, por carpetas dentro de cada una, incluyendo capas de datos, dominio, y presentación, cada una con subcapas según requiera. De este modo, creando un código escalable y fácil de mantener.

2.1.1.3.3. Navegación entre pantallas

En base a lo expresado en el capítulo 8 en Moore (2025), la navegación es la forma de cambiar entre pantallas (p. 235). En el caso de *Flutter*, Napoli (2019) indicaba que el *widget Navigator* gestiona una pila de rutas entre páginas. Con el cual, se encarga de mandar datos de la página destino y de regreso a la original. Y, para navegar entre páginas se requiere utilizar métodos. En general, este es uno de los componentes esenciales dentro de una aplicación móvil, debido a que una buena navegación se traduce a una buena experiencia de usuario, otorgada por un acceso a la información óptimo (pp. 177-178).

2.1.1.3.4. Gestión de estado de la aplicación

En el capítulo 6, *State Management Fundamentals*, de Moore (2025), se menciona que el estado es cualquier dato interno, con la capacidad de variar y actualizar la interfaz de usuario de la aplicación. Específicamente, en *Flutter* se tiene un estado local, el cual gestiona dentro de la clase de ese *widget*, a diferencia contra el estado de la *App* que es compartido en diferentes partes. Es posible manejar el estado con *stateful widgets*, sin embargo, existen paquetes que amplían la funcionalidad, entre estos, se encuentran:

Provider, *Riverpod*, *Business Logic Component (BLoC)*, *GetIt*, *Redux*, *MobX*. Los más relevantes son: *Provider*, propiedad de *Google*, provee acceso del estado a los *widgets* inferiores; *BLoC*, facilita la implementación del patrón de diseño, cuyo propósito es separar la interfaz de usuario de la lógica empresarial (pp. 163-165, 167).

Payne (2019) indica que *BLoC* fue creado por *Google*, así que naturalmente fue adoptado por la comunidad de *Flutter*. Además, señala que hay una amplia comunidad, dado que es un patrón sólido. Sin embargo, hay que tener presente que hay que saber dónde se debe inyectar un *BLoC* (pp. 200-201). En este sentido, Napoli (2019) considera que las clases *BLoC* son independientes de la plataforma y no dependen de paquetes o clases específicas (p. 432).

2.1.1.4. Diseño funcional del sistema móvil

Armijos et al. (2021) indican que el mercado de aplicaciones móviles ha crecido y, con ello, también las funcionalidades y servicios ofrecidos, por lo cual, el desarrollo de estos debe ir encaminado a garantizar la experiencia de usuario (p. 18). Por ello, en base a lo mencionado en Casado (2021), el diseño *UX*, podrá mostrar los problemas que los usuarios experimentan con los productos o servicios. Incluyendo, el diseño de interfaz de usuario, la disciplina con enfoque hacia los elementos propios de la interfaz, otorgando funcionalidades con la que los usuarios interactuaran. Así, el diseño se centra en el funcionamiento, y la interacción del humano y la máquina, como unión y auge del diseño *UI/UX* (pp. 24, 57, 208). Considerar el diseño *UI/UX*, es relevante como indica Yunhee (2024) integrar *UX/UI*, con accesibilidad es de especial atención para optimizar la experiencia de los usuarios, con fin de fortalecer resultados empresariales, y el comercio electrónico eficiente e inclusivo (p. 10).

2.1.1.4.1. Diseño de interfaz de usuario (UI)

En base a Camburn et al. (2017), empieza desde el concepto de un prototipo, el cual esquivale al concepto o la representación de un producto o diseño final. Esto es relevante, dado que permite plasmar la idea por medio de maquetas; y, en consecuencia, obtener *feedback* (pp. 1, 12). En este sentido, antes del desarrollo como menciona Pressman y Maxim (2019) la creación de un prototipo y su revisión con el cliente, durante la

programación sirve para evitar perder tiempo con un desarrollo mal direccionado. En este sentido, es de bastante utilidad crear de interfaz de usuario, por medio de un prototipo funcional. Especialmente, porque se pueden evitar malentendidos entre los desarrolladores, y las partes interesadas (p. 62).

Según Casado (2021) la interfaz de usuario es aquella que se le aplica elementos diseño, estética y estilo. Siendo habitual antes de pasar a lenguaje de código (p. 57). En este sentido, las consideraciones a tomar en cuenta, en la interfaz de usuario de una aplicación móvil, como se menciona en Pressman y Maxim (2019) al diseñar interfaces se debe tomar en cuenta las limitaciones físicas, impuestas por los dispositivos móviles. Dentro de una interfaz, se tienen componentes visuales. Se debe considerar especial detalle, en las características necesarias para adaptar la apariencia, en distintos aspectos, tanto gráficos, como de texto, e incluso en controles de la *IU* funcionen correctamente en diferentes tipos de pantalla (pp. 227, 257).

2.1.1.4.2. *Diseño de experiencia de usuario (UX)*

Según Casado (2021) se trata del componente vinculado a la experiencia de usuario, alude a quien hace uso del servicio o producto final. Por ende, se busca identificar los problemas o sensaciones experimentadas por los usuarios (pp. 14-15, 24, 76). Esto se apoya con lo mencionado en Pressman y Maxim (2019), el diseño de experiencia de usuario significa la búsqueda de mejora constante del producto creado. Igualmente, otorgando una interacción entre producto y usuarios, con sensación usable, accesible y placentera. En consecuencia, para ser efectivo, debe considerarse al inicio del ciclo de vida del proyecto, teniendo en cuenta la acción y expectativa del usuario en cada etapa del desarrollo (pp. 233-234). En conjunto a lo mencionado por Casado (2021) quien cito a Morville (2004) detallaba que la *UX* tiene por fin ser valioso, útil, localizable, creíble, deseable, accesible, y utilizable (pp. 14-15, 24, 76).

Dentro de un *software*, Pressman y Maxim (2019) destaca que la accesibilidad es aquella que permite dar la interacción ideal, para percibir, entender y poder navegar e interactuar con los productos informáticos. Siendo un objetivo el proporcionar herramientas

de *hardware* o *software*, para que reduzcan o eliminen los impedimentos a los usuarios al usar una aplicación (p. 237). A la par, Casado (2021) expresa que las experiencias a diseñar también deben ser inclusivas al tomar en cuenta primero a aquellos usuarios extremos, tales como personas de bajo nivel educativo, poco poder adquisitivo, y usuarios con algún tipo de discapacidad, pueda ser de movilidad, visual, o auditiva (pp. 200-201).

2.1.1.4.3. Capacidades del dispositivo móvil

Por lo mencionado Velásquez et al. (2019), las funcionalidades asociadas a una aplicación comprenden, el conjunto de funcionalidades o tecnologías comunes, comprendiendo conectividad inalámbrica como *WiFi*, *GPS*, *Bluetooth*, tecnología *NFC*; componentes de *hardware* tales como batería, tarjetas *SIM*, memorias; sensores como de movimiento, luz, localización; y otros elementos. Siendo importante entender los requisitos, para construir las historias de usuarios, para posterior probar las funcionalidades de la *App* móvil (pp. 44-45). A la par, coincide con Thomas et al. (2018), donde se indica que las aplicaciones poseen un conjunto de características, las cuales aprovechan las capacidades del dispositivo con buen rendimiento, funcionalidades sin internet, e incluso funcionamiento en segundo plano que tiene la capacidad de notificar al usuario (p. 589).

2.1.1.5. Almacenamiento y servicios *backend*

Como lo explican Cole et al. (2007), citado en Kuzmiakova (2023), los datos tienen un origen, provienen de bases de datos y se almacenan versátilmente desde diferentes fuentes. La forma para almacenarlos busca facilitar su consulta, extracción o análisis. Los sistemas de almacenamiento, el análisis de negocios y los almacenes de datos son términos que se refieren al diseño, la implementación y el uso de almacenes de datos (p. 10). Va de la mano, con lo dicho en Hervás (2023) Conjunto de información almacenada, la cual otorga la manipulación de datos. En general, comprende el conjunto de datos estructurados y no redundantes, organizados independientemente del uso y posterior implementación. Mismos que deben ser flexibles, sin que las aplicaciones influyan en el diseño *backend*; adaptándose óptimamente frente a futuras características, manteniendo una amplia disponibilidad; y mostrando los datos según el tipo de usuario (pp. 14-15).

2.1.1.5.1. Bases de datos en la nube: Firebase

Titmus (2021) describe a las aplicaciones nativas en la nube como escalables, débilmente acopladas, resilientes, gestionables y observables (p. 6). Esto se relaciona, bajo la perspectiva de uso, donde los sistemas de base de datos en la nube proporcionan a los proyectos, como indica Silva et al. (2022), *Firebase* otorga una tecnología rápida y eficiente. Se trata a un sistema de gestión de datos, el cual permite operar con grandes volúmenes de datos no estructurados. También, entre otras características de este *backend* útil para aplicaciones móviles, comprende el almacenamiento de archivos, uso de base de datos y análisis de estos. Fácil uso de funciones, optimización de la autenticación de usuarios y el envío flexible de notificaciones (p. 3).

2.1.1.5.2. Bases de datos local: SQLite

Según Lemahieu et al. (2018) las bases de datos locales son aquellas que usualmente funcionan en dispositivos móviles. Mismos que siempre permanecen conectados, requieren tener un procesamiento limitado, almacenamiento y duración de batería (pp. 133-134). En este contexto, fue creado *SQLite*, que como mencionan Allen y Owens (2011) se trata de una solución de acceso abierto, con las capacidades de una base de datos relacional embebidas, diseñada para que las aplicaciones manejen los datos sin sobrecargarse. Posee ciertas características y soporte para *SQL* estándar, entre otras funciones asociadas. En otras palabras, equivale a una base de datos incrustada, la cual coexiste dentro de la aplicación, sirviendo de proceso para la misma (pp. 1, 8).

2.1.1.5.3. Comunicación con servicios externos: APIs

Una *API*, según Lemahieu et al. (2018) se refiere a una tecnología de servicios *web*, la cual permite que los sistemas de bases de datos interactúan con otras aplicaciones. Un servicio *web* podrá depender directamente de una base de datos subyacente para ejecutar su funcionalidad. Además, la propia base de datos puede comportarse como proveedor o consumidor de servicios *web*. Por tanto, procedimientos almacenados o disparadores pueden incluir llamadas a servicios externos, lo cual amplía sus capacidades operativas. A continuación, cabe destacar que *REST* ha sido ampliamente adoptado como arquitectura de

servicios por su simplicidad, al estar construido sobre *HTTP*. Esta característica permite su invocación directa desde navegadores y facilita su uso en servicios *web* básicos o puntuales. En consecuencia, *REST* se ha convertido en el enfoque preferido por plataformas modernas para ofrecer *APIs* a desarrolladores externos (pp. 678, 681, 1144).

2.1.1.5.4. Seguridad de datos y autenticación

El acceso a sistemas de bases de datos debe garantizarse mediante mecanismos sólidos de autenticación. Según Kuzmiakova (2023), quien cita a Jarke y Koch (1984), indican que se requiere que todos los usuarios utilicen credenciales como contraseñas o métodos biométricos, incluyendo lectores de huellas, para ingresar al sistema, ya que esta información es esencial para asignar derechos de acceso a consultas y reportes (p. 31). Asimismo, Kuzmiakova (2023) resalta que los sistemas de gestión de bases de datos implementan mecanismos complejos que permiten la concurrencia segura de múltiples usuarios sin comprometer la integridad del sistema (p. 32). Por otra parte, Kuzmiakova (2023), al citar a Kao y Garcia-Molina (1994), sostiene que la administración de respaldos y recuperaciones es clave, ya que los *DBMS* ofrecen protocolos de seguridad y capacidades tanto estándar como personalizadas para la restauración de datos tras fallas como pérdidas de energía o sectores dañados del disco. Esta funcionalidad resulta indispensable para mantener la seguridad general de un sitio *web*. Finalmente, la autora añade que los *DBMS* aceptan consultas de usuarios finales mediante diversas configuraciones de red, lo que permite el acceso incluso desde navegadores *web* (p. 32).

2.1.2. Aplicación web

Según Hernández-Bejarano y Baquero-Rey (2020) una aplicación *web* constituye un programa informático diseñado para acceder a través de la red, como Internet, mediante el uso de un navegador *web* en particular. Las aplicaciones *web* no requieren instalación en el equipo, lo que significa una gran ventaja en términos de espacio y accesibilidad por lo que son utilizadas desde cualquier dispositivo con Internet. Este modelo ayuda la visualización de sitios *web* que integran texto, imágenes, videos y otros contenidos multimedia,

permitiendo a los usuarios navegar a través de enlaces y así lograr la satisfacción y experiencia de usuario (p. 17).

2.1.2.1. Lenguaje de Programación

Llerena-Izquierdo (2023), lo define como un conglomerado de simbologías que permiten crear algoritmos que resuelven problemas específicos, a través de una secuencia de instrucciones que el computador sigue para realizar una tarea. Así, un lenguaje de programación cumple una arquitectura sintáctica para ser ejecutado o procesado en el computador, siendo la sintaxis el conjunto de reglas usadas para crear el programa (p. 77).

2.1.2.1.1. PHP

Celi-Párraga et al. (2023) afirma que es posible crear aplicaciones *web* complejas con un diseño agradable, facilitado por el uso de *frameworks* CSS como *Bootstrap* y librerías como *jQuery* para agilizar la interactividad y también crear aplicaciones *web*, desde el *Frontend* hasta el *Backend* con *PHP* (p. 67). En cambio, Hernández-Bejarano y Baquero-Rey (2020) destacan que el desarrollo se construye sobre pilares como *HTML*, conocido como un lenguaje de marcado para sitios *web*, y *CSS*, un lenguaje de estilo en cascada encargado de la presentación de los documentos. Dentro del diseño *web*, la parte visual de la página *web*, *PHP* se presenta como un lenguaje de programación de código abierto, apropiado al desarrollo *web* dinámico, cuya integración al usar *HTML* favorece a la construcción de páginas con mayor funcionalidad (pp. 13, 69, 71, 142).

2.1.2.1.2. JavaScript

Manifiesta Celi-Párraga et al. (2023) que esta tecnología agiliza la interacción con el usuario, lo cual, permite que varios tipos de aplicaciones *web*, se ejecuten en distintas plataformas. Por otro lado, para crear la interacción en una página *web* se necesita de un lenguaje de programación y el formato *CSS*, ya sea para algo básico como una calculadora, ese evento sería controlado en el código de *JavaScript*. Además, este constituye una herramienta fundamental para el *Frontend*, ya que dispone de *frameworks* y librerías que facilitan la gestión y el trabajo de los desarrolladores (p. 39).

2.1.2.1.3. CSS

Según Celi-Párraga et al. (2023) en el desarrollo del *Frontend*, CSS es un lenguaje de programación encargado del diseño de la página *web*, ya que este permite dar color, tamaño, posición, fuente de la página *web* (p. 23) . Por otro lado, Gauchat (2017), detalla que el CSS es un facilitador de etiquetas enfocado en convertir y mejorar la sección visual de una página *web*, ya que se usa para dar a la página su estructura y CSS (p. 83).

2.1.2.2. Herramienta de Desarrollo

Moreno-Pérez (2018) indica que no se utiliza un único programa, sino un conjunto de herramientas para facilitar el desarrollo de la aplicación. A este conjunto, se le denomina un entorno de desarrollo. Entre ellos destaca el *IDE*, cuyas siglas en ingles son *Integrated Development Environment*, puesto que facilita las tareas de desarrollo de *software* y se componen del editor de texto, compilador y constructor de interfaz gráfica. Por otro lado, el entorno de desarrollo en la nube se aloja en un servidor en internet, cuya principal ventaja es que el trabajo se puede hacer desde cualquier lugar y dispositivo (pp. 36-37).

2.1.2.2.1. React

En Altadill-Izura (2022) se menciona que *React* se encuentra en pleno uso, dada su utilidad en las mejores empresas del mundo y relevantes de los últimos años, dada su robusto conjunto de módulos que poseen la capacidad de elaborar y desarrollar casi cualquier necesidad desde cero y con el menor esfuerzo posible. Una característica única de *react* es que permite organizar componentes de interfaz web y posteriormente nuevamente hacer uso de estos. Así, aportando al uso de *HTML* y mediante la aplicación de un *DOM* virtual, facilita su uso. Es una excelente alternativa para desarrollar aplicaciones *web* o móviles mediante el uso de la librería *React Native*, ya que el aprendizaje comprende los principios fundamentales para la total aplicación de *JavaScript* (pp. 72, 84, 316).

2.1.2.2.2. Vite

Según Hoyos (2024), hace referencia que *Vite* tiene como concepto propio que es una Aplicación *React* mediante la gestión de una Plataforma de Desarrollo de nivel dado si enfoque y su versatilidad en el campo actual. También menciona que se logra esta

singularidad mediante el uso de un servidor nativo de desarrollo *ESM*, que comúnmente es conocido por su rapidez y las actualizaciones siempre a disposición para el o los usuarios antes o durante el desarrollo. Dicha combinación crea una sinergia y experiencia mucho más eficiente, ya que ayuda la comprensión de la lógica al momento de trabajar con desarrollos complejos y robustos mediante el uso de componentes propios de cómodo acceso y amplio uso (pp. 28-29).

2.1.2.2.3. Node.js

Bocangel-Weydert et al. (2024), explican que node.js es una multiplataforma de código abierto, ya que se trata de un entorno de código que ejecuta y permite el uso de *JavaScript*, mediante su diseño que permite trabajar de manera eficiente. De este modo, *Node.js* es actualmente una excelente opción para la construcción de *APIs* y servidores *web*, puesto que gestiona los microservicios de manera escalable, ya que permite el uso de *JavaScript* en el ecosistema de cliente-servidor. En cuanto al desarrollo ayuda mayormente en el manejo de paquetes, para facilitar su uso en lo posterior, puesto que permite a los desarrolladores reutilizar el mismo y facilitar de trabajo de manera sencilla (pp. 211 - 212).

2.1.2.3. Software de Programación

Lira-Camargo y Lira-Camargo (2021) señalan que el *software* permite llevar a cabo tareas concretas en una computadora. El *software*, por su parte, proporciona las operaciones y es el fundamento del desarrollo tecnológico. Asimismo, engloba aplicaciones y sistemas operativos. Por último, el *software* de programación brinda interfaces amigables para el usuario e incorpora herramientas que permiten desarrollar programas informáticos empleando diversas alternativas y lenguajes de programación (pp. 10-11).

2.1.2.3.1. Visual Studio Code

Bin-Uzayr (2023), recalca de *Visual Studio Code*, como una herramienta de desarrollo de programación, misma que ofrece simplicidad mediante una gama de combinaciones que permiten tener una experiencia mucho más fluida, ya sea para programación básica o avanzada. El autor también manifiesta que el editor incluye integrado el soporte necesario para funciones de autocompletado como *IntelliSense*, lo cual

es una semántica de código en donde predomina la comprensión, navegación y refactorización de código. También es compatible para trabajar con tecnologías tales como *HTML*, *CSS*, *Sass*, *Less* y *JSON*, lo cual, mediante la integración de paquetes en distintos repositorios, ofrece un excelente flujo de trabajo, sea para codificación o levantamiento de producción como *Git*, esto gracias a su flexibilidad de aplicaciones nativas (p. 2).

2.1.2.3.2. Sublime Text

Según Chanchlani y Gune (2025), aunque *Sublime Text* pueda ser menos preferido que otros editores, sigue siendo una opción que muchos desarrolladores tienen en cuenta dada su velocidad y capacidad de personalización. Siendo que este editor se emplea para tareas de codificación, mientras que sistemas de gestión de contenido como *WordPress* y *Drupal* están orientados a los usuarios, que necesitan gestionar sitios *web* y adaptarse a distintas necesidades del desarrollo (pp. 4121, 4123-4124).

2.1.2.3.3. Android Studio

Con base a lo mencionado por Usca-Paguay (2023), uno de los entornos de desarrollo fundamentales es *Android Studio*, puesto que hace posible la creación de *software* dirigido a varios dispositivos, desde equipos de menor escala hasta plataformas como televisores. Siendo que *Google*, impulso su desarrollo como una herramienta de código abierto, la cual cuenta con actualizaciones constantes de las librerías y con sus respectivas guías. Además, destaca que cuenta con una amplia gama de desarrolladores para todo tipo de aplicaciones, permitiendo una contribución recursiva y constante, creando diversas aplicaciones utilizadas por millones de usuarios en todo el mundo (p. 11).

2.1.2.4. Base de Datos

Según Saltos-Viteri y Franco-Bayas (2019), el uso prolongado de bases de datos ha permitido consolidar al modelo relacional como el predominante, pues alcanzó maduración y consistencia, ganándose el respaldo de distintos desarrolladores a lo largo del tiempo. Asimismo, este modelo ha estado vinculado al uso de *SQL*, caracterizada por una lógica comprensible y estructurada; de igual forma, en la formación en sistemas de información se tiende a incorporar tanto modelos relacionales como documentales. Sin embargo, las

herramientas para bases de datos documentales resultan menos intuitivas y requieren mayor dominio de sus instrucciones para explotarla a conveniencia (p. 29).

2.1.2.4.1. Relacional

Saltos-Viteri y Franco-Bayas (2019) sostienen que el modelo relacional ha sido clave en el desarrollo de sistemas transaccionales y que aún no es posible dejarlo en su totalidad, dado que mantiene unas amplias capacidades para el almacenamiento de información en el sector empresarial. Asimismo, señalan que la transición de aprendizaje puede exigir desaprender costumbres del modelo relacional; por ello, contar con herramientas adecuadas y de manejo fácil, posibilitan una correcta modelación de datos y un manejo más eficiente mediante servicios *API* (pp. 46-47).

2.1.2.4.2. MySQL

Gillfillan (2003) explica que *MySQL* emplea *SQL* para crear bases de datos y para agregar, manipular y recuperar datos según criterios específicos; además, estas operaciones se apoyan en el lenguaje de definición de datos, mediante instrucciones como *CREATE*, que permiten crear, modificar o eliminar tablas y estructuras. Asimismo, destaca la importancia de la codificación *Unicode* y de las reglas de ordenación de caracteres para el tratamiento correcto de la información textual (pp. 40, 65, 532 y 783). En esa misma línea, Soto-Fernández et al. (2021) señalan que *MySQL* es un sistema de gestión de bases de datos relacionales que organiza la información en tablas separadas y se sustenta en el lenguaje *SQL* estandarizado (p. 402).

2.1.2.4.3. Firebase

Saltos-Viteri y Franco-Bayas (2019) enfatizan que, con el auge de *NoSQL*, surgieron modelos alternativos como las bases de datos orientadas a documentos, con fin de manejar grandes volúmenes de datos (pp. 38-39). En este sentido, Tomás et al. (2019), destacaron que *Firebase* es una plataforma propiedad de *Google*, siendo un *MBaaS*, el cual integra servicios en la nube para quienes desarrolla aplicaciones de tipo móvil o *web*. Entre estas funcionalidades, mencionaron que *Authentication* es aquella que facilita la autenticación de usuarios desde el *backend*, *Cloud Firestore*, alternativa a *Realtime Database* aporta un

modelo de datos intuitivo, para consultas rápidas siendo de mayor escalabilidad, *Cloud Functions* permite ejecutar código del lado del servidor, *Crashlytics* sirve para detectar fallos y *Test Lab* para probar la aplicación en dispositivos reales o virtuales, con una gestión sencilla e integrada del *backend* (pp. 1-2, 6-8).

2.1.2.5. Metodología de Desarrollo

Molina-Ríos y Pedreira-Souto (2019) plantean que las metodologías de desarrollo *web* constituyen procesos estandarizados que orientan el trabajo del equipo mediante actividades, métodos y herramientas destinadas al desarrollo y mejora continua de los sistemas de información. En este sentido, una metodología reúne técnicas y procedimientos que permiten organizar el proceso de desarrollo y alcanzar el producto esperado. Asimismo, los autores señalan que estas metodologías han evolucionado desde enfoques tradicionales hacia propuestas ágiles e híbridas, con el fin de responder mejor a los cambios y a la complejidad propia del desarrollo de aplicaciones *web* (p. 12).

2.1.2.5.1. Metodología Tradicionales

Según López-de-Sosoaga-López-de-Robles et al. (2015) explica que las metodologías tradicionales se caracterizan por una estructura educativa imponente y un rol rígido del tutor, quien planifica y organiza unilateralmente el contenido, mientras que el estudiante asume un papel pasivo y de obediencia. Por lo tanto, estas metodologías se enfocan en la adquisición memorística de contenidos fragmentados, trabajando en horarios fuertes y en ambientes de aula aislados. Después de haber sido el enfoque dominante durante una larga proyección de tiempo, se contraponen a metodologías modernas, ya que las tradicionales tienden a limitar la reflexión crítica activa de los oyentes, enfocándose en una motivación en particular (pp. 400-403, 410).

2.1.2.5.2. Modelo en Cascada

Según Delgado-Olivera y Díaz-Alonso (2021), el Modelo en Cascada es un enfoque lineal para el desarrollo de *software* que abarca actividades fundamentales del proceso tales como especificación, desarrollo, evolución y validación, organizadas en fases separadas. Cada etapa debe ser aprobada por el usuario, generando su documentación, ya

que este enfoque inicia con la definición de requisitos, seguida por el diseño del *software*. Posteriormente, se procede con la implementación, las respectivas pruebas, la integración y las del sistema completo, para culminar con la operación y el mantenimiento, que representa la fase de mayor duración dentro ciclo de vida del *software* (pp. 46-47).

2.1.3. Gestión de Emprendimientos

Rivera-Barberán et al. (2024) comprende los procedimientos requeridos para el cumplimiento de objetivos a largo plazo, tiene por fin, hacer la organización competitiva y retornando un crecimiento positivo que genere empleo. Equivale a una herramienta modular, que otorga modularidad, mejorando la administración y operación de las empresas. Siendo importante aparta garantizar el correcto funcionamiento, reflejado en su productividad y rentabilidad interna (pp. 84-85). Godoy-Rosero (2021), toda organización que interactúa con el mercado. El emprendimiento, transformara su gestión, para así permanecer o incrementar su rentabilidad, mediante procesos optimizadores. Manteniendo clientes y accediendo a nuevos mercados, en base a sus objetivos planteados (p. 342) .

2.1.3.1. Gestión de Operaciones y Logística

Álvarez-Parodys et al. (2023) menciona que toda organización gestiona y controlara de forma eficiente el movimiento de recursos de cualquier tipo, con fin de asegurar el adecuado funcionamiento operacional. En conjunto, dentro de la empresa, la logística, es el área clave responsable de garantizar que el producto llegue al cliente indicado, a la ubicación y tiempo, procurando además que se haga con la calidad, cantidad y al menor costo posible (p. 12). A la par, la gestión operativa según Bueno-Tacuri y Jácome-Ortega (2021) es la que incrementa la capacidad productiva de las organizaciones, dando un enfoque en la administración de bienes y servicios logrando generar con calidad y sostenibilidad (p. 243).

2.1.3.1.1. Procesos productivos

Menciona Bueno-Tacuri y Jácome-Ortega (2021), parte operativa empieza con la categorización de actividades que apoyan procesos estratégicos, mediante una ventaja para incrementar la capacidad productiva de organizaciones (p. 335). Procesos, los cuales,

según Calle-Herencia (2022) pueden automatizarse total o parcialmente, permitiendo que las empresas los hagan más eficientes para apoyar otras estrategias. Estos procesos, puntualmente son los pedidos, al integrar la parte administrativa con la de los pedidos, como procesos productivos enfocados bajo una automatización, ayudan a reducir costos, y así no solo vender productos, sino experiencias (pp. 71-72, 76).

Este proceso estaría compuesto, como se menciona por Kocaoglu y Acar (2016), por tres fases esenciales: Recepción y registro del pedido, que abarca la preparación de la orden, verificación de datos y transmisión de la información; estado del proceso del pedido, que implica la validación de crédito, disponibilidad de inventario y programación interna; y finalmente, el seguimiento y entrega del pedido, que comprende la preparación de envíos, despacho y monitoreo del estado hasta la recepción por parte del cliente, incluyendo reportes de estatus durante todo el ciclo (pp. 213-215).

2.1.3.1.2. Control de calidad

De acuerdo con Bueno-Tacuri y Jácome-Ortega (2021), la gestión de operaciones es el centro, encargada de la gestión de bienes y servicios, con el enfoque de producir con calidad, pensado en el bienestar del consumidor, asegurando así la sostenibilidad y las ganancias de la organización. La dirección de operaciones conforma un pilar estratégico para el éxito empresarial, unificando una conexión directa con la gestión operativa de recursos (pp. 343-344). Lo cual, es clave para cumplir y garantizar estándares de calidad establecidos en cada pedido y producto. Para Valenzuela-Velasco et al. (2024), la calidad productiva influye en la reducción de pérdidas por vencimiento o tiempo de vida útil (p. 195).

2.1.3.1.3. Optimización de inventarios

De acuerdo con Hernandez (2011), cualquier entidad productiva debería tener operaciones fluidas, al equilibrar entre entrada y salida de recursos. Sin embargo, las unidades de procesamiento pueden presentar capacidades desequilibradas. A mayor capacidad, no hay problemas de producción; pero una capacidad administrativa excesiva podría indicar sobreinversión, mientras que una capacidad inferior genera cuellos de botella. El excedente se convierte en un pasivo y genera costos adicionales como almacenaje,

impuestos y vigilancia, subrayando la importancia de alinear producción con los recursos disponibles para optimizar la gestión económica y garantizar la rentabilidad (pp. 4-5, 16, 20). Además, según Valenzuela-Velasco et al. (2024), implementar correctamente una gestión de inventarios permite responder con agilidad a las fluctuaciones del mercado (p. 195).

2.1.3.2. Gestión de Talento Humano

Alarcón y Hernández (2024) quienes citan a Chiavenato (2009) detallan que para poder dirigir los aspectos de las personas y recursos convergen políticas y prácticas realizadas en la gestión de talento humano. Cuyos procesos puntuales son: reclutamiento, selección, capacitación, recompensas y evaluación de desempeño (p. 2). Además, en el capítulo 3 de Carriel et al. (2017), el enfoque de las empresas en talento humano será más tradicional, y en otras más dinámico. Siendo que la administración de personal ve de forma individual al colaborador, destacando de entre un grupo, por sus capacidades intelectuales y habilidades (p. 53). También, según Santana y Deodato (2024) se expande al entorno laboral positivo, donde los empleados progresan personal y profesionalmente (p. 167).

2.1.3.2.1. Reclutamiento y selección

El enfoque presentado por Alarcón y Hernández (2024) busca encontrar y atraer candidatos para reclutarlos, con fin de conformar equipos que aporten al progreso de la organización. Siendo que el propósito de la empresa será retenerlos aplicando estrategias de potenciamiento personal, mejorando habilidades, y su participación productiva dentro del ambiente laboral. Permitiendo, integrar por medio de políticas y prácticas enfocadas a la selección, reclutamiento, orientación, retribuyendo el esfuerzo, desarrollando, auditando y dando seguimiento al nuevo personal (pp. 1, 6).

2.1.3.2.2. Capacitación y evaluación de desempeño

Según Marcó et al. (2016) La dirección busca orientar los esfuerzos humanos para lograr objetivos, creando condiciones adecuadas para que el personal cumpla tareas y rinda bien. Esto requiere liderazgo, comunicación y motivación, además de habilidades técnicas que se fortalecen con capacitación y práctica constante (pp. 45-47). Esta idea se respalda con lo mencionado por Alarcón y Hernández (2024), quienes señalan que capacitar

maximiza y aporta capacidades al emprendedor (p. 10). Por lo tanto, sería un proceso de evaluación y monitoreo del desempeño, el cual, garantiza que las actividades se desarrollen correctamente, evitando desviaciones y aplicando correcciones para mantener el rumbo hacia el éxito a largo plazo.

2.1.3.2.3. Cultura organizacional

Según Santana y Deodato (2024), la cultura organizacional es el factor para moldear comportamientos, actitudes, aptitudes y procesos dentro de la empresa. Siendo que la cultura, se mostrara en la motivación, la satisfacción laboral y el compromiso de los colaboradores. De esta forma, la evidencia muestra que impacta en la capacidad de atraer, retener y desarrollar talento humano, fortaleciendo su desempeño individual y colectivo. Además, se evidencia en estudios de caso que, por ejemplo, un 56 % de microempresarios perciben contar con una cultura organizacional fortalecida, lo cual se traduce en mayor productividad, motivación sostenida y retención de empleados, consolidando su efecto estratégico dentro de la gestión del talento humano (pp. 162, 170, 173).

2.1.3.3. Gestión Legal y Aspectos Regulatorios

Paredes-Guerrero et al. (2021) previo o posterior a la creación de un emprendimiento, es relevante analizar aspectos administrativos, legales y operativos, dado que tener una buena estructura hará que perdure en el tiempo. Por ende, para mantenerse al día con las obligaciones legales evita contratiempos por aspectos legales, tributarios o ciertos permisos, en consecuencia, no afectar a la operatividad empresarial. Cumplir a cabalidad, los varios aspectos legales, tras la creación del emprendimiento y posterior constitución legal de la empresa, se traduce en lograr unas bases legales solidas (pp. 21, 23). La Organización Internacional del Trabajo (2015), da una estrategia para combinar medidas, las cuales, reduzcan el peso regulatorio de formación en micro y pequeñas empresas, otorguen incentivos por formalizarse y ofrezcan una mejor fiscalización (p. 7).

2.1.3.3.1. Constitución legal de la empresa

Guerrero-Velástegui et al. (2023) para permitir el desarrollo y funcionamiento adecuado de las empresas, el procedimiento es el establecimiento de un marco legal. Siendo la base propuesta por la organización jurídica, estructurada hacia una empresa o negocio, definiendo la forma legal de la empresa, tales como, las sociedades, corporaciones, asociaciones, entre otras entidades legales (pp. 3, 21). La OIT (2015) determina que, tras el registro de una empresa, se incorporara como una entidad legal, con plenas capacidades legales para interactuar con entidades públicas o privadas (p. 55).

2.1.3.3.2. Licencias

Son la principal autorización para operar. Como indican Ujianti et al. (2024) una licencia es una garantía legal para certificar la actividad realizada por una persona o negocio, siendo similar a un permiso o certificado de registro comercial (p. 8880). Otra definición señala que “Una licencia comercial es un permiso de una agencia gubernamental para participar en una actividad comercial específica” (Cornell Law School, 2021). Asimismo, Paredes-Guerrero et al. (2021) definen la licencia como un documento emitido por una entidad pública, en favor de controlar la actividad comercial. Además, señalan que, en beneficio para los emprendedores, los trámites para su obtención pueden realizarse en línea (p. 33). En cuanto a su trámite, la OIT (2015) indica que, posterior a la solicitud del empresario con la información requerida, en algunos países los funcionarios municipales establecen un plazo para la concesión de licencias de funcionamiento (p. 58).

2.1.3.3.3. Permisos

Los permisos, según la OIT (2015) son autorizaciones complementarias y específicas para poder ejercer sus actividades. En este sentido, muchos se han mantenido, reformulado, eliminado, o se han eximido a ciertos tipos de empresas (p. 57). Pueden definirse como “aprobaciones necesarias para acciones específicas” (Anwar, 2025). Al respecto, Paredes-Guerrero et al. (2021), recalcan que los permisos son documentos emitidos por entidades públicas que permiten ejercer una actividad comercial. Siendo que pueden ser tanto como el permiso de funcionamiento a ante el Ministerio de Gobierno,

también el Permiso de funcionamiento de la Agencia Nacional de Regulación, Control y Vigilancia Sanitaria, e incluso de Cuerpo de bomberos (pp. 33, 34).

2.1.3.4. Gestión de Redes y Ecosistemas Emprendedores

La gestión de redes para un ecosistema de emprendimientos, la podemos definir tomando de referencia lo mencionado por Gancedo y Ibáñez-Maraña (2021), esta se entiende como el impulso de entornos donde la existencia de empresas sociales consolidadas favorece el surgimiento de nuevas iniciativas, generando un ecosistema sólido basado en principios y valores sociales. Además, se caracteriza por su dimensión colectiva, que fomenta la colaboración, la cooperación y la articulación con redes locales y el entorno (pp. 6-7). El ecosistema de emprendimientos es definido por Torres y Cardona (2023) como la interacción entre aspectos políticos, sociales, económicos, culturales y ambientales, los cuales son el habilitador hacia el desarrollo de las actividades emprendedoras dentro de una región definida. Aspecto el cual ha cobrado relevancia (p. 1).

2.1.3.4.1. Alianzas estratégicas

En base a Gancedo y Ibáñez-Maraña (2021) se establecen canales de comunicación y diálogo con grupos de interés, buscando transparencia y difundiendo el ideario de la entidad social (p. 13). Rojas-López et al. (2014) lo resume en: *“Las alianzas estratégicas son uniones formales entre dos o más organizaciones... para formar asociaciones que fortalecen la competitividad”* (p. 281). Carvajal-Camperos et al. (2021) las define como acuerdos cooperativos entre empresas autónomas e independientes que se unen para metas compartidas o resolver problemas mediante interacción sostenida. Apoyando al crecimiento colaborativo, dando independencia en varios ámbitos. Cuyo propósito, será mejorar la competitivamente, tras crear valor y sinergias mediante un contrato temporal que define la participación de cada socio. Implica compartir conocimientos, recursos y capacidades, cuyo compromiso es alcanzar logros colectivamente, dada su dificultad individual (pp. 297, 304).

2.1.3.4.2. Incubadoras y aceleradoras

La OIT (2021) expresa que la función de mayor importancia que proporcionan las incubadoras y aceleradoras, es la prestación de apoyo referente al crecimiento empresarial, concebido íntegramente para nuevos emprendedores y *startups*. Siendo que estas invierten tiempo y esfuerzo en robustecer el entorno emprendedor local. En sí, vienen a ser cómo centros de formación para el emprendimiento (pp. 17, 23). Amador-Martínez et al. (2023), quienes referenciando a ONU Mujeres (2022), indico que el objetivo de autonomía económica de mujeres emprendedoras está pensado para el fortalecimiento de sus negocios. Así mismo, aplican acciones para mejorar sus condiciones de emprendimiento, como se evidenció en Guatemala, replicables en otros contextos. Dentro de estos resultados se destaca una fuerte resiliencia económica, usos de redes de mercado, enlaces comerciales, abriendo oportunidades a la expansión productiva (pp. 5–6).

2.1.3.4.3. Emprendimiento social colaborativo

Según Gancedo y Ibáñez-Maraña (2021) es una alternativa aceptada de forma voluntaria, donde las personas que ejercen el emprendimiento, se plantean el objetivo vinculado bajo una demanda o necesidad social, tiene la aspiración de conseguirla en conjunto, dado que comparten en común la visión y las necesidades reales para emprender. Siendo un aporte que busca generar un impacto positivo, ampliado hacia el entorno de quienes desarrollan sus actividades bajo la cara voluntad de emprender juntas (pp. 15, 16). En este sentido, Amador-Martínez et al. (2023), menciona que parte del éxito del negocio, radica en el empoderamiento de las mujeres que emprenden, pues abarca dimensiones de igualdad de género. Además, sus experiencias y resiliencia impactan positivamente en su economía mediante prácticas orientadas al bien común. Ya que las mujeres, son protagonistas clave en nichos de mercado, aportando al desarrollo económico, a nivel regional y nacional (pp. 123-124, 136).

2.2. Predicción Científica

H0: La implementación de la aplicación *web* y móvil no incide en el fortalecimiento de la gestión de los emprendimientos en la empresa “100 Mujeres” del cantón Santo Domingo.

H1: La implementación de la aplicación *web* y móvil incide en el fortalecimiento de la gestión de los emprendimientos en la empresa “100 Mujeres” del cantón Santo Domingo.

.

3. METODOLOGÍA

3.1. Enfoque, alcance, diseño y tipo de investigación

La realización del presente trabajo de titulación fue implementada, mediante el uso de una metodología cuantitativa, este enfoque fue seleccionado porque permite la recolección y el análisis sistemático de datos numéricos, lo que hace más fácil rastrear el impacto y la eficacia de la aplicación *web* y móvil. Igualmente, Medina-Romero, et al. (2023) destacan que el enfoque por medio del método cuantitativo, es clave para respaldar o refutar las hipótesis, bajo el análisis de datos estadísticos, para cuantificar variables, relacionar e identificar diferencias de interés significativo (p. 15).

Por otra parte, el alcance determina cuán a fondo se estudia el objeto de estudio, Arias-Gonzales y Covinos-Gallardo (2021) señalan los alcances: exploratorio, descriptivo, correlacional y explicativo, y precisan que el alcance aplicativo no lo asumen dentro de esta categoría, porque los alcances expresan la profundidad del estudio, mientras que aplicar alude a la intención de intervenir para mejorar un problema (p. 69). Sin embargo, Mamani et al. (2024) lo integran como nivel, e incorporan el nivel aplicativo, el cual implica usar los hallazgos para atender problemas prácticos del mundo real y mejorar procesos o intervenciones (p. 136). Así, aunque algunos autores lo clasifican distinto, para esta investigación se adopta un alcance a nivel aplicativo de la versión ampliada de los niveles.

En el enfoque cuantitativo de la investigación, el diseño permite comprobar relaciones entre variables. Los diseños experimentales permiten verificar cuantitativamente, tal como lo describen Arias-Gonzales y Covinos-Gallardo (2021), es un proceso que implica manejar variables, donde la variable independiente es el factor o condiciones que se prueban bajo control, y provocan a la vez efectos sobre la variable dependiente (pp. 73-74). Asimismo, Mamani et al. (2024) precisan que dentro de los diseños experimentales distinguen tres grupos: experimental puro, cuasi experimental y preexperimental. El experimental puro supone control total, asignación aleatoria y la participación de dos o más grupos. Por otro lado, el cuasi experimental, control no completo, con dos o más grupos

comparables. En el preexperimental el control es mínimo y se trabaja con un solo grupo, conocido como grupo experimental, aplicando mediciones antes y después del tratamiento (pp. 137-138, 140-141), de acuerdo con lo anteriormente mencionado, se optó por este diseño.

Por su parte, el tipo de investigación se enfocó hacia la investigación aplicada, misma que tiene como propósito, dar a proponer una solución que fortalezca la gestión y autonomía comercial hacia las mujeres emprendedoras. Con este fin, se aplica lo detallado en Tarrillo-Saldaña et al. (2024), donde se indica que la investigación aplicada se encarga de abordar problemas específicos, y de desarrollar soluciones puntuales, a través de la aplicación práctica del conocimiento existente (p. 58).

A su vez, se incorpora el uso de la investigación de campo, en la cual, los datos se recogen directamente de la realidad social del grupo de estudio, sin manipulación artificial del entorno. Según lo distinguido en Escudero-Sánchez y Cortez-Suárez (2018) permite la obtención de nuevos conocimientos, logrando aplicar manipulaciones controladas (p. 20), por ende, es ideal para el estudio en la empresa 100 Mujeres.

3.2. Unidad de análisis

La unidad de análisis corresponde a las personas involucradas, durante el proceso de distribución y comercialización de productos de la empresa “100 Mujeres”, considerando específicamente una población censal con total de 109 mujeres emprendedoras, las cuales, realizan ventas a partir de los pedidos de bandejas realizados. Esto se evidencia en la tabla 1, los tipos de emprendedoras y las cantidades involucradas, mismas que desempeñan su trabajo en sectores de la ciudad de Santo Domingo.

Tabla 1. Población

Categoría	Tipo de Emprendedora	Descripción	Cantidad estimada
1	Emprendedoras en puntos móviles	Venta presencial	31
2	Emprendedoras en puntos fijos	Venta por medios digitales	56
3	Emprendedoras nuevas	Entrenamiento para incorporarse	22
Total			109

Fuente: Datos proporcionados del modelo operativo del proyecto “100 Mujeres Trabajando”.

Debido al tamaño accesible de la población, se optó por aplicar un estudio o muestreo censal, debido a su mayor precisión, con fin de reducir el margen de error muestral asociado a la selección de una muestra. Esto implicó la recolección de información del total de las emprendedoras disponibles. Según lo estipulado por Pérez-Flores (2024), el estudio censal es aquel que permite obtener una representación exhaustiva de la población de interés, la cual, implica el estudio total de la población a estudiar. Diferenciando a elegir una muestra representativa de la misma (p. 67), otros autores lo denominan muestreo censal. Esta elección, considerando la naturaleza del proyecto, en conjunto con las emprendedoras activas que se encuentren en puntos móviles y fijos. Por lo tanto, el censo fue aplicado a una muestra total de 109 emprendedoras.

3.3. Técnicas e instrumentos de investigación

Es relevante indicar que las técnicas a utilizar durante la recolección de información, frente a lo reflejado por la población analizada, según lo expresado por Medina-Romero, Rojas-León, et al. (2023), las técnicas de investigación vienen a ser el procedimiento para recopilar y analizar la información bajo un sentido investigativo. Esto va de la mano con los instrumentos de investigación, los cuales son usados durante el proceso de investigación, siendo de beneficio para obtener información precisa y confiable del tema de estudio, obteniendo finalmente, conclusiones acertadas (p. 13).

De este modo, se utilizó la técnica de recolección de datos correspondiente a la encuesta, la cual fue realizada a un total de 109 participantes. Adicionalmente, se utilizó *Microsoft Forms* como herramienta de investigación, misma que se aplicó en dos ocasiones, en una fase previa y en una posterior. De acuerdo con Gallardo-García (2025), la encuesta es la técnica que posibilita recolectar información de forma sistemática y estructurada a un gran número de individuos, posibilitando la obtención de opiniones, actitudes y características de los participantes. Además, el cuestionario es el instrumento compuesto de preguntas estructuradas, las cuales permiten generar respuestas cuantificables. Que fueron respondidas, bajo un conjunto limitado de respuestas, tal como opción múltiple o de tipo *Likert* y administrados bajo un medio físico o digital (pp. 10, 19, 44, 73).

Esto posibilita el acceso a información directa, lo que es beneficioso para establecer la perspectiva del estudio, con el propósito de reunir información. Este cuestionario fue diseñado para medir factores tras el uso de la tecnología, en términos de la gestión del emprendimiento bajo su proceso operativo.

3.4. Técnicas de análisis de datos

El análisis de datos en la investigación cuantitativa se apoya del análisis estadístico, el cual examina hipótesis y comprende relaciones entre las variables. Según Mamani et al. (2024) se emplea estadística descriptiva para resumir y describir las características de los datos; en contraposición, la estadística inferencial permite generalizar a la población a partir de la muestra. A su vez, las pruebas estadísticas funcionan como un procedimiento para tomar decisiones o extraer conclusiones sobre una población con base en datos muestrales, y, cuando corresponde probar hipótesis sobre características poblacionales (pp. 13, 159, 162).

Con base en el planteamiento, se aplicó análisis estadístico descriptivo, el cual permitió la depuración, tabulación y organización automática de las respuestas individuales obtenidas, utilizando técnicas de distribución de frecuencias relativas (frecuencias y porcentajes) y gráficos de barras agrupadas para interpretar los resultados de manera clara y comprensible, apoyándose en la herramienta *Microsoft Excel*.

Posterior a la recopilación, los datos fueron preparados para su análisis, según lo expresado por Rivadeneira-Pacheco et al. (2020), existen varios *softwares* estadísticos que permiten analizar proyecciones relacionadas al tema de investigación y facilitando el análisis tanto descriptivo como inferencial (p. 24). En este sentido, para el análisis de datos se empleó la estadística inferencial, por medio de la técnica de regresión logística binaria mediante la herramienta *IBM SPSS*. A la par, Rahman y Mukhtadir (2021) destacan a *SPSS*, por la posibilidad de extracción de conclusiones y predicciones, a partir de los datos estadísticos. Permitiendo hacer análisis complejos, gestionar los datos y documentar sus resultados en graficas o diagramas (p. 302). Siendo así, esencial para la comparativa de los resultados obtenidos entre la fase inicial (*pretest*) y final (*posttest*).

3.5. Operacionalización de las variables

Tabla 2. Aplicación móvil variable independiente

Conceptualizaciones	Dimensión	Indicadores	Preguntas	Herramienta
Según Randhawa (2022) una aplicación móvil es un <i>software</i> que normalmente se ejecuta en dispositivos móviles personales, tal como teléfonos inteligentes o tabletas. Consecuentemente, para su funcionamiento se compone de una interfaz gráfica de usuario, almacenar datos de forma local, y permite la comunicación, tanto entre dispositivos como con servidores empresariales a través de redes adecuadas. Con este sentido, se tienen <i>frameworks</i> de interfaz gráfica, los cuales proporcionan bibliotecas, con controles de interacción móvil para la	Plataforma de ejecución	<i>Android</i>	¿Qué celular tienes, <i>Android</i> o <i>iOS</i> ?	Encuesta a Emprendedora
		<i>iOS</i>	¿Te gustaría usar la aplicación en otro dispositivo?	Encuesta a Emprendedora
		Multiplataforma	¿Qué tan necesario crees que es usar la <i>App</i> en varios dispositivos?	Encuesta a Emprendedoras
	Entorno de desarrollo móvil	<i>Flutter</i>	¿Qué tan de acuerdo estás con que la <i>App</i> funcione igual en cualquier dispositivo?	Encuesta a Emprendedoras
		<i>Dart</i>	¿Qué funciones considera más importantes para garantizar los pedidos y seguimiento de estos?	Reunión de planificación del sprint con la Administradora
		Librerías y paquetes externos (<i>Pub.dev</i>)		
	Arquitectura de la aplicación	<i>Clean Architecture</i>	¿Qué tan importante consideras que una aplicación tenga una estructura ordenada y clara?	Encuesta a emprendedoras
		Estructura modular del código	¿Te gustaría que el sistema te guíe paso a paso para hacer pedidos o registros?	Encuesta a emprendedoras
		Navegación entre pantallas	¿Qué tan fácil te resulta el uso de una aplicación móvil?	Encuesta a emprendedoras
		Gestión de estado de la aplicación	¿Te gustaría ver en tiempo real si tu pedido está aceptado o en camino?	Encuesta a Emprendedoras

interfaz de usuario. En otras palabras, las aplicaciones proporcionan necesidades básicas para la interacción del usuario, acceso al almacenamiento de datos y soporte para la comunicación de datos. En apoyo a múltiples sistemas móviles, surgieron aplicaciones móviles híbridas, y bajo otro enfoque que únicamente necesita un código, por un lenguaje del entorno de desarrollo, que se traducirá a versiones nativas de las plataformas compatibles (pp. 4, 57, 113, 222).	<i>Diseño funcional del sistema móvil</i>	Diseño de interfaz de usuario (<i>UI</i>)	¿Qué tan satisfecha estas con el diseño, cumple con lo que esperabas de la aplicación?	Encuesta a Emprendedoras
			¿Te resulta fácil usar una <i>App</i> para hacer tus pedidos?	Encuesta a emprendedoras
			Considerando a emprendedoras con diferentes niveles de educación, ¿Qué criterios de diseño considera se han de tomar en cuenta?	Reunión de planificación del sprint con la Administradora
		Diseño de experiencia de usuario (<i>UX</i>)	¿Qué tan fácil te resulta entender los íconos e imágenes en una aplicación?	Encuesta a emprendedoras
			¿Qué tanto estás de acuerdo, en que una aplicación pensada para emprendedoras considere el uso de letras grandes y colores claros?	Encuesta a Emprendedoras
			¿Te gustaría que puedas calcular el total de tu pedido automáticamente?	Encuesta a emprendedoras
	<i>Almacenamiento y servicios backend</i>	Capacidades del dispositivo móvil	¿Te gustaría recibir notificaciones cuando tu pedido esté listo?	Encuesta a Emprendedoras
		<i>Firestore</i>	¿Te da confianza compartir tus datos en una <i>App</i> ?	Encuesta a emprendedoras
		<i>SQLite</i>	¿Qué tan necesario consideras poder usar la <i>App</i> sin internet?	Encuesta a emprendedoras
		<i>APIs</i>	¿Qué medidas considera necesarias para validar los comprobantes de pago subidos por las emprendedoras y garantizar la seguridad de esta información?	Reunión de planificación del sprint con la Administradora
		Seguridad de datos y autenticación	¿Qué tan segura te sentirías al compartir tu comprobante de pago en la <i>App</i> ?	Encuesta a Emprendedoras

Tabla 3. Aplicación *web* – variable independiente.

Conceptualización	Dimensión	Indicadores	Preguntas	Herramientas
Márquez-Coca et al. (2023) explican que las aplicaciones <i>web</i> se fundamentan en el modelo cliente-servidor, en el que las interacciones se producen por medio de un flujo de solicitudes y respuestas entre ambos componentes. En este sentido, el desarrollo <i>web</i> combina la programación del lado del cliente y del servidor, la gestión de bases de datos y el uso de <i>frameworks</i> que extienden sus capacidades. Además, estos sistemas consideran aspectos como la usabilidad, accesibilidad, lo cual facilita la creación, el mantenimiento y la conexión óptima de los sitios web hacia los usuarios (pp. 9, 26, 32). Por su parte, Hernández-Bejarano y Baquero-Rey (2020) señalan que estas aplicaciones operan a partir de la relación entre el navegador, que muestra la información <i>web</i> , y el servidor, que permite acceder a ella, mediante un esquema de solicitudes y respuestas que forma parte de sus características básicas (pp. 20, 176).	<i>Lenguaje de programación</i>	<i>PHP</i>	¿Cuál tan familiarizado se encuentra con el uso de aplicaciones <i>Web</i> ?	Reunión de planificación del <i>sprint</i> con la Administradora Encuesta a Emprendedora
		<i>JavaScript</i>	¿Cómo te sientes al usar tecnología para tu negocio, como páginas <i>Web</i> ?	
		<i>CSS</i>		
	<i>Herramientas de Desarrollo</i>	<i>React</i>	¿Considera que el desarrollo de una aplicación <i>web</i> , de acuerdo a sus requerimientos, puede fortalecer la gestión de operaciones y logística en su empresa?	Reunión de planificación del <i>sprint</i> con la Administradora
		<i>Vite</i>		
		<i>Node.js</i>		
	<i>Software de Programación</i>	<i>Visual Studio Code</i>	¿Considera que la aplicación de nuevas tecnologías puede facilitar la gestión de talento humano? ¿Qué tan difícil es para ti enviar y recibir información digitalmente?	Reunión de planificación del <i>sprint</i> con la Administradora Encuesta a Emprendedora
		<i>Sublime Text</i>		
		<i>Android Studio</i>		
	<i>Base de Datos</i>	<i>Relacional</i>	¿Está de acuerdo en almacenar toda la información de sus emprendedoras y productos en una aplicación tecnológica?	Reunión de planificación del <i>sprint</i> con la Administradora
		<i>MySQL</i>		
		<i>Firebase</i>		
	<i>Metodología de Desarrollo de Software</i>	<i>Metodologías tradicionales</i>	¿Cómo guarda la información de las emprendedoras?	Reunión de planificación del <i>sprint</i> con la Administradora
		<i>Modelo en Cascada</i>	¿Con qué frecuencia organiza la información de su emprendimiento? ¿Cuál es el proceso para registrar la información de las emprendedoras?	Encuesta a Emprendedora Reunión de planificación del <i>sprint</i> con la Administradora

Tabla 4. Gestión de emprendimientos – variable dependiente.

Conceptualizaciones	Dimensión	Indicadores	Preguntas	Herramienta
La gestión de emprendimientos, infiriendo lo mencionado en Rivera-Barberán et al. (2024) , comprende el conjunto de procedimientos para el crecimiento de una empresa, permiten ser competitiva a largo plazo, y generar empleo. Esta gestión empresarial es relevante para el crecimiento sostenible de cualquier emprendimiento emergente. En este sentido, incluye “planificar, administrar y organizar los recursos”. Buscando garantizar operatividad, se tendrá un aspecto administrativo, compuesto por fases, etapas y elementos, e interrelacionado en un proceso integral, lo cual, sustenta la importancia de estructurar la gestión en dimensiones como operaciones, talento humano, aspectos legales y redes colaborativas. Asimismo, una gestión eficiente permite a empresas, ser activas en el mercado. La gestión permite la operatividad modular de las empresas (pp. 10, 17, 61-62, 84-85).	<i>Gestión de Operaciones y Logística</i>	Procesos productivos	¿Qué tan satisfecha te sientes con la forma actual de pedir tus pedidos? ¿Con que frecuencia, realizan pedidos actualmente?	Encuesta a Emprendedoras Encuesta a Emprendedoras
		Control de calidad	¿Qué medidas de control de calidad considera importantes para asegurar la entrega de pedidos? ¿Qué tan satisfecha estás con la forma actual de hacer tus pedidos?	Reunión de planificación del sprint con la Administradora Encuesta a Emprendedoras
		Optimización de inventarios	¿Qué cambios cree usted que la app debe incluir para mejorar la gestión de pedidos?	Reunión de planificación del sprint con la Administradora
			¿La empresa tiene planes de producir otros productos?	Reunión de planificación del sprint con la Administradora
	<i>Gestión de Talento Humano</i>	Reclutamiento y selección	¿Con qué datos se siente más cómoda registrándose?	Encuesta a Emprendedoras
		Capacitación y evaluación de desempeño	¿Qué tan necesario sería capacitarte si cambia la forma de realizar pedidos?	Encuesta a emprendedoras
		Cultura organizacional	¿Pueden las emprendedoras, vender otros productos además de los de la empresa?	Reunión de planificación del sprint con la Administradora
	<i>Gestión Legal y Aspectos Regulatorios</i>	Constitución legal de la empresa	¿Cómo está constituida legalmente la empresa?	Reunión de planificación del sprint con la Administradora
			¿Todas las emprendedoras venden lo mismo o productos distintos?	Reunión de planificación del sprint con la Administradora
		Licencias Permisos	¿Qué licencias o permisos son los necesarios para garantizar el trabajo de las emprendedoras?	Reunión de planificación del sprint con la Administradora
	<i>Gestión de Redes y Ecosistemas Emprendedores</i>	Alianzas estratégicas	¿Cómo evalúa la empresa la coordinación y apoyo entre emprendedoras?	Reunión de planificación del sprint con la Administradora
		Incubadoras y aceleradoras	¿Se da algún tipo de apoyo inicial a las emprendedoras?	Reunión de planificación del sprint con la Administradora
		Emprendimiento social colaborativo	¿Qué tan importante es coordinarse entre emprendedoras para repartirse las zonas de venta?	Reunión de planificación del sprint con la Administradora

4. RESULTADOS

4.1. Resultado del primer objetivo específico

La encuesta y la reunión de planificación del *sprint* con la administradora, detalladas en el anexo 3, fueron examinadas y valoradas por los expertos mencionados en la tabla 5, en la que se ofrece una explicación minuciosa de los porcentajes de las respuestas obtenidas y gestionadas en la etapa del *pretest*.

Tabla 5. Expertos en la validación de instrumentos para la recopilación de datos.

Nombres	Título académico	Área
Carrión Bósquez Nelson Geovany	Pg.D. <i>Business Administration</i>	Administración de Negocios.
Ulloa Meneses Luis Javier	Mg. En Informática Empresarial	Sistemas

4.1.1. Aplicación de las encuestas a las Emprendedoras

En la primera etapa, se hizo la encuesta previa a la implementación de la propuesta de intervención, la cual fue realizada a las emprendedoras de la empresa “100 Mujeres”.

4.1.1.1. Presentación de resultados de la encuesta a las emprendedoras

En base al recopilado de datos realizado a las 109 emprendedoras de “100 Mujeres”, mediante la aplicación de una encuesta en la fase del *pretest*, se pudo evidenciar que, respecto a la forma actual de pedir sus pedidos, el 81.65% se siente “poco satisfecha” o “medianamente satisfecha”, frente al 18.35% que se considera “satisfecha”. Por otro lado, el conjunto del 36.70% y 18.35%, considera su gestión de pedidos como “normal” y “difícil”. asimismo, la información de su emprendimiento la organizan, “ocasionalmente” un 72.48% y el 9.17% la hace “raramente”, mientras que solo el 18.35% señalaba hacerlo entre “frecuentemente” y “muy frecuentemente”. Estos resultados permitieron identificar limitaciones en la gestión de emprendimientos antes de la implementación de la propuesta. Adicionalmente, se evidencia que *Android* es el dispositivo más usado, orientando la prioridad de desarrollo. Más del 68% está de acuerdo con usar la *App* en varios dispositivos, aunque no es indispensable para todas. Más del 80% coincide en estructura ordenada, guía paso a paso, notificaciones y estado de pedidos en tiempo real.

4.1.1.2. Resultados de la encuesta a las emprendedoras

La tabla 6, se muestran los resultados de manera porcentual, generados a partir de las encuestas realizadas, durante la fase de *pretest*, indicando las preguntas realizadas y asignando sus escalas y porcentajes respectivos.

Tabla 6. Datos obtenidos del resultado de la encuesta pre-test

N°	Preguntas	Escala y %					Figura
		A. Android	B. iOS	-	-	-	
1	¿Qué celular tienes para usar la App?	100.00%	0.00%				
2	¿Te gustaría usar la aplicación en otro dispositivo?	A. Totalmente de acuerdo 22.94%	B. De acuerdo 44.95%	C. Neutral 22.94%	D. Poco de acuerdo 9.17%	E. Nada de acuerdo 0.00%	
3	¿Qué tan necesario crees que es usar la App en varios dispositivos?	A. Muy necesario 13.76%	B. Necesario 26.61%	C. Medianamente necesario 22.94%	D. Poco necesario 22.94%	E. Innecesario 13.76%	
4	¿Está usted de acuerdo con que la App funcione igual en cualquier dispositivo?	A. Totalmente de acuerdo 44.95%	B. De acuerdo 41.28%	C. Neutral 9.17%	D. Poco de acuerdo 4.59%	E. Nada de acuerdo 0.00%	
5	¿Qué tan importante consideras que una aplicación tenga una estructura ordenada y clara?	A. Muy importante 36.70%	B. Importante 54.13%	C. Medianamente importante 9.17%	D. Poco importante 0.00%	E. Nada importante 0.00%	
6	¿Te gustaría que el sistema te guíe paso a paso para hacer pedidos o registros?	A. Totalmente de acuerdo 54.13%	B. De acuerdo 45.87%	C. Neutral 0.00%	D. Poco de acuerdo 0.00%	E. Nada de acuerdo 0.00%	
7	¿Qué tan fácil te resulta el uso de una aplicación móvil?	A. Muy fácil 9.17%	B. Fácil 63.30%	C. Normal 27.52%	D. Difícil 0.00%	E. Muy difícil 0.00%	
8	¿Qué tan de acuerdo estas con ver en tiempo real si tu pedido está aceptado o en camino?	A. Totalmente de acuerdo 45.87%	B. De acuerdo 54.13%	C. Neutral 0.00%	D. Poco de acuerdo 0.00%	E. Nada de acuerdo 0.00%	
9	¿Qué tan fácil te resulta entender los íconos e imágenes en una aplicación?	A. Muy fácil 22.94%	B. Fácil 58.72%	C. Normal 13.76%	D. Difícil 4.59%	E. Muy difícil 0.00%	
10	¿Qué tan de acuerdo está en que una aplicación móvil, considere el uso de letras grandes y colores claros?	A. Totalmente de acuerdo 27.52%	B. De acuerdo 67.89%	C. Neutral 4.59%	D. Poco de acuerdo 0.00%	E. Nada de acuerdo 0.00%	
11	¿Te gustaría calcular el total a pagar de tu pedido automáticamente?	A. Totalmente de acuerdo 49.54%	B. De acuerdo 45.87%	C. Neutral 4.59%	D. Poco de acuerdo 0.00%	E. Nada de acuerdo 0.00%	

12	¿Te gustaría recibir notificaciones cuando tu pedido esté listo?	A. Totalmente de acuerdo	B. De acuerdo	C. Neutral	D. Poco de acuerdo	E. Nada de acuerdo	
		54.13%	36.70%	9.17%	0.00%	0.00%	
13	¿Te da confianza compartir tus datos en una App?	A. Totalmente de acuerdo	B. De acuerdo	C. Neutral	D. Poco de acuerdo	E. Nada de acuerdo	
		22.94%	31.19%	27.52%	18.35%	0.00%	
14	¿Qué tan necesario consideras poder usar la App sin internet?	A. Muy necesario	B. Necesario	C. Medianamente necesario	D. Poco necesario	E. Innecesario	
		13.76%	58.72%	9.17%	18.35%	0.00%	
15	¿Qué tan segura te sentirías al compartir tu comprobante de pago en la App?	A. Totalmente segura	B. Segura	C. Moderadamente segura	D. Poco segura	E. Nada segura	
		36.70%	54.13%	4.59%	4.59%	0.00%	
16	¿Cómo te sientes al usar tecnología para tu negocio, como páginas Web?	A. Totalmente de acuerdo	B. De acuerdo	C. Neutral	D. Poco de acuerdo	E. Nada de acuerdo	
		18.35%	49.54%	18.35%	4.59%	9.17%	
17	¿Qué tan difícil es para ti enviar y recibir información digitalmente?	A. Muy complicado	B. Complicado	C. Neutral	D. Poco complicado	E. Nada complicado	
		4.59%	4.59%	35.78%	27.52%	27.52%	
18	¿Con qué frecuencia organiza la información de su emprendimiento?	A. Muy frecuentemente	B. Frecuentemente	C. Ocasionalmente	D. Raramente	E. Nunca	
		4.59%	13.76%	72.48%	9.17%	0.00%	
19	¿Qué tan satisfecha te sientes con la forma actual de pedir sus pedidos?	A. Totalmente satisfecha	B. Satisfecha	C. Medianamente satisfecha	D. Poco satisfecha	E. Nada satisfecha	
		0.00%	18.35%	44.95%	36.70%	0.00%	
20	¿Con qué frecuencia realizan pedidos actualmente?	A. Siempre	B. Casi siempre	C. A veces	D. Casi nunca	E. Nunca	
		9.17%	41.28%	49.54%	0.00%	0.00%	
21	¿Qué tan fácil te resulta gestionar sus pedidos?	A. Muy Fácil	B. Fácil	C. Normal	D. Difícil	E. Muy difícil	
		0.00%	44.95%	36.70%	18.35%	0.00%	
22	¿Con qué datos se siente más cómoda registrándose?	A. Solo correo	B. Solo número	C. Ambos	D. Otro	E. Prefiero no compartir	
		13.76%	44.95%	36.70%	0.00%	4.59%	
23	¿Qué tan necesario sería capacitarte si cambia la forma de realizar pedidos?	A. Muy necesario	B. Necesario	C. Medianamente necesario	D. Poco necesario	E. Innecesario	
		58.72%	36.70%	4.59%	0.00%	0.00%	
24	¿Cuál es su Edad?	A. 18 - 25	B. 26 - 30	C. 30 - 40	D. 40 - 50	E. 50 o más	
		18.35%	22.94%	22.94%	35.78%	9.17%	
25	¿Cuál es su Género?	A. Femenino	B. Masculino	C. Prefiere no decirlo	-	-	
		100.00%	0.00%	0.00%			
26	¿Tiene alguna Discapacidad?	A. Auditiva	B. Visual	C. Física	D. Otro	E. Ninguna	
		0.00%	0.00%	0.00%	4.59%	95.41%	

4.1.2. Identificación de los procesos en la gestión de emprendimientos.

A través de la reunión de planificación del *sprint*, efectuada con la gerenta de la empresa “100 mujeres”, detallada en el anexo 13. Se brindó información acerca del funcionamiento y operación de la organización. En concreto, los procesos de la empresa para la gestión de emprendimientos son: gobierno y preparación; promoción y comunidad; vinculación y participación; pedido y pago; y, por último, operación y cierre.

4.1.3. Diagrama BPD del proceso para la gestión de emprendimientos

Se establecieron ciertos procesos claves a gestionar emprendimientos, mismo que abarca desde las necesidades de la empresa y para las emprendedoras, a las actividades relacionadas con el ecosistema colaborativo. A través del uso de la notación (*BPMN 2.0*), se pudo representar el diagrama de procesos del negocio (*BPM*), en la figura 4, 5, 6, 7, 8 y 9, el flujo de actividades que se ejecutan, permitiendo visualizar de forma clara todo lo involucrado en cada etapa del proceso.

Por consiguiente, el diagrama *BPD* se utilizó como una representación del funcionamiento operativo para el desarrollo de la propuesta de intervención (aplicación *web* y móvil para la gestión de emprendimientos). Asimismo, su uso fue especialmente relevante en la ingeniería de requerimientos bajo un enfoque ágil, particularmente durante reunión de planificación del *sprint* y en la elaboración de historias de usuario. Cabe precisar que el diagrama *BPD* no fue considerado como un artefacto propio del enfoque de investigación descrito en la metodología de la investigación.

Figura 4. Diagrama BPD de gestión de emprendimientos.

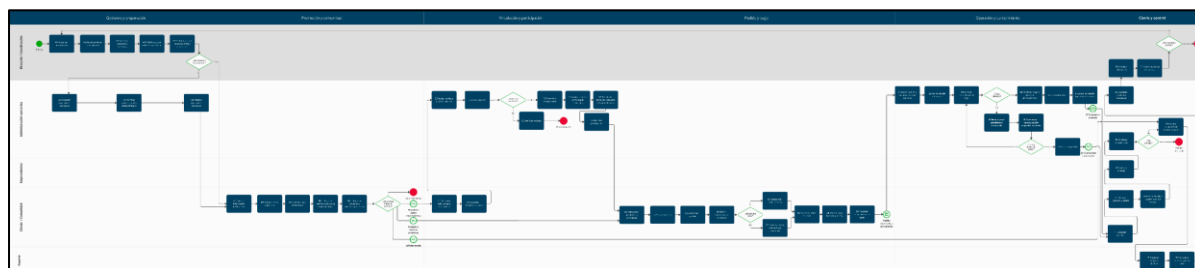


Figura 5. Proceso de Gobierno y preparación

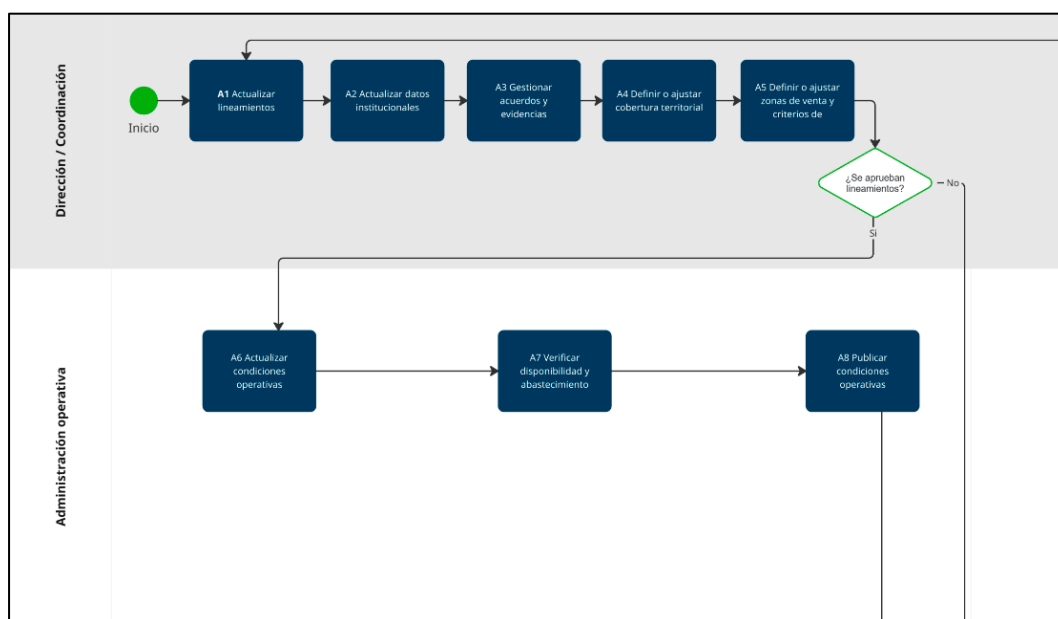


Figura 6. Proceso de Promoción y comunidad

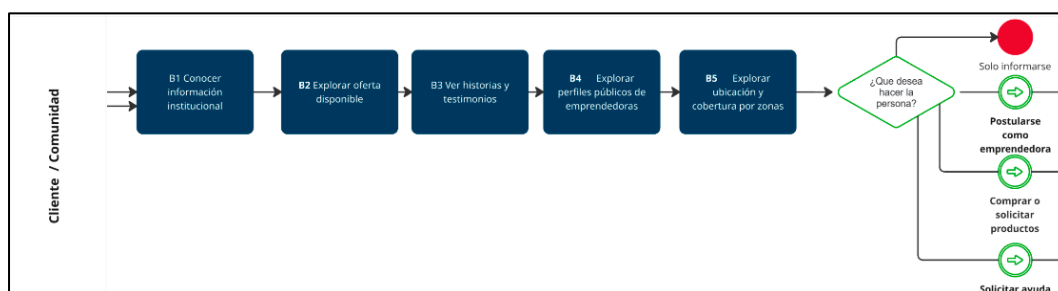


Figura 7. Proceso de Vinculación y participación

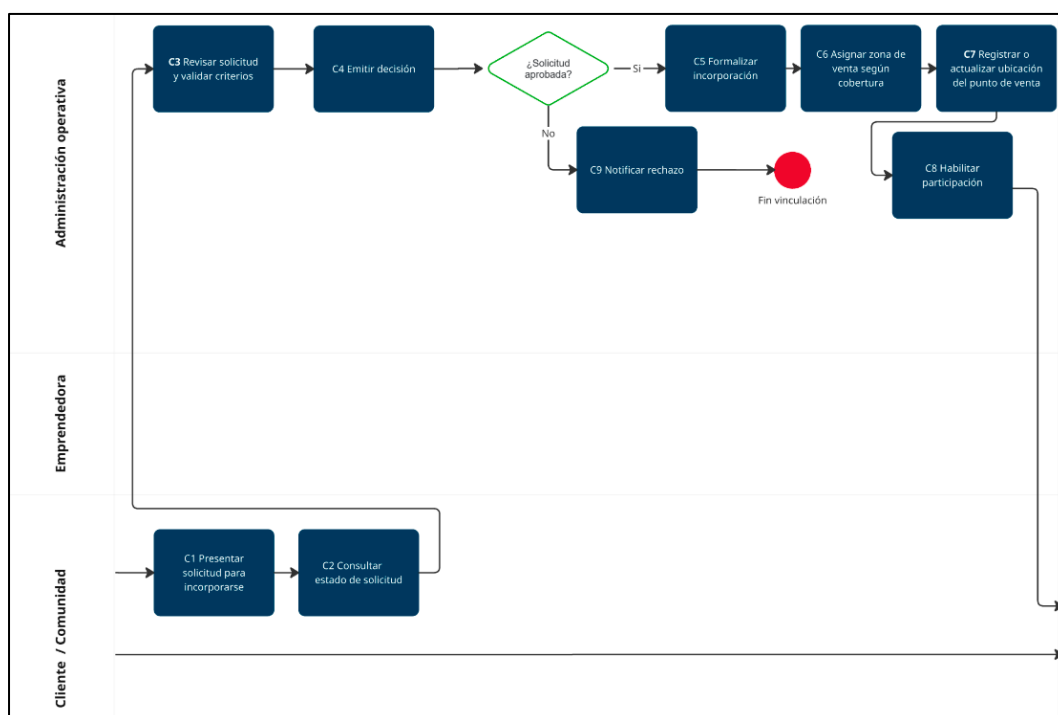


Figura 8. Proceso de Pedido y pago

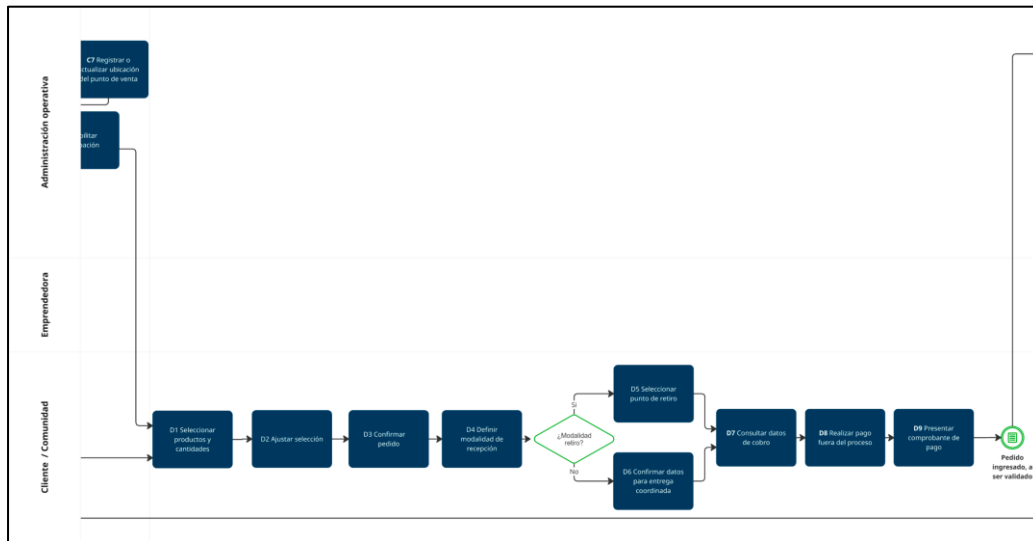
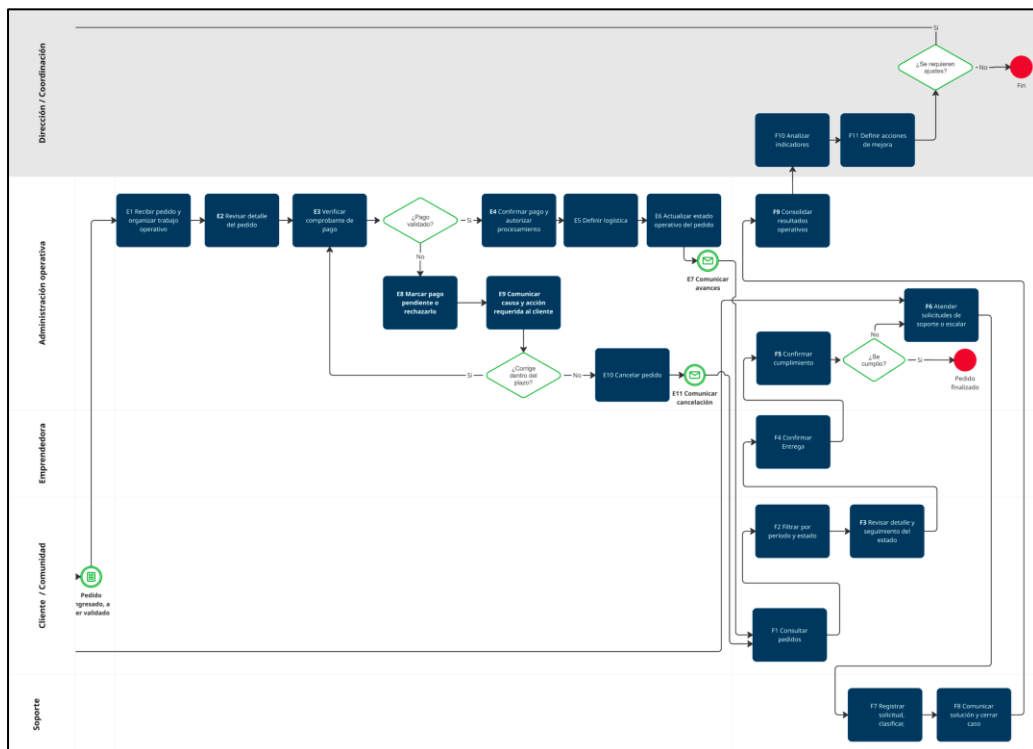


Figura 9. Procesos finales de Operación y cierre.



4.2. Resultado del segundo objetivo específico

Como parte inicial del desarrollo de la aplicación bajo el marco de trabajo *Scrum*, se tomaron las decisiones técnicas necesarias para *web* y *móvil*, considerando además la necesidad de separar responsabilidades entre sus componentes. Para ello, se revisaron y analizaron las alternativas de herramientas a emplear, tanto en: almacenamiento de datos

para ambos entornos, patrones arquitectónicos, tecnologías de *frontend* y *backend*, servicios en la nube y mecanismos de gestión de estado.

4.2.1. Comparación de almacenamiento para web y móvil

El almacenamiento de datos constituye un componente esencial del sistema, ya que condiciona la organización de la información, las operaciones de consulta y la capacidad de adaptación ante cambios estructurales. En este sentido, se comparó un enfoque relacional, representado por *MySQL*, y un enfoque documental *NoSQL*, representado por *Cloud Firestore*. La síntesis comparativa se muestra en la tabla 7.

Tabla 7. Comparación de almacenamiento para la aplicación web y móvil

Criterio	Almacenamiento	
	<i>MySQL (SQL)</i> ^a	<i>Cloud Firestore (NoSQL)</i> ^{bc}
Paradigma	Relacional	<i>NoSQL</i> documental
Unidad de organización	Tablas y filas	Colecciones y documentos
Estructura del esquema	Definida	Flexible
Formato de datos	Registros tabulares	Documentos <i>JSON</i>
Operaciones destacadas	Transacciones, índices y vistas	<i>API</i> documental y operaciones <i>CRUD</i>
Escalabilidad	Menor flexibilidad al crecer	Escalado horizontal
Rasgo funcional relevante	Integridad relacional	Actualización en tiempo real
Consultas	Consultas complejas y relacionales	Lecturas y escrituras directas
Sincronización en tiempo real	Requiere implementación adicional	Integrada entre clientes
Cambio del esquema	Requiere migraciones	Admite variación por documento
Seguridad / transacciones	Integridad y transacciones fuertes	Reglas y operaciones por lotes
Uso recomendado	Datos estructurados y reportes	Datos dinámicos y tiempo real

Nota. Adaptado de: ^a Skrzypiec y Plechawska-Wójcik (2023), ^b Semma et al. (2023), ^c Ariani et al. (2025).

4.2.1.1. Elección de almacenamiento de datos (web)

En la aplicación web, *MySQL* resultó la alternativa más conveniente por su naturaleza relacional y por las funciones asociadas al manejo de transacciones, índices y vistas. Estas características favorecen el tratamiento de información estructurada y la ejecución de consultas orientadas al control administrativo y a la elaboración de reportes.

4.2.1.2. Elección de almacenamiento de datos (móvil)

Para la aplicación móvil se optó por *Cloud Firestore*, ya que su modelo documental, la flexibilidad del esquema y la actualización en tiempo real se ajustan mejor a la gestión de pedidos, usuarios y cambios de estado dentro de una interfaz con interacción continua.

4.2.2. Comparación de patrones arquitectónicos

Los patrones arquitectónicos permiten organizar responsabilidades y delimitar la relación entre la lógica de negocio, la interfaz y el acceso a datos. Por ello, *Clean Architecture*, *MVC* y *MVVM* se utilizaron como referentes conceptuales para contrastar distintas formas de estructurar el sistema. La síntesis comparativa se muestra en la tabla 8.

Tabla 8. Comparación de patrones arquitectónicos

Criterio	Patrones arquitectónicos		
	<i>Clean Architecture</i> ^a	<i>MVC</i> ^b	<i>MVVM</i> ^c
Organización general	Capas: dominio, datos y presentación	Modelo, vista y controlador	Modelo, ViewModel y vista
Separación principal	Negocio e interfaces	Componentes por propósito	Presentación y lógica
Componente de coordinación	Casos de uso y dominio	Controlador	ViewModel
Beneficio destacado	Mantenibilidad	Estructura y claridad	Modularidad y pruebas
Resultado reportado	Mejora de mantenibilidad	Separación funcional en web	Facilita actualizaciones y pruebas
Dependencia dominante	Hacia reglas y contratos	Coordinada por el controlador	Mediada por el ViewModel
Desventaja principal	Mayor esfuerzo inicial	Puede concentrar lógica	Exige control adicional del estado

Nota. Fuente de: ^a Arias-Orezano et al. (2021), ^b Solis-Galván et al. (2022), ^c Ariani et al. (2025).

4.2.2.1. Patrón arquitectónico (web)

En el módulo web no se realizó una adopción estricta de alguno de los patrones comparados, sino una arquitectura modular por capas técnicas con una aproximación parcial a *Clean Architecture*. Esto se observa en la separación funcional entre páginas y componentes para la presentación, y servicios y utilidades para la autenticación, el acceso a datos y parte de la lógica compartida, aunque todavía sin una capa de dominio formal e independiente.

4.2.2.2. Patrón arquitectónico (móvil)

En la aplicación móvil sí se adoptó *Clean Architecture*, porque permite separar la lógica de negocio de los detalles de infraestructura e interfaz y, con ello, fortalecer la mantenibilidad del sistema. En este sentido, Arias-Orezano et al. (2021) reportan mejoras de 7 % en la capacidad de análisis y 56 % en estabilidad al aplicar arquitectura limpia junto con la norma *ISO/IEC 25010* (p. 15).

4.2.3. Tecnologías de *frontend*

Las tecnologías de *frontend* determinan la forma en que se construyen las interfaces, la reutilización de componentes y la experiencia de interacción con el sistema. Por ello, se compararon las alternativas seleccionadas para *web* y móvil, a fin de justificar técnicamente la elección realizada en cada entorno.

4.2.3.1. Tecnologías de *frontend* de *web*

En el entorno *web* se compararon *Angular* y *React*, debido a que ambas opciones permiten construir interfaces modernas, modulares y escalables. La síntesis de sus características principales se presenta en la tabla 9.

Tabla 9. Comparación de tecnologías de *frontend web*

Criterio	Tecnologías de <i>frontend web</i>	
	<i>Angular</i>^a	<i>React</i>^b
<i>Naturaleza</i>	<i>Framework</i>	Biblioteca de interfaz
<i>Organización asociada</i>	<i>MVC</i>	Compatible con <i>MVC</i> o <i>MVVM</i>
<i>Lenguaje principal</i>	<i>TypeScript</i>	<i>JavaScript</i> con <i>JSX</i>
<i>Rasgo distintivo</i>	Inyección de dependencias y <i>Material</i>	<i>Virtual DOM</i>
<i>Código</i>	Limpio y testeable	Legible y mantenible
<i>Flexibilidad</i>	Más integrada	Más abierta
<i>Sintaxis de interfaz</i>	Plantillas y directivas	<i>JavaScript</i> con <i>JSX</i>
<i>Soporte de UI</i>	<i>Angular Material</i>	Componentes reutilizables
<i>Actualización visual</i>	Integrada en el <i>framework</i>	<i>Virtual DOM</i> selectivo

Nota. Fuente de: ^{a,b} Skrzypiec y Plechawska-Wójcik (2023).

La interfaz web se desarrolló con *React* y *Vite*, debido a que esta combinación favorece una construcción más dinámica y modular de la aplicación. Su enfoque centrado en la interfaz, el uso de *JSX* y el empleo del virtual *DOM* contribuyen a disponer de componentes reutilizables y de un mantenimiento más flexible.

4.2.3.2. Tecnologías de *frontend* de móvil

En la tabla 10, se visualiza la para el entorno móvil la comparativa entre *Flutter* y *React Native*, debido a que ambas herramientas permiten el desarrollo multiplataforma.

Tabla 10. Comparación de tecnologías de *frontend* móvil

Criterio	Tecnologías de <i>frontend</i> móvil	
	<i>Flutter</i>^{ab}	<i>React Native</i>^b
<i>Tipo de tecnología</i>	SDK y <i>framework</i>	Estructura multiplataforma
<i>Lenguaje principal</i>	<i>Dart</i>	<i>JavaScript</i>
<i>Construcción de interfaz</i>	<i>Widgets</i>	Componentes
<i>Compilación</i>	Motor <i>Skia</i>	Rendimiento cercano a nativo
<i>Rasgo distintivo</i>	Interfaces expresivas y flexibles	Componentes similares a <i>HTML</i>
<i>Enfoque de desarrollo</i>	<i>Widgets</i> y render propio	Componentes y complementos
<i>Comunidad / adopción</i>	Popularidad creciente en <i>GitHub</i>	Alta visibilidad en <i>GitHub</i>

Nota. Fuente de: ^a Solis-Galván et al. (2022), ^b Gülcüoğlu et al. (2021).

En el entorno móvil, *Flutter* fue la tecnología seleccionada por el uso de *Dart*, la construcción basada en widgets y la posibilidad de generar aplicaciones para *iOS* y *Android* desde un mismo código fuente. A ello se suma la reducción del tiempo de desarrollo sin afectar el rendimiento de la aplicación.

4.2.4. Tecnologías de *backend* y servicios en la nube

Las tecnologías de *backend* y los servicios en la nube permiten gestionar la lógica del sistema, procesar solicitudes, almacenar información y automatizar tareas complementarias. Por ello, se analizaron alternativas del lado servidor y de integración *cloud* tanto para el módulo *web* como para la aplicación móvil.

4.2.4.1. Tecnologías de *backend* (web)

Para el *backend web* se consideraron *Node.js* y *Django*, dado que representan enfoques distintos de construcción del lado del servidor. La síntesis comparativa se muestra en la tabla 11.

Tabla 11. Comparación de tecnologías de *backend web*

Criterio	Tecnologías de <i>backend web</i>	
	<i>Node.js</i>^a	<i>Django</i>^b
<i>Base tecnológica</i>	Runtime de <i>JavaScript</i>	Framework en <i>Python</i>
<i>Modelo de ejecución</i>	Asíncrono y orientado a eventos	<i>MVT</i> como variante de <i>MVC</i>
<i>Gestión de solicitudes</i>	<i>Event loop</i> no bloqueante	View coordina <i>model</i> y <i>template</i>

<i>Fortalezas</i>	Manejo de concurrencia	Separación funcional en <i>web</i>
<i>Uso representativo</i>	Aplicaciones escalables	Aplicaciones <i>web</i> con vistas y plantillas
<i>Base / motor</i>	Motor V8	<i>Framework</i> con <i>MVT</i>
<i>Enfoque de uso</i>	Servicios y concurrencia	Vistas, modelos y plantillas

Nota. Fuente de: ^a Odeniran et al. (2023), ^b Solis-Galván et al. (2022).

En la comparación realizada, *Node.js* se consideró la alternativa más afín al enfoque de *backend web*, debido a su modelo asíncrono y a su compatibilidad con el ecosistema *JavaScript*. Sin embargo, en la configuración vigente del módulo web, la operación activa depende principalmente de servicios de *Firebase* consumidos desde el cliente *React*; por ello, *Node.js* se mantiene como referencia tecnológica de soporte y no como el *backend* transaccional predominante del *runtime* actual.

4.2.4.2. Servicios en la nube para *backend* (móvil)

En el entorno móvil, la decisión no recayó en una sola base de datos, sino en un conjunto de servicios integrados para autenticación, almacenamiento, sincronización, automatización y notificaciones. En este caso, la solución seleccionada fue *Firebase*. La síntesis funcional se presenta en la tabla 12.

Tabla 12. Servicios de *Firebase* considerados para el *backend* móvil

Servicio	<i>Función principal</i>	<i>Aporte al proyecto</i>
<i>Firebase Authentication</i> ^a	Registro e inicio de sesión	Control de acceso de usuarios
<i>Cloud Firestore</i> ^{ab}	Datos y sincronización	Pedidos, usuarios y estados
<i>Firebase Storage</i> ^a	Almacenamiento de archivos	Comprobantes y adjuntos
<i>Cloud Functions</i> ^a	Lógica <i>serverless</i>	Validaciones y automatización
<i>Cloud Messaging</i> ^a	Notificaciones	Avisos y recordatorios

Nota. Fuente de: ^a Ariani et al. (2025), ^b Semma et al. (2023).

Para el *backend* móvil se recurrió a *Firebase* como plataforma de soporte, debido a que reúne en un mismo entorno los servicios necesarios para autenticación, datos, archivos, automatización y notificaciones. Asimismo, Ariani et al. (2025) reportan tiempos de respuesta de 120 ms en lectura y 140 ms en escritura, lo que respalda su viabilidad operativa en escenarios de uso en tiempo real (pp. 63, 70).

4.2.5. Gestión de estado en la aplicación móvil

La gestión de estado es relevante en el desarrollo móvil porque permite coordinar la interacción entre la interfaz, la lógica de procesamiento y el acceso a los datos. Para sustentar esta decisión, se compararon los enfoques *BLoC* y *MVVM*, como se presenta en la tabla 13, tomando en cuenta que la implementación real del proyecto utiliza el ecosistema *flutter_bloc* con predominio de *Cubit* y uso puntual de *Bloc*.

Tabla 13. Comparación de enfoques de gestión de estado en móvil

Criterio	Gestión de estado en móvil	
	<i>BLoC</i>^a	<i>MVVM</i>^b
<i>Organización</i>	Presentación, <i>BLoC</i> y datos	<i>Model</i> , <i>ViewModel</i> y <i>View</i>
<i>Componente intermedio</i>	<i>BLoC</i>	<i>ViewModel</i>
<i>Separación destacada</i>	Capas funcionales	Presentación y negocio
<i>Beneficio reportado</i>	Organización modular	Modularidad y mantenibilidad
<i>Interacción con la vista</i>	Emite estados a la interfaz	Expone datos a la vista
<i>Adecuación funcional</i>	Flujos complejos y asíncronos	Presentación desacoplada
<i>Nivel de uso en el proyecto</i>	Implementado en la <i>App</i> móvil	Referencia comparativa

Nota. Fuente de: ^a Solis-Galván et al. (2022), ^b Ariani et al. (2025).

En materia de gestión de estado, la aplicación móvil se apoya en la familia *BLoC*, implementada principalmente mediante *Cubit* y, de forma puntual, mediante *Bloc*. Esta elección resulta coherente con la organización modular del proyecto, favorece la separación entre interfaz y lógica de estado y permite coordinar de mejor manera la sesión, los pedidos, el perfil, las notificaciones y los módulos administrativos.

4.3. Resultado del tercer objetivo específico: Aplicación Móvil y Web.

4.3.1. Nomenclatura y Logotipo

En la figura 10, se observa el logotipo proporcionado por la empresa, y utilizado al implementar la solución, ya que expresa de forma clara y directa la identidad del proyecto “100 Mujeres Trabajando”. Se trata de un ícono relacionado con la preparación de alimentos, que simboliza el trabajo y la colaboración de las emprendedoras involucradas. Reforzando la visibilidad de la organización, la cual se mantendrá constante en los distintos apartados de la aplicación *web* y móvil.

Figura 10. Logo de la empresa 100 Mujeres Trabajando.



4.3.2. Marco de Trabajo Scrum

Para la elaboración de la aplicación móvil y aplicación *web*, para la gestión de emprendimientos, se decidió hacer uso del marco de trabajo ágil *Scrum*. El cual, como indican Schwaber y Sutherland (2020), se trata de un marco ágil que estructura el trabajo mediante roles y eventos definidos, apoyándose en principios de empirismo y Lean, para que individuos y equipos generen valor creando soluciones adaptables a problemas complejos (pp. 1, 3). Lo anterior mencionado, será empleado con la finalidad de alcanzar de la forma más eficiente los objetivos previamente fijados.

4.3.3. Sprint 1

4.3.3.1. Sprint 1 – Planificación (*Sprint Planning*)

Según lo señalado por Schwaber y Sutherland (2020), cada *sprint* comienza con una planificación colaborativa para definir el trabajo a realizar. Esta fase inicial organiza y guía el desarrollo dentro de un evento con duración máxima de un mes, asegurando que cada nuevo *sprint* inicie de inmediato tras la finalización del anterior para mantener consistencia (p. 8). Así, se sostiene que esta fase es crucial para alcanzar el éxito durante la implementación de la aplicación para móviles y páginas *web*. En relación con el enfoque laboral Scrum, se implementaron acciones como: Realizar cuatro *sprints*, basados en las exigencias y las prioridades del cliente.

4.3.3.1.1. Roles

En *Scrum*, como se indica en la tabla 14, se establecieron los roles específicos para la ejecución del marco de trabajo. Según SCRUMstudy (2022) existen tres roles esenciales que juntos conforman el núcleo del equipo, para garantizar el cumplimiento de los objetivos del proyecto. En primer lugar, el *Product Owner* maximiza el valor del proyecto y actúa como voz del cliente. Además, el *Scrum Master*, el cual, facilita el proceso y vela porque se aplique correctamente el marco de trabajo. Y, por último, el *Scrum Team* se encarga de comprender los requisitos y desarrollar el producto acordado (pp. 40-41).

Tabla 14. Distribución de los Roles asignados.

Área	Persona	Roles
Gerente de la empresa 100 Mujeres Trabajando	MSc. Ximena Toro	Dueño el Producto (<i>Product Owner</i>)
Docente de la PUCESD	Mg. Willian Ocampo	<i>Scrum Master</i>
Desarrollador <i>Frontend</i> y <i>Backend</i>	Marco Silva	Equipo de Desarrollo y Testeo
Diseñador <i>Web</i> y <i>Tester Web</i>	Leonardo Diaz	Equipo de Desarrollo y Testeo

4.3.3.1.2. Modelo Vista y Gestión de Estado con Clean Architecture

La solución se estructuró con separación entre presentación, lógica de negocio y acceso a datos. En móvil esta organización se aproxima a *Clean Architecture* con capas definidas, en *web*, la separación es funcional entre vistas, componentes, contexto y servicios, por lo que su adopción es parcial.

- **Presentación:** agrupa pantallas, *widgets* y componentes responsables de la interacción, navegación y visualización de la información.
- **Dominio y lógica:** reúne entidades, casos de uso y reglas de negocio; en la *web* también se apoya en mapeadores, ayudas y servicios compartidos.
- **Datos e integración:** concentra repositorios, *datasources* y servicios para la autenticación, consulta, persistencia y conexión con recursos remotos.
- **Gestión de estado:** se apoya principalmente en *flutter_bloc* en móvil, y en *useState* con uso acotado de *Context API* en la aplicación *web*.

4.3.3.1.3. *Infraestructura Cloud*

Para la infraestructura en la nube se implementó *Firebase*, como plataforma principal. La gestión de datos recae en *Cloud Firestore* y el control de acceso en *Authentication*. Además, la solución incorpora servicios complementarios para notificaciones, funciones auxiliares, protección del cliente y archivos.

- **Cloud Firestore:** centraliza la información de usuarios, perfiles, pedidos, catálogos y notificaciones del sistema.
- **Authentication:** administra el registro, inicio de sesión, verificación y control de acceso según el tipo de usuario.
- **Servicios auxiliares:** se emplean *Cloud Functions*, *Firebase Cloud Messaging* y *App Check* para procesos complementarios dentro de la solución.
- **Servidor de archivos:** su manejo se apoya en *Cloudinary* y en soporte de almacenamiento integrado en la infraestructura implementada.

4.3.3.1.4. *Parametrización*

Se definió la estructura del proyecto siguiendo los principios de *Clean Architecture* conforme a la tecnología de cada plataforma, centralizando el arranque, las rutas, la configuración y la integración con servicios externos. Esta organización favorece la mantenibilidad del sistema y permite una distribución clara de responsabilidades técnicas.

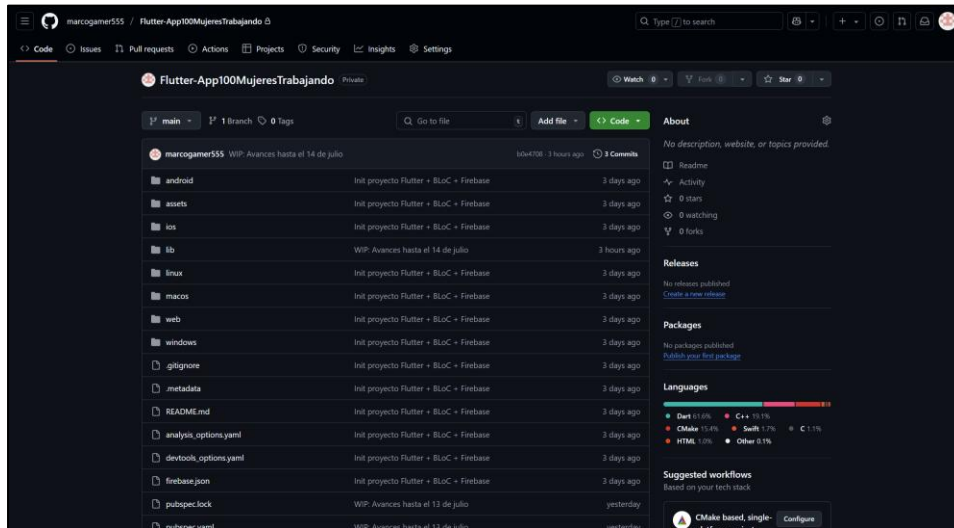
- **Estructura móvil:** se organiza en *lib/modules*, separando *app*, *core* y *features*; en varias funcionalidades se distinguen *data*, *domain* y *presentation*.
- **Estructura web:** se distribuye en *src* mediante *pages*, *components*, *context*, *services*, *helpers*, *mappers*, *config* y *constants*.

4.3.3.1.5. *Control de Versiones*

Se utilizó *GitHub* como herramienta de control de versiones, permitiendo colaboración simultánea, gestión de ramas y revisiones de código. Facilita la administración remota del repositorio y sus actualizaciones, permitiendo guardar avances, gestionar cambios y asegurar la continuidad del trabajo sin contratiempos. Además, favorece la

revisión de aportes de manera estructurada y segura, impulsando la productividad y la integración del equipo. Este proceso se refleja en la figura 11.

Figura 11. Repositorio *GitHub*, para la aplicación “100 Mujeres Trabajando”.



4.3.3.1.6. Product Backlog

Toda planificación ágil precisa contar con unas bases bien definidas, dicho aquello, este proyecto fue elaborado estableciendo un producto *backlog*. Este elemento reúne funciones requeridas por el sistema, mismo que es construido a partir los requerimientos requeridos tras recopilar la información. Identificando, organizando y priorizando el trabajo a realizar en *sprints*. La tabla 15, 16 y 17, resumen la estructura con la estimación, prioridad y riesgo de cada historia de usuario e historias técnicas, esto se detalló en el anexo 4.

Tabla 15. *Product Backlog* (Aplicación móvil)

Nº	Nombre de la historia	Valor Estimado	Prioridad	Riesgo de desarrollo
1	Pantalla de Bienvenida	1	Media	Bajo
2	Configuración de accesibilidad y preferencias de visualización	5	Media	Medio
3	Selección de Rol	1	Media	Bajo
4	Registro de Usuario	5	Alta	Medio
5	Ingreso de Usuario	5	Alta	Medio
6	Solicitud de Acceso para Administradoras	3	Alta	Bajo
7	<i>Dashboard</i> Principal y Navegación por Rol	8	Muy Alta	Moderado
8	Visualización del Perfil de Usuario	3	Alta	Bajo
9	Cerrar Sesión del Usuario	3	Media	Bajo
10	Recuperación de Contraseña	3	Alta	Bajo
11	Edición del Perfil de Usuario	5	Alta	Moderado
12	Nuevas presentaciones de productos	2	Alta	Bajo

13	Realizar pedido de bandejas productos	5	Muy Alta	Moderado
14	Adjuntar comprobante de pago	3	Alta	Bajo
15	Ampliación del pedido: tipos de entrega	5	Alta	Moderado
16	Selección de punto de retiro	5	Alta	Moderado
17	Gestión de pedidos por parte del administrador	8	Muy Alta	Alto
18	Verificación de pago (<i>admin</i>)	5	Alta	Moderado
19	Notificaciones <i>push</i>	10	Muy Alta	Alto
20	Consulta de pedidos por fecha (cliente)	3	Moderada	Bajo
21	Consulta de pedidos por fecha (administrador)	3	Moderada	Bajo
22	Agendamiento de entregas	5	Alta	Moderado
23	Gestión de zonas de venta	5	Alta	Moderado
24	Solicitud de nueva emprendedora del proyecto	5	Alta	Moderado
25	Registro de acuerdos privados de ubicación	3	Moderada	Moderado
26	Mapa de puntos de venta cercanos	8	Muy Alta	Alto
27	Ver perfiles de emprendedoras	3	Alta	Bajo
28	Testimonios o mensajes motivacionales	2	Baja	Bajo
29	Visualización de estadísticas (sin exportar)	5	Moderada	Alto
30	Registro de ubicación por la emprendedora	8	Moderada	Alto

Tabla 16. *Product Backlog* (Plataforma Web)

Nº	Nombre de la historia	Valor Estimado	Prioridad	Riesgo de desarrollo
1	Menú Principal – Llamado a la Acción	5	Alta	Alto
2	Registro de Emprendedoras	3	Moderada	Alto
3	Login de Emprendedoras	4	Muy Alta	Medio
4	<i>Login</i> para Administradora	5	Alta	Alto
5	Perfil de Emprendedoras	4	Muy Alta	Alto
6	Realizar Pedido	5	Moderada	Medio
7	Registro de pagos	4	Alta	Alto
8	Historial de Pedidos	3	Alta	Alto
9	Administrador <i>Dashboard</i>	5	Media	Medio
10	<i>Admin</i> Gestión Pedidos	5	Muy Alta	Alto
11	<i>Admin</i> Gestión Productos	4	Alta	Alto
12	<i>Admin</i> Gestión Usuarios	3	Media	Media

Tabla 17. *Product Backlog* Técnico

Nº	Nombre de la historia	Valor Estimado	Prioridad	Riesgo de desarrollo
1	Seguridad y privacidad de la aplicación móvil	8	Alta	Alto
2	Seguridad y privacidad de la aplicación <i>web</i>	8	Alta	Alto

4.3.3.1.7. Estimación

De acuerdo con la evaluación realizada, se optó por la estimación mediante puntos de historia, usando una escala de *Fibonacci* que abarca de 1 a 34, donde 1 representa el

menor esfuerzo y complejidad y 34 el mayor. Tras asignar los puntos a cada historia, se distribuyeron para estimar las tareas necesarias. Para estimar las tareas de desarrollo de forma colaborativa y objetiva, se empleó la técnica *Planning Poker* utilizando la secuencia de *Fibonacci* como escala de puntos de historia.

Esta metodología, ampliamente aplicada en proyectos ágiles, permite llegar a un consenso entre los miembros del equipo mediante la comparación y debate de las valoraciones individuales. Lo mencionado por Sahni (2022), se comprende que el uso de esta secuencia facilita valorar las complejidades significativas, a fin de llegar a consensos en estimación de puntos adecuados, evitando aproximaciones imprecisas (pp. 2-3). De igual forma, Chauhan et al. (2023). reconoce que el *Planning Poker* integra el juicio profesional, la comparación por analogía y la descomposición de tareas, logrando estimaciones rápidas y confiables (pp. 2649, 2652). La aplicación de este proceso colaborativo contribuyó a organizar y valorar cada historia de usuario de acuerdo con su dificultad relativa, optimizando así la planificación del *backlog* del proyecto.

4.3.3.1.8. Velocidad de Desarrollo

Al determinar el alcance del *sprint*, se consideran las estimaciones de las historias de usuario conforme a lo especificado por el *Product Owner*. Se estima que la primera historia tiene 1 punto; la segunda, 5; la tercera, 1; y las cuarta y quinta también tienen 5. Las historias sexta, séptima y octava se estiman en 3 puntos cada una; por último, la novena historia se estimaron en 3 puntos. Esto produce un total de 29 puntos relacionados con la historia que deben completarse en un mes o en cuatro semanas.

4.3.3.1.9. Escenarios de Prueba

Una vez que se han terminado las historias de usuario, es necesario implementar los escenarios de prueba para verificar que el sistema ha cumplido con las funcionalidades solicitadas, las cuales se evidencian en el anexo 4. En este sentido, Cohn (2006) indica que las condiciones de satisfacción constituyen, en esencia, pruebas de aceptación de alto nivel, y que su especificación mediante ejemplos aporta beneficios al trabajo de los

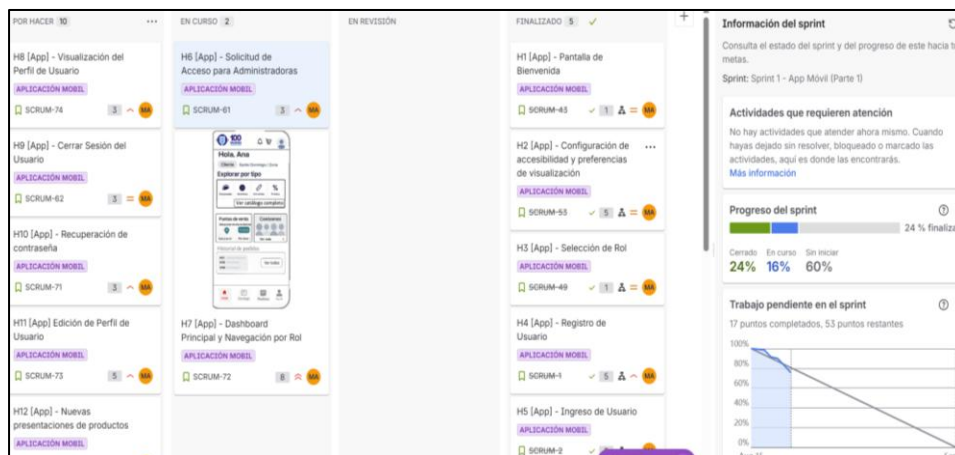
programadores (pp. 228-229). En este sentido, el anexo 5 detalla las pruebas de aceptación a las funcionalidades, las cuales fueron verificadas en la reunión con el *Product Owner*.

4.3.3.1.10. Gestión de tareas de ingeniería

El manejo de incidencias y el seguimiento de los progresos de cada trabajo, se observa en el registro de las tareas completadas en la herramienta colaborativa *Jira*, que posibilita la organización y gestión de las tareas a través de un tablero *kanban*, como se puede ver en la figura 12.

La herramienta, contiene los apartados: "*to do*" contiene distintas actividades e historias a llevarse a cabo. la sección "*progress*", define todas las tareas en desarrollo actual del equipo. Finalmente, en "*done*", se enumera todas las tareas ya completadas. Adicionalmente, el gráfico de trabajo pendiente muestra el progreso avanzado.

Figura 12. Gestión de Incidencias *Jira*



4.3.3.1.11. Sprint Backlog

La tabla 18 muestra que, se eligieron cinco historias de usuarios que se consideraban más esenciales. Para el primer *sprint*, en conjunto, se estimaron 21 puntos. Según Schwaber y Sutherland (2020) el objetivo del *sprint backlog* es organizar las tareas que los desarrolladores tienen que llevar a cabo, mediante un plan visible y accionable para alcanzar el objetivo establecido y entregar el incremento previsto (pp. 6, 8-9, 12).

Tabla 18. Sprint backlog 1

Sprint Backlog					
Sprint: 1					
Objetivo: Implementar el flujo base de acceso y pedidos (bienvenida, autenticación por rol, perfil, catálogo, creación de pedidos, comprobante y gestión inicial de pedidos y productos).					
Historia	Est	Categoría	Tarea	Est	Estado
H1 [App] – Pantalla de Bienvenida	1	Navegación	Configurar ruta inicial (<i>StartupPage/AppRoutes</i>) hacia <i>WelcomePage</i> cuando no hay sesión.	0.25	Completado
		UI	Construir <i>WelcomePage</i> (logo, frase y botones: Registrarse/ Iniciar sesión/Visitante).	0.5	Completado
		Lógica	Conectar acciones de botones y modo visitante hacia <i>Home</i> .	0.25	Completado
H2 [App] – Configuración de accesibilidad y preferencia de visualización	5	Lógica / Estado	Crear <i>SettingsCubit</i> (<i>themeMode</i> + <i>textScaleFactor</i>) y estado reactivo.	1.5	Completado
		UI / Accesibilidad	Implementar <i>SettingsDialog</i> con slider de tamaño de texto y selector de tema.	2	Completado
		Integración UI	Aplicar preferencias al <i>MaterialApp</i> (<i>themeMode</i> y <i>TextScaler</i>).	1.5	Completado
H3 [App] - Selección de Rol	1	UI	Pantalla de selección de rol (Cliente/Emprendedora) con botón continuar.	0.5	Completado
		Validación	Validar selección obligatoria y mostrar mensaje si falta.	0.25	Completado
		Navegación	Navegar al registro correspondiente según rol.	0.25	Completado
H4 [App] - Registro de Usuario	5	UI	Formulario de registro (datos personales + credenciales).	2	Completado
		Validación	Validaciones de campos (requeridos/formato) y <i>feedback</i> .	1	Completado
		Firestore / Datos	Crear usuario (<i>Auth</i>) y guardar perfil/rol en <i>Firestore</i> .	2	Completado
H5 [App] - Ingreso de Usuario	5	UI	Pantalla <i>Login</i> (cédula/contraseña) con mensajes de error.	2	Completado
		Lógica	Autenticación por documento (<i>lookup</i>) y manejo de sesión.	2	Completado
		Navegación	Redirección <i>post-login</i> al <i>dashboard</i> según rol.	1	Completado
H6 [App] - Solicitud de Acceso para Administradoras	3	UI	Pantalla <i>AccessRequest</i> (ingreso de código + acceso a <i>wizard</i>).	1	Completado
		UI / Validación	<i>AdminWizard</i> : formulario + validaciones básicas.	1.25	Completado
		Firestore / Datos	Crear cuenta <i>admin</i> / consumir invitación (flujo implementado).	0.75	Completado
H7 [App] - Dashboard Principal y Navegación por Rol	3	UI	<i>Dashboard/Home</i> con opciones visibles por rol (cliente/ emprendedora/ <i>admin</i> /visitante).	1.25	Completado
		Lógica / Estado	<i>HomeCubit</i> : cargar usuario, rol y estado inicial.	1	Completado
		Navegación	Configurar rutas y restricciones por rol en <i>AppRoutes</i> .	0.75	Completado

H8 [App] - Visualización del Perfil de Usuario	3	UI	Perfil: mostrar nombre, cédula, rol, teléfono y foto (si existe).	1	Completado
		Lógica / Estado	<i>ProfileCubit</i> : cargar datos y detectar campos faltantes.	1	Completado
		<i>Firebase</i> / Datos	Lectura de perfil desde <i>Firestore/Storage</i> (si aplica).	1	Completado
H9 [App] - Cerrar Sesión del Usuario	3	UI/UX	Agregar botón 'Cerrar sesión' + diálogo de confirmación.	1	Completado
		<i>Firebase</i> / Datos	Ejecutar <i>signOut</i> y limpiar datos locales de sesión.	1	Completado
		Navegación	Redirigir a <i>Welcome/Login</i> y limpiar <i>stack</i> (bloquear <i>back</i>).	1	Completado
H10 [App] - Recuperación de contraseña	1	UI	<i>ForgotPasswordPage</i> con opción buscar por correo o documento.	0.5	Completado
		Lógica/ <i>Firebase</i>	Buscar cuenta y disparar flujo de recuperación (<i>email/reset</i>).	0.5	Completado
H11 [App] - Edición de Perfil de Usuario	1	UI	Pantalla <i>EditProfile</i> : formulario con datos actuales.	0.4	Completado
		<i>Firebase</i> / Datos	Actualizar campos (y foto si aplica) + validaciones.	0.6	Completado
H12 [App] - Nuevas presentaciones de productos	5	UI	<i>Workspace</i> de catálogo <i>admin</i> : listado por categorías + tarjetas de producto.	2	Completado
		UI / Validación	<i>ProductForm</i> : presentación, precio, variantes, visible/oculto, validaciones.	1.5	Completado
		<i>Firebase</i> / Datos	CRUD en <i>Firestore</i> + carga de imagen (<i>Cloudinary/Storage fallback</i>).	1.5	Completado
H13 [App] - Realizar pedido de bandejas productos	5	Navegación	Flujo catálogo → detalle → carrito → <i>checkout</i>	1.5	Completado
		UI / Validación	<i>CheckoutFromCart</i> : captura datos requeridos y confirmación.	2	Completado
		<i>Firebase</i> / Datos	Crear pedido en <i>Firestore</i> con estado inicial y <i>timestamps</i> .	1.5	Completado
H14 [App] - Adjuntar comprobante de pago	3	UI	Sección de comprobante: botón subir, <i>thumbnail</i> y visor.	1	Completado
		Lógica	Seleccionar/validar archivo y controlar estados (en revisión/ <i>pending</i>).	1	Completado
		<i>Firebase</i> / Storage	Subir comprobante y actualizar <i>order (proofUrl + paymentStatus)</i> .	1	Completado
H15 [App] - Ampliación del pedido: tipos de entrega	5	UI	Selector Retiro/Entrega y campos condicionales <i>checkout</i> .	2	Completado
		Lógica / Estado	<i>DeliveryConfigCubit</i> : cargar <i>config (costo/flags)</i> y validarla.	1.5	Completado
		<i>Firebase</i> / Datos	Guardar <i>deliveryMethod + address/pickup</i> en el pedido.	1.5	Completado
H16 [App] - Selección de punto de retiro	5	UI	<i>Dropdown</i> /lista de puntos de retiro en <i>checkout (pickup_points)</i> .	2	Completado
		UI	Mostrar datos del punto en detalle/historial (dirección/foto).	1.5	Completado
		UX	Acción 'Cómo llegar' usando <i>mapsUrl</i> (si existe).	1.5	Completado

H17 [App] - Gestión de pedidos por parte del administrador	8	UI	Admin Pedidos: lista operativa con <i>badges</i> (estado, pago, método).	2.5	Completado
		UI	Detalle <i>admin</i> (cliente, <i>items</i> , entrega/retiro, comprobante).	2	Completado
		Lógica	Acciones para cambio de estado (recibido→entregado/cancelado)	1.5	Completado
		Firestore / Datos	Actualización de pedido en <i>Firestore</i> y refresco en cliente.	1.5	Completado
		Backend / Placeholder	Dejar listo el evento para notificación (<i>Cloud Function</i>).	0.5	Completado

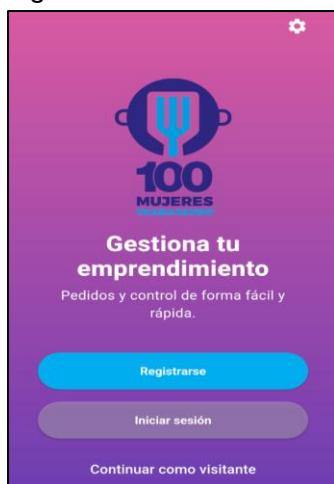
4.3.3.2. Sprint 1 - Reuniones diarias (Daily Scrum)

Se llevaron a cabo reuniones empleando la planificación efectuada en el *sprint* 1, diario, cuyo tiempo máximo fue de 20 minutos por día. En estos, se registraron las actividades, obstáculos durante el desarrollo de las tareas, y se determinaron las próximas actividades. Para tal fin, con el fin de mantener un seguimiento eficaz de las tareas del equipo de desarrollo, se implementó el uso de la plataforma *Jira*, vinculada al desarrollo de *software* relacionado con las historias de usuario, este recurso posibilita observar el progreso de cada tarea, estableciendo las prioridades a medida que se avanza en el *sprint*.

4.3.3.2.1. Historia de Usuario 1: Pantalla de Bienvenida

El anexo 4 reúne la historia correspondiente a la pantalla de bienvenida, desde la cual se presenta el acceso inicial a la aplicación y se delimitan las tres alternativas de continuidad: registro, ingreso y visita. La figura 13 sitúa este punto de partida y muestra una entrada clara al flujo general de uso.

Figura 13. Interfaz móvil de pantalla de bienvenida a la App



La implementación de este comportamiento se distribuye entre la composición visible y el control de estado. Las figuras 14 y 15 muestran que *welcome_page.dart* organiza la vista inicial, mientras *WelcomeCubit* y *WelcomeState* sostienen la respuesta ante cada acción y conducen la navegación sin trasladar esa lógica al plano visual.

Figura 14. Código de *welcome_page.dart*

```

1 // modules/features/auth/presentation/pages/welcome_page.dart
2 import 'package:flutter/material.dart';
3 import 'package:flutter_bloc/flutter_bloc.dart';
4
5 import '../../../core/config/app_theme.dart';
6 import '../../../core/config/app_routes.dart';
7 import '../../../core/widgets/settings_dialog.dart';
8 import '../../../core/widgets/layout/app_logo.dart';
9 import '../../../core/widgets/actions/buttons/primary_button.dart';
10 import '../../../blocks/welcome_cubit.dart';
11
12 You, 3 months ago | 1 author (You)
13 class WelcomePage extends StatelessWidget {
14   const WelcomePage({super.key});
15
16   @override
17   StatefulElement createElement() => _WelcomePageStateElement();
18 }
19
20 You, 2 months ago | 1 author (You)
21 class _WelcomePageStateElement extends StatefulElement {
22   @override
23   Widget build(BuildContext context) {
24     return BlocListener<WelcomeCubit, WelcomeState>(
25       listener: (context, state) {
26         // Siempre saltamos el foco justo antes de navegar.
27         FocusScope.of(context).unfocus();
28
29         if (state is ContinueAsGuest) {
30           context.read<WelcomeCubit>().reset();
31           Navigator.pushNamedAndRemoveUntil(
32             context,
33             AppRoutes.home,
34             (_) => false,
35           );
36         } else if (state is GoToRegister) {
37           context.read<WelcomeCubit>().reset();
38           Navigator.pushReplacementNamed(context, AppRoutes.roleSelection);
39         } else if (state is GoToLogin) {
40           context.read<WelcomeCubit>().reset();
41           Navigator.pushReplacementNamed(context, AppRoutes.login);
42         }
43       },
44     );
45   }
46 }

```

Figura 15. Código de *welcome_cubit.dart* y *welcome_state.dart*

```

1 // modules/features/auth/presentation/blocs/welcome_cubit.dart
2 import 'package:flutter_bloc/flutter_bloc.dart';
3
4 part 'welcome_state.dart';
5
6 You, 3 months ago | 1 author (You)
7 class WelcomeCubit extends Cubit<WelcomeState> {
8   WelcomeCubit() : super(WelcomeInitial());
9
10   void continueAsGuest() => emit(ContinueAsGuest());
11   void goToRegister() => emit(GoToRegister());
12   void goToLogin() => emit(GoToLogin());
13
14   // << NUEVO: para evitar que el listener navegue
15   void reset() => emit(WelcomeInitial());
16 }

```

```

1 lib > modules > features > auth > presentation > blocs > welcome_state.dart
2 You, 3 months ago | 1 author (You)
3 // modules/features/auth/presentation/blocs/welcome_state.dart
4 part of 'welcome_cubit.dart';
5
6 abstract class WelcomeState {}
7
8 class WelcomeInitial extends WelcomeState {}
9
10 class ContinueAsGuest extends WelcomeState {}
11
12 class GoToLogin extends WelcomeState {}
13
14 class GoToRegister extends WelcomeState {}

```

4.3.3.2.2. Historia de Usuario 2: Configuración de accesibilidad y preferencias de visualización

Dentro del anexo 4, esta historia incorpora la configuración de accesibilidad como un recurso transversal para ajustar la lectura y la apariencia general de la aplicación. En la figura 16, se ubica el diálogo desde el cual se modifican el tamaño de texto y el tema, integrando estas preferencias sin interrumpir el flujo principal.

Figura 16. Interfaz móvil del *widget* de configuración de Accesibilidad

El comportamiento inmediato del módulo se apoya en *settings_dialog.dart* y *settings_cubit.dart*, presentados en las figuras 17 y 18. En este bloque se articula tanto la interacción del diálogo como la actualización reactiva del estado, lo que permite que los cambios se reflejen de forma consistente.

Figura 17. Código de *settings_dialog.dart*

```
lib > modules > core > widgets > settings_dialog.dart > ...
You 2 hours ago | 1 author (You)
1 // modules/core/widgets/settings_dialog.dart
2 import 'package:flutter/material.dart';
3 import 'package:flutter_bloc/flutter_bloc.dart';
4
5 import '../blocs/settings_cubit.dart';
6
7 You 3 hours ago | 1 author (You)
7 class SettingsDialog extends StatelessWidget {
8   const SettingsDialog({super.key});
9
10  @override
11  Widget build(BuildContext context) {
12    return AlertDialog(
13      title: const Text('Accesibilidad'),
14      content: SingleChildScrollView(
15        child: BlocBuilder<SettingsCubit, SettingsState>(
16          builder: (context, state) {
17            return Column(
18              mainAxisAlignment: MainAxisAlignment.min,
19              children: [
20                const Text(
21                  'Vista previa:',
22                  style: TextStyle(fontWeight: FontWeight.bold),
23                ), // Text
24                const SizedBox(height: 8),
25                Text(
26                  '100 MUJERES TRABAJANDO',
27                  style: TextStyle(
28                    fontSize:
29                      18 *
30                      state.textScaleFactor.clamp(
31                        SettingsCubit.minTextScale,
32                        SettingsCubit.maxTextScale,
```

Figura 18. Código de *settings_cubit.dart*

```
lib > modules > core > blocs > settings_cubit.dart > ...
You 3 hours ago | 1 author (You)
1 // modules/core/blocs/settings_cubit.dart
2 import 'package:flutter_bloc/flutter_bloc.dart';
3
4 import '../services/storage/settings_storage.dart';
5
6 enum AppThemeMode { system, light, dark }
7
8 You 6 months ago | 1 author (You)
8 class SettingsState {
9   final double textScaleFactor;
10   final AppThemeMode themeMode;
11
12   SettingsState({
13     required this.textScaleFactor,
14     required this.themeMode,
15   });
16
17   SettingsState copyWith({
18     double? textScaleFactor,
19     AppThemeMode? themeMode,
20   }) {
21     return SettingsState(
22       textScaleFactor: textScaleFactor ?? this.textScaleFactor,
23       themeMode: themeMode ?? this.themeMode,
24     );
25   }
26 }
27
28 You 3 hours ago | 1 author (You)
28 class SettingsCubit extends Cubit<SettingsState> {
29   static const double minTextScale = 0.8;
30   static const double maxTextScale = 1.2;
31
32   final SettingsStorage _storage;
33
34   SettingsCubit({required SettingsStorage storage})
35     : _storage = storage,
36       super(
37         SettingsState(
38           textScaleFactor: _normalizeScale(storage.getTextScaleFactor()),
39           themeMode: _normalizeTheme(storage.getThemeMode()),
```

La persistencia y la aplicación global se completan con *settings_storage.dart* y *app.dart*, visibles en las figuras 19 y 20. Con ello, las preferencias no quedan restringidas a una pantalla aislada, sino que se conservan y se propagan al resto de la aplicación.

Figura 19. Código de *settings_storage.dart*

```
lib > modules > core > services > storage > settings_storage.dart > ...
You, 3 hours ago | 1 author (You)
1 import 'package:shared_preferences/shared_preferences.dart';
2
3 You, 3 hours ago | 1 author (You)
4 class SettingsStorage {
5     static const _textScaleKey = 'settings_text_scale';
6     static const _themeModeKey = 'settings_theme_mode';
7
8     final SharedPreferences _prefs;
9
10    SettingsStorage(this._prefs);
11
12    double? getTextScaleFactor() => _prefs.getDouble(_textScaleKey);
13
14    String? getThemeMode() => _prefs.getString(_themeModeKey);
15
16    Future<void> setTextScaleFactor(double value) async {
17        await _prefs.setDouble(_textScaleKey, value);
18    }
19
20    Future<void> setThemeMode(String mode) async {
21        await _prefs.setString(_themeModeKey, mode);
22    }
23 }
```

Figura 20. Código de *app.dart*

```
lib > modules > app > app.dart > ...
You, 3 hours ago | 1 author (You)
1 // lib/modules/app/app.dart
2 import 'package:flutter/foundation.dart';
3 import 'package:flutter/material.dart';
4 import 'package:flutter_bloc/flutter_bloc.dart';
5 import 'package:flutter_localizations/flutter_localizations.dart';
6
7 import '../core/blocs/settings_cubit.dart';
8 import '../core/config/app_theme.dart';
9 import '../core/config/app_routes.dart';
10 import '../features/startup/startup_routes.dart';
11 import '../core/navigation/app_navigator.dart';
12 import '../core/config/feature_flags.dart' as ff;
13 import '../core/widgets/appcheck_status_badge.dart';
14 import '../core/widgets/network/connectivity_bud.dart';
15 import '../core/widgets/layout/ios_boot_guard.dart';
16 import '../core/widgets/layout/tap_outside_unfocus.dart';
17 import '../core/widgets/network/connectivity_banner.dart'; // empuja contenido
18
19 You, 3 hours ago | 1 author (You)
20 class App extends StatelessWidget {
21     const App({super.key});
22
23     @override
24     Widget build(BuildContext context) {
25         return BlocBuilder<SettingsCubit, SettingsState>(
26             builder: (context, settings) {
27                 return MaterialApp(
28                     title: 'See Mujeres Trabajando',
29                     theme: AppTheme.lightTheme,
30                     debugShowCheckedModeBanner: false,
31                     darkTheme: AppTheme.darkTheme,
32                     themeMode: () {
33                         switch (settings.themeMode) {
34                             case AppThemeMode.light:
35                                 return ThemeMode.light;
36                             case AppThemeMode.dark:
37                                 return ThemeMode.dark;
38                             case AppThemeMode.system:
39                                 return ThemeMode.system;
40                         }
41                     },
42                     navigatorKey: AppNavigator.navigatorKey,
```

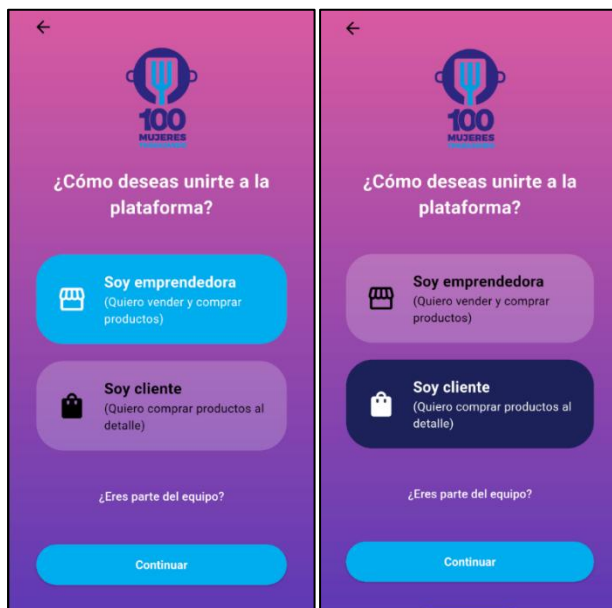
4.3.3.2.3. Historia de Usuario 3: Selección de Rol

Tal como se detalla en el anexo 4, la selección de rol funciona como un punto de bifurcación previo al registro. La figura 21, 22 y 23 presentan la pantalla desde la cual la usuaria define el perfil con el que continuará, de modo que el flujo posterior quede alineado con el tipo de participación esperado.

Figura 21. Interfaz móvil de selección de rol



Figura 22. Interfaz móvil de selección del rol Emprendedora / Cliente

Figura 23. Estructura de la interfaz `role_selection_page.dart`: disposición del logotipo

```

lib > modules > features > auth > presentation > pages > role_selection_page.dart > ...
15 class RoleSelectionPage extends StatelessWidget {
16   Widget build(BuildContext context) {
17     // ...
18     builder: (context, state) {
19       return Column(
20         children: [
21           RoleButton(
22             icon: Icons.storefront,
23             title: 'Soy emprendedora',
24             subtitle:
25               '(Quiero vender y comprar productos)',
26             selected:
27               state.selectedRole ==
28                 UserRole.entrepreneur,
29             color: AppTheme.blue,
30             onTap: () {
31               context
32                 .read(RoleSelectionCubit>()
33                 .selectRole(
34                   UserRole.entrepreneur,
35                 );
36             },
37           ), // RoleButton
38           const SizedBox(height: 20),
39           RoleButton(
40             icon: Icons.shopping_bag,
41             title: 'Soy cliente',
42             subtitle:
43               '(Quiero comprar productos al detalle)',
44             selected:
45               state.selectedRole ==
46                 UserRole.client,
47             color: AppTheme.purple,
48             onTap: () {
49               context
50                 .read(RoleSelectionCubit>()
51                 .selectRole(
52                   UserRole.client,
53                 );
54             },
55           ), // RoleButton
56         ],
57       );
58     },
59   );
60 }

```

El soporte de esta decisión se concentra en *role_selection_cubit.dart*, mostrado en la figura 24. Desde este componente se conserva la selección activa, se valida el avance y se evita que la pantalla asuma por sí sola la lógica de continuidad del proceso.

Figura 24. Definición del *enum UserRole* y de las clases *RoleSelectionCubit* y *RoleSelectionState*

```

1 // features/auth/presentation/blocs/role_selection_cubit.dart
2 import 'package:flutter_bloc/flutter_bloc.dart';
3
4 part 'role_selection_state.dart';
5
6 enum UserRole { none, client, entrepreneur, admin }
7
8 class RoleSelectionCubit extends Cubit<RoleSelectionState> {
9   RoleSelectionCubit() : super(RoleSelectionState());
10
11   void selectRole(UserRole role) {
12     emit(state.copyWith(selectedRole: role));
13   }
14
15   void continueToRegister() {
16     if (state.selectedRole == UserRole.none) {
17       emit(state.copyWith(error: 'Por favor selecciona un rol.'));
18     } else {
19       emit(state.copyWith(continueToRegister: true));
20     }
21   }
22 }
23
24 // features/auth/presentation/blocs/role_selection_state.dart
25 part of 'role_selection_cubit.dart';
26
27 class RoleSelectionState {
28   final UserRole selectedRole;
29   final bool continueToRegister;
30   final String? error;
31
32   RoleSelectionState({
33     this.selectedRole = UserRole.none,
34     this.continueToRegister = false,
35     this.error,
36   });
37
38   RoleSelectionState copyWith({
39     UserRole? selectedRole,
40     bool? continueToRegister,
41     String? error,
42   }) {
43     return RoleSelectionState(
44       selectedRole: selectedRole ?? this.selectedRole,
45       continueToRegister: continueToRegister ?? false,
46       error: error,
47     );
48   }
49 }

```

4.3.3.2.4. Historia de Usuario 4: Registro de usuario

El anexo 4 recoge el registro de usuario como un proceso escalonado que diferencia el alta de cliente y de emprendedora desde el primer paso. Las figuras 25 y 26 muestran la captura inicial de credenciales, con una organización que reduce la carga de información y mantiene visible el avance del proceso.

Figura 25. Interfaz móvil del registro (Paso 1) para rol Emprendedora

Figura 26. Interfaz móvil del registro (Paso 1) para rol Cliente

←

Registrándote como: Cliente

Paso 1: Crear tu cuenta

Solo pediremos lo necesario para crear tu cuenta.

Correo electrónico

Contraseña

Continuar

La segunda etapa del registro se desarrolla en las figuras 27 y 28, donde se completa el perfil básico según el rol y se incorpora la aceptación de la política de privacidad. Este bloque da cierre al alta desde una secuencia progresiva y más comprensible para la usuaria.

Figura 27. Interfaz móvil del registro (Paso 2) para rol Emprendedora

←

Registrándote como: Emprendedora

Paso 2: Completar tu perfil

Completa tu perfil para disfrutar mejor la plataforma.

Nombre completo

Dirección

Nombre del negocio (opcional)

☐ Acepto la Política de Privacidad

Crear cuenta

Figura 28. Interfaz móvil del registro (Paso 2) para rol Cliente

←

Registrándote como: Cliente

Paso 2: Completar tu perfil

Completa tu perfil para disfrutar mejor la plataforma.

Nombre completo

Teléfono de contacto

☐ Acepto la Política de Privacidad

Crear cuenta

La estructura de la pantalla y la coordinación del flujo se sostienen en *register_page.dart*, *register_bloc.dart* y *register_cubit.dart*, visibles en la figura 29, 30 y 31.

En conjunto, estos componentes articulan la transición entre pasos, las validaciones y la retroalimentación del envío.

Figura 29. Código de *register_page.dart*

```
lib > modules > features > auth > presentation > pages > register_page.dart > % register_page.dart > build
16 class RegisterPage extends StatelessWidget {
17   Widget build(BuildContext context) {
18     color: AppTheme.white,
19     fontWeight: FontWeight.bold,
20     fontSize: 16,
21   }, // TextStyle
22   ), // Text
23   const SizedBox(height: 8),
24   Text(
25     cubitState.step == 1
26     ? 'Paso 1: Crear tu cuenta'
27     : 'Paso 2: Completar tu perfil',
28     style: const TextStyle(
29       color: AppTheme.white,
30       fontWeight: FontWeight.bold,
31       fontSize: 20,
32     ), // TextStyle
33   ), // Text
34   const SizedBox(height: 4),
35   Text(
36     cubitState.step == 1
37     ? 'Solo pediremos lo necesario para crear tu cuenta.'
38     : 'Completa tu perfil para disfrutar mejor la plataforma.',
39     style: const TextStyle(
40       color: Colors.white70,
41       fontSize: 14,
42     ), // TextStyle
43   ), // Text
44   const SizedBox(height: 16),
45   StepIndicator(
46     currentStep: cubitState.step,
47     totalSteps: 2,
48   ), // StepIndicator
49 },
50 // Column
```

Figura 30. Código de *register_bloc.dart*

```
lib > modules > features > auth > presentation > blocs > register_bloc.dart
1 // lib/modules/features/auth/presentation/blocs/register_bloc.dart
2
3 import 'package:flutter_bloc/flutter_bloc.dart';
4 import 'package:domain/entities/user_entity.dart';
5 import 'package:domain/usecases/register_user.dart';
6
7 abstract class RegisterEvent {}
8
9 void 6 months ago [author]
10 class SubmitRegisterEvent extends RegisterEvent {
11   final UserEntity user;
12   final String password;
13   final String role;
14
15   SubmitRegisterEvent({
16     required this.user,
17     required this.password,
18     required this.role,
19   });
20
21 abstract class RegisterBlocState {}
22
23 class RegisterInitial extends RegisterBlocState {}
24
25 class RegisterLoading extends RegisterBlocState {}
26
27 class RegisterSuccess extends RegisterBlocState {}
28
29 void 6 months ago [author]
30 class RegisterFailure extends RegisterBlocState {
31   final String error;
32   RegisterFailure(this.error);
33 }
34
35 void 6 months ago [author]
36 class RegisterBloc extends Bloc<RegisterEvent, RegisterBlocState> {
37   final RegisterUser registerUser;
38 }
```

Figura 31. Código de *register_cubit.dart*

```
lib > modules > features > auth > presentation > blocs > register_cubit.dart
1 // lib/modules/features/auth/presentation/blocs/register_cubit.dart
2
3 import 'package:flutter/material.dart';
4 import 'package:flutter_bloc/flutter_bloc.dart';
5
6 part 'register_state.dart';
7
8 void 6 months ago [author]
9 class RegisterCubit extends Cubit<RegisterCubitState> {
10   final String role;
11
12   RegisterCubit([required this.role])
13     : super(RegisterCubitState(selectedDocumentType: 'Cédula'));
14
15   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
16   final GlobalKey<FormState> _roleKey = GlobalKey<FormState>();
17
18   final TextEditingController celularController = TextEditingController();
19   final TextEditingController emailController = TextEditingController();
20   final TextEditingController phoneController = TextEditingController();
21   final TextEditingController passwordController = TextEditingController();
22   final TextEditingController nameController = TextEditingController();
23   final TextEditingController addressController = TextEditingController();
24   final TextEditingController businessNameController = TextEditingController();
25
26   bool acceptedPrivacy = false;
27
28   String selectedDocumentType = 'Cédula';
29
30   void onDocumentTypeChanged(String? type) {
31     selectedDocumentType = type ?? 'Cédula';
32     celularController.clear();
33     emailController.clear();
34     nameController.clear();
35     selectedDocumentType = selectedDocumentType,
36     isStepValid = false;
37   }
38
39   void setAcceptedPrivacy(bool? value) {
40     acceptedPrivacy = value ?? false;
41     emailController.clear();
42     isStepValid = false;
43   }
44 }
```

A nivel de dominio y datos, la figura 32 y 33 trasladan la operación hacia el caso de uso de registro y su repositorio. Esta separación permite que la creación de cuenta mantenga una frontera clara entre la interacción de la pantalla y la persistencia del usuario.

Figura 32. Código de *register_user.dart*

```
lib > modules > features > auth > domain > usecases > register_user.dart > ...
You, 5 months ago | 1 author (You)
1 import '../entities/user_entity.dart';
2 import '../repositories/auth_repository.dart';
3
4 You, 5 months ago | 1 author (You)
5 class RegisterUser {
6   final AuthRepository repository;
7
8   RegisterUser(this.repository);
9
10  Future<void> call(UserEntity user, String password, String role) {
11    return repository.register(user, password, role);
12  }
13 }
```

Figura 33. Código de *auth_repository_impl.dart*

```
lib > modules > features > auth > data > repositories > auth_repository_impl.dart > ...
You, last month | 1 author (You)
1 // modules/features/auth/data/repositories/auth_repository_impl.dart
2 import '../domain/entities/user_entity.dart';
3 import '../domain/repositories/auth_repository.dart';
4 import '../datasources/remote_auth_datasource.dart';
5 import '../models/user_model.dart';
6
7 You, last month | 1 author (You)
8 class AuthRepositoryImpl implements AuthRepository {
9   final RemoteAuthDataSource remote;
10  AuthRepositoryImpl(this.remote);
11
12  @override
13  Future<void> register(UserEntity user, String password, String role) {
14    return remote.register(UserModel.fromEntity(user), password, role);
15  }
16
17  @override
18  Future<UserEntity> login(String identifier, String password) {
19    // Delegamos toda la resolución (correo/documento) al DataSource.
20    return remote.login(identifier, password);
21  }
22
23  @override
24  Future<void> sendEmailVerification() => remote.sendEmailVerification();
25
26  @override
27  Future<void> signOut() => remote.signOut();
28
29  @override
30  Future<void> reloadUser() => remote.reloadUser();
31 }
```

4.3.3.2.5. Historia de Usuario 5: Ingreso de usuario

En el anexo 4, el ingreso de usuario se presenta como la puerta de acceso para recuperar el contexto de sesión y conducir a la usuaria al entorno que le corresponde. La figura 34 resume esta entrada mediante un formulario directo, coherente con el resto del módulo de autenticación.

Figura 34. Interfaz móvil de inicio de sesión

The image shows a mobile application interface for login. At the top, there is a back arrow and a logo with a blue circle containing a white 'U' and the text '100 MUJERES' below it. The title 'Iniciar sesión' is centered. Below the title are two input fields: 'Usuario' and 'Contraseña'. The 'Contraseña' field has an eye icon to toggle visibility. Below these fields is a link that says '¿Olvidaste tu contraseña?'. At the bottom, there is a large 'Iniciar sesión' button and a footer that says 'Tu información está protegida.'

La resolución del acceso se organiza entre *login_page.dart*, *login_cubit.dart* y *login_bloc.dart*, desarrollados en las figuras 35, 36 y 37. Este bloque concentra la construcción del formulario, la validación previa y el manejo de estados de carga, éxito o error durante el envío.

Figura 35. Código de *login_page.dart*

```
lib > modules > features > auth > presentation > pages > login_page.dart > ...
You 4 months ago | 1 author (You)
21 class LoginPage extends StatelessWidget {
22   const LoginPage({super.key});
23
24   @override
25   Widget build(BuildContext context) {
26     return MultiBlocProvider(
27       providers: [
28         BlocProvider(create: (_) => LoginCubit()),
29         BlocProvider.value(value: context.read(LoginBloc)),
30       ],
31       child: const _LoginContent(),
32     ); // MultiBlocProvider
33   }
34 }
35
36 You 4 months ago | 1 author (You)
37 class _LoginContent extends StatefulWidget {
38   const _LoginContent();
39
40   @override
41   State<_LoginContent> createState() => _LoginContentState();
42 }
43
44 You 5 hours ago | 1 author (You)
45 class _LoginContentState extends State<_LoginContent> {
46   final _userController = TextEditingController();
47   final _passwordController = TextEditingController();
48   final _userFocus = FocusNode();
49   final _passwordFocus = FocusNode();
50 }
```

Figura 36. Código de *login_cubit.dart*

```
lib > modules > features > auth > presentation > blocs > login_cubit.dart > ...
2 import 'package:flutter/material.dart';
3 import 'package:flutter_bloc/flutter_bloc.dart';
4
5 You 2 months ago | 1 author (You)
6 class LoginCubit extends Cubit<bool> {
7   LoginCubit() : super(false);
8
9   final formKey = GlobalKey<FormState>();
10
11   void onFieldChanged(
12     TextEditingController user,
13     TextEditingController password,
14   ) {
15     emit(_validate(user, password));
16   }
17
18   bool _validate(TextEditingController user, TextEditingController password) {
19     final userValid = user.text.trim().isNotEmpty; // feasible: correo o cédula
20     final passwordValid = password.text.trim().length >= 4;
21     return userValid && passwordValid;
22   }
23
24   // Regexp from internet online
25   String validateUser(String value) {
26     if (value == null || value.trim().isEmpty) {
27       return 'usuario obligatorio';
28     }
29     final emailRegex = RegExp(r'^[a-zA-Z0-9]{1,30}@[a-zA-Z0-9]{1,30}\.([a-zA-Z]{2,4})$');
30     final isValid = emailRegex.hasMatch(value.trim());
31     final isValid = int.tryParse(value.trim()) != null;
32
33     if (!isValid && !isValid) {
34       return 'correo o cédula no válida';
35     }
36     return null;
37   }
38
39   String validatePassword(String value) {
40     if (value == null || value.trim().isEmpty) {
41       return 'contraseña obligatoria';
42     }
43     if (value.trim().length < 4) {
44       return 'mínimo 4 caracteres';
45     }
46   }
47 }
```

Figura 37. Código de *login_bloc.dart*

```
lib > modules > features > auth > presentation > blocs > login_bloc.dart > ...
You 2 months ago | 1 author (You)
1 // lib/modules/features/auth/presentation/blocs/login_bloc.dart
2 import 'package:flutter_bloc/flutter_bloc.dart';
3 import 'package:domain/usecases/login_user.dart';
4 import 'package:domain/entities/user_entity.dart';
5
6 abstract class LoginEvent {}
7
8 You 2 months ago | 1 author (You)
9 class LoginSubmittedEvent extends LoginEvent {
10   /// Puede ser correo o documento (cédula/RUC/pasaporte).
11   final String identifier;
12   final String password;
13
14   LoginSubmittedEvent({
15     required this.identifier,
16     required this.password,
17   });
18 }
19
20 abstract class LoginBlocState {}
21
22 class LoginInitial extends LoginBlocState {}
23 class LoginLoading extends LoginBlocState {}
24 class LoginSuccess extends LoginBlocState {
25   final UserEntity user;
26   LoginSuccess(this.user);
27 }
28
29 You 6 months ago | 1 author (You)
30 class LoginFailure extends LoginBlocState {
31   final String error;
32   LoginFailure(this.error);
33 }
34
35 You 2 months ago | 1 author (You)
36 class LoginBloc extends Bloc<LoginEvent, LoginBlocState> {
37   final _loginUser = loginUser;
38 }
```

La figura 38 y 39 completan la historia desde el caso de uso *LoginUser* y la fuente remota de autenticación. De esta manera, la verificación de credenciales y la recuperación del perfil quedan desacopladas de la vista y se integran al flujo con mayor consistencia.

Figura 38. Código de *login_user.dart*

```
lib > modules > features > auth > domain > usecases > login_user.dart > ...
You, 2 months ago | 1 author (You)
1 // lib/modules/features/auth/domain/usecases/login_user.dart
2 import '../entities/user_entity.dart';
3 import '../repositories/auth_repository.dart';
4
5 You, 2 months ago | 1 author (You)
6 class LoginUser {
7   final AuthRepository repository;
8
9   LoginUser(this.repository);
10
11   /// Acepta correo o documento como `identifier`.
12   Future<UserEntity> call(String identifier, String password) {
13     return repository.login(identifier, password);
14   }
15 }
```

Figura 39. Código de *remote_auth_datasource.dart*

```
lib > modules > features > auth > data > datasources > remote_auth_datasource.dart > ...
You, 2 months ago | 1 author (You)
9 class RemoteAuthDataSource {
10   final FirebaseAuth _auth;
11   final Firestore _db;
12
13   static const _usersCol = 'users';
14   static const _cedulasCol = 'cedulas';
15
16   RemoteAuthDataSource(this._auth, this._db);
17
18   // --- Helpers ---
19
20   bool _looklikeEmail(String input) =>
21     RegExp(r'^[a-zA-Z0-9]{1,30}@[a-zA-Z0-9]{2,5}\.com$').hasMatch(input);
22
23   Future<String> _resolveEmailFromIdentifier(String identifier) async {
24     final id = identifier.trim();
25     if (_looklikeEmail(id)) return id.toLowerCase();
26
27     // Si es cédula (solo dígitos), intenta resolver desde Firestore
28     final isCedula = int.tryParse(id) != null;
29     if (isCedula) {
30       try {
31         final cedulaDoc = await _db.collection(_cedulasCol).doc(id).get();
32         if (cedulaDoc.exists) {
33           final data = cedulaDoc.data();
34           final email = (data['email'] as String?).trim();
35           if (email != null && email.isNotEmpty) return email.toLowerCase();
36
37           final uid = data['uid'] as String?;
38           if (uid != null) {
39             final userDoc = await _db.collection(_usersCol).doc(uid).get();
40             if (userDoc.exists) {
41               final userEmail = (userDoc.data()?['email'] as String?).trim();
42               if (userEmail != null && userEmail.isNotEmpty) return userEmail.toLowerCase();
43             }
44           }
45         }
46       } catch (_) {}
47       // Ignorado: si las reglas no permiten leer, seguimos con el fallback
48     }
49   }
```

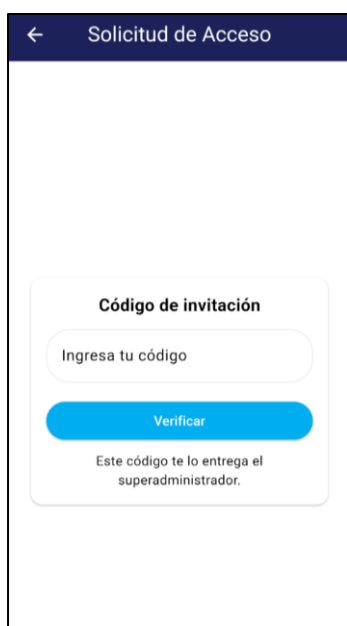
4.3.3.2.6. Historia de Usuario 6: Solicitud de Acceso para Administradoras

Tal como se desarrolla en el anexo 4, la solicitud de acceso para administradoras se diferencia del registro general desde la elección inicial y la verificación del código de invitación. Las figuras 40 y 41 ubican este acceso especial y muestran el inicio del recorrido previo a la creación de la cuenta administrativa.

Figura 40. Interfaz móvil de acceso administrativo desde selección de rol



Figura 41. Interfaz móvil “Solicitud de Acceso”: ingreso de código de invitación y acción “Verificar”



Una vez validada la invitación, el flujo continúa con el asistente de alta mostrado en la figura 42 y 43. Estas vistas ordenan la captura de credenciales y perfil en pasos sucesivos, lo que hace más comprensible una operación que requiere mayor control.

Figura 42. Interfaz móvil del asistente administrativo: “Crear cuenta administrativa – Paso 1: Credenciales”

The screenshot shows a mobile app interface for creating an administrative account. The title bar at the top is dark blue with a back arrow and the text "Solicitud de Acceso". Below this, a pink header contains the title "Crear cuenta administrativa". Under the header, a progress indicator shows "Paso 1: Credenciales" as the active step. The main form area is white and contains two input fields: "Correo real (obligatorio)" and "Contraseña". Below the email field, there is a red error message "Correo no válido". Below the password field, there is a red error message "Mínimo 6 caracteres". A note below the password field states: "Usaremos tu correo para verificar la cuenta. Luego podrás habilitar 2FA." At the bottom of the form, there are two buttons: "Cancelar" and "Siguiente".

Figura 43. Interfaz móvil del asistente administrativo: “Crear cuenta administrativa – Paso 2: Perfil”

The screenshot shows the same mobile app interface, but for Step 2: Perfil. The title bar and header are the same. The progress indicator now shows "Paso 2: Perfil" as the active step. The main form area is white and contains four input fields: "Nombre completo", "Teléfono (recomendado)", "Cédula" (with a dropdown menu showing "Documento de Iden..."), and "Dirección (opcional)". At the bottom of the form, there are two buttons: "Atrás" and "Finalizar".

La conmutación entre la verificación y el asistente, así como el control del proceso, se sostienen en *access_request_page.dart* y *access_request_cubit.dart*, visibles en las figuras 44 y 45. Desde aquí se coordinan los cambios de estado y la continuidad del flujo administrativo.

Figura 44. Código de `access_request_page.dart`

```

12 class AccessRequestPage extends StatelessWidget {
13   widget build(BuildContext context) {
14     return BlocConsumer<AccessRequestCubit, AccessRequestStates>(
15       listener: (p, e) => p.error != e.error && e.error != null,
16       listener: (context, state) {
17         if (state.error != null && !state.loading) {
18           ScaffoldMessenger.of(context).showSnackBar(
19             SnackBar(content: Text(state.error!.replaceFirst('Exception: ', ''))),
20           );
21         }
22       },
23     );
24   }
25   builder: (context, state) {
26     widget body;
27   }
28   if (state.current != null) {
29     switch (state.current!.status) {
30       case AccessRequestStatus.pending:
31         body = const AccessRequestCard(
32           icon: Icons.hourglass_bottom_rounded,
33           title: 'Solicitud enviada',
34           subtitle: 'Estamos revisando tu solicitud. Te avisaremos cuando haya una decisión.',
35         );
36       case AccessRequestStatus.approved:
37         body = const AccessRequestCard(
38           icon: Icons.check_circle_rounded,
39           title: 'Solicitud aprobada',
40           subtitle: '¡Felicitaciones! Ya puedes acceder al sistema.',
41         );
42       case AccessRequestStatus.rejected:
43         body = const AccessRequestCard(
44           icon: Icons.cancel_rounded,
45           title: 'Solicitud rechazada',
46           subtitle: 'Puedes contactar a soporte si consideras que hay un error.',
47         );
48     }
49   } else {
50     body = (state.inviteVerified == true)
51       ? const AdminCard() // aquí usamos el wizard
52       : const InviteCard();
53   }
54 }

```

Figura 45. Código de `access_request_cubit.dart`

```

12 class AccessRequestCubit extends Cubit<AccessRequestStates> {
13   Future<AccessRequestStates> createAccount() {
14     // Aquí se crea la cuenta directamente y se consume el código
15     // de verificación de la cuenta
16     // Se llama al método createAccount() de la clase AccessRequestStates
17     // Se llama al método createAccount() de la clase AccessRequestStates
18     // Se llama al método createAccount() de la clase AccessRequestStates
19     // Se llama al método createAccount() de la clase AccessRequestStates
20     // Se llama al método createAccount() de la clase AccessRequestStates
21     // Se llama al método createAccount() de la class AccessRequestStates
22     // Se llama al método createAccount() de la class AccessRequestStates
23     // Se llama al método createAccount() de la class AccessRequestStates
24     // Se llama al método createAccount() de la class AccessRequestStates
25     // Se llama al método createAccount() de la class AccessRequestStates
26     // Se llama al método createAccount() de la class AccessRequestStates
27     // Se llama al método createAccount() de la class AccessRequestStates
28     // Se llama al método createAccount() de la class AccessRequestStates
29     // Se llama al método createAccount() de la class AccessRequestStates
30     // Se llama al método createAccount() de la class AccessRequestStates
31     // Se llama al método createAccount() de la class AccessRequestStates
32     // Se llama al método createAccount() de la class AccessRequestStates
33     // Se llama al método createAccount() de la class AccessRequestStates
34     // Se llama al método createAccount() de la class AccessRequestStates
35     // Se llama al método createAccount() de la class AccessRequestStates
36     // Se llama al método createAccount() de la class AccessRequestStates
37     // Se llama al método createAccount() de la class AccessRequestStates
38     // Se llama al método createAccount() de la class AccessRequestStates
39     // Se llama al método createAccount() de la class AccessRequestStates
40     // Se llama al método createAccount() de la class AccessRequestStates
41     // Se llama al método createAccount() de la class AccessRequestStates
42     // Se llama al método createAccount() de la class AccessRequestStates
43     // Se llama al método createAccount() de la class AccessRequestStates
44     // Se llama al método createAccount() de la class AccessRequestStates
45     // Se llama al método createAccount() de la class AccessRequestStates
46     // Se llama al método createAccount() de la class AccessRequestStates
47     // Se llama al método createAccount() de la class AccessRequestStates
48     // Se llama al método createAccount() de la class AccessRequestStates
49     // Se llama al método createAccount() de la class AccessRequestStates
50     // Se llama al método createAccount() de la class AccessRequestStates
51     // Se llama al método createAccount() de la class AccessRequestStates
52     // Se llama al método createAccount() de la class AccessRequestStates
53     // Se llama al método createAccount() de la class AccessRequestStates
54     // Se llama al método createAccount() de la class AccessRequestStates
55     // Se llama al método createAccount() de la class AccessRequestStates
56     // Se llama al método createAccount() de la class AccessRequestStates
57     // Se llama al método createAccount() de la class AccessRequestStates
58     // Se llama al método createAccount() de la class AccessRequestStates
59     // Se llama al método createAccount() de la class AccessRequestStates
60     // Se llama al método createAccount() de la class AccessRequestStates
61     // Se llama al método createAccount() de la class AccessRequestStates
62     // Se llama al método createAccount() de la class AccessRequestStates
63     // Se llama al método createAccount() de la class AccessRequestStates
64     // Se llama al método createAccount() de la class AccessRequestStates
65     // Se llama al método createAccount() de la class AccessRequestStates
66     // Se llama al método createAccount() de la class AccessRequestStates
67     // Se llama al método createAccount() de la class AccessRequestStates
68     // Se llama al método createAccount() de la class AccessRequestStates
69     // Se llama al método createAccount() de la class AccessRequestStates
70     // Se llama al método createAccount() de la class AccessRequestStates
71     // Se llama al método createAccount() de la class AccessRequestStates
72     // Se llama al método createAccount() de la class AccessRequestStates
73     // Se llama al método createAccount() de la class AccessRequestStates
74     // Se llama al método createAccount() de la class AccessRequestStates
75     // Se llama al método createAccount() de la class AccessRequestStates
76     // Se llama al método createAccount() de la class AccessRequestStates
77     // Se llama al método createAccount() de la class AccessRequestStates
78     // Se llama al método createAccount() de la class AccessRequestStates
79     // Se llama al método createAccount() de la class AccessRequestStates
80     // Se llama al método createAccount() de la class AccessRequestStates
81     // Se llama al método createAccount() de la class AccessRequestStates
82     // Se llama al método createAccount() de la class AccessRequestStates
83     // Se llama al método createAccount() de la class AccessRequestStates
84     // Se llama al método createAccount() de la class AccessRequestStates
85     // Se llama al método createAccount() de la class AccessRequestStates
86     // Se llama al método createAccount() de la class AccessRequestStates
87     // Se llama al método createAccount() de la class AccessRequestStates
88     // Se llama al método createAccount() de la class AccessRequestStates
89     // Se llama al método createAccount() de la class AccessRequestStates
90     // Se llama al método createAccount() de la class AccessRequestStates
91     // Se llama al método createAccount() de la class AccessRequestStates
92     // Se llama al método createAccount() de la class AccessRequestStates
93     // Se llama al método createAccount() de la class AccessRequestStates
94     // Se llama al método createAccount() de la class AccessRequestStates
95     // Se llama al método createAccount() de la class AccessRequestStates
96     // Se llama al método createAccount() de la class AccessRequestStates
97     // Se llama al método createAccount() de la class AccessRequestStates
98     // Se llama al método createAccount() de la class AccessRequestStates
99     // Se llama al método createAccount() de la class AccessRequestStates
100    // Se llama al método createAccount() de la class AccessRequestStates

```

La validación del código y su consumo posterior se apoyan en

`verify_admin_invite.dart` e `invite_codes_remote_datasource.dart`, presentados en las figuras 46 y 47. Este cierre técnico refuerza que la habilitación administrativa no dependa solo de la interfaz, sino de reglas de control y persistencia.

Figura 46. Código de `verify_admin_invite.dart`

```

1 import '../entities/invite_code.dart';
2 import '../repositories/invite_codes_repository.dart';
3
4 class VerifyAdminInvite {
5   final InviteCodesRepository repo;
6   VerifyAdminInvite(this.repo);
7
8   Future<InviteCode> call(String raw) async {
9     final code = raw.trim();
10    if (code.length < 8) {
11      throw Exception('Código inválido');
12    }
13    final invite = await repo.verify(code);
14    if (!invite.isActive) throw Exception('Código ya usado/inactivo');
15    if (invite.expiresAt != null && invite.expiresAt!.isBefore(DateTime.now())) {
16      throw Exception('Código expirado');
17    }
18    return invite;
19  }
20 }

```

Figura 47. Código de *invite_codes_remote_datasource.dart*

```

1 import 'package:cloud_firestore/cloud_firestore.dart';
2 import '../models/invite_code_model.dart';
3
4 class InviteCodesRemoteDataSource {
5   final FirebaseFirestore firestore;
6   InviteCodesRemoteDataSource(this.firestore);
7
8   Future<InviteCodeModel> verify(String code) async {
9     final ref = firestore.collection('admin_invites').doc(code);
10    final snap = await ref.get();
11    if (!snap.exists) throw Exception('Código inválido');
12    return InviteCodeModel.fromDoc(code, snap);
13  }
14
15   Future<void> consume(String code, String uid) async {
16     final ref = firestore.collection('admin_invites').doc(code);
17     await firestore.runTransaction(tx) async {
18       final snap = await tx.get(ref);
19       if (!snap.exists) throw Exception('Código inválido');
20
21       final data = snap.data();
22       final isActive = (data['isActive'] ?? data['active'] ?? true) == true && data['usedAt'] == null;
23       if (!isActive) throw Exception('Código ya usado/inactivo');
24
25       tx.update(ref, {
26         'isActive': false,
27         'usedBy': uid,
28         'usedAt': FieldValue.serverTimestamp(),
29         'usedCount': (data['usedCount'] ?? 0) + 1,
30       });
31     });
32  }
33 }

```

4.3.3.2.7. Historia de Usuario 7: Dashboard Principal y Navegación por Rol

El anexo 4 muestra que el *dashboard* principal no responde a una única vista, sino a un acceso diferenciado según la condición de la usuaria. Las figuras 48, 49 y 50 reúnen las variantes más representativas del inicio para cliente o emprendedora, visitante y administración, con una navegación adaptada a cada contexto.

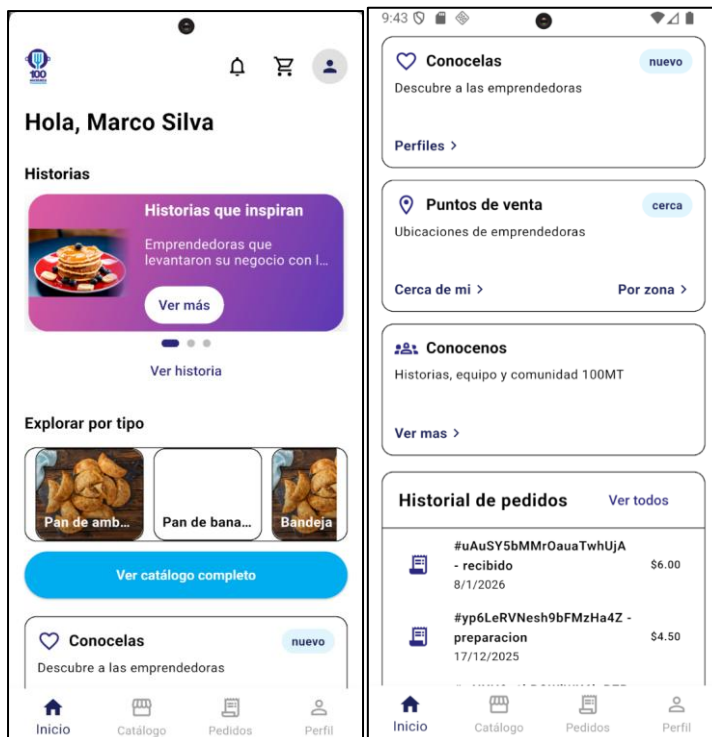
Figura 48. Interfaz móvil del *dashboard* para cliente o emprendedora

Figura 49. Interfaz móvil de bienvenida para visitante



Figura 50. Interfaz móvil del *dashboard* administrativo



La base de esta organización se distribuye entre el enrutamiento inicial y la composición de los contenedores principales, visibles en las figuras 51, 52, 53 y 54. En este conjunto se decide la ruta de entrada, se estructura el home general y se mantiene separado el espacio administrativo.

Figura 51. Código de *app_routes.dart*

```

lib > modules > core > config > app_routes.dart > AppRoutes > generateRoute
24 class AppRoutes {
25
26   static String initialRouteFor(User? user) {
27     if (user == null) return welcome;
28     if (_shouldForceEmailVerification(user)) return emailVerification;
29     final storedRole = _sessionStorage?.getRole(user.uid);
30     if (storedRole != null && isAdminRole(storedRole)) return admin;
31     return home;
32   }
33
34   static Route<dynamic> generateRoute(RouteSettings settings) {
35     final name = settings.name ?? '';
36     final user = FirebaseAuth.instance.currentUser;
37     final shouldForceEmailVerification =
38       _shouldForceEmailVerification(user) && !name.startsWith('/auth');
39
40     if (name == StartupRoutes.startup) {
41       return StartupRoutes.onGenerateRoute(
42         settings,
43         initialRouteResolver: initialRouteFor,
44       );
45     }
46
47     if (shouldForceEmailVerification) {
48       return AuthRoutes.onGenerateRoute(
49         const RouteSettings(name: AuthRoutes.emailVerification),
50       );
51     }
52
53     if (name.startsWith('/auth')) {
54       return AuthRoutes.onGenerateRoute(settings);
55     }
56     if (name.startsWith(OrgProfileRoutes.about)) {
57       return OrgProfileRoutes.onGenerateRoute(settings);
58     }
59     if (name.startsWith('/home')) {
60       return HomeRoutes.onGenerateRoute(settings);
61     }
62   }
63 }

```

Figura 52. Código de *home_page.dart*

```

lib > modules > features > home > presentation > pages > home_page.dart > ...
22 class HomePage extends StatefulWidget {
23   const HomePage({super.key});
24   @override
25   State<HomePage> createState() => _HomePageState();
26 }
27
28 You 4 days ago | 1 author (You)
29 class _HomePageState extends State<HomePage> {
30   int _tabIndex = 0;
31   bool _showVerificationReminder = false;
32   String? _reminderContact;
33
34   void _goToTab(int index) {
35     if (index == _tabIndex) return;
36     setState(() => _tabIndex = index);
37   }
38
39   @override
40   void initState() {
41     super.initState();
42     WidgetsBinding.instance.addPostFrameCallback((_) => _evaluateVerificationReminder());
43   }
44
45   Future<void> _evaluateVerificationReminder() async {
46     final user = FirebaseAuth.instance.currentUser;
47     if (user == null) return;
48     if (user.emailVerified) return;
49     if (!_requiresEmailVerification(user)) return;
50     final storage = RepositoryProvider.of<VerificationGraceStorage>(context, listen: false);
51     final uid = user.uid;
52     if (!storage.isValidDeadline(uid)) return;
53     if (!storage.shouldShowReminder(uid)) return;
54     await storage.markReminderShown(uid);
55     if (!mounted) return;
56     setState(() {
57       _showVerificationReminder = true;
58       _reminderContact = user.email ?? 'tu cuenta';
59     });
60   }
61 }

```

Figura 53. Código de *home_tab.dart*

```

lib > modules > features > home > presentation > pages > tabs > home_tab.dart > ...
24 class HomeTab extends StatelessWidget {
25   final VoidCallback onOpenCatalog;
26   final VoidCallback onOpenOrders;
27   const HomeTab({
28     super.key,
29     required this.onOpenCatalog,
30     required this.onOpenOrders,
31   });
32
33   @override
34   Widget build(BuildContext context) {
35     final authUser = FirebaseAuth.instance.currentUser;
36     final profileState = context.watch<ProfileCubit>().state;
37     final sessionStorage = context.read<SessionStorage>();
38     final userKind = _resolveUserKind();
39     authUser: authUser,
40     profileState: profileState,
41     sessionStorage: sessionStorage,
42   );
43   final infoCards = _buildInfoCards(context, userKind, authUser);
44
45   return Scaffold(
46     appBar: AppBar(
47       color: AppTheme.white,
48       title: CustomScrollView(
49         slivers: [
50           SliverToBoxAdapter(
51             child: Column(
52               crossAxisAlignment: CrossAxisAlignment.start,
53               children: [
54                 HomeTopBar(
55                   showNotifications: userKind != HomeUserKind.guest,
56                   onNotifications:
57                     userKind == HomeUserKind.guest
58                       ? null
59                       : () => safePushNamed(
60                         context,
61                         ProfileRoutes.notifications,
62                         fallbackTitle: 'Notificaciones',
63                         fallbackDescription:
64                           'Inicia sesión para ver tus notificaciones.',
65                       ),
66                 ), // HomeTopBar
67                 const GreetingHeader(),
68               ], // Column
69             ), // SliverToBoxAdapter

```

Figura 54. Código de `admin_shell.dart`

```

15 class AdminShell extends StatelessWidget {
16   Widget build(BuildContext context) {
17   }
18 }
19
20 return TabBackGuard(
21   currentIndex: index,
22   onRequestRoot: () => context.read<AdminNavCubit>().setTab(0),
23   child: Scaffold(
24     appBar: AdminTopBar(
25       title: titles[index],
26       useTopSafeArea: false,
27       onNotifications:
28         () => safePushNamed(
29           context,
30           ProfileRoutes.notifications,
31           fallbackTitle: 'Notificaciones',
32           fallbackDescription: 'Estamos preparando las notificaciones.',
33         ),
34       showBadge: index == 0,
35       photoUrl: photoUrl,
36       displayName: displayName,
37       email: email,
38     ), // AdminTopBar
39     body: IndexedStack(index: index, children: pages),
40     bottomNavigationBar: NavigationBar(
41       // You, 2 months ago • Text: Implement Feature Flags for AppCheck badge.
42       selectedIndex: index,
43       onDestinationSelected: context.read<AdminNavCubit>().setTab,
44       destinations: const [
45         NavigationDestination(
46           icon: Icon(Icons.dashboard_outlined),
47           label: 'Panel',
48         ), // NavigationDestination
49         NavigationDestination(
50           icon: Icon(Icons.receipt_long_outlined),
51           label: 'Pedidos',
52         ), // NavigationDestination
53         NavigationDestination(
54           icon: Icon(Icons.category_outlined),
55           label: 'Catálogo',
56         ),
57       ],
58     ),
59   ),
60 );

```

4.3.3.2.8. Historia de Usuario 8: Visualización del Perfil de Usuario

Dentro del anexo 4, la visualización del perfil reúne la consulta de datos personales y las variantes de acceso disponibles según el tipo de sesión. Las figuras 55, 56 y 57 presentan tanto la vista general del perfil como la edición accesible desde ella y la restricción aplicada cuando se ingresa como invitado.

Figura 55. Interfaz móvil del perfil de usuario

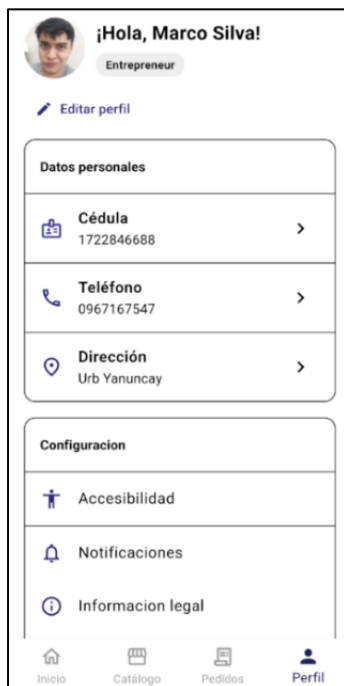


Figura 56. Interfaz móvil de Editar perfil

← Editar perfil

Foto de perfil

 Consejo: usa una imagen nítida, rostro centrado.

 Cambiar

Datos básicos

Nombre
Marco Silva

Dirección
Urb Yanuncay

Contacto

Teléfono
(+593) 0967167547

Cancelar Guardar cambios

Figura 57. Interfaz móvil del perfil en modo invitado



El soporte técnico del módulo se concentra en *profile_overview_cubit.dart*, *profile_overview_tab.dart* y *watch_user_stream.dart*, mostrados en las figuras 58, 59 y 60. Este bloque mantiene sincronizada la información visible del perfil y ordena la consulta continua de los datos de cuenta.

Figura 58. Código de *profile_overview_cubit.dart*

```

lib > modules > features > profile > presentation > blocs > profile_overview_cubit.dart > ProfileState
8 sealed class ProfileState {}
9 class ProfileLoading extends ProfileState {}
10 class ProfileGuest extends ProfileState {} // sin uid/session
11 class ProfileEmpty extends ProfileState {} // sin doc
12 You 4 months ago | 1 author (You)
13 class ProfileLoaded extends ProfileState {
14   final UserProfile profile;
15   final int completed;
16   final int total;
17   final List<String> missing;
18   ProfileLoaded(this.profile) {
19     : completed = profile.completedCount(),
20       total = profile.totalCount(),
21       missing = profile.missingFields();
22 }
23 You 4 months ago | 1 author (You)
24 class ProfileError extends ProfileState {
25   final String message;
26   ProfileError(this.message);
27 }
28 You 2 months ago | 1 author (You)
29 class ProfileCubit extends Cubit<ProfileState> {
30   final WatchUserProfile _watch;
31   final SessionStorage? _sessionStorage;
32   StreamSubscription? _sub;
33   String? _currentuid;
34
35   ProfileCubit(this._watch, {SessionStorage? sessionStorage})
36     : _sessionStorage = sessionStorage,
37       super(ProfileLoading());
38
39   /// uid=null => invitado
40   void watch(String? uid) {
41     _sub?.cancel();
42     if (uid == null || uid.isEmpty) {
43       emit(ProfileGuest());
44     } else if (_currentuid != uid) {
45       emit(ProfileLoading());
46     }
47 }

```

Figura 59. Código de *profile_overview_tab.dart*

```

lib > modules > features > profile > presentation > pages > tabs > profile_overview_tab.dart > ProfileTab > build
17
18 You 2 weeks ago | 1 author (You)
19 class ProfileTab extends StatelessWidget {
20   const ProfileTab({super.key});
21 }
22 @override
23 Widget build(BuildContext context) {
24   return Scaffold(
25     child: ColoredBox(
26       color: Theme.of(context).scaffoldBackgroundColor,
27     ),
28     child: BlocBuilder<ProfileCubit, ProfileState>(
29       builder: (context, state) {
30         if (state is ProfileLoading) return const _Loading();
31         if (state is ProfileGuest) return const _Guest();
32         if (state is ProfileError) return _Error(message: state.message);
33         if (state is ProfileEmpty) return const _Empty();
34
35         final s = state as ProfileLoaded;
36         return CustomScrollView(
37           slivers: [
38             SliverToBoxAdapter(
39               child: Column(
40                 crossAxisAlignment: CrossAxisAlignment.start,
41                 children: [
42                   ProfileHeader(
43                     displayName: (s.profile.name?.isEmpty == true)
44                       ? s.profile.name!
45                       : 'Usuario',
46                     role: s.profile.role,
47                     photoUrl: s.profile.photoUrl,
48                   ), // ProfileHeader
49
50                   // Botón Editar
51                   Padding(
52                     padding: const EdgeInsets.symmetric(horizontal: 16),
53                     child: TextButton.icon(
54                       onPressed: () =>
55                         Navigator.pushNamed(context, ProfileRoutes.edit),
56                       icon: const Icon(Icons.edit),
57                       label: const Text('Editar perfil'),
58                     ), // TextButton.icon
59                   // // Padding
60                 ],
61               ), // SliverToBoxAdapter
62             ),
63           ],
64         );
65       },
66     ),
67   );
68 }

```

Figura 60. Código de *watch_user_stream.dart*

```

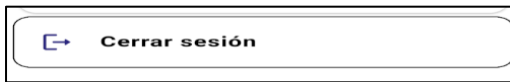
lib > modules > features > profile > domain > usecases > watch_user_stream.dart > ...
1 import '../entities/user_profile.dart';
2 import '../repositories/profile_read_repository.dart';
3
4 You 4 months ago | 1 author (You)
5 class WatchUserProfile {
6   final ProfileRepository repository;
7   WatchUserProfile(this.repository);
8
9   Stream<UserProfile> call(String uid) => repository.watchProfile(uid);
10 }

```

4.3.3.2.9. Historia de Usuario 9: Cerrar Sesión del Usuario

El cierre de sesión, documentado en el anexo 4, se articula como una acción deliberada y confirmada antes de abandonar el entorno autenticado. Las figuras 61 y 62 muestran tanto el acceso a esta opción como el diálogo que evita salidas accidentales y refuerza la decisión de retorno al espacio público.

Figura 61. Interfaz móvil de la opción cerrar sesión

Figura 62. Interfaz móvil: cuadro de diálogo de confirmación de cierre de sesión (confirmar/cancelar) invocado desde *LogoutTile*

La operación se sostiene en el componente visual de cierre y en la utilidad encargada de limpiar la sesión, según se observa en las figuras 63 y 64. Con ello, la historia conserva un recorrido breve en interfaz, pero respaldado por una ejecución ordenada del *signOut* y la redirección posterior.

Figura 63. Código de *profile_logout_tile.dart*

```
lib > modules > features > profile > presentation > widgets > profile_logout_tile.dart > ...
7 class LogoutTile extends StatelessWidget {
8   const LogoutTile({super.key});
9
10  @override
11  Widget build(BuildContext context) {
12    final scheme = Theme.of(context).colorScheme;
13    final idquest = FirebaseAuth.instance.currentUser?.isAnonymous ?? false;
14    final titleText = idquest ? 'Volver al inicio' : 'Cerrar sesión';
15    final contentText = idquest
16      ? '¿Quieres volver a la pantalla de bienvenida?'
17      : '¿Deseas cerrar la sesión?';
18
19    return Padding(
20      padding: const EdgeInsets.fromLTRB(16, 0, 16, 16),
21      child: Container(
22        decoration: BoxDecoration(
23          color: scheme.surface,
24          borderRadius: BorderRadius.circular(12),
25          border: Border.all(color: scheme.outlineVariant),
26        ), // BoxDecoration
27        child: ListTile(
28          leading: Icon(Icons.logout_rounded, color: scheme.primary),
29          title: Text(
30            titleText,
31            style: const TextStyle(fontWeight: FontWeight.w700),
32          ), // Text
33          onTap: () async {
34            final confirm = await showDialog<bool>({
35              context: context,
36              builder: (dialogContext) => AlertDialog(
37                title: Text(titleText),
38                content: Text(contentText),
39                actions: [
40                  TextButton(
41                    onPressed: () => Navigator.of(dialogContext).pop(false),
42                    child: const Text('Cancelar'),
43                  ), // TextButton
44                  FilledButton(
45                    onPressed: () => Navigator.of(dialogContext).pop(true),
46                    child: Text(titleText),
47                  ), // FilledButton
48                ],
49              ),
50            );
51            if (confirm == true) {
52              // Cerrar sesión
53            }
54          },
55        ),
56      ),
57    );
58  }
59 }
```

Figura 64. Código de *logout_helper.dart*

```
lib > modules > features > profile > presentation > utils > logout_helper.dart > ...
1 import 'package:firebase_auth/firebase_auth.dart';
2 import 'package:flutter/widgets.dart';
3 import 'package:flutter_bloc/flutter_bloc.dart';
4
5 import '.....core/config/app_routes.dart';
6 import '.....core/services/storage/session_storage.dart';
7
8 /// Cierra la sesión actual y limpia el rol guardado.
9 /// Si [navigate] es true, navega a la pantalla de bienvenida.
10 FutureCvoid performLogout(
11   BuildContext context, {
12   bool navigate = true,
13 }) async {
14   final navigator = Navigator.of(context);
15   final storage = context.read<SessionStorage>();
16
17   final uid = FirebaseAuth.instance.currentUser?.uid;
18   await FirebaseAuth.instance.signOut();
19   if (uid != null) {
20     await storage.clearRole(uid);
21   }
22   if (navigate && navigator.mounted) {
23     navigator.pushNamedAndRemoveUntil(
24       AppRoutes.welcome,
25       (_) => false,
26     );
27   }
28 }
```

4.3.3.2.10. Historia de Usuario 10: Recuperación de Contraseña

Tal como se recoge en el anexo 4, la recuperación de contraseña inicia desde el acceso al enlace correspondiente y ofrece más de una vía para localizar la cuenta. Las figuras 65, 66 y 67 sitúan esta entrada al proceso mediante la búsqueda por documento o por correo, lo que amplía las posibilidades de recuperación.

Figura 65. Interfaz móvil de acceso a recuperación de contraseña

The screenshot shows a mobile app interface with a purple gradient background. At the top left is a back arrow. In the center is the '100 MUJERES' logo. Below the logo is the title 'Iniciar sesión'. There are two input fields: 'Usuario' and 'Contraseña' (with an eye icon for toggling visibility). Below the password field is a link that says '¿Olvidaste tu contraseña?'. At the bottom is a large 'Iniciar sesión' button and a small text line 'Tu información está protegida.'

Figura 66. Interfaz móvil de Recuperar acceso en modo Documento

The screenshot shows the 'Recuperar acceso' screen. At the top left is a back arrow. The title is 'Recuperar acceso' with the subtitle 'Encuentra tu cuenta y elige cómo recuperarla.' Below this are two buttons: 'Documento' (selected with a checkmark) and 'Correo'. Under the heading 'Tipo de documento', there are three buttons: 'Cédula' (selected with a checkmark), 'RUC', and 'Pasaporte'. Below these is a text input field labeled 'Cédula'. At the bottom is a 'Buscar cuenta' button. A small text note at the very bottom states: 'Usa el tipo correcto: Cédula (10 dígitos), RUC (13 dígitos) o Pasaporte (A-Z/0-9, 6-12).'

Figura 67. Interfaz móvil de Recuperar acceso en modo Correo



La continuidad del flujo se observa desde la figura 68 hasta la figura 72, donde se desarrollan la selección del canal, el envío del enlace por correo, la verificación telefónica y el cambio final de contraseña. Este conjunto da sentido secuencial a una historia que requiere varias etapas antes de restablecer el acceso.

Figura 68. Interfaz móvil de Recuperar acceso

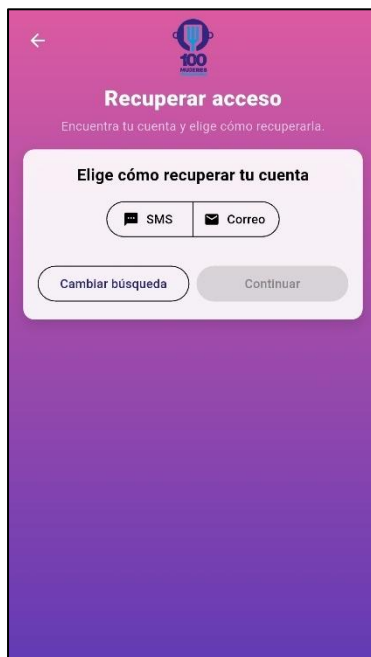


Figura 69. Interfaz móvil de Recuperar acceso

← 

Recuperar acceso

Encuentra tu cuenta y elige como recuperarla.

Revisa tu correo

Te enviamos un enlace a m***@msn.com

Va confirmé

Cambiar método

Figura 70. Interfaz móvil de Recuperar acceso

← 

Recuperar acceso

Encuentra tu cuenta y elige cómo recuperarla.

Confirma tu teléfono

Ingresa el número que termina como ***7547.

Teléfono móvil

09xxxxxxxx, +5939xxxxxxxx, 5939xxxxxxxx

Enviar SMS

Cambiar método

Figura 71. Interfaz móvil de Recuperar acceso

← 

Recuperar acceso

Encuentra tu cuenta y elige cómo recuperarla.

Verificación por SMS

Enviamos un código a ***7547.

1 [] [] [] [] [] []

Confirmar código

Reenviar código

Cambiar método

Figura 72. Interfaz móvil de Recuperar acceso



La orquestación visible del proceso se concentra en *forgot_password_page.dart* y *password_reset_cubit.dart*, presentados en las figuras 73 y 74. Desde estos componentes se controlan las transiciones entre etapas y la respuesta inmediata que la interfaz ofrece en cada caso.

Figura 73. Código de *forgot_password_page.dart*

```

10 // module > features > auth > presentation > pages > forgot_password_page.dart > %ForgotPasswordPageState
11
12 class _ForgotPasswordPageState extends State<ForgotPasswordPage> {
13   Widget build(BuildContext context) {
14     child: IconButton(
15       icon: const Icon(Icons.arrow_back, color: Colors.white),
16       onPressed: () {
17         final consumed = _handleBackInside(state, cubit);
18         if (consumed) Navigator.of(context).pop();
19       },
20     ), // Scaffold
21     // Positioned
22     if (state.loading)
23       Container(
24         color: Colors.white,
25         child: const Center(child: CircularProgressIndicator()),
26       ), // Container
27     // Stack
28     // Scaffold
29     // PageScope
30   }, // BlocConsumer
31 }
32
33 // widget _buildBody()
34 // BuildContext context;
35 // PasswordResetState state;
36 // PasswordResetCubit cubit;
37
38 // final scheme = Theme.of(context).colorScheme;
39 // final colors = Theme.of(context).brightness == Brightness.dark;
40 // // 0) Router
41 // if (state.loading) {
42 //   return FutureBuilder(
43 //     future: state.finderNode,
44 //     builder: (context, snapshot) {
45 //       if (snapshot.hasData) {
46 //         onRouteChanged: cubit.resetFinderNode,
47 //         onSubmit: cubit.findAccount,
48 //       }
49 //     },
50 //   );
51 // }
52
53 // // 1) Login metodo
54 // if (state.selectedChannel == null) {
55 //   final hasPhone =
56 //     (state.channels.contains('phone') ||
57 //      state.channels.contains('sms')) &&

```

Figura 74. Código de *password_reset_cubit.dart*

```

10 // module > features > auth > presentation > bloc > password_reset_cubit.dart > %PasswordResetCubit
11
12 class PasswordResetCubit extends Cubit<PasswordResetState> {
13   void resetFinderNode() => emit(
14     state.copyWith(
15       isLoading: false,
16       channels: const [],
17       maskedEmail: null,
18       maskedPhone: null,
19       phoneId: null,
20       verificationId: null,
21       selectedChannel: null,
22       awaitingNewPassword: false,
23       readPhoneEntry: false,
24       cachedAuthProof: null,
25       error: null,
26       info: null,
27     ),
28   );
29
30   void cancelNewPassword() =>
31     emit(state.copyWith(awaitingNewPassword: false, cachedAuthProof: null));
32
33   // ===== 100LUP =====
34   FutureOr<void> findAccount(String identifier) async {
35     // ===== 100LUP =====
36     // ===== 100LUP =====
37     // ===== 100LUP =====
38     // ===== 100LUP =====
39     // ===== 100LUP =====
40     // ===== 100LUP =====
41     // ===== 100LUP =====
42     // ===== 100LUP =====
43     // ===== 100LUP =====
44     // ===== 100LUP =====
45     // ===== 100LUP =====
46     // ===== 100LUP =====
47     // ===== 100LUP =====
48     // ===== 100LUP =====
49     // ===== 100LUP =====
50     // ===== 100LUP =====
51     // ===== 100LUP =====
52     // ===== 100LUP =====
53     // ===== 100LUP =====
54     // ===== 100LUP =====
55     // ===== 100LUP =====
56     // ===== 100LUP =====
57     // ===== 100LUP =====
58     // ===== 100LUP =====
59     // ===== 100LUP =====
60     // ===== 100LUP =====
61     // ===== 100LUP =====
62     // ===== 100LUP =====
63     // ===== 100LUP =====
64     // ===== 100LUP =====
65     // ===== 100LUP =====
66     // ===== 100LUP =====
67     // ===== 100LUP =====
68     // ===== 100LUP =====
69     // ===== 100LUP =====
70     // ===== 100LUP =====
71     // ===== 100LUP =====
72     // ===== 100LUP =====
73     // ===== 100LUP =====
74     // ===== 100LUP =====
75     // ===== 100LUP =====
76     // ===== 100LUP =====
77     // ===== 100LUP =====
78     // ===== 100LUP =====
79     // ===== 100LUP =====
80     // ===== 100LUP =====
81     // ===== 100LUP =====
82     // ===== 100LUP =====
83     // ===== 100LUP =====
84     // ===== 100LUP =====
85     // ===== 100LUP =====
86     // ===== 100LUP =====
87     // ===== 100LUP =====
88     // ===== 100LUP =====
89     // ===== 100LUP =====
90     // ===== 100LUP =====
91     // ===== 100LUP =====
92     // ===== 100LUP =====
93     // ===== 100LUP =====
94     // ===== 100LUP =====
95     // ===== 100LUP =====
96     // ===== 100LUP =====
97     // ===== 100LUP =====
98     // ===== 100LUP =====
99     // ===== 100LUP =====
100    // ===== 100LUP =====

```

El cierre técnico se apoya en *recovery_lookup.dart* y en las funciones *backend recoveryLookup.ts* y *finishPasswordReset.ts*, desarrolladas en las figuras 75, 76 y 77. Así, la recuperación combina la guía ofrecida en la aplicación con operaciones seguras resueltas fuera de la pantalla.

Figura 75. Código de *recovery_lookup.dart*

```

1  You, 4 months ago | 1 author (You)
2  import '../data/datasources/password_reset_remote_datasource.dart';
3  import '../repositories/password_reset_repository.dart';
4
5  You, 4 months ago | 1 author (You)
6  class RecoveryLookup {
7    final PasswordResetRepository repo;
8    RecoveryLookup(this.repo);
9
10   Future<RecoveryLookupResult> call(String raw) {
11     return repo.recoveryLookup(raw.trim());
12   }
13 }

```

Figura 76. Código de *recoveryLookup.ts*

```

functions > src > functions > recoveryLookup.ts ...
1  You, 4 months ago | 1 author (You)
2  import { onCall, HttpsError } from "firebase-functions/v2/https";
3  import * as logger from "firebase-functions/logger";
4  import { db } from "../config/firebase.js";
5  import { maskEmail, maskPhoneId } from "../utils/masking.js";
6  import { normalizeIdentifier } from "../utils/normalize.js";
7  import { rateLimit } from "../utils/rateLimit.js";
8
9  export const recoveryLookup = onCall(
10    {
11      region: "us-central1",
12      enforceAppCheck: true,
13      cors: true,
14      invoker: "public",
15    },
16    async (req) => {
17      try {
18        // Solo "peek", valida como pero NO escribe
19        await rateLimit(req.app.token, { mode: "peek", limit: 30 });
20
21        const raw = (req.data?.identifier ?? "").toString();
22        if ((raw || raw.trim()).length < 1) {
23          throw new HttpsError("invalid-argument", "identifier required");
24        }
25
26        const { email, cedula } = normalizeIdentifier(raw);
27        let userDoc: FirebaseFirestore.DocumentSnapshot | null = null;
28
29        if (email) {
30          const q = await db
31            .collection("users")
32            .where("emailLowercase", "==", email)
33            .limit(1)
34            .get();
35          if (!q.empty) userDoc = q.docs[0];
36        }
37
38        if (userDoc && cedula) {
39          const q2 = await db
40            .collection("users")
41            .where("cedula", "==", cedula)
42            .limit(1)
43            .get();
44        }
45      }
46    }
47  );

```

Figura 77. Código de *finishPasswordReset.ts*

```

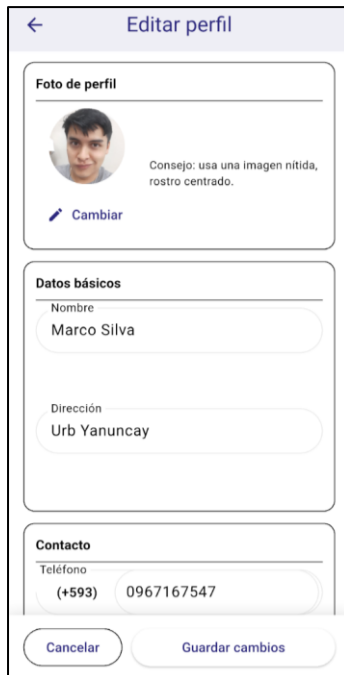
functions > src > functions > finishPasswordReset.ts ...
1  You, 4 months ago | 1 author (You)
2  import { onCall, HttpsError } from "firebase-functions/v2/https";
3  import * as logger from "firebase-functions/logger";
4  import { db } from "../config/firebase.js";
5  import { getAuth } from "firebase-admin/auth";
6
7  function normalizeIdentifier(raw: string): { email?: string; cedula?: string } {
8    const s = raw.trim().toLowerCase();
9    if (!s) return {};
10    if (s.includes("@")) return { email: s };
11    return { cedula: s.replace(/10/g, "") };
12  }
13
14  export const finishPasswordReset = onCall(
15    {
16      region: "us-central1",
17      enforceAppCheck: true,
18      cors: true,
19      invoker: "public",
20    },
21    async (req) => {
22      try {
23        // Aquí sí "commit": cuenta intento real de reset
24        await rateLimit(req.app.token, { mode: "commit", limit: 5 });
25
26        const newPassword = (req.data?.newPassword ?? "").toString();
27        const idToken = (req.data?.idToken ?? "").toString().trim();
28        const rawIdentifier = (req.data?.identifier ?? req.data?.cedula ?? "").toString().trim();
29
30        if (!idToken || newPassword.length < 6) {
31          throw new HttpsError("invalid-argument", "Bad input");
32        }
33
34        const decoded = await getAuth().verifyIdToken(idToken, true);
35        const phoneFromToken = decoded.phone_number || "";
36        if (!phoneFromToken) {
37          throw new HttpsError("failed-precondition", "Phone token required");
38        }
39      }
40    }
41  );

```

4.3.3.2.11. Historia de Usuario 11: Edición de Perfil de Usuario

El anexo 4 desarrolla la edición de perfil como una extensión directa de la consulta de datos de cuenta. La figura 78 y 79 presentan la pantalla de modificación y el selector de imagen, de modo que la actualización del perfil se mantenga en un entorno concentrado y comprensible.

Figura 78. Interfaz móvil de la pantalla Editar perfil (*/profile/edit*)



← Editar perfil

Foto de perfil

 Consejo: usa una imagen nítida, rostro centrado.

 Cambiar

Datos básicos

Nombre
Marco Silva

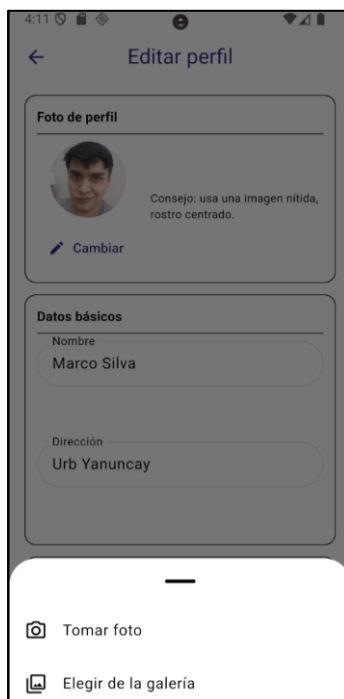
Dirección
Urb Yanuncay

Contacto

Teléfono
(+593) 0967167547

Cancelar Guardar cambios

Figura 79. Interfaz móvil del selector de imagen (*PickImageSheet*)



4:11

← Editar perfil

Foto de perfil

 Consejo: usa una imagen nítida, rostro centrado.

 Cambiar

Datos básicos

Nombre
Marco Silva

Dirección
Urb Yanuncay

 Tomar foto

 Elegir de la galería

La construcción de esta experiencia y el control de cambios se organizan en *profile_edit_page.dart* y *profile_editor_cubit.dart*, visibles en las figuras 80 y 81. En este bloque se resuelven tanto la composición de la vista como la validación previa al guardado.

Figura 80. Código de *profile_edit_page.dart*

```
lib > modules > features > profile > presentation > pages > profile_edit_page.dart > EditProfilePageState
32 class _EditProfilePageState extends State<EditProfilePage> {
33   final _nameCtrl = TextEditingController();
34   final _phoneCtrl = TextEditingController();
35   final _addrCtrl = TextEditingController();
36
37   // FocusNodes para auto-scroll y control fino del foco
38   final _nameNode = FocusNode();
39   final _addrNode = FocusNode();
40   final _phoneNode = FocusNode();
41
42   final _scroll = ScrollController();
43
44   @override
45   void dispose() {
46     _nameCtrl.dispose();
47     _phoneCtrl.dispose();
48     _addrCtrl.dispose();
49     _nameNode.dispose();
50     _addrNode.dispose();
51     _phoneNode.dispose();
52     _scroll.dispose();
53     super.dispose();
54   }
55
56   Future<bool> _confirmDiscard(BuildContext context) async {
57     final c = context.read<ProfileEditCubit>();
58     if (!c.hasUnsavedChanges) return true;
59
60     final ok = await showDialog<bool>({
61       context: context,
62       builder:
63         (_) => AlertDialog(
64           title: const Text('Descartar cambios'),
65           content: const Text(
66             '¿Quieres cambios sin guardar. ¿Deseas descartarlos?'
67           ), // Text
68           actions: [
69             TextButton(
70               onPressed: () => Navigator.pop(context, false),
71               child: const Text('Seguir editando'),

```

Figura 81. Código de *profile_editor_cubit.dart*

```
lib > modules > features > profile > presentation > blocs > profile_editor_cubit.dart > ProfileEditCubit
89 class ProfileEditCubit extends Cubit<ProfileEditState> {
90   final UpdateUserProfile _update;
91   final ProfileWriteRepository _repo;
92   final IFirebaseStorageService _cloudinary;
93   final FirebaseStorageService? _firebaseFallback;
94   final CloudinaryMaintenanceApi? _maintenance;
95
96   ProfileEditCubit({
97     required String uid,
98     required UpdateUserProfile update,
99     required ProfileWriteRepository repo,
100     IFirebaseStorageService? cloudinary,
101     FirebaseStorageService? firebaseFallback,
102     CloudinaryMaintenanceApi? maintenance,
103   }) : _update = update,
104       _repo = repo,
105       _cloudinary = cloudinary ?? CloudinaryStorageService(),
106       _firebaseFallback =
107         StorageFlags.enableFirebaseFallback
108           ? (firebaseFallback ?? firebaseFallbackService())
109           : null,
110       _maintenance = maintenance ?? CloudinaryMaintenanceApi(),
111       super(ProfileEditState(uid: uid, loading: true));
112
113   Future<void> bootstrap() async {
114     emit(state.copyWith(loading: true, error: null));
115     try {
116       final map = await _repo.fetchProfileOnce(state.uid) ?? {};
117       emit(
118         state.copyWith(
119           loading: false,
120           role: (map['role'] ?? '').toString(),
121           name: (map['name'] ?? '').toString(),
122           phone: (map['phone'] ?? '').toString(),
123           address: (map['addressline'] ?? '').toString(),
124           photoId: (map['photoId'] ?? '').toString(),
125           photoPublicId: (map['photoPublicId'] ?? '').toString(),
126           dirty: false,
127         ),
128       );
129     } catch (e, st) {
130       if (kDebugMode) {
131         debugPrint('bootstrap() error: $e\n$st');
132       }

```

Las figuras 82 y 83 trasladan la operación hacia el caso de uso de actualización y la fuente remota de escritura. Con ello, la edición de perfil queda sostenida por una separación clara entre la interacción del formulario y la persistencia de los cambios.

Figura 82. Código de *update_user_profile.dart*

```
lib > modules > features > profile > domain > usecases > update_user_profile.dart > ...
1 You, 4 months ago | 1 author (You)
  import '../repositories/profile_write_repository.dart';
2
3 You, 4 months ago | 1 author (You)
  class UpdateUserProfile {
4    final ProfileWriteRepository repo;
5    UpdateUserProfile(this.repo);
6
7    Future<void> call({
8      required String uid,
9      required Map<String, dynamic> patch,
10     String? authDisplayName,
11     String? authPhotoUrl,
12   }) async {
13     await repo.updateProfile(uid: uid, patch: patch);
14     await repo.syncAuth(
15       uid: uid,
16       displayName: authDisplayName,
17       photoUrl: authPhotoUrl,
18     );
19   }
20 }
```

Figura 83. Código de *profile_write_remote_datasource.dart*

```
lib > modules > features > profile > data > datasources > profile_write_remote_datasource.dart > ...
1 You, 4 months ago | 1 author (You)
  import 'package:cloud_firestore/cloud_firestore.dart';
2 import 'package:firebase_auth/firebase_auth.dart';
3 import '../core/config/collections.dart';
4
5 You, 4 months ago | 1 author (You)
  class ProfileWriteRemoteDataSource {
6    final FirebaseFirestore firestore;
7    final FirebaseAuth auth;
8    ProfileWriteRemoteDataSource(this.firestore, this.auth);
9
10    Future<Map<String, dynamic>> fetchOnce(String uid) async {
11      final doc = await firestore.collection(Col.users).doc(uid).get();
12      return doc.data();
13    }
14
15    Future<void> updateProfile({
16      required String uid,
17      required Map<String, dynamic> patch,
18    }) async {
19      await firestore.collection(Col.users).doc(uid).set(patch, SetOptions(merge: true));
20    }
21
22    Future<void> syncAuth({
23      required String uid,
24      String? displayName,
25      String? photoUrl,
26    }) async {
27      final u = auth.currentUser;
28      if (u != null) {
29        await u.updateProfile(
30          displayName: displayName,
31          photoUrl: photoUrl,
32        );
33      }
34    }
35  }
```

4.3.3.2.12. Historia de Usuario 12: Gestión de productos del catálogo (presentaciones y variantes)

Dentro del anexo 4, la gestión de productos del catálogo se presenta desde el espacio administrativo y el formulario con el que se crean o actualizan productos, presentaciones y variantes. Las figuras 84, 85, 86 y 87 sitúan este trabajo desde la interfaz y muestran cómo la edición del catálogo se concentra en un mismo flujo operativo.

Figura 84. Interfaz móvil del *workspace* de catálogo

Figura 85. Interfaz móvil del formulario de producto en creación

Nuevo producto

Datos básicos
Este producto se verá como parte de esta categoría en el catálogo y filtros del mapa.

Empanadas Panes Judías

[Ver todas las categorías](#)

Nombre del producto
Ej.: Bandeja 6 empanadas

Visible Oculto

Define si se muestra en pedidos y pantallas públicas.

Presentación y precio
Configura cómo vendes este producto.

Bandeja Unidad Combo

Descartar Guardar producto

Figura 86. Interfaz móvil del formulario de producto en edición

Editar producto

Contenido por unidad

Unidades por bandeja: 6

Precio base

Precio normal de esta presentación: Descu... 1.5

Sabores y variantes
Opcional. Para sabores (queso, carne, pollo) o tipos de masa (verde, harina).

Nombre del sabor: queso

Nombre del sabor: pollo

Nombre del sabor: carne

+ Agregar sabor / variante

Descartar Guardar producto

Figura 87. Interfaz móvil de gestión de presentaciones del producto

Presentaciones

Buscar producto

Bandeja 6 empanadas Ricas

Nueva presentación

Nombre: Combo Empanadas x5

Precio: 1

Cantidad mínima: 5

Activo

Cancelar Guardar

La lógica que mantiene este módulo se apoya en *product_form_cubit.dart*, *save_product_usecase.dart* y *products_remote_datasource.dart*, según se observa en las figuras 88, 89 y 90. Este conjunto permite que la administración del producto conserve validación, intención de negocio y persistencia sin dispersarse en la vista.

Figura 88. Código de *product_form_cubit.dart*

```
lib > modules > features > catalog > presentation > admin > screens > product_form > product_form_cubit.dart > ProductFormCubit
23 class ProductFormCubit extends Cubit<ProductFormState> {
24   void onNameChanged(String value) {
25     // You, 2 months ago | 1 author | 1 product form and product list
26     emit(
27       state.copyWith(
28         name: value,
29         dirty: true,
30         resetError: true,
31         resetSuccess: true,
32       ),
33     );
34   }
35
36   void onUnitTypeChanged(UnitType unitType) {
37     emit(
38       state.copyWith(
39         unitType: unitType,
40         dirty: true,
41         resetError: true,
42         resetSuccess: true,
43       ),
44     );
45   }
46
47   void onUnitTypePackChanged(String value) {
48     emit(
49       state.copyWith(
50         unitTypePack: value,
51         dirty: true,
52         resetError: true,
53         resetSuccess: true,
54       ),
55     );
56   }
57
58   void onBasePriceChanged(String value) {
59     emit(
60       state.copyWith(
61         basePrice: value,
62         dirty: true,
63         resetError: true,
64         resetSuccess: true,
65       ),
66     );
67   }
68
69   void onVisibilityToggled(bool isVisible) {
70     emit(
71       state.copyWith(
72         isVisible: isVisible,
73         dirty: true,
74         resetError: true,
75         resetSuccess: true,
76       ),
77     );
78   }
79
80   void onDescriptionChanged(String value) {
81     emit(
82       state.copyWith(
83         description: value,
84         dirty: true,
85         resetError: true,
86         resetSuccess: true,
87       ),
88     );
89   }
90
91   void addVariant() {
92     emit(
93       state.copyWith(
94         variants: [...state.variants, AdminProductVariant.empty()],
95         dirty: true,
96         resetError: true,
97         resetSuccess: true,
98       ),
99     );
100   }
101
102   void updateVariant(int index, String value) {
103     emit(
104       state.copyWith(
105         variants: [...state.variants, AdminProductVariant.empty()],
106         dirty: true,
107         resetError: true,
108         resetSuccess: true,
109       ),
110     );
111   }
112
113   void removeVariant(int index) {
114     if (index < 0 || index >= state.variants.length) return;
115     final updated = [...state.variants, AdminProductVariant.empty()].remove(index);
116     emit(
117       state.copyWith(
118         variants: updated,
119         dirty: true,
120         resetError: true,
121         resetSuccess: true,
122       ),
123     );
124   }
125
126   Future<void> importImageFromUrl(String url) {
127     Future<void> submit() async {
128       if (!state.isValid) {
129         emit(
130           state.copyWith(
131             errorMessage: 'Completa los campos obligatorios',
132             saveSuccess: false,
133           ),
134         );
135       }
136       return;
137     }
138   }
139 }
```

Figura 89. Código de *save_product_usecase.dart*

```
lib > modules > features > catalog > domain > usecases > admin > save_product_usecase.dart > SaveProductUseCase
1 import '../entities/admin_product.dart';
2 import '../repositories/products_repository.dart';
3
4 You, 2 months ago | 1 author (You)
5 class SaveProductUseCase {
6   final ProductsRepository repository;
7   SaveProductUseCase(this.repository);
8
9   Future<void> call(AdminProduct product) {
10     final isNew = product.id.isEmpty || product == AdminProduct.empty();
11     if (isNew) {
12       return repository.createProduct(product);
13     }
14     return repository.updateProduct(product);
15   }
16 }
```

Figura 90. Código de *products_remote_datasource.dart*

```
lib > modules > features > catalog > data > datasources > products_remote_datasource.dart > ProductsRemoteDataSourceImpl > update
7 abstract class ProductsRemoteDataSource {
8   Stream<List<AdminProductModel>> watchAll();
9   Stream<List<AdminProductModel>> watchByCategory(String categoryId);
10  Future<AdminProductModel> getById(String id);
11  Future<String> create(AdminProductModel model);
12  Future<void> update(AdminProductModel model);
13  Future<void> delete(String id);
14 }
15
16 You, 2 months ago | 1 author (You)
17 class ProductsRemoteDataSourceImpl implements ProductsRemoteDataSource {
18   final FirebaseFirestore firestore;
19   ProductsRemoteDataSourceImpl(this.firestore);
20
21   CollectionReference<Map<String, dynamic>> get _collection =>
22     firestore.collection(Col.products);
23
24   /// Query base que cumple las reglas públicas: solo productos publicados.
25   Query<Map<String, dynamic>> get _activeProductsQuery => _collection
26     .where('isActive', isEqualTo: true)
27     .where('isVisible', isEqualTo: true);
28
29   @override
30   Stream<List<AdminProductModel>> watchAll() {
31     print(
32       '[FIRESTORE] admin_products.watchAll = isActive=true & isVisible=true',
33     );
34     return _activeProductsQuery.snapshots().map((snapshot) {
35       print(
36         '[FIRESTORE] admin_products.watchAll = snapshot.size=${snapshot.docs.length}',
37       );
38       return snapshot.docs
39         .map((doc) => AdminProductModel.fromJson(doc.data(), doc.id))
40         .toList();
41     });
42   }
43
44   @override
45   Stream<List<AdminProductModel>> watchByCategory(String categoryId) {
46     print(
47       '[FIRESTORE] admin_products.watchByCategory = categoryId=${categoryId}',
48     );
49     return _activeProductsQuery
50       .where('categoryId', isEqualTo: categoryId)
51       .snapshots().map((snapshot) {
52         print(
53           '[FIRESTORE] admin_products.watchByCategory = snapshot.size=${snapshot.docs.length}',
54         );
55         return snapshot.docs
56           .map((doc) => AdminProductModel.fromJson(doc.data(), doc.id))
57           .toList();
58       });
59   }
60 }
```


4.3.3.2.13. Historia de Usuario 13: Realizar pedido de bandejas productos

El anexo 4 organiza la realización del pedido como una secuencia continua que parte del catálogo, pasa por el detalle del producto, incorpora el carrito y culmina en el *checkout* y su confirmación. Las figuras 91, 92, 93, 94 y 95 permiten seguir este recorrido y muestran cómo la compra se sostiene en pasos reconocibles para la usuaria.

Figura 91. Interfaz móvil de entrada al catálogo

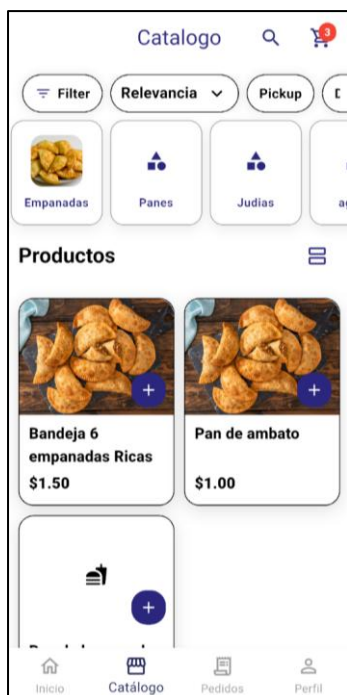


Figura 92. Interfaz móvil del detalle del producto



Figura 93. Interfaz móvil del carrito de compras

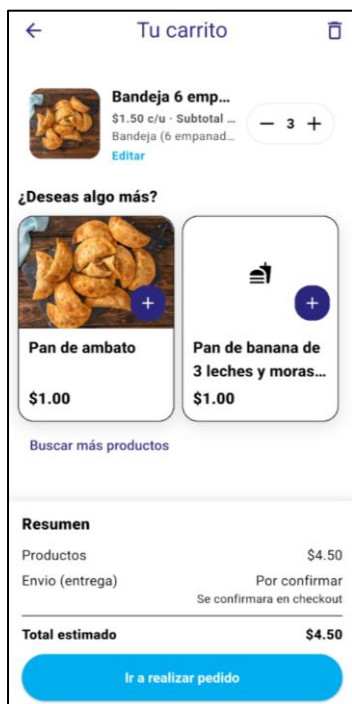


Figura 94. Interfaz móvil del checkout del pedido

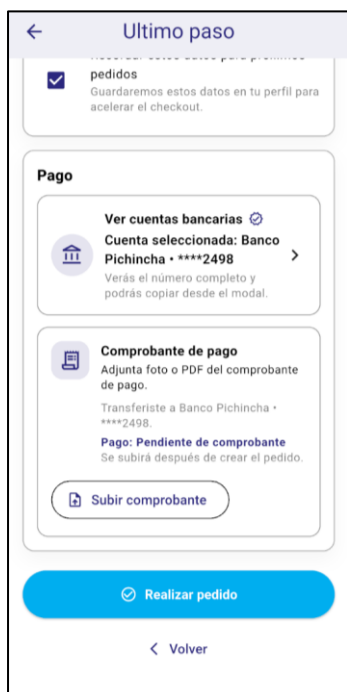
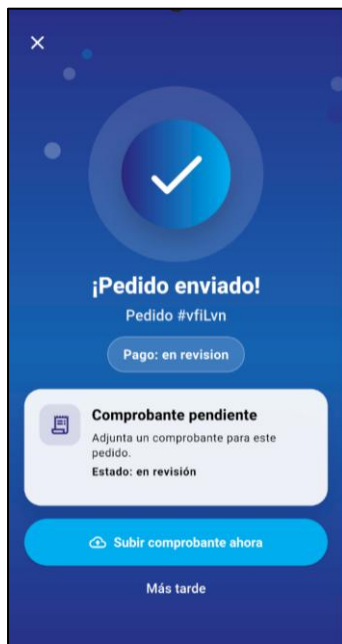


Figura 95. Interfaz móvil de confirmación del pedido



El soporte de esta historia se concentra en *checkout_from_cart_cubit.dart*, *create_order.dart* y *orders_remote_datasource.dart*, visibles en las figuras 96, 97 y 98. A través de estos componentes se validan los datos del pedido, se formaliza la operación en dominio y se registra la información correspondiente.

Figura 96. Código de *checkout_from_cart_cubit.dart*

```

378 class CheckoutFromCartCubit extends Cubit<CheckoutFromCartState> {
379   Future<void> _processPickedProof({
380     status: PaymentProofFlowStatus.ready,
381     localBytes: processedBytes,
382     contentType: processedMime,
383     fileName: processedName,
384     remoteUrl: null,
385     errorMessage: null,
386   });
387   }
388   }
389   }
390   }
391   }
392   }
393   }
394   }
395   }
396   }
397   }
398   }
399   }
400   }
401   }
402   }
403   }
404   }
405   }
406   }
407   }
408   }
409   }
410   }
411   }
412   }
413   }
414   }
415   }
416   }
417   }
418   }
419   }
420   }
421   }
422   }
423   }
424   }
425   }
426   }
427   }
428   }
429   }
430   }
431   }
432   }
433   }
434   }
435   }
436   }
437   }
438   }
439   }
440   }
441   }
442   }
443   }
444   }
445   }
446   }
447   }
448   }
449   }
450   }
451   }
452   }
453   }
454   }
455   }
456   }
457   }
458   }
459   }
460   }
461   }
462   }
463   }
464   }
465   }
466   }
467   }
468   }
469   }
470   }
471   }
472   }
473   }
474   }
475   }
476   }
477   }
478   }
479   }
480   }
481   }
482   }
483   }
484   }
485   }
486   }
487   }
488   }
489   }
490   }
491   }
492   }
493   }
494   }
495   }
496   }
497   }
498   }
499   }
500   }
501   }
502   }
503   }
504   }
505   }
506   }
507   }
508   }
509   }
510   }
511   }
512   }
513   }
514   }
515   }
516   }
517   }
518   }
519   }
520   }
521   }
522   }
523   }
524   }
525   }
526   }
527   }
528   }
529   }
530   }
531   }
532   }
533   }
534   }
535   }
536   }
537   }
538   }
539   }
540   }
541   }
542   }
543   }
544   }
545   }
546   }
547   }
548   }
549   }
550   }
551   }
552   }
553   }
554   }
555   }
556   }
557   }
558   }
559   }
560   }
561   }
562   }
563   }
564   }
565   }
566   }
567   }
568   }
569   }
570   }
571   }
572   }
573   }
574   }
575   }
576   }
577   }
578   }
579   }
580   }
581   }
582   }
583   }
584   }
585   }
586   }
587   }
588   }
589   }
590   }
591   }
592   }
593   }
594   }
595   }
596   }
597   }
598   }
599   }
600   }
601   }
602   }
603   }
604   }
605   }
606   }
607   }
608   }
609   }
610   }
611   }
612   }
613   }
614   }
615   }
616   }
617   }
618   }
619   }
620   }
621   }
622   }
623   }
624   }
625   }
626   }
627   }
628   }
629   }
630   }
631   }
632   }
633   }
634   }
635   }
636   }
637   }
638   }
639   }
640   }
641   }
642   }
643   }
644   }
645   }
646   }
647   }
648   }
649   }
650   }
651   }
652   }
653   }
654   }
655   }
656   }
657   }
658   }
659   }
660   }
661   }
662   }
663   }
664   }
665   }
666   }
667   }
668   }
669   }
670   }
671   }
672   }
673   }
674   }
675   }
676   }
677   }
678   }
679   }
680   }
681   }
682   }
683   }
684   }
685   }
686   }
687   }
688   }
689   }
690   }
691   }
692   }
693   }
694   }
695   }
696   }
697   }
698   }
699   }
700   }
701   }
702   }
703   }
704   }
705   }
706   }
707   }
708   }
709   }
710   }
711   }
712   }
713   }
714   }
715   }
716   }
717   }
718   }
719   }
720   }
721   }
722   }
723   }
724   }
725   }
726   }
727   }
728   }
729   }
730   }
731   }
732   }
733   }
734   }
735   }
736   }
737   }
738   }
739   }
740   }
741   }
742   }
743   }
744   }
745   }
746   }
747   }
748   }
749   }
750   }
751   }
752   }
753   }
754   }
755   }
756   }
757   }
758   }
759   }
760   }
761   }
762   }
763   }
764   }
765   }
766   }
767   }
768   }
769   }
770   }
771   }
772   }
773   }
774   }
775   }
776   }
777   }
778   }
779   }
780   }
781   }
782   }
783   }
784   }
785   }
786   }
787   }
788   }
789   }
790   }
791   }
792   }
793   }
794   }
795   }
796   }
797   }
798   }
799   }
800   }
801   }
802   }
803   }
804   }
805   }
806   }
807   }
808   }
809   }
810   }
811   }
812   }
813   }
814   }
815   }
816   }
817   }
818   }
819   }
820   }
821   }
822   }
823   }
824   }
825   }
826   }
827   }
828   }
829   }
830   }
831   }
832   }
833   }
834   }
835   }
836   }
837   }
838   }
839   }
840   }
841   }
842   }
843   }
844   }
845   }
846   }
847   }
848   }
849   }
850   }
851   }
852   }
853   }
854   }
855   }
856   }
857   }
858   }
859   }
860   }
861   }
862   }
863   }
864   }
865   }
866   }
867   }
868   }
869   }
870   }
871   }
872   }
873   }
874   }
875   }
876   }
877   }
878   }
879   }
880   }
881   }
882   }
883   }
884   }
885   }
886   }
887   }
888   }
889   }
890   }
891   }
892   }
893   }
894   }
895   }
896   }
897   }
898   }
899   }
900   }
901   }
902   }
903   }
904   }
905   }
906   }
907   }
908   }
909   }
910   }
911   }
912   }
913   }
914   }
915   }
916   }
917   }
918   }
919   }
920   }
921   }
922   }
923   }
924   }
925   }
926   }
927   }
928   }
929   }
930   }
931   }
932   }
933   }
934   }
935   }
936   }
937   }
938   }
939   }
940   }
941   }
942   }
943   }
944   }
945   }
946   }
947   }
948   }
949   }
950   }
951   }
952   }
953   }
954   }
955   }
956   }
957   }
958   }
959   }
960   }
961   }
962   }
963   }
964   }
965   }
966   }
967   }
968   }
969   }
970   }
971   }
972   }
973   }
974   }
975   }
976   }
977   }
978   }
979   }
980   }
981   }
982   }
983   }
984   }
985   }
986   }
987   }
988   }
989   }
990   }
991   }
992   }
993   }
994   }
995   }
996   }
997   }
998   }
999   }
1000  }

```

Figura 97. Código de *create_order.dart*

```

lib > modules > features > orders > domain > usecases > create_order.dart > ...
1  You, last month | 1 author (You)
2  import '../entities/order.dart';
3  import '../repositories/orders_repository.dart';
4
5  You, last month | 1 author (You)
6  class CreateOrderUseCase {
7    final OrdersRepository repo;
8    CreateOrderUseCase(this.repo);
9
10   Future<String> call(Order order) => repo.createOrder(order);
11 }

```

Figura 98. Código de *orders_remote_datasource.dart*

```

26 class OrdersRemoteDataSource {
27   final FirebaseFirestore _firestore;
28   final FirebaseAuth _auth;
29   OrdersRemoteDataSource(this._firestore, this._auth);
30
31   Future<OrgProfileModel> fetchOrgProfile() async {
32     final doc = await _firestore.collection(Col.orgProfile).doc('main').get();
33     if (!doc.exists) return null;
34     return OrgProfileModel.fromDoc(doc);
35   }
36
37   Future<List<PaymentAccountModel>> fetchPaymentAccounts() async {
38     final snap =
39       await _firestore
40         .collection(Col.paymentAccounts)
41         .where('isActive', isEqualTo: true)
42         .get();
43     final accounts =
44       snap.docs
45         .map((d) => PaymentAccountModel.fromMap({'id': d.id, ...d.data()}))
46         .toList();
47     return accounts.where((a) => a.isActive).toList(growable: false);
48   }
49
50   Future<String> createOrder(Order order) async {
51     final uid = _auth.currentUser?.uid;
52     final data = order.toJson();
53     final now = FieldValue.serverTimestamp();
54     final doc = await _firestore.collection(Col.orders).add({
55       ...data,
56       'userId': uid ?? order.userId,
57       'paymentStatus': order.paymentStatus,
58       'paymentProofUrl': order.paymentProofUrl,
59       'proofUrl': order.proofUrl,
60       'createdAt': now,
61       'updatedAt': now,
62     });
63     return doc.id;
64   }

```

4.3.3.2.14. Historia de Usuario 14: Adjuntar comprobante de pago

Tal como se detalla en el anexo 4, adjuntar el comprobante de pago forma parte del seguimiento del pedido una vez creada la orden. Las figuras 99 y 100 ubican el acceso a esta acción desde el historial y el detalle, integrando el comprobante como parte visible del estado del pago.

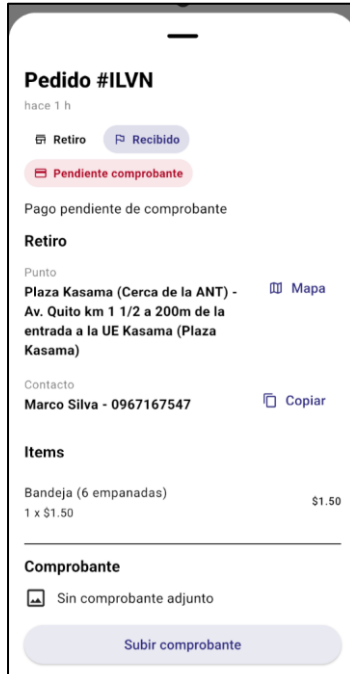
Figura 99. Interfaz móvil de *MyOrdersPage* (vista móvil)

Figura 100. Interfaz móvil de *OrderDetailPage* (vista móvil)

La selección del archivo y el control del proceso se desarrollan en *payment_proof_section.dart* y *payment_proof_cubit.dart*, presentados en las figuras 101 y 102. Este bloque mantiene la interacción ordenada y evita que la pantalla principal concentre toda la lógica del manejo del comprobante.

Figura 101. Código de *payment_proof_section.dart*

```

10 class PaymentProofSection extends StatelessWidget {
11   final Order order;
12   final PaymentProofState proofState;
13   final bool canUpload;
14   final VoidCallback onPick;
15   final VoidCallback onUpload;
16   final VoidCallback onViewRemote;
17   final VoidCallback onViewLocal;
18   final VoidCallback onClearLocal;
19   final VoidCallback? onReplace;
20
21   const PaymentProofSection({
22     super.key,
23     required this.order,
24     required this.proofState,
25     required this.canUpload,
26     required this.onPick,
27     required this.onUpload,
28     required this.onViewRemote,
29     required this.onViewLocal,
30     required this.onClearLocal,
31     this.onReplace,
32   });
33
34   @override
35   Widget build(BuildContext context) {
36     final remoteUrl = _remoteProof(order, proofState);
37     final localBytes = proofState.proof.localBytes;
38     final hasLocal = localBytes != null;
39     final hasRemote = remoteUrl.isNotEmpty;
40     final isBusy = proofState.isBusy;
41     final isUploading = proofState.isUploading;
42     final busyLabel = proofState.busyLabel;
43     final canSend = canUpload && !isBusy;
44
45     final stage = hasLocal
46       ? 'A'

```

Figura 102. Código de *payment_proof_cubit.dart*

```

lib > modules > features > orders > presentation > bloc > payment_proof_cubit.dart > %PaymentProofCubit
19 class PaymentProofCubit extends Cubit<PaymentProofState> {
20   final UploadPaymentProofUseCase _upload;
21   final UpdateOrderPaymentProofUseCase _attach;
22   final ImagePickerService _picker;
23
24   PaymentProofCubit({
25     required String orderId,
26     required String initialPaymentStatus,
27     String? initialProofUrl,
28     required UploadPaymentProofUseCase upload,
29     required UpdateOrderPaymentProofUseCase attach,
30     ImagePickerService? picker,
31     PaymentProofViewState? initialProofState,
32   }) : _upload = upload,
33       _attach = attach,
34       _picker = picker ?? ImagePickerService(),
35       super(
36         PaymentProofState(
37           orderId: orderId,
38           paymentStatus: initialPaymentStatus,
39           proof:
40             initialProofState ??
41             PaymentProofViewState(
42               status:
43                 (initialProofUrl ?? '').isNotEmpty
44                 ? PaymentProofFlowStatus.attached
45                 : PaymentProofFlowStatus.idle,
46               remoteUrl: initialProofUrl,
47             ), // PaymentProofViewState
48         ), // PaymentProofState
49       );
50
51   Future<void> pickFromGallery() => _pick(AppImageSource.gallery); // You, last month • Fast, enhance checkout success page with animat...
52   Future<void> pickFromCamera() => _pick(AppImageSource.camera);
53   Future<void> pickProofFromGallery() => pickFromGallery();
54   Future<void> pickProofFromCamera() => pickFromCamera();
55   Future<void> pickProofFromFilePicker() => pickFromFile();
56   Future<void> pickFromFile() async {
57
58   }
59   Future<void> _pick(AppImageSource source) async { ...
60
61   Future<void> _processPicked({ ...
62
63

```

Las figuras 103, 104 y 105 completan la historia con los casos de uso que suben y asocian el comprobante, así como con la actualización remota del pedido. De este modo, la evidencia de pago no queda como un elemento visual aislado, sino vinculada a la estructura de datos del proceso.

Figura 103. Código de *upload_payment_proof.dart*

```

lib > modules > features > orders > domain > usecases > upload_payment_proof.dart > ...
You, last month | 1 author (You)
1 import 'dart:typed_data';
2
3 import '../..//core/services/storage/i_file_storage_service.dart';
4 import '../repositories/orders_repository.dart';
5
6 You, last month | 1 author (You)
7 class UploadPaymentProofUseCase {
8   final OrdersRepository repo;
9   UploadPaymentProofUseCase(this.repo);
10
11   Future<UploadedFile> call({
12     required String orderId,
13     required Uint8List bytes,
14     required String contentType,
15     String? fileName,
16   }) async {
17     return repo.uploadPaymentProof(
18       orderId: orderId,
19       bytes: bytes,
20       contentType: contentType,
21       fileName: fileName,
22     );
23   }
24 }

```

Figura 104. Código de *update_order_payment_proof.dart*

```

lib > modules > features > orders > domain > usecases > update_order_payment_proof.dart > ...
You, last month | 1 author (You)
1 import '../repositories/orders_repository.dart';
2
3 You, last month | 1 author (You)
4 class UpdateOrderPaymentProofUseCase {
5   final OrdersRepository repo;
6   UpdateOrderPaymentProofUseCase(this.repo);
7
8   Future<void> call({
9     required String orderId,
10    required String paymentProofUrl,
11    required String paymentProofPublicId,
12    required int paymentProofSizeBytes,
13    required String paymentStatus,
14  }) {
15    return repo.attachPaymentProof(
16      orderId: orderId,
17      paymentProofUrl: paymentProofUrl,
18      paymentProofPublicId: paymentProofPublicId,
19      paymentProofSizeBytes: paymentProofSizeBytes,
20      paymentStatus: paymentStatus,
21    );
22  }
23 }

```

Figura 105. Código de `orders_remote_datasource.dart`

```

16 class OrdersRemoteDataSource {
17   Future<void> attachPaymentProof({
18     required String orderId,
19     required String paymentProofUrl,
20     required String paymentProofPublicId,
21     required int paymentProofSizeBytes,
22     required String paymentStatus,
23   }) async {
24     await firestore.collection(Col.orders).doc(orderId).set(
25       {
26         'paymentProofUrl': paymentProofUrl,
27         'paymentProofPublicId': paymentProofPublicId,
28         'paymentProofSizeBytes': paymentProofSizeBytes,
29         'paymentProofUploadDate': FieldValue.serverTimestamp(),
30         'paymentStatus': paymentStatus,
31         'updatedAt': FieldValue.serverTimestamp(),
32       },
33       SetOptions(merge: true),
34     );
35   }
36
37   Future<MyOrdersFetchResult> fetchMyOrders({
38     required MyOrdersQuery query,
39     required int limit,
40     DateTime? startAfterAt,
41     String? startAfterId,
42   }) async {
43     final userId = (query.userId ?? '').trim();
44     final phone = (query.phoneNormalized ?? '').trim();
45     if (userId.isEmpty && phone.isEmpty) {
46       return const MyOrdersFetchResult(orders: [], hasMore: false);
47     }
48
49     final sortField =
50       query.sortField == MyOrdersSortField.scheduledAt
51         ? 'scheduledAt'
52         : 'createdAt';
53     final descending = query.descending;
54     Query<Map<String, dynamic>> base = firestore
55       .collection(Col.orders)
56       .orderBy(sortField, descending: descending)
57       .orderBy(FieldPath.documentId);

```

4.3.3.2.15. Historia de Usuario 15: Ampliación del pedido: tipos de entrega

El anexo 4 presenta la ampliación del pedido mediante la incorporación de tipos de entrega que modifican el contenido del *checkout*. La figura 106 y 107 muestran el selector entre retiro y entrega coordinada, así como la aparición de los campos complementarios cuando la modalidad lo requiere.

Figura 106. Interfaz móvil de selección del método de entrega

Figura 107. Interfaz móvil del *checkout* con método “Entrega” y Datos de contacto.

The screenshot shows a mobile application interface for the final step of checkout, titled "Ultimo paso". It is divided into two main sections: "Entrega" (Delivery) and "Datos de contacto" (Contact Information).

Entrega Section:

- Método de entrega:** A section for selecting the delivery method.
 - Retiro en punto:** An option with a location pin icon and the text "Pasa a recoger cuando esté listo".
 - Entrega coordinada:** A selected option with a checkmark icon and the text "Sujeta a disponibilidad. Confirmar:". Below this is a text field containing "Urb Yanuncay".
- Entrega coordinada (sujeta a disponibilidad):** A note stating "Confirmamos costo antes de enviar."

Datos de contacto Section:

- Nombre de contacto:** A text field containing "Marco Silva".
- Telefono:** A text field containing "(+593) 0967167547".
- Recordar estos datos para proximos pedidos:** A checkbox that is checked, with the text "Guardaremos estos datos en tu perfil para acelerar el checkout."

La respuesta inmediata de la interfaz se sostiene en *delivery_method_selector.dart* y en *checkout_from_cart_cubit.dart*, visibles en la figura 108 y 109. Con ello, la selección del método y la actualización del formulario se mantienen articuladas dentro del mismo flujo.

Figura 108. Código de *delivery_method_selector.dart*

```
lib > modules > features > orders > presentation > widgets > delivery_method_selector.dart > DeliveryMethodSelector > build
4 class DeliveryMethodSelector extends StatelessWidget {
5   final DeliveryMethod value;
6   final ValueChanged<DeliveryMethod> onChanged;
7
8   const DeliveryMethodSelector({
9     super.key,
10    required this.value,
11    required this.onChanged,
12  });
13
14  @override
15  Widget build(BuildContext context) {
16    final cs = Theme.of(context).colorScheme;
17    final options = [
18      (
19        label: 'Retiro en punto',
20        value: DeliveryMethod.pickup,
21        icon: Icons.store_mall_directory_outlined,
22        helper: 'Pasa a recoger cuando esté listo',
23      ),
24      (
25        label: 'Entrega coordinada',
26        value: DeliveryMethod.delivery,
27        icon: Icons.local_shipping_outlined,
28        helper: 'Sujeta a disponibilidad. Confirmamos hora y costo.',
29      ),
30    ];
31  };
32
33  return Column(
34    crossAxisAlignment: CrossAxisAlignment.start,
35    children: [
36      const Text(
37        'Método de entrega',
38        style: TextStyle(fontWeight: FontWeight.w600),
39      ), // Text
40      const SizedBox(height: 6),
41      Wrap(
42        spacing: 10,
43        runSpacing: 0,
44        children:
45          options.map((opt) {
46            final selected = value == opt.value;
47            return ChoiceChip(
48              labelPadding: const EdgeInsets.symmetric(
49                horizontal: 12,
50                vertical: 10,
51              ), // EdgeInsets.symmetric
```


Figura 109. Código de *checkout_from_cart_cubit.dart*

```

lib > modules > features > orders > presentation > blocs > CheckoutFromCartCubit > CheckoutFromCartCubit
570 class CheckoutFromCartCubit extends Cubit<CheckoutFromCartState> {
571   String? _selectedPaymentAccountSelection();
572   return availableFirstId;
573 }
574
575 void setDeliveryMethod(DeliveryMethod method) {
576   // You, last month * feat: enhance order creation and payment proof
577   final st = state;
578   if (st.isSubmitting) return;
579   final result = _calculateDeliveryCost(
580     method: method,
581     config: st.deliveryConfig,
582   );
583   emit(
584     st.copyWith(
585       deliveryMethod: method,
586       pickupPoint: method == DeliveryMethod.pickup ? st.pickupPoint : null,
587       addressLine: method == DeliveryMethod.delivery ? st.addressLine : '',
588       deliveryCost: result.cost,
589       isDeliveryCostConfigured: result.configured,
590       isDeliveryCostLoading: false,
591     ),
592   );
593 }
594
595 void setPickupPoint(PickupPoint? point) {
596   final st = state;
597   if (st.isSubmitting) return;
598   emit(st.copyWith(pickupPoint: point));
599 }
600
601 void setPaymentAccount(String? accountId) {
602   final st = state;
603   if (st.isSubmitting) return;
604   final exists =
605     accountId != null && st.paymentAccounts.any((a) => a.id == accountId);
606   emit(st.copyWith(selectedPaymentAccountId: exists ? accountId : null));
607 }
608
609 void setAddressLine(String value) {
610   final st = state;
611   if (st.isSubmitting) return;
612   emit(st.copyWith(addressLine: value));
613 }
614
615 void setRecipient(required String name, required String phone) {

```

El respaldo funcional se completa con *get_delivery_config.dart* y *calculate_delivery_cost.dart*, presentados en la figura 110 y 111. Estas piezas permiten que la modalidad elegida no solo cambie la pantalla, sino también las condiciones con las que se procesa el pedido.

Figura 110. Código de *get_delivery_config.dart*

```

lib > modules > features > orders > domain > usecases > get_delivery_config.dart > GetDeliveryConfig
You, 3 weeks ago | 1 author (You)
1 import '../entities/delivery_config.dart';
2 import '../repositories/delivery_config_repository.dart';
3
4 class GetDeliveryConfig {
5   // You, 3 weeks ago * feat(delivery): Add delivery configuration and ...
6   final DeliveryConfigRepository repo;
7   GetDeliveryConfig(this.repo);
8
9   Future<DeliveryConfig> call() async {
10    final config = await repo.getDeliveryConfig();
11    if (config != null) {
12      print(
13        '[DELIVERY_CONFIG] loaded enabled=${config.enabled} fee=${config.flatFee}',
14      );
15    } else {
16      print('[DELIVERY_CONFIG] not_found');
17    }
18    return config;
19  }

```

Figura 111. Código de *calculate_delivery_cost.dart*

```

lib > modules > features > orders > domain > usecases > calculate_delivery_cost.dart > ...
You, 3 weeks ago | 1 author (You)
1 import '../entities/delivery_config.dart';
2 import '../entities/order.dart';
3
4 class DeliveryCostResult {
5   final double? cost;
6   final bool configured;
7
8   const DeliveryCostResult({
9     required this.cost,
10    required this.configured,
11  });
12 }
13
14 class CalculateDeliveryCost {
15   DeliveryCostResult call({
16     required DeliveryMethod method,
17     DeliveryConfig? config,
18   }) {
19     if (method == DeliveryMethod.pickup) {
20       print('[DELIVERY_COST] method=pickup cost=0 configured=true');
21       return const DeliveryCostResult(cost: 0, configured: true);
22     }
23
24     final enabled = config?.enabled == true;
25     final fee = (config?.flatFee ?? 0);
26     if (enabled && fee > 0) {
27       print('[DELIVERY_COST] method=delivery cost=$fee configured=true');
28       return DeliveryCostResult(cost: fee, configured: true);
29     }

```

4.3.3.2.16. Historia de Usuario 16: Selección de punto de retiro

Tal como se recoge en el anexo 4, la selección del punto de retiro se integra al *checkout* y permanece visible durante la consulta posterior del pedido. Las figuras 112, 113 y 114 reúnen la elección del punto, su reflejo en el historial y la visualización de la dirección correspondiente en el detalle.

Figura 112. Interfaz móvil de selección del punto de retiro



Figura 113. Interfaz móvil de *MyOrdersPage*

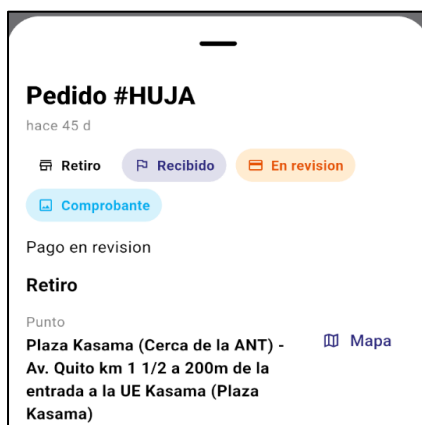


Figura 114. Interfaz móvil de *MyOrderDetailSheet*



La implementación que respalda este comportamiento se distribuye entre *pickup_point_dropdown.dart* y *my_order_detail_sheet.dart*, mostrados en las figuras 115 y 116. De esta manera, la elección inicial y la consulta posterior comparten una lógica consistente sin duplicar responsabilidades.

Figura 115. Código de pickup_point_dropdown.dart

```

18 > modules > features > orders > presentation > widgets > pickup_point_dropdown.dart > % PickupPointDropdown
You last month | 1 author (You)
1 import 'package:flutter/material.dart';
2 import '../pickup_points/domain/entities/pickup_point.dart';
3
4 You last month | 1 author (You)
5 class PickupPointDropdown extends StatelessWidget {
6   final List<PickupPoint> items;
7   final PickupPoint? selected;
8   final ValueChanged<PickupPoint?> onChanged;
9   final String? helperText;
10  final String? errorText;
11
12  const PickupPointDropdown({
13    super.key,
14    required this.items,
15    required this.selected,
16    required this.onChanged,
17    this.helperText,
18    this.errorText,
19  });
20
21  @override
22  Widget build(BuildContext context) {
23    return DropdownButtonFormField<PickupPoint>(
24      // Flutter 3.3+: use initialValue
25      initialValue: selected,
26      isExpanded: true,
27      decoration: const InputDecoration(
28        labelText: 'Punto de retiro',
29        border: OutlineInputBorder(),
30        isDense: true,
31      ), copyWith(helperText: helperText, errorText: errorText), // InputDecoration
32      hint: const Text(
33        'Selecciona un punto de retiro',
34        maxLines: 1,
35        overflow: TextOverflow.ellipsis,
36      ), // Text
37      selectedItemBuilder:
38        (context) =>

```

Figura 116. Código de my_order_detail_sheet.dart

```

18 > modules > features > orders > presentation > widgets > my_order_detail_sheet.dart > % MyOrderDetailSheetState > buildDeliveryRow
29 class MyOrderDetailSheetState extends State<MyOrderDetailSheet> {
30   MyOrderDetailSheet buildDeliveryRow(Order order) {
31     label: 'Direccion',
32     value: order.deliveryAddress ?? 'Sin direccion',
33   };
34 }
35
36 You last month | 1 author (You)
37 final pickup = _pickupPoint;
38 final name = pickup?.name ?? 'Punto de retiro';
39 final address = pickup?.address ?? '';
40 final value = address.isEmpty ? name : '$name - $address';
41 final mapUrl = pickup?.mapUrl;
42
43 return KeyRow(
44   label: 'Punto',
45   value: value,
46   actions: [
47     if (mapUrl != null && mapUrl.isNotEmpty)
48       TextButton.icon(
49         onPressed: () => _openMap(mapUrl),
50         icon: const Icon(Icons.map_outlined, size: 16),
51         label: const Text('Mapa'),
52       ), // TextButton.icon
53   ], // KeyRow
54 ); // KeyRow
55
56 String _resolveProofUrl(Order order, PaymentProofState proofState) {
57   final proof = proofState.proof.remoteUrl ?? '';
58   if (proof.isEmpty) return proof;
59   return order.proofUrl ?? '';
60 }
61
62 bool _canUploadProof(
63   String paymentStatus,
64   String proofUrl,
65   PaymentProofState proofState,
66 ) {
67   final normalized = paymentStatus.toLowerCase();
68   final allowed = normalized == PaymentStatus.pendingProof ||
69     normalized == 'pending_payment' ||
70     normalized == PaymentStatus.pendingValidation ||
71     normalized == PaymentStatus.inReview;
72 }

```

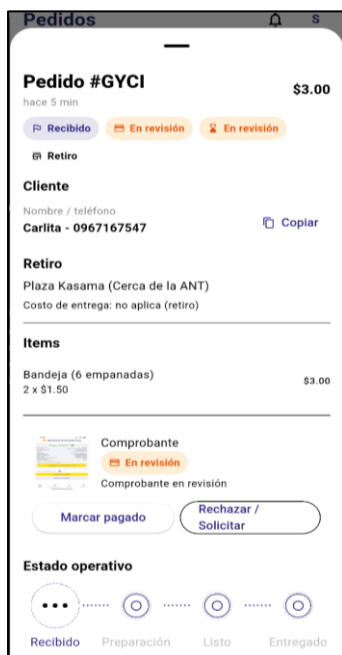
4.3.3.2.17. Historia de Usuario 17: Gestión de pedidos por parte del administrador

El anexo 4 sitúa la gestión de pedidos por parte del administrador en una bandeja operativa desde la cual se revisa el listado y se abre el detalle de cada orden. Las figuras 117 y 118 muestran este espacio de trabajo y reflejan la forma en que la administración concentra el seguimiento del pedido.

Figura 117. Interfaz móvil de la bandeja administrativa de pedidos



Figura 118. Interfaz móvil del detalle administrativo del pedido



La interfaz y el control de estados se apoyan en *admin_orders_page.dart*, *admin_orders_list_cubit.dart* y *admin_order_detail_bloc.dart*, desarrollados en las figuras 119, 120 y 121. Este bloque articula la carga de la bandeja, el filtrado y la respuesta ante las acciones ejecutadas sobre cada pedido.

Figura 119. Código de `admin_orders_page.dart`

```

lib > modules > features > admin_orders > presentation > pages > admin_orders_page.dart > _AdminOrdersPageState
51 class _AdminOrdersPageState extends State<AdminOrdersPage> {
179 Future<void> _openDetail(BuildContext context, Order order) async {
180   print('[ADMIN_ORDERS] open_order id=${order.id}');
181   await showModalBottomSheet<void>({
182     context: context,
183     isScrollControlled: true,
184     showDragHandle: true,
185     builder:
186       (_) => BlocProvider(
187         create:
188           (_) => AdminOrderDetailBloc(
189             orderId: order.id ?? '',
190             initial: order,
191             getOrder: widget.getOrderById,
192             updateOperationalStatus: widget.updateStatus,
193             updatePaymentStatus: widget.updatePayment,
194             updateSchedule: widget.scheduleOrder,
195             updateDeliveryCost: widget.updateDeliveryCost,
196             updateAdminNote: widget.updateAdminNote,
197           ), // AdminOrderDetailBloc
198         child: SizedBox(
199           height: MediaQuery.of(context).size.height * 0.9,
200           child: AdminOrderDetailSheet(
201             orderId: order.id ?? '',
202             initial: order,
203             getPickupPoints: widget.getPickupPoints,
204           ), // AdminOrderDetailSheet
205         ), // SizedBox
206       ), // BlocProvider
207   );
208 }

```

Figura 120. Código de `admin_orders_list_cubit.dart`

```

lib > modules > features > admin_orders > presentation > blocs > admin_orders_list_cubit.dart > AdminOrdersListState
1 import 'dart:async';
2
3 import 'package:flutter_bloc/flutter_bloc.dart';
4
5 import '../core/entities/order_status_consts.dart';
6 import '../core/entities/validation_phone_normalizer.dart';
7 import '../domain/entities/order.dart';
8 import '../domain/repositories/admin_orders_repository.dart';
9 import '../domain/usecases/get_admin_orders.dart';
10
11 enum AdminOrderChip { all, newOnes, payment, process, completed, canceled }
12
13 You last month | 1 author (You)
14 class AdminOrdersListState {
15   You last month > feat: Implement Admin Orders feature with repos...
16   const _sentinel = Object();
17 }
18
19 You last month | 1 author (You)
20 class AdminOrdersListCubit extends Cubit<AdminOrdersListState> {
21   final GetAdminOrders _getOrders;
22   Timer? _searchTimer;
23   int _requestSeq = 0;
24
25   AdminOrdersListCubit(this._getOrders) : super(AdminOrdersListState.initial());
26
27   Future<void> load([bool refresh = false]) async {
28
29   }
30   Future<void> loadMore() async {
31
32   }
33   Future<void> selectChip(AdminOrderChip chip) async {
34
35   }
36   Future<void> setDateRange(DateTime? from, DateTime? to) async {
37
38   }
39   Future<void> setDeliveryMethod(DeliveryMethod? method) async {
40
41   }
42   Future<void> applyFilters() {
43
44   }
45   void setSearch(String term) {
46
47   }
48   Future<void> clearFilters() async {
49
50   }
51   AdminOrdersQuery _buildQuery() {
52
53   }
54   String? _extractPhone(String raw) {
55
56   }
57   Future<void> _fetch([required bool reset]) async {
58
59   }
60 }

```

Figura 121. Código de `admin_order_detail_bloc.dart`

```

lib > modules > features > admin_orders > presentation > blocs > admin_order_detail_bloc.dart > ...
1 import 'package:flutter_bloc/flutter_bloc.dart';
2 import '../core/entities/order_status_consts.dart';
3 import '../domain/entities/order.dart';
4 import '../domain/services/operational_status_policy.dart';
5 import '../domain/usecases/get_admin_order_by_id.dart';
6 import '../domain/usecases/update_order_admin_note.dart';
7 import '../domain/usecases/update_order_delivery_cost.dart';
8 import '../domain/usecases/update_payment_status.dart';
9 import '../domain/usecases/schedule_order.dart';
10 import '../domain/usecases/update_order_status.dart';
11
12 enum AdminOrderActionType {
13   load,
14   operationalStatus,
15   paymentStatus,
16   schedule,
17   deliveryCost,
18   adminNote,
19   notifyClient,
20 }
21
22 enum AdminOrderRequestStatus { idle, loading, success, failure }
23
24 You last month | 1 author (You)
25 class AdminOrderDetailState {
26
27 }
28 sealed class AdminOrderDetailEvent {}
29
30 class AdminOrderStarted extends AdminOrderDetailEvent {}
31
32 You last month | 1 author (You)
33 class AdminOrderOperationalStatusTapped extends AdminOrderDetailEvent {
34
35 }
36
37 You last month | 1 author (You)
38 class AdminOrderPaymentStatusTapped extends AdminOrderDetailEvent {
39
40 }
41
42 You last month | 1 author (You)
43 class AdminOrderScheduleTapped extends AdminOrderDetailEvent {
44
45 }

```

Las figuras 122 y 123 trasladan la historia hacia el caso de uso de consulta y la fuente de datos de *Firestore*. Con ello, la operación administrativa conserva un enlace claro entre la revisión visible del pedido y la actualización de su información.

Figura 122. Código de *get_admin_orders.dart*

```
lib > modules > features > admin_orders > domain > usecases > get_admin_orders.dart > GetAdminOrders
You, last month | 1 author (You)
1 import '../repositories/admin_orders_repository.dart';
2
3 class GetAdminOrders {
4   final AdminOrdersRepository repo;
5   GetAdminOrders(this.repo);
6
7   Future<AdminOrdersPage> call({
8     required AdminOrdersQuery query,
9     int limit = 20,
10    DateTime? startAfterCreatedAt,
11    String? startAfterId,
12  }) {
13    return repo.fetchOrders(
14      query: query,
15      limit: limit,
16      startAfterCreatedAt: startAfterCreatedAt,
17      startAfterId: startAfterId,
18    );
19  }
20 }
```

Figura 123. Código de *admin_orders_firestore_datasource.dart*

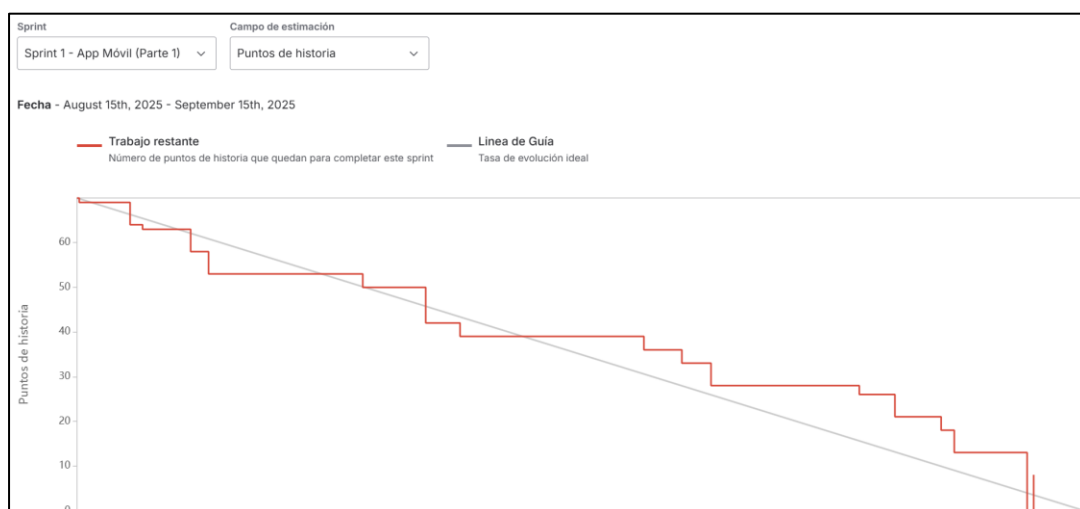
```
lib > modules > features > admin_orders > data > datasources > admin_orders_firestore_datasource.dart > AdminOrdersFirestoreDataSource
1 import 'package:cloud_firestore/cloud_firestore.dart' hide Order;
2
3 import '../config/collections.dart';
4 import '../entities/order_status_consts.dart';
5 import '../orders/data/models/order_model.dart';
6 import '../orders/domain/entities/order.dart' as domain;
7 import '../domain/repositories/admin_orders_repository.dart';
8
9 You, last month | 1 author (You)
10 class AdminOrdersFetchResult {
11   final List<domain.Order> orders;
12   final DateTime? lastCreatedAt;
13   final String? lastId;
14   final bool hasMore;
15
16   const AdminOrdersFetchResult({
17     required this.orders,
18     required this.hasMore,
19     this.lastCreatedAt,
20     this.lastId,
21   });
22
23 You, last month | 1 author (You)
24 class AdminOrdersFirestoreDataSource {
25   final FirebaseFirestore firestore;
26   AdminOrdersFirestoreDataSource(this.firestore);
27
28   CollectionReference<Map<String, dynamic>> get _orders =>
29     firestore.collection(Col.orders);
30
31   Future<AdminOrdersFetchResult> fetchOrders({
32     You, last month - feat: Implement Admin Orders feature with repos...
33
34   Future<domain.Order> getOrderById(String id) async {
35
36   Future<void> updateOperationalStatus({
37
```

En la capa de datos, este componente ejecuta la operación remota o local de Gestión de pedidos por parte del administrador para cumplir la tarea principal de la historia y sostener el flujo del proyecto. Ejecuta lecturas o escrituras y retorna resultados que el repositorio consume, manteniendo la *UI* lejos de llamadas directas a servicios.

4.3.3.3. Gráfico de trabajo pendiente del Sprint 1

En la figura 124, se puede visualizar el trabajo pendiente de las tareas definidas al principio del *sprint*. La gráfica del trabajo restante fue creada a través de *Jira Software*, lo cual posibilitó estar al tanto del avance de las actividades a llevar a cabo.

Figura 124. Gráfica de trabajo restante del 1er *Sprint* en *Jira*.



4.3.3.4. Sprint 1 – Revisión (*Sprint Review*)

Después de concluir el primer *sprint*, se llevó a cabo una reunión con la totalidad del equipo de desarrollo y la gerente de "100 Mujeres". El propósito de esta reunión era mostrar el progreso del producto funcional y, en particular, verificar si se habían alcanzado los objetivos establecidos, que se enfocaban en los módulos de módulos de Autenticación y acceso (bienvenida, selección de rol, registro, inicio de sesión y solicitud administrativa), Perfil de usuario (visualización, edición, recuperación de contraseña y cierre de sesión), Catálogo administrativo (gestión de productos) y Gestión de pedidos (creación del pedido, tipo de entrega, punto de retiro y bandeja administrativa de pedidos).

Se empleó la herramienta *Jira* para supervisar el progreso, gestionar las tareas y generar el Gráfico de Trabajo Pendiente (consultar figura 124), que mostró que las actividades habían terminado.

4.3.3.5. Sprint 1 – Retrospectiva (*Sprint Retrospective*)

Se consideraron tres preguntas para el análisis retrospectivo de los *sprints*, que brindarán una comprensión clara y precisa de cómo se desarrolló el primer *sprint*. Como se muestra en la tabla 19.

Tabla 19. Sprint 1 retrospectiva

Aspectos exitosos del sprint	Mejoras para aplicar	Errores en el sprint
Se consolidó un flujo completo de acceso con bienvenida, rol, registro, <i>login</i> y recuperación, junto con gestión de productos y pedidos con comprobante, entrega y retiro.	Conviene reforzar pruebas integrales entre <i>app</i> y bandeja <i>admin</i> , unificar validaciones de formularios y medir rendimiento para reducir fricción en perfil, <i>checkout</i> y seguimiento de pedidos.	Existieron inconsistencias de estado entre pantallas y validaciones no homogéneas, además de ajustes de configuración que afectaron continuidad del desarrollo en acceso, catálogo y atención de pedidos.

4.3.4. Sprint 2

4.3.4.1. Sprint 2 – Planificación (*Sprint Planning*)

En base a la tabla 20 y 21, se describen las historias de usuario y la historia técnica consideradas para dar lugar a el segundo *sprint* backlog móvil, las cuales tienen una alta probabilidad de ser desarrolladas. Asimismo, se detallaron las tareas de cada historia de usuario que integra el *sprint*.

Tabla 20. Sprint 2 retrospectiva

Sprint Backlog					
Sprint: 2					
Objetivo: Completar funcionalidades operativas y de expansión (verificación de pago, notificaciones, filtros por fecha, agendamiento, zonas, solicitudes, mapa/perfiles y componentes de contenido/estadísticas).					
Historia	Est	Categoría	Tarea	Est	Estado
H18 [App] - Verificación de pago (<i>admin</i>)	5	UI	Botones en detalle <i>admin</i> : 'Marcar pagado' / 'Rechazar' con evidencia.	2	Completado
		Firestore / Datos	Implementar <i>updatePaymentStatus (paidAt/ paidBy/ reason)</i> en <i>datasource</i> .	2	Completado
		Seguridad	Revisar reglas <i>Firestore</i> y detectar campos bloqueados para verificación de pago.	1	Completado
H19 [App] - Notificaciones <i>push</i>	8	Firestore / FCM	Inicializar <i>PushService</i> (permisos, <i>token</i>) en <i>startup</i> .	2.5	Completado
		Backend	<i>Cloud Function</i> : generar notificaciones por creación/actualización de pedidos.	2.5	Completado
		UI	Pantalla <i>NotificationsPage</i> + navegación desde campana.	2	Completado
		Datos	Persistir tokens/ <i>notifications</i> en subcolecciones de usuario.	1	Completado
H20 [App] - Consulta de pedidos por fecha (cliente)	3	UI	Historial con filtros por período/rango y botón aplicar/limpiar.	1	Completado
		Lógica / Estado	<i>MyOrdersCubit</i> : aplicar filtros y actualizar lista.	1.5	Completado

		<i>Firestore / Datos</i>	Consulta <i>Firestore</i> por usuario + rango de fechas.	0.5	Completado
H21 [App] - Consulta de pedidos por fecha (administrador)	3	<i>UI</i>	Selector de rango de fechas en <i>Admin > Pedidos</i> .	1	Completado
		<i>Firestore / Datos</i>	Consulta <i>Firestore</i> por rango (y conteo/lista).	1.5	Completado
		<i>UI / Navegación</i>	Abrir detalle desde resultados.	0.5	Completado
H22 [App] - Agendamiento de entregas	5	<i>UI</i>	Agendar entrega: <i>date/time picker</i> y notas en pedido.	2	Completado
		<i>Firestore / Datos</i>	Guardar <i>scheduledAt</i> y reflejar en Pedidos > Programados del cliente.	2	Completado
		<i>Placeholder</i>	Dejar listo disparador para notificación por reprogramación.	1	Completado
H23 [App] - Gestión de zonas de venta	5	<i>UI</i>	<i>AdminSalesZonesPage</i> : listado de zonas + crear/editar.	2	Completado
		Lógica / Validación	Validar nombre/ubicación y estados (activo).	1.5	Completado
		<i>Firestore / Datos</i>	<i>CRUD</i> en <i>Firestore</i> + desactivación (<i>isActive=false</i>) en lugar de <i>delete</i> .	1.5	Completado
H24 [App] - Solicitud de nueva emprendedora del proyecto	5	<i>UI</i>	Formulario de solicitud de emprendedora (datos + documentos + fotos).	2	Completado
		Validación	Validaciones de requeridos y mensajes de error.	1.5	Completado
		<i>Firestore / Storage</i>	Guardar solicitud en <i>Firestore</i> + subir adjuntos.	1.5	Completado
H25 [App] - Registro de acuerdos privados de ubicación	3	<i>UI</i>	<i>AdminLocationAgreementsPage</i> : listado de acuerdos y estados.	1	Completado
		<i>UI/Adjuntos</i>	Formulario crear/editar + anexos opcionales (foto/ <i>PDF</i> /nota).	1	Completado
		Seguridad	Identificar necesidad de reglas para colección <i>location_agreements</i> .	1	Completado
H26 [App] - Mapa de puntos de venta cercanos	8	<i>UI</i>	<i>MapPage</i> : mapa referencial con pines + listado/contador.	3	Completado
		<i>UI/Filtros</i>	Filtros por zona/tipo y estado vacío con 'Limpiar filtros'.	2	Completado
		Navegación	Ficha resumida y <i>CTA</i> para abrir perfil desde mapa/lista.	1	Completado
		<i>UX</i>	Mantener estado (filtros/ <i>zoom</i>) al volver del perfil.	1	Completado
		<i>Placeholder</i>	Recibir argumentos <i>mode (near/zone)</i> sin filtro de cercanía aún.	1	Completado
H27 [App] - Ver perfiles de emprendedoras	3	<i>UI</i>	Listado 'Conócelas' (<i>cards</i>) + búsqueda simple si aplica.	1	Completado
		<i>UI</i>	Detalle de emprendedora (<i>info</i> + productos + <i>tags</i>).	1.5	Completado
		Lógica / Datos	Favoritos y acciones de contacto (copiar/llamar/ <i>WhatsApp</i>).	0.5	Completado
H28 [App] - Testimonios o mensajes motivacionales	2	<i>UI</i>	<i>StoriesCarousel</i> en <i>Home</i> (<i>swipe</i> /flechas) con mensajes de impacto.	1	Completado
		Navegación / Datos	Abrir <i>StoryDetailPage</i> y cargar historias (local/ <i>Firestore</i>).	1	Completado

H29 [App] - Visualización de estadísticas (sin exportar)	5	UI	<i>AdminDashboard</i> : KPIs/resumen (pedidos y productos más vendidos).	2	Completado
		Firestore / Datos	<i>Datasource</i> de estadísticas básicas con rango fijo (últimos 30 días).	2	Completado
		Placeholder	Botón <i>Analytics</i> marcado como <i>ComingSoon</i> (sin vista detallada).	1	Completado
H30 [App] - Registro de ubicación por la emprendedora	8	UI	Formulario ubicación: dirección/barrio/sector + descripción + foto.	3	Completado
		Validación	Validar campos y archivo (formato/tamaño) + <i>feedback</i> .	2	Completado
		Firestore / Storage	Guardar/actualizar <i>entrepreneur_location</i> en <i>Firestore</i> + subir imagen.	2	Completado
		UX	Reemplazo de ubicación previa + refresco en mapa/perfil.	1	Completado

Tabla 21. Sprint Backlog Técnico Móvil

Historia	Est	Categoría	Tarea	Est	Estado
HT1- Seguridad y privacidad de la aplicación móvil	8	Desarrollo	Definir reglas de acceso por autenticación, propiedad y rol en <i>firestore.rules</i> para proteger la actualización de perfiles y permisos.	2	Completado
		Desarrollo	Implementar verificación de integridad de la aplicación con <i>AppCheckInitializer</i> antes del consumo de servicios remotos.	1	Completado
		Desarrollo	Incorporar limitación de intentos en funciones sensibles mediante <i>rateLimit()</i> para prevenir abuso y saturación.	1	Completado
		Desarrollo	Validar rol administrativo en funciones críticas con <i>requireAdmin(uid)</i> antes de permitir cambios de configuración.	1	Completado
		Desarrollo	Asegurar la carga protegida de imágenes de perfil con <i>uploadProfileImage</i> , validando autenticación, formato y tamaño.	1	Completado
		Desarrollo	Construir perfiles públicos con datos controlados mediante <i>rebuildPublicEntrepreneur</i> , separando información pública de la interna.	1	Completado
		Desarrollo	Gestionar la verificación de cuenta por correo y el período de gracia con <i>EmailVerificationCubit</i> .	1	Completado

4.3.4.1.1. Historia de usuario 18: Verificación de pago (admin)

Tal como se desarrolla en el anexo 4, la verificación de pago administrativa se resuelve dentro del detalle del pedido y contempla tanto la aprobación como el rechazo del comprobante. La figura 125 y 126 muestran el estado previo y posterior a la revisión, junto con el diálogo utilizado cuando es necesario registrar un motivo de rechazo.

Figura 125. Interfaz móvil del detalle administrativo del pedido (*AdminOrderDetailSheet*) con sección de pago: (a) estado previo a revisión; (b) estado posterior marcado como “Pagado”.

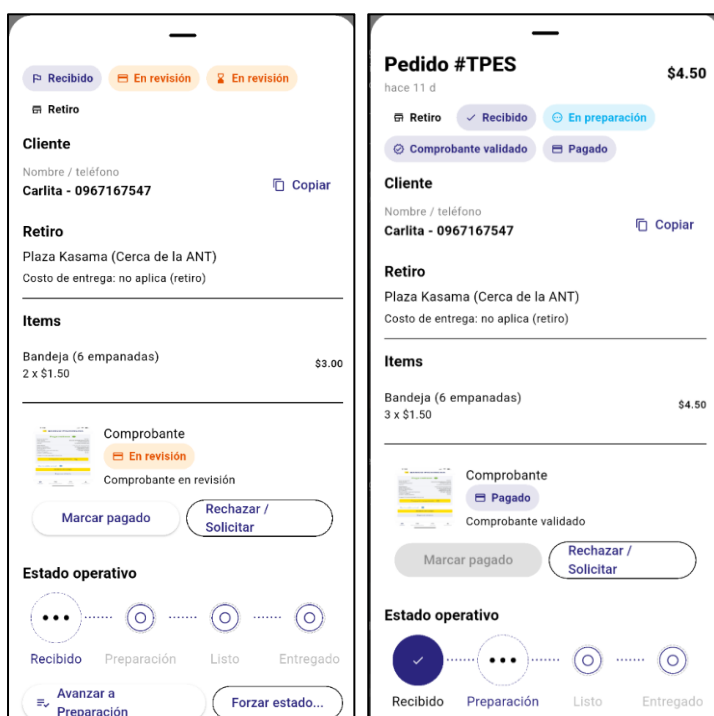
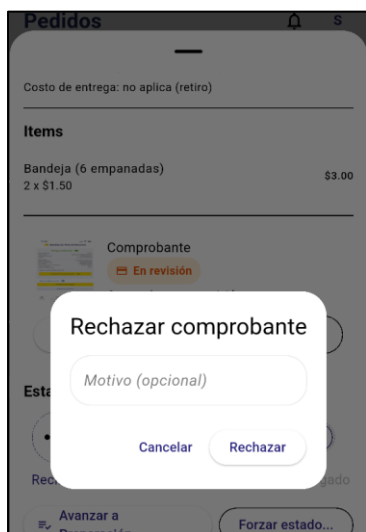


Figura 126. Interfaz móvil del diálogo “Rechazar comprobante” con campo de motivo y acciones de confirmación/cancelación.



La integración de estas acciones se concentra en *admin_order_detail_sheet.dart*, visible en la figura 127. Desde este componente se enlaza la evidencia de pago con las decisiones que modifican su estado dentro del flujo administrativo.

Figura 127. Código de *admin_order_detail_sheet.dart*: integración de *OrderPaymentProofSection*, *callbacks* (*_markPaid*, *_rejectPayment*) y confirmaciones previas al despacho del evento.

```
lib > modules > features > admin_orders > presentation > widgets > admin_order_detail_sheet.dart > % AdminOrderDetailSheetState > build
33 class AdminOrderDetailSheetState extends State<AdminOrderDetailSheet> {
48 Widget build(BuildContext context) {
125 OrderPaymentProofSection(
126   proofUrl: order.proofUrl,
127   hasProof: order.hasProof,
128   paymentStatus: order.paymentStatus,
129   onViewProof: () => _openProof(context, order.proofUrl),
130   onMarkPaid:
131     state.isActionLoading(AdminOrderActionType.paymentStatus)
132     ? null
133     : () => _markPaid(context),
134   onReject:
135     state.isActionLoading(AdminOrderActionType.paymentStatus)
136     ? null
137     : () => _rejectPayment(context),
138   loadingPayment: state.isActionLoading(
139     AdminOrderActionType.paymentStatus,
140   ), // OrderPaymentProofSection
141 }

144 Future<void> _markPaid(BuildContext context) async {
145   context.read<AdminOrderDetailSheet>().add(
146     AdminOrderPaymentStatusEvent(
147       nextPaymentStatus: PaymentStatusConst.paid,
148       reason: context,
149       paidAt: DateTime.now(),
150     ), // AdminOrderPaymentStatusEvent
151   );
152 }

153 Future<void> _rejectPayment(BuildContext context) async {
154   final reasonId = context.read<AdminOrderDetailSheet>().add(
155     AdminOrderPaymentStatusEvent(
156       nextPaymentStatus: PaymentStatusConst.rejected,
157       reason: context,
158       reviewedBy: context,
159     ), // AdminOrderPaymentStatusEvent
160   );
161   context.read<AdminOrderDetailSheet>().add(
162     AdminOrderPaymentStatusEvent(
163       nextPaymentStatus: PaymentStatusConst.paid,
164       reason: context,
165       paidAt: DateTime.now(),
166     ), // AdminOrderPaymentStatusEvent
167   );
168 }
}
```

El cambio de estado se formaliza mediante *update_payment_status.dart* y *admin_orders_firestore_datasource.dart*, presentados en las figuras 128 y 129. Así, la validación del pago queda respaldada por una operación controlada y por la persistencia correspondiente en el pedido.

Figura 128. Código de *update_payment_status.dart*: caso de uso para actualizar estado de pago.

```
lib > modules > features > admin_orders > domain > usecases > update_payment_status.dart > ...
You, last month | 1 author (You)
1 import '../core/entities/order_status_consts.dart';
2 import '../repositories/admin_orders_repository.dart';
3 You, last month > feat: Implement Admin Orders feature with repos...
You, last month | 1 author (You)
4 class UpdateOrderPaymentStatusUseCase {
5   final AdminOrdersRepository repo;
6   UpdateOrderPaymentStatusUseCase(this.repo);
7
8   Future<void> call({
9     required String orderId,
10    required String nextPaymentStatus,
11    DateTime? paidAt,
12    String? paidBy,
13    String? rejectionReason,
14    String? reviewedBy,
15  }) {
16    final normalized =
17      PaymentStatusConst.normalize(nextPaymentStatus) ?? nextPaymentStatus;
18    return repo.updatePaymentStatus(
19      orderId: orderId,
20      nextPaymentStatus: normalized,
21      paidAt: paidAt,
22      paidBy: paidBy,
23      rejectionReason: rejectionReason,
24      reviewedBy: reviewedBy,
```

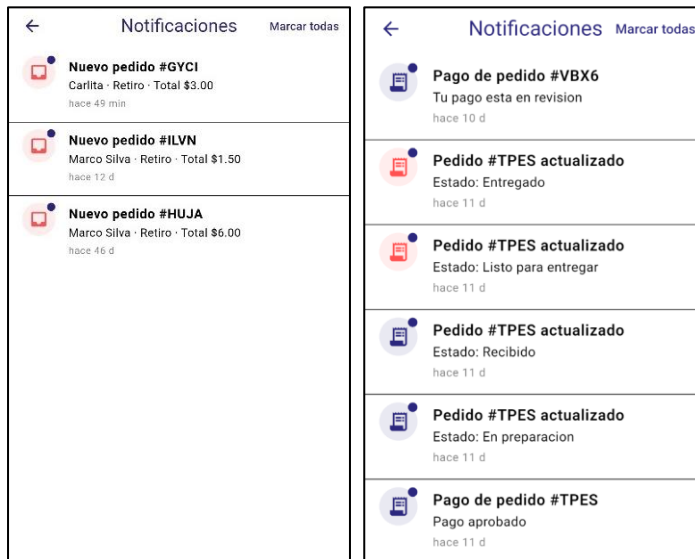
Figura 129. Código de `admin_orders_firestore_datasource.dart`: `updateMap` y `update` de pago en `orders`.

```
lib > modules > features > admin_orders > data > datasources > admin_orders_firestore_datasource.dart > AdminOrdersFirestoreDataSource
23 class AdminOrdersFirestoreDataSource {
110 Future<void> updatePaymentStatus({
111   required String orderId,
112   required String nextPaymentStatus,
113   DateTime? paidAt,
114   String? paidBy,
115   String? rejectionReason,
116   String? reviewedBy,
117 }) async {
118   final normalized = PaymentStatusConst.normalize(nextPaymentStatus) ?? nextPaymentStatus;
119   final data = <String, dynamic>{
120     'paymentStatus': normalized,
121     'updatedAt': FieldValue.serverTimestamp(),
122     'paymentReviewedAt': FieldValue.serverTimestamp(),
123   };
124   if (reviewedBy != null && reviewedBy.isNotEmpty) {
125     data['paymentReviewedBy'] = reviewedBy;
126   }
127   if (normalized == PaymentStatusConst.pagado) {
128     data['paidAt'] = Timestamp.fromDate(paidAt ?? DateTime.now());
129     if (paidBy != null && paidBy.isNotEmpty) {
130       data['paidBy'] = paidBy;
131     }
132     data['paymentRejectedReason'] = FieldValue.delete();
133   } else if (normalized == PaymentStatusConst.rechazado) {
134     if (rejectionReason != null && rejectionReason.isNotEmpty) {
135       data['paymentRejectedReason'] = rejectionReason;
136     }
137   } else {
138     data['paymentRejectedReason'] = FieldValue.delete();
139   }
140   await _collection.update(orderId, data);
141 }
142 }
```

4.3.4.1.2. Historia de usuario 19: Notificaciones push

El anexo 4 incorpora las notificaciones *push* como un medio para mantener informada a la usuaria sobre los eventos relevantes del ciclo de pedidos. La figura 130 presenta la bandeja donde estos avisos se consultan desde la aplicación y se integran al recorrido cotidiano.

Figura 130. Interfaz móvil de la bandeja de notificaciones



La gestión visible del módulo se organiza en `notifications_page.dart`, `notifications_cubit.dart` y `notifications_badge_cubit.dart`, visibles en las figuras 131, 132 y 133. En este bloque se articulan la carga del listado, la paginación y el conteo de notificaciones pendientes.

Figura 131. Código de *notifications_page.dart*: UI y conexión con estado de notificaciones.

```

16 class _NotificationsPageState extends State<NotificationsPage> {
17
18   @override
19   void initState() {
20     super.initState();
21     _loadInitial();
22   }
23
24   void _loadInitial() {
25     Future<void> loadInitial() async {
26       try {
27         final res = await _getNotifications(limit: 20);
28         if (_token != myToken || isClosed) return;
29         emit(
30           state.copyWith(
31             loading: false,
32             items: res.items,
33             hasMore: res.hasMore,
34             lastDate: res.lastDate,
35             lastId: res.lastId,
36             errorMessage: res.items.isEmpty ? null : null,
37           ),
38         );
39       } catch (e) {
40         if (_token != myToken || isClosed) return;
41         emit(
42           state.copyWith(
43             loading: false,
44             errorMessage: e.toString(),
45             items: const [],
46             hasMore: true,
47             lastDate: null,
48             lastId: null,
49           ),
50         );
51       }
52     }
53   }
54
55   @override
56   Widget build(BuildContext context) {
57     return BlocListener<NotificationsCubit, NotificationsState>{
58       listener: (prev, curr) => {
59         prev.markAllMessage != curr.markAllMessage ||
60         prev.markAllError != curr.markAllError,
61       },
62       listener: (context, state) {
63         final messenger = ScaffoldMessenger.of(context);
64         if (state.markAllMessage != null) {
65           messenger.showSnackBar(
66             SnackBar(content: Text(state.markAllMessage!)),
67           );
68           context.read<NotificationsCubit>().clearMarkAllFeedback();
69           return;
70         }
71         if (state.markAllError != null) {
72           messenger.showSnackBar(
73             SnackBar(content: Text(state.markAllError!)),
74           );
75           context.read<NotificationsCubit>().clearMarkAllFeedback();
76         }
77       },
78     ).child(
79       Scaffold(
80         appBar: AppBar(
81           title: const Text('Notificaciones'),
82           actions: [
83             BlocBuilder<NotificationsCubit, NotificationsState>{
84               builder: (context, state) {
85                 final unread = state.unreadCount;
86                 final showButton = unread > 0 || state.markingAll;
87                 if (!showButton) return const SizedBox.shrink();
88                 final enabled = unread > 0 && !state.markingAll;
89                 return TextButton(
90                   onPressed: enabled
91                     ? () => context.read<NotificationsCubit>().markAll()
92                     : null,
93                 );
94               },
95             ),
96           ],
97         ),
98       ),
99     );
100   }
101 }

```

Este componente en *notifications_page.dart* configura la pantalla asociada a Notificaciones *push* para mantener informada a la usuaria con avisos del ciclo de pedidos.

Figura 132. Código de *notifications_cubit.dart*: carga, paginación y marcado de lectura.

```

lib > modules > features > notifications > presentation > blocs > notifications_cubit.dart > NotificationsCubit > loadInitial
100 class NotificationsCubit extends Cubit<NotificationsState> {
101
102   Future<void> loadInitial() async {
103     try {
104       You, 3 weeks ago • feat: Implement notifications feature with CRUD...
105       final res = await _getNotifications(limit: 20);
106       if (_token != myToken || isClosed) return;
107       emit(
108         state.copyWith(
109           loading: false,
110           items: res.items,
111           hasMore: res.hasMore,
112           lastDate: res.lastDate,
113           lastId: res.lastId,
114           errorMessage: res.items.isEmpty ? null : null,
115         ),
116       );
117     } catch (e) {
118       if (_token != myToken || isClosed) return;
119       emit(
120         state.copyWith(
121           loading: false,
122           errorMessage: e.toString(),
123           items: const [],
124           hasMore: true,
125           lastDate: null,
126           lastId: null,
127         ),
128       );
129     }
130   }
131 }

```

Figura 133. Código de *notifications_badge_cubit.dart*: inicio del conteo de no leídas.

```

lib > modules > features > notifications > presentation > blocs > notifications_badge_cubit.dart > NotificationsBadgeCubit
2
3 import 'package:flutter_bloc/flutter_bloc.dart';
4
5 import 'package:domain/usecases/watch_unread_notifications_count.dart';
6
7 You, 3 weeks ago [1 author] feat
8 class NotificationsBadgeState {
9   final int unread;
10   final bool loading;
11
12   const NotificationsBadgeState({required this.unread, required this.loading});
13
14   factory NotificationsBadgeState.initial() =>
15     const NotificationsBadgeState(unread: 0, loading: true);
16 }
17
18 You, 3 weeks ago [1 author] feat
19 class NotificationsBadgeCubit extends Cubit<NotificationsBadgeState> {
20   final WatchUnreadNotificationsCount _watch;
21   StreamSubscription<int>? _sub;
22
23   NotificationsBadgeCubit(this._watch)
24     : super(NotificationsBadgeState.initial());
25
26   void start() {
27     You, 3 weeks ago • feat: Implement notifications feature with CRUD...
28     _sub?.cancel();
29     emit(NotificationsBadgeState(unread: state.unread, loading: false));
30     _sub = _watch.listen((count) {
31       emit(NotificationsBadgeState(unread: count, loading: false));
32     });
33   }
34
35   @override
36   Future<void> close() {
37     _sub?.cancel();
38     return super.close();
39   }
40 }

```

El soporte de mensajería y persistencia se completa con *notifications_push_service.dart*, *fcm_remote_datasource.dart* y *orderNotifications.ts*, mostrados en las figuras 134, 135 y 136. De esta forma, la historia vincula la recepción del aviso con el registro del token y la generación del evento en *backend*.

Figura 134. Código de *notifications_push_service.dart*: *listeners* de mensajería y apertura.

```
lib > modules > features > notifications_push > presentation > notifications_push_service.dart > NotificationsPushService > initialize
16 class NotificationsPushService {
54   Future<void> initialize() async {
100     _tokenSub = _messaging.onTokenRefresh.listen((token) async {
101       _currentToken = token;
102       final uid = _currentUserId;
103       if (uid != null) {
104         await _saveTokenForUser(uid, token);
105       } else {
106         debugPrint('[PUSH] token refresh ignored (signed out)');
107       }
108     });
109     FirebaseMessaging.onMessage.listen(
110       (message) => unawaited(_handleForegroundMessage(message)),
111     );
112     FirebaseMessaging.onMessageOpenedApp.listen(
113       (message) => unawaited(_deepLinkHandler.handleRemoteMessage(message)),
114     );
115     final initial = await _messaging.getInitialMessage();
116     if (initial != null) {
117       await _deepLinkHandler.handleRemoteMessage(initial);
118     }
119     debugPrint('[PUSH] init_success');
120   }
121 }
```

Figura 135. Código de *fcm_remote_datasource.dart*

```
lib > modules > features > notifications_push > data > datasources > fcm_remote_datasource.dart > FcmRemoteDataSource
You, 3 weeks ago | I author (You)
1 import 'package:cloud_firestore/cloud_firestore.dart';
2
3 import '../domain/entities/fcm_token_entity.dart';
4
5 You, 3 weeks ago | I author (You)
5 class FcmRemoteDataSource {
6   final FirebaseFirestore _firestore;
7   FcmRemoteDataSource(this._firestore);
8
9   CollectionReference<Map<String, dynamic>> _collection(String userId) {
10     return _firestore
11       .collection('users')
12       .doc(userId)
13       .collection('fcm_tokens')
14       .withConverter<Map<String, dynamic>>({
15         fromFirestore: (snap, _) => snap.data() ?? <String, dynamic>{},
16         toFirestore: (map, _) => map,
17       });
18   }
19
20   Future<void> saveToken({
21     required String userId,
22     required FcmTokenEntity token,
23   }) async {
24     final ref = _collection(userId).doc(token.token);
25     final snap = await ref.get();
26     final base = <String, dynamic>{
27       'token': token.token,
28       'platform': token.platform,
29       'isActive': token.isActive,
30       'updatedAt': FieldValue.serverTimestamp(),
31       if (token.deviceInfo != null) 'deviceInfo': token.deviceInfo,
32     };
33
34     if (snap.exists) {
35       await ref.set(base, SetOptions(merge: true));
36     } else {
37       await ref.set({
38         ...base,
39         'createdAt': FieldValue.serverTimestamp(),
40       });
41     }
42   }
43 }
```

La fuente remota materializa la interacción con el *backend* en Notificaciones *push*. Materializa mediante una función *notifyAdminsOnOrderCreated* (disparador por creación de pedido).

Figura 136. Código de *orderNotifications.ts*

```

46 import { const notifyAdminsOnOrderCreated = onDocumentCreated(
47   {
48     region: "us-central1",
49     document: "orders/{orderId}",
50   },
51   async (event) => {
52     try {
53       const snap = event.data;
54       if (!snap) return;
55
56       const data = snap.data() || {};
57       const orderId = (event.params.orderId ?? "").toString();
58       const shortId = formatShortOrderId(orderId);
59       const title = `Nuevo pedido ${shortId}`;
60       const body = buildAdminNewOrderBody(data);
61
62       const draft: NotificationDraft = {
63         title,
64         body,
65         type: "new_order_admin",
66         priority: "high",
67         target: { targetType: "admin_orders", orderId },
68         dedupKey: `new_order_admin:${orderId}`,
69         orderId,
70       };
71
72       const adminUids = await fetchAdminUids();
73       if (adminUids.length === 0) return;
74
75       await Promise.all(
76         adminUids.map((uid) => dispatchNotificationToUser(uid, draft)),
77       );

```

4.3.4.1.3. Historia de usuario 20: Consulta de pedidos por fecha (cliente)

Tal como se detalla en el anexo 4, la consulta de pedidos por fecha para cliente se construye sobre el historial y la hoja de filtros asociada. Las figuras 137, 138 y 139 muestran el listado de pedidos y la selección del período, lo que facilita recuperar órdenes dentro de un rango temporal concreto.

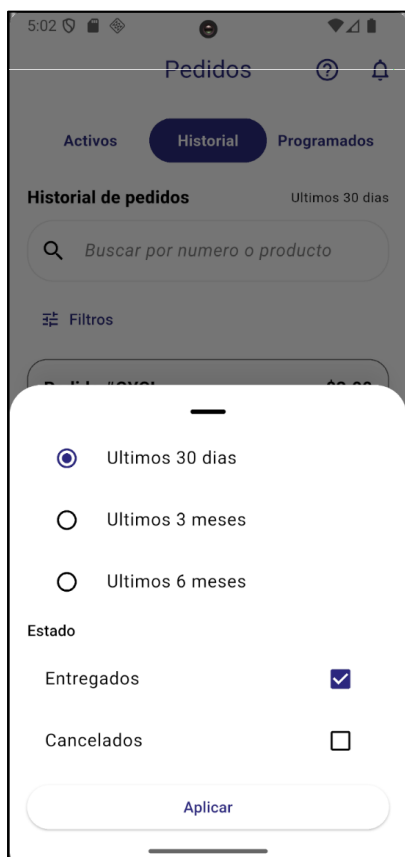
Figura 137. Interfaz móvil de *MyOrdersPage* (ruta */orders*), vista de los pedidos activos.



Figura 138. Interfaz móvil del flujo de filtros por fecha/estado desde *MyOrdersPage* (estado previo a aplicar filtros).



Figura 139. Interfaz móvil de *MyOrdersHistoryFiltersSheet*, selección de período/rango.



La aplicación de estos criterios se apoya en *my_orders_history_filters_sheet.dart*, *my_orders_cubit.dart*, *my_orders_page.dart* y *get_my_order_history.dart*, desarrollados en las figuras 140, 141, 142 y 143. En conjunto, estos componentes ordenan la selección de filtros, su envío y la obtención del historial resultante.

Figura 140. Código de *my_orders_history_filters_sheet.dart*: sheet de filtros por período/estado.

```
lib > modules > features > orders > presentation > widgets > my_orders_history_filters_sheet.dart > % MyOrdersHistoryFiltersSheetState
22 class MyOrdersHistoryFiltersSheetState {
23   // ...
24   @override
25   Widget build(BuildContext context) {
26     final hasChanges = !timeFilters();
27     return Scaffold(
28       child: ListView(
29         padding: const EdgeInsets.fromLTRB(16, 8, 16, 24),
30         children: [
31           Row(
32             children: [
33               Expanded(
34                 child: Text(
35                   'Filtros por',
36                   style: Theme.of(context).textTheme.titleMedium,
37                   textTheme:
38                     TextTheme(
39                       titleMedium:
40                         TextStyle(
41                           color: Colors.grey,
42                           fontWeight: FontWeight.w400,
43                         ),
44                     ),
45                 ),
46               Expanded(
47                 child: Text(
48                   'Período',
49                   style: Theme.of(context).textTheme.titleMedium,
50                   textTheme:
51                     TextTheme(
52                       titleMedium:
53                         TextStyle(
54                           color: Colors.grey,
55                           fontWeight: FontWeight.w400,
56                         ),
57                     ),
58                 ),
59               Expanded(
60                 child: Text(
61                   'Estado',
62                   style: Theme.of(context).textTheme.titleMedium,
63                   textTheme:
64                     TextTheme(
65                       titleMedium:
66                         TextStyle(
67                           color: Colors.grey,
68                           fontWeight: FontWeight.w400,
69                         ),
70                     ),
71                 ),
72             ],
73           ),
74           const SizedBox(height: 8),
75           Text(
76             'Período',
77             style: Theme.of(context).textTheme.titleMedium,
78             textTheme:
79               TextTheme(
80                 titleMedium:
81                   TextStyle(
82                     color: Colors.grey,
83                     fontWeight: FontWeight.w400,
84                   ),
85               ),
86           ),
87           const SizedBox(height: 8),
88           RadioGroup<int>Period(
89             groupValue: period,
90             onChanged: (value) {
91               if (value == null) return;
92               setState(() => _period = value);
93             },
94           ),
95           child: Column(
96             children: [
97               for (final period in HistoryPeriod.values)
98                 RadioListTile<int>HistoryPeriod(
99                   value: period,
100                   groupValue: period,
101                   onChanged: (value) {
102                     if (value == null) return;
103                     setState(() => _period = value);
104                   },
105                 ),
106             ],
107           ),
108         ],
109       ),
110     );
111   }
112 }
```

Figura 141. Código de *my_orders_cubit.dart*: aplicación de filtros por fecha.

```
lib > modules > features > orders > presentation > blocs > my_orders_cubit.dart > % MyOrdersCubit
176 class MyOrdersCubit extends Cubit<MyOrdersState> {
177   // ...
178   Future<void> applyHistoryFilters(HistoryFilters filters) async {
179     // You, last month + Refactor: restructure orders feature
180     if (isClosed) return;
181     if (_sameHistoryFilters(filters, state.historyFilters)) return;
182     print(
183       '[MY_ORDERS] date_filter_changed period=${filters.period} statuses=${filters.statuses}',
184     );
185     emit(state.copyWith(historyFilters: filters));
186     await loadHistory(refresh: true);
187   }
188   Future<void> clearHistoryFilters() async {
189     if (isClosed) return;
190     print('[MY_ORDERS] clear_filters');
191     emit(state.copyWith(historyFilters: HistoryFilters.initial()));
192     await loadHistory(refresh: true);
193   }
194 }
```

Figura 142. Código de *my_orders_page.dart*: vínculo entre UI y acciones del cubit.

```
lib > modules > features > orders > presentation > pages > my_orders_page.dart > % MyOrdersPage > build
23 class MyOrdersPage extends StatelessWidget {
24   // ...
25   Widget build(BuildContext context) {
26     // ...
27     onPressed: () => Navigator.pushNamed(context, OrdersRoutes.help),
28     // ...
29     child: Column(
30       children: [
31         Row(
32           children: [
33             Expanded(
34               child: OrdersSegmentedControl(
35                 selected: state.tab,
36                 onChanged: (tab) =>
37                   context.read<MyOrdersCubit>().selectTab(tab),
38               ),
39             Expanded(
40               child: AnimatedSwitcher(
41                 duration: const Duration(milliseconds: 220),
42                 child: Builder<BuildContext, State> (context, state) {
43                   // ...
44                 },
45               ),
46             ),
47           ],
48         ),
49         Column(
50           children: [
51             // ...
52             // Scaffold
53           ],
54         ),
55       ],
56     ),
57   }
58   Widget _buildTab(BuildContext context, MyOrdersState state) {
59     switch (state.tab) {
60       case MyOrdersTab.active:
61         return _ActiveOrdersTab(
62           key: const ValueKey('active'),
63         );
64       // ...
65     }
66   }
67 }
```

Por último, la orquestación del flujo se realiza con la lógica de estado en Consulta de pedidos por fecha cliente.

Figura 143. Código de *get_my_order_history.dart*: caso de uso de historial.

```
lib > modules > features > orders > domain > usecases > get_my_order_history.dart > GetMyOrderHistory

You last month | 1 author (You)
1 import '.../core/entities/order_status_consts.dart';
2 import '.../repositories/orders_repository.dart';
3 import 'get_my_orders.dart';
4
5 You last month | 1 author (You)
6 class GetMyOrderHistory {
7   final GetMyOrders _getMyOrders;
8   GetMyOrderHistory(this._getMyOrders);
9
10  Future<MyOrdersPage> call({
11    required String userId,
12    String? phone,
13    DateTime? dateFrom,
14    DateTime? dateTo,
15    List<String>? statusFilter,
16    int limit = 20,
17    DateTime? startAfterAt,
18    String? startAfterId,
19  }) {
20    return _getMyOrders(
21      userId: userId,
22      phone: phone,
23      dateFrom: dateFrom,
24      dateTo: dateTo,
25      statusFilter:
26        statusFilter ??
27        const [
28          OrderStatusConst.entregado,
29          OrderStatusConst.cancelado,
30        ],
31      sortField: MyOrdersSortField.createdAt,
32      descending: true,
33      limit: limit,
34      startAfterAt: startAfterAt,
35      startAfterId: startAfterId
36    );
37  }
38}
```

4.3.4.1.4. Historia de usuario 21: Consulta de pedidos por fecha (administrador)

En el anexo 4, la consulta de pedidos por fecha para administración amplía la revisión operativa de la bandeja al incorporar un rango temporal de búsqueda. La figura 144 sitúa esta funcionalidad dentro del espacio administrativo desde el que se controlan los pedidos.

Figura 144. Interfaz móvil de pedidos *admin* con filtro por fecha



El soporte técnico se distribuye entre *orders_filters_bar.dart*, *admin_orders_list_cubit.dart* y *admin_orders_firestore_datasource.dart*, visibles en las figuras 145, 146 y 147. Este bloque articula la selección del rango, la aplicación de filtros y la consulta paginada de los resultados.

Figura 145. Código de *orders_filters_bar.dart*

```

183 class OrdersFiltersSheetState extends State<OrdersFiltersSheet> {
184   Future<DateTimeRange?> pickCustomRange() {
185     // You, last month + Refactor order list controls and filters; Improv...
186     BuildContext context,
187     DateTimeRange? initial,
188   ) async {
189     final now = DateTime.now();
190     final picked = await showDatePicker(
191       context: context,
192       firstDate: DateTime(now.year - 1),
193       lastDate: DateTime(now.year + 1),
194       initialDateRange: initial,
195     );
196     if (picked == null) return null;
197     return DateTimeRange(
198       start: startOfDay(picked.start),
199       end: endOfDay(picked.end),
200     );
201   }
202
203   String _rangeLabel(DateTimeRange range) {
204     final start =
205       '${range.start.day.toString().padLeft(2, '0')}/${range.start.month.toString().padLeft(2, '0')}';
206     final end =
207       '${range.end.day.toString().padLeft(2, '0')}/${range.end.month.toString().padLeft(2, '0')}';
208     return '$start - $end';
209   }
210 }

```

Figura 146. Código de *admin_orders_list_cubit.dart*

```

90 class AdminOrdersListCubit extends Cubit<AdminOrdersListState> {
156   Future<void> applyFilters({
157     required DateTime? dateFrom,
158     required DateTime? dateTo,
159     required DeliveryMethod? deliveryMethod,
160   }) async {
161     if (isClosed) return;
162     final sameDates = state.dateFrom == dateFrom && state.dateTo == dateTo;
163     final sameMethod = state.deliveryMethod == deliveryMethod;
164     if (sameDates && sameMethod) return;
165
166     print(
167       '[ADMIN_ORDERS] filters_changed date=$dateFrom..$dateTo delivery=$deliveryMethod',
168     );
169     emit(
170       state.copyWith(
171         dateFrom: dateFrom,
172         dateTo: dateTo,
173         deliveryMethod: deliveryMethod,
174         loading: true,
175         loadingMore: false,
176         lastCreatedAt: null,
177         lastId: null,
178         hasMore: true,
179       ),
180     );
181     await _fetch(reset: true);
182   }

```

Figura 147. Código de *admin_orders_firestore_datasource.dart*

```

23 class AdminOrdersFirestoreDataSource {
24   Future<AdminOrdersFetchResult> fetchOrders({
36     Query<Map<String, dynamic>> q = _orders // You, last month + feat: Enhance order management with admin notes...
37     .orderBy('createdAt', descending: true)
38     .orderBy(FieldPath.documentId);
39
40     if (query.statuses != null && query.statuses.isNotEmpty) {
41       q = q.where('status', whereIn: query.statuses);
42     }
43     if (query.paymentStatuses != null && query.paymentStatuses.isNotEmpty) {
44       q = q.where('paymentStatus', whereIn: query.paymentStatuses);
45     }
46     if (query.deliveryMethod != null) {
47       q = q.where('deliveryMethod', isEqualTo: query.deliveryMethod.name);
48     }
49     if (query.dateFrom != null) {
50       q = q.where('createdAt', isGreaterThanOrEqualTo: Timestamp.fromDate(query.dateFrom));
51     }
52     if (query.dateTo != null) {
53       q = q.where('createdAt', isLessThanOrEqualTo: Timestamp.fromDate(query.dateTo));
54     }
55     if (query.searchPhoneExact != null && query.searchPhoneExact!.isNotEmpty) {
56       q = q.where('recipientPhone', isEqualTo: query.searchPhoneExact);
57     }
58     if (startAfterCreatedAt != null && startAfterId != null) {
59       q = q.startAfter(
60         Timestamp.fromDate(startAfterCreatedAt),
61         startAfterId,
62       );
63     }

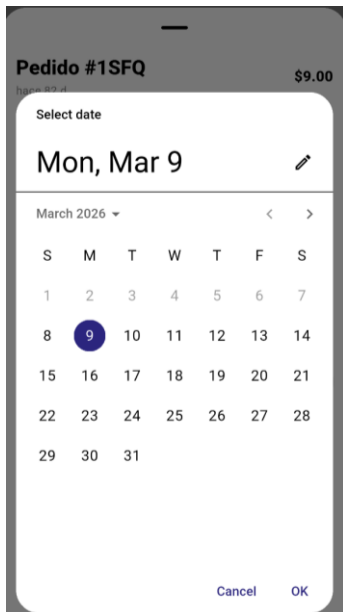
```

El *datasource* resuelve la ejecución de cumplir la tarea principal de la historia y sostener el flujo del proyecto en la capa de datos de consulta de pedidos por fecha administrador.

4.3.4.1.5. Historia de usuario 22: Agendamiento de entregas

Tal como se recoge en el anexo 4, el agendamiento de entregas se resuelve desde el detalle administrativo del pedido mediante la selección de una fecha programada. La figura 148 muestra este punto de interacción y lo integra al seguimiento operativo de la orden.

Figura 148. Interfaz móvil de agendamiento de entregas



La programación se articula a través del caso de uso *schedule_order.dart* y del bloque de detalle administrativo, presentados en las figuras 149 y 150. Con ello, la decisión tomada en la interfaz queda conectada con la acción que formaliza el cambio de agenda.

Figura 149. Código de *schedule_order.dart*

```
lib > modules > features > admin_orders > domain > usecases > schedule_order.dart > UpdateOrderScheduleUseCase
You, last month | 1 author (You)
1 import '../repositories/admin_orders_repository.dart';
2
3 You, last month | 1 author (You)
4 class UpdateOrderScheduleUseCase { You, last month * feat: Enhance order management
5   final AdminOrdersRepository repo;
6   UpdateOrderScheduleUseCase(this.repo);
7
8   Future<void> call({
9     required String orderId,
10    required DateTime scheduledAt,
11  }) {
12    return repo.updateSchedule(
13      orderId: orderId,
14      scheduledAt: scheduledAt,
15    );
16  }
```

Figura 150. Código de *admin_order_detail_bloc.dart*

```
lib > modules > features > admin_orders > presentation > blocs > admin_order_detail_bloc.dart >
103 class AdminOrderScheduleTapped extends AdminOrderDetailEvent {
104   final DateTime scheduledAt;
105   AdminOrderScheduleTapped(this.scheduledAt);
106 }
107
```

La persistencia del agendamiento y la emisión del evento asociado se observan en las figuras 151 y 152, correspondientes a *admin_orders_firestore_datasource.dart* y *orderNotifications.ts*. Así, la fecha programada no solo se registra en el pedido, sino que también puede reflejarse en avisos posteriores.

Figura 151. Código de *admin_orders_firestore_datasource.dart*: *updateSchedule* en *orders/{orderId}*.

```
lib > modules > features > admin_orders > data > datasources > admin_orders_firestore_datasource.dart >
23 class AdminOrdersFirestoreDataSource {
144 Future<void> updateSchedule({
145   required String orderId,
146   required DateTime scheduledAt,
147 }) async {
148   final patch = <String, dynamic>{
149     'scheduledAt': Timestamp.fromDate(scheduledAt),
150     'updatedAt': FieldValue.serverTimestamp(),
151   };
152   _logPatch('update_schedule', orderId, patch);
153   await _orders.doc(orderId).update(patch);
154 }

```

Figura 152. Código de *functions/src/functions/notifications/orderNotifications.ts*, emisión de evento tipo *schedule_update* ante cambios en *scheduledAt*.

```
functions > src > functions > notifications > orderNotifications.ts > buildOrderUpdateDrafts
331 function buildOrderUpdateDrafts(
375   const prevScheduled = asTimestamp(before["scheduledAt"]);
376   const nextScheduled = asTimestamp(after["scheduledAt"]);
377   if (
378     prevScheduled?.toMillis() !== nextScheduled?.toMillis() &&
379     nextScheduled
380   ) {
381     const scheduledIso = nextScheduled.toDate().toISOString();
382     drafts.push({
383       title: `Pedido ${formatShortOrderId(orderId)} reprogramado`,
384       body: `Nueva fecha: ${scheduledIso}`,
385       type: "schedule_update",
386       priority: "normal",
387       target: { targetType: "scheduled_orders", orderId },
388       dedupeKey: `schedule_update:${orderId}:${scheduledIso}`,
389       orderId,
390       createdAt,
391     });
392   }

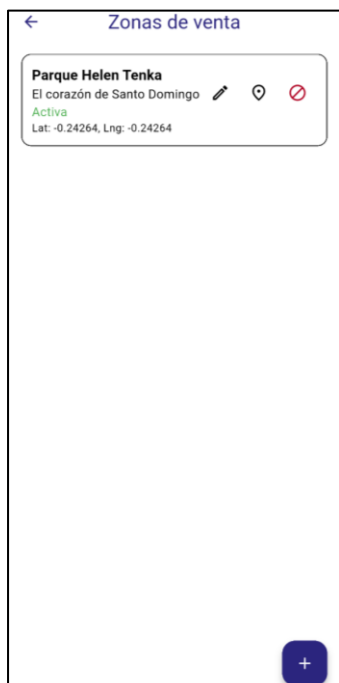
```

Esta implementación conecta un *endpoint callable* asociado a Agendamiento de entregas. Ejecuta programar entregas y reflejarlo en listas y avisos con control de intentos y validación de contexto, reduciendo exposición de reglas sensibles en la aplicación.

4.3.4.1.6. Historia de usuario 23: Gestión de zonas de venta

El anexo 4 desarrolla la gestión de zonas de venta como un módulo administrativo orientado a organizar áreas de cobertura y asignaciones operativas. La figura 153 presenta la vista general de este espacio y sitúa las acciones principales dentro del mismo entorno.

Figura 153. Interfaz móvil de gestión de zonas de venta



La estructura del listado, la administración del estado y las operaciones de persistencia se distribuyen entre *admin_sales_zones_page.dart*, *sales_zones_cubit.dart* y *sales_zones_remote_datasource.dart*, según muestran las figuras 154, 155 y 156. Este bloque concentra la operación cotidiana del módulo sin trasladarla a varias pantallas aisladas.

Figura 154. Código de *admin_sales_zones_page.dart*: estructura de listado.

```

183 class _ZoneTile extends StatelessWidget { // TODO: last month + test: integrate Google Maps and implement actions
184   final SalesZone zone;
185   final VoidCallback onEdit;
186   final VoidCallback onAssign;
187   final VoidCallback onDeactivate;
188
189   const _ZoneTile({
190     required this.zone,
191     required this.onEdit,
192     required this.onAssign,
193     this.onDeactivate,
194   });
195
196   @override
197   Widget build(BuildContext context) {
198     final theme = Theme.of(context);
199     final scheme = theme.colorScheme;
200     return Container(
201       padding: const EdgeInsets.all(12),
202       decoration: BoxDecoration(
203         color: scheme.surface,
204         borderRadius: BorderRadius.circular(12),
205         border: Border.all(color: scheme.outlineVariant),
206       ), // BoxDecoration
207       child: Column(
208         crossAxisAlignment: CrossAxisAlignment.start,
209         children: [
210           Row(
211             children: [
212               Expanded(
213                 child: Column(
214                   crossAxisAlignment: CrossAxisAlignment.start,
215                   children: [
216                     Text(
217                       zone.name,
218                       style: theme.textTheme.titleMedium?.copyWith(
219                         fontWeight: FontWeight.w700,
220                       ),
221                     ),

```

En `admin_sales_zones_page.dart` se organiza la pantalla usada por Gestión de zonas de venta para administrar zonas y asignaciones operativas. Su aporte es mantener el módulo estable, mejorar legibilidad del código y facilitar la integración con el resto de las funcionalidades.

Figura 155. Código de `sales_zones_cubit.dart`: *load/save/assign/deactivate*.

```
lib > modules > features > sales_zones > presentation > bitoc > sales_zones_cubit.dart > % SalesZonesCubit
43 class SalesZonesCubit extends Cubit<SalesZonesState> {
60 Future<void> load() async {
61   emit(state.copyWith(loading: true, error: null, info: null));
62   try {
63     final zones = await _getZones();
64     emit(SalesZonesState(loading: false, zones: zones));
65   } catch (e) {
66     emit(
67       state.copyWith(
68         loading: false,
69         error: FirebaseException.friendly(
70           e,
71           generic: 'no se pudieron cargar las zonas',
72         ),
73       ),
74     );
75   }
76 }
77
78 Future<void> save({
79   String? id,
80   required String name,
81   String? description,
82   LocationPoint? center,
83   bool isActive = true,
84 }) async {
85   if (name.trim().isEmpty) {
86     emit(state.copyWith(error: 'el nombre es obligatorio'));
87     return;
88   }
89   emit(state.copyWith(saving: true, error: null, info: null));
90   try {
91     final zone = SalesZone(
92       id: id ?? '',
93       name: name.trim(),
94     );
95   }
```

Figura 156. Código de `sales_zones_remote_datasource.dart`: operaciones *CRUD* Firestore.

```
lib > modules > features > sales_zones > data > datasources > sales_zones_remote_datasource.dart > % SalesZonesRemoteDataSource
You, last month | 1 author (You)
1 import 'package:cloud_firestore/cloud_firestore.dart';
2
3 import '../..//core/config/collections.dart';
4 import '../models/sales_zone_model.dart';
5
6 class SalesZonesRemoteDataSource {
7   final FirebaseFirestore _firestore;
8   SalesZonesRemoteDataSource(this._firestore);
9
10  Future<List<SalesZoneModel>> fetchAll() async {
11    You, last month < feat: integrate google maps and implement entre...
12    final snap = await _firestore.collection(col.salesZones).get();
13    return snap.docs.map(SalesZoneModel.fromDoc).toList(growable: false);
14  }
15
16  Future<SalesZoneModel> create(SalesZoneModel model) async {
17    final ref = await _firestore.collection(col.salesZones).doc(model.toUp());
18    final saved = await ref.get();
19    return SalesZoneModel.fromDoc(saved);
20  }
21
22  Future<SalesZoneModel> update(SalesZoneModel model) async {
23    final ref = _firestore.collection(col.salesZones).doc(model.id);
24    await ref.set(model.toUp(), SetOptions(merge: true));
25    final saved = await ref.get();
26    return SalesZoneModel.fromDoc(saved);
27  }
28 }
```

Las figuras 157 y 158 completan la historia desde los casos de uso que resuelven la asignación y el guardado principal de la zona. Con ello, la administración de cobertura mantiene una base funcional diferenciada de la vista.

Figura 157. Código del caso de uso de `assign_zone_to_entrepreneur.dart`

```
lib > modules > features > sales_zones > domain > usecases > assign_zone_to_entrepreneur.dart > ...
You, last month | 1 author (You)
1 import '../repositories/sales_zone_repository.dart';
2
3 You, last month | 1 author (You)
4 class AssignZoneToEntrepreneur {
5   final SalesZoneRepository repository;
6   AssignZoneToEntrepreneur(this.repository);
7
8   Future<void> call({
9     required String entrepreneurId,
10    required String zoneId,
11  }) {
12    return repository.assignZoneToEntrepreneur(
13      entrepreneurId: entrepreneurId,
14      zoneId: zoneId,
15    );
16  }
17 }
```


Figura 158. Código del caso de uso de `save_sales_zone.dart`

```

lib > modules > features > sales_zones > domain > usecases > save_sales_zone.dart > ...
You, last month | 1 author (You)
1 import '../entities/sales_zone.dart';
2 import '../repositories/sales_zone_repository.dart';
3
4 You, last month | 1 author (You)
5 class SaveSalesZone {
6   final SalesZoneRepository repository;
7   SaveSalesZone(this.repository);
8
9   Future<SalesZone> call(SalesZone zone) {
10    if (zone.id.isEmpty) return repository.create(zone);
11    return repository.update(zone);
12  }

```

4.3.4.1.7. Historia de usuario 24: Solicitud de nueva emprendedora del proyecto

Tal como se presenta en el anexo 4, la solicitud de nueva emprendedora se organiza desde una puerta de acceso informativa, el formulario de postulación y la confirmación posterior al envío. Las figuras 159, 160 y 161 permiten seguir este recorrido y ubicar la solicitud como un flujo propio dentro de la aplicación.

Figura 159. Interfaz móvil de acceso a la solicitud de emprendedora

← Solicitud emprendedora

Proyecto 100 Mujeres

Envía tu solicitud para ser emprendedora del proyecto. No afecta tu cuenta actual.

Aun no envías una solicitud.
Completa el formulario para enviarla.

[Crear solicitud](#)

Figura 160. Interfaz móvil del formulario de solicitud de emprendedora

← Solicitud emprendedora

Datos personales

Documento y tipo se mantienen iguales a tu cuenta. Para cambiarlos, solicita soporte.

Nombre completo

Carlita

Sincronizado desde tu cuenta

Tipo de documento

Cédula

Cédula / documento

1722846688

Teléfono

0967167547

Sincronizado desde tu cuenta

Correo de contacto (opcional)

Dirección personal (opcional)

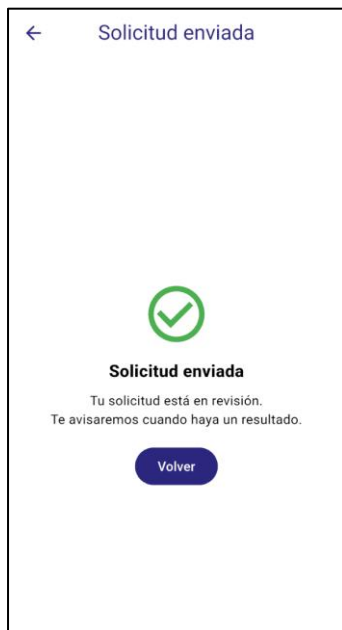
Urb Yanuncay

Sincronizada desde tu cuenta

Nombre del negocio (opcional)

Datos del punto de venta (desde tu ubicación registrada)

Figura 161. Interfaz móvil de confirmación de solicitud enviada



La captura de adjuntos, las validaciones y el envío se sostienen en *entrepreneur_request_form_page.dart* y *entrepreneur_request_form_cubit.dart*, visibles en las figuras 162 y 163. Este bloque mantiene ordenada la información ingresada antes de formalizar la postulación.

Figura 162. Código de *entrepreneur_request_form_page.dart*: para la selección y manejo de adjuntos.

```

123 class EntrepreneurRequestFormPageState
124 widget build(BuildContext context) {
125     label: const Text('Agregar documento'),
126     // ElevatedButton.icon
127     const SizedBox(width: 8),
128     Text('${state.documents.length} adjuntos'),
129   ],
130   // Row
131   const SizedBox(height: 8),
132   _AttachmentsList(
133     items: state.documents,
134     onRemove:
135       (item) =>
136         context
137           .read<EntrepreneurRequestFormCubit>()
138             .removeDocument(item),
139   ), // _AttachmentsList
140   const SizedBox(height: 16),
141   const Text(
142     'Fotos del punto de venta (máximo 1)',
143     style: TextStyle(fontWeight: FontWeight.w700, fontSize: 16),
144   ), // Text
145   const SizedBox(height: 8),
146   Row(
147     children: [
148       ElevatedButton.icon(
149         onPressed: uploading ? null : _addStallPhoto,
150         icon: const Icon(Icons.photo_camera_outlined),
151         label: const Text('Agregar foto'),
152       ), // ElevatedButton.icon
153       const SizedBox(width: 8),
154       Text('${state.stallPhotos.length} fotos'),
155     ],
156   ), // Row
157   const SizedBox(height: 8),
158   _AttachmentsList(
159     items: state.stallPhotos,
160     onRemove:
161       (item) =>

```

Figura 163. Código de *entrepreneur_request_form_cubit.dart*: validaciones y *submit()*.

```
lib > modules > features > entrepreneur_requests > presentation > blocs > entrepreneur_request_form_cubit.dart > EntrepreneurRequestFormCubit
119 Future<EntrepreneurRequest> submit({
120   required ApplicantInfo applicant,
121   String? notes,
122 }) async {
123   final uid = userId;
124   if (uid == null || uid.isEmpty) {
125     throw Exception('Debes iniciar sesion para enviar la solicitud');
126   }
127   emit(state.copyWith(submitting: true, error: null));
128   try {
129     final res = await submitRequest(
130       userId: uid,
131       applicant: applicant,
132       documents: state.documents,
133       stallPhotos: state.stallPhotos,
134       notes: notes,
135     );
136     emit(state.copyWith(submitting: false));
137     return res;
138   } on PendingRequestExistsException {
139     emit(state.copyWith(submitting: false));
140     rethrow;
141   } on MissingStallPhotoException catch (e) {
142     emit(state.copyWith(submitting: false, error: e.toString()));
143     rethrow;
144   } catch (e) {
145     emit(state.copyWith(submitting: false, error: e.toString()));
146     rethrow;
147   }
148 }
```

El respaldo funcional se completa con *submit_entrepreneur_request.dart*, *entrepreneur_requests_remote_datasource.dart* y *entrepreneur_request_repository_impl.dart*, presentados en las figuras 164, 165 y 166. Así, la solicitud queda articulada entre reglas de envío, persistencia y coordinación de fuentes de datos.

Figura 164. Código de *submit_entrepreneur_request.dart*: reglas de envío.

```
lib > modules > features > entrepreneur_requests > domain > usecases > submit_entrepreneur_request.dart > SubmitEntrepreneurRequest
You last month | author (You)
1 import '../entities/applicant_info.dart';
2 import '../entities/entrepreneur_request.dart';
3 import '../entities/request_attachment.dart';
4 import '../errors/entrepreneur_request_errors.dart';
5 import '../repositories/entrepreneur_request_repository.dart';
6
7 class SubmitEntrepreneurRequest {
8   final EntrepreneurRequestRepository repository;
9   SubmitEntrepreneurRequest(this.repository);
10
11   Future<EntrepreneurRequest> call({
12     required String userId,
13     required ApplicantInfo applicant,
14     required List<RequestAttachment> documents,
15     required List<RequestAttachment> stallPhotos,
16     String? notes,
17   }) async {
18     if (stallPhotos.isEmpty) throw MissingStallPhotoException();
19
20     final pending = await repository.findPendingForUser(userId);
21     if (pending != null &&
22         pending.status == EntrepreneurRequestStatus.pending) {
23       throw PendingRequestExistsException(pending);
24     }
25
26     final entity = EntrepreneurRequest(
27       id: '',
28       createdById: userId,
29       applicant: applicant,
30       documents: documents,
31       stallPhotos: stallPhotos,
32       status: EntrepreneurRequestStatus.pending,
33       notes: notes,
34     );
35     return repository.createRequest(entity);
36   }
37 }
```

Figura 165. Código de *entrepreneur_requests_remote_datasource.dart*: consulta/creación en *Firestore*.

```
lib > modules > features > entrepreneur_requests > data > datasources > entrepreneur_requests_remote_datasource.dart > EntrepreneurRequestsRemoteDataSource
6 class EntrepreneurRequestsRemoteDataSource {
34 Future<EntrepreneurRequestModel?> findPendingForUser(String uid) async {
35     final snap =
36         await _col
37             .where('createdByUid', isEqualTo: uid)
38             .where('status', isEqualTo: 'pending')
39             .limit(1)
40             .get();
41     if (snap.docs.isEmpty) return null;
42     return EntrepreneurRequestModel.fromDoc(snap.docs.first);
43 }
44
45 Future<EntrepreneurRequestModel> createRequest(
46     EntrepreneurRequestModel model,
47 ) async {
48     final now = FieldValue.serverTimestamp();
49     final doc = await _col.add({
50         ...model.toMap(),
51         'createdAt': now,
52         'updatedAt': now,
53     });
54     final snap = await doc.get();
55     return EntrepreneurRequestModel.fromDoc(snap);
56 }
57 }
```

Figura 166. Código de *entrepreneur_request_repository_impl.dart*: coordinación de *datasources*.

```
lib > modules > features > entrepreneur_requests > data > repositories > entrepreneur_request_repository_impl.dart > EntrepreneurRequestRepositoryImpl
12 class EntrepreneurRequestRepositoryImpl
13 implements EntrepreneurRequestRepository {
14     final EntrepreneurRequestsRemoteDataSource remote;
15     final EntrepreneurRequestsMediaDataSource media;
16
17     EntrepreneurRequestRepositoryImpl(
18         required this.remote,
19         required this.media,
20     );
21
22     @override
23     Future<EntrepreneurRequest?> fetchLatestForUser(String uid) {
24         return remote.fetchLatestForUser(uid);
25     }
26
27     @override
28     Future<EntrepreneurRequest?> findPendingForUser(String uid) {
29         return remote.findPendingForUser(uid);
30     }
31
32     @override
33     Future<EntrepreneurRequest?> createRequest(EntrepreneurRequest request) {
34         final model = _toModel(request);
35         return remote.createRequest(model);
36     }
37
38     @override
39     Future<RequestAttachment> uploadFile(
40         required String ownerId,
41         required Uint8List bytes,
42         required String contentType,
43         String? type,
44         bool isStillPhoto = false,
45     ) {
46         return media.upload(
47             ownerId: ownerId,
48             bytes: bytes,
49             contentType: contentType,
50             type: type,
51             isStillPhoto: isStillPhoto,
52         );
53     }
54 }
```

4.3.4.1.8. Historia de usuario 25: Registro de acuerdos privados de ubicación

El anexo 4 integra el registro de acuerdos privados de ubicación como un módulo orientado a la administración documental y al seguimiento de su estado. Las figuras 167, 168 y 169 muestran el listado de acuerdos, su detalle y el formulario desde el cual se registran o actualizan los datos relevantes.

Figura 167. Interfaz móvil del listado de acuerdos de ubicación

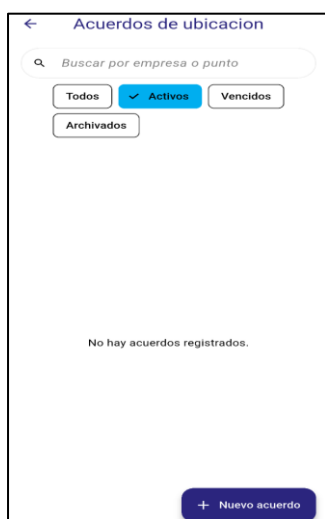


Figura 168. Interfaz móvil del detalle del acuerdo de ubicación

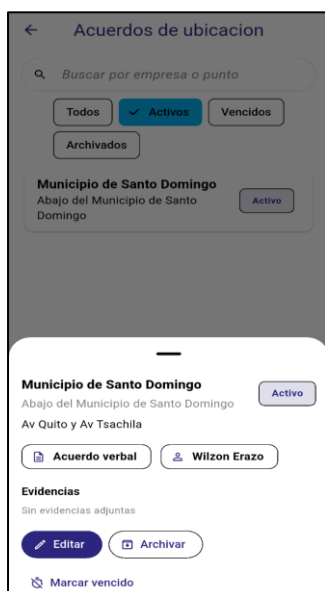


Figura 169. Interfaz móvil del formulario de acuerdo de ubicación

La organización de la pantalla y el control del formulario se apoyan en *admin_location_agreements_page.dart* y *location_agreement_form_cubit.dart*, visibles en las figuras 170 y 171. En este bloque se concentra la operatividad del módulo y la validación previa al guardado.

Figura 170. Código de *admin_location_agreements_page.dart*: *AppBar*, filtros y carga inicial.

```

lib > modules > features > location_agreements > presentation > pages > admin_location_agreements_page.dart > AdminLocationAgreementsPageState
30 class AdminLocationAgreementsPageState
31 {
32   @override
33   Widget build(BuildContext context) {
34     return Scaffold(
35       appBar: AppBar(
36         title: const Text('Acuerdos de ubicación'),
37       ), // AppBar
38       floatingActionButton: FloatingActionButton.extended(
39         onPressed: _openCreateForm,
40         icon: const Icon(Icons.add),
41         label: const Text('Nuevo acuerdo'),
42       ), // FloatingActionButton.extended
43       body: BlocConsumer<AdminLocationAgreementsCubit,
44         AdminLocationAgreementsState>(
45         listener: (p, c) => p.error != c.error,
46         listener: (context, state) {
47           if (state.error != null) {
48             ScaffoldMessenger.of(context).showSnackBar(
49               SnackBar(content: Text(state.error!)),
50             );
51           }
52         },
53         builder: (context, state) {
54           final items = state.filtered;
55           return Column(
56             children: [
57               Padding(
58                 padding: const EdgeInsets.fromLTRB(16, 12, 16, 6),
59                 child: TextField(
60                   controller: _searchCtrl,
61                   decoration: InputDecoration(
62                     prefixIcon: const Icon(Icons.search),
63                     hintText: 'Buscar por empresa o punto',
64                     suffixIcon: _searchCtrl.text.isEmpty
65                       ? null

```

Figura 171. Código de *location_agreement_form_cubit.dart*

```

lib > modules > features > location_agreements > presentation > blocs > location_agreement_form_cubit.dart > LocationAgreementFormCubit
43 class LocationAgreementFormCubit extends Cubit<LocationAgreementFormState> {
44   void init(LocationAgreement? agreement) {
45     // You, last month + first location agreement's management ...
46     emit(
47       state.copyWith(
48         initial: agreement,
49         evidences: agreement?.evidences ?? const [],
50       ),
51     );
52   }
53   void removeEvidence(AgreementEvidence evidence) {
54     final next = [...state.evidences].remove(evidence);
55     emit(state.copyWith(evidences: next));
56   }
57   Future<AgreementEvidence> addEvidence({
58     required Uint8List bytes,
59     required String contentType,
60     String? note,
61   }) async {
62     emit(state.copyWith(uploading: true, error: null));
63     try {
64       final evidence = await uploadEvidence(
65         agreementId: state.initial?.id.isNotNull ? true
66           ? state.initial.id
67           : uploadKey,
68         bytes: bytes,
69         contentType: contentType,
70         note: note,
71       );
72       final next = [...state.evidences, evidence];
73       emit(state.copyWith(evidences: next, uploading: false));
74       return evidence;
75     } catch (e) {
76       emit(state.copyWith(uploading: false, error: e.toString()));
77       rethrow;
78     }
79   }
80   Future<LocationAgreement> save({
81     required String companyName,
82   }) async {
83     emit(state.copyWith(saving: true, error: null));
84     try {
85       final payload = LocationAgreement(

```

Las figuras 172 y 173 trasladan la historia al caso de uso de guardado y a la fuente de datos que ejecuta consultas y actualizaciones. Con ello, el acuerdo de ubicación queda respaldado por una estructura que separa el registro visible de su persistencia

Figura 172. Código de `save_location_agreement.dart`

```

1 import '../entities/location_agreement.dart';
2 import '../repositories/location_agreement_repository.dart';
3 import '../utils/location_agreement_normalizer.dart';
4 import 'find_duplicate_location_agreement.dart';
5
6 class SaveLocationAgreement {
7   final LocationAgreementRepository repository;
8   final FindDuplicateLocationAgreement findDuplicate;
9   SaveLocationAgreement(this.repository, this.findDuplicate);
10
11   Future<LocationAgreement> call(
12     LocationAgreement agreement, {
13     required String? userId,
14     bool ignoreDuplicate = false,
15   }) async {
16     final duplicate = buildAgreementDuplicate(
17       agreement.companyName,
18       agreement.pointName,
19     );
20     final normalized = agreement.copyWith(
21       agreementType: LocationAgreementType.invalid(agreement.agreementType),
22       agreementStatus: LocationAgreementStatus.archived,
23       companyName: normalized.companyName,
24       pointName: normalized.pointName,
25       status: LocationAgreementStatus.archived,
26       agreementStatus: LocationAgreementStatus.archived,
27     );
28     if (!ignoreDuplicate && normalized.status != LocationAgreementStatus.archived) {
29       final dup = await findDuplicate.call(
30         companyName: normalized.companyName,
31         pointName: normalized.pointName,
32         excludeId: normalized.id.isEmpty ? null : normalized.id,
33       );
34       if (dup != null) throw DuplicateLocationAgreementException(dup);
35     }
36     if (normalized.id.isEmpty) {
37       return repository.create(normalized, userId: userId);
38     }
39     return repository.update(normalized, userId: userId);
40   }
41 }

```

Figura 173. Código de `location_agreements_remote_datasource.dart`: consultas, `upsert` y estado.

```

1 class LocationAgreementsRemoteDataSource {
2   Future<List<LocationAgreementModel>> fetchAll() async {
3     try {
4       final snap = await _col.orderBy('createdAt', descending: true).get();
5       return snap.doc.map((doc) => LocationAgreementModel.fromDoc(doc)).toList();
6     } catch (_) {}
7     final snap = await _col.get();
8     return snap.doc.map((doc) => LocationAgreementModel.fromDoc(doc)).toList();
9   }
10
11   Future<LocationAgreementModel?> findById(String id) async {
12     String? docId;
13     String? excludedId;
14     final snap = await _col.where('docId', isEqualTo: id).limit(1).get();
15     if (snap.docs.isNotEmpty) {
16       docId = snap.docs.first.docId;
17       final data = snap.docs.first.data;
18       final status = data['status'] ?? 'active';
19       if (status == LocationAgreementStatus.archived) continue;
20       return LocationAgreementModel.fromDoc(snap.doc);
21     }
22     return null;
23   }
24
25   Future<LocationAgreementModel> create(
26     LocationAgreementModel data, {
27     String? userId,
28   }) async {
29     final doc = LocationAgreementModel.toDoc(data, userId: userId);
30     final snap = await _col.add(doc);
31     final docId = snap.docId;
32     final data = snap.data;
33     final status = data['status'] ?? 'active';
34     if (status == LocationAgreementStatus.archived) continue;
35     return LocationAgreementModel.fromDoc(snap.doc);
36   }
37
38   Future<LocationAgreementModel> update(
39     LocationAgreementModel data, {
40     String? userId,
41   }) async {
42     final docId = data.docId;
43     final snap = await _col.doc(docId).get();
44     final data = snap.data;
45     final status = data['status'] ?? 'active';
46     if (status == LocationAgreementStatus.archived) continue;
47     final snap = await _col.doc(docId).update(data);
48     return LocationAgreementModel.fromDoc(snap.doc);
49   }
50 }

```

4.3.4.1.9. Historia de usuario 26: Mapa de puntos de venta cercanos

El anexo 4 desarrolla el mapa de puntos de venta cercanos como una herramienta de exploración territorial apoyada en filtros y resultados visibles sobre el mapa. Las figuras 174 y 175 muestran tanto la vista principal como el estado sin resultados, permitiendo comprender cómo se comporta el módulo ante distintos criterios de búsqueda.

Figura 174. Interfaz móvil del mapa de puntos de venta

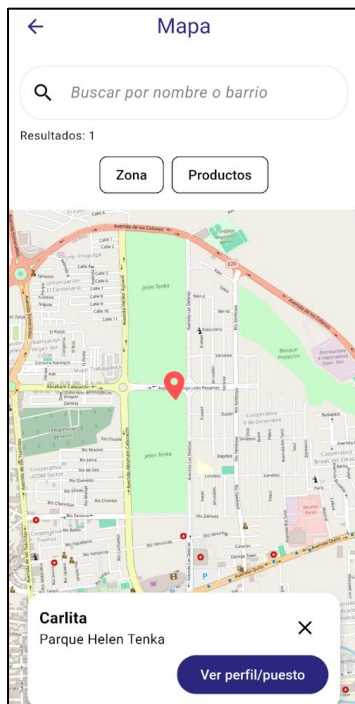
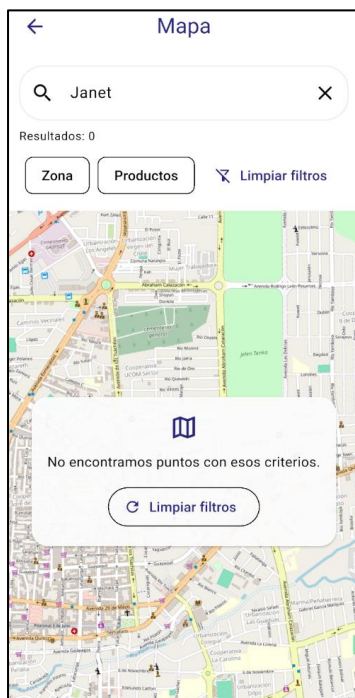


Figura 175. Interfaz móvil del estado sin resultados en el mapa



La entrada al mapa, su enrutamiento y la coordinación del estado se apoyan en *home_tab.dart*, *map_routes.dart*, *map_page.dart* y *map_cubit.dart*, visibles en las figuras 176, 177, 178 y 179. En este conjunto se define el acceso al módulo y la forma en que la interfaz responde a los filtros aplicados.

Figura 176. Código de *home_tab.dart*: acceso “Puntos de venta” con modos *near/zone*.

```
lib > modules > features > home > presentation > pages > tabs > home_tab.dart > HomeTab > _buildInfoCards
24 class HomeTab extends StatelessWidget {
132 List<HomeInfoCardData> _buildInfoCards(
271 HomeInfoCardData(
272 title: 'Puntos de venta', You, 2 weeks ago + Título: refactor: dashboard HU7
273 subtitle: 'Ubicaciones de emprendedoras',
274 trailing: const HomeInfoPill(text: 'cerca'),
275 icon: Icons.location_on_outlined,
276 onTap: () => openMap(),
277 actions: [
278 HomeInfoActionData(label: 'Cerca de mí', onTap: () => openMap(mode: 'near')),
279 HomeInfoActionData(label: 'Por zona', onTap: () => openMap(mode: 'zone')),
280 ],
281 ), // HomeInfoCardData
```

En *home_tab.dart* se organiza una pieza de soporte usado por Mapa de puntos de venta cercanos para explorar puntos de venta con filtros y navegación a detalle.

Figura 177. Código de *map_routes.dart*: ruta */map* y armado de *EntrepreneursMapPage*.

```
lib > modules > features > entrepreneur_map > map_routes.dart > MapRoutes
You, last month | 1 author (You)
1 import 'package:cloud_firestore/cloud_firestore.dart';
2 import 'package:flutter/material.dart';
3 import 'package:flutter_bloc/flutter_bloc.dart';
4
5 import '../entrepreneur_profiles/data/datasources/entrepreneur_favorites_datasource.dart';
6 import '../entrepreneur_profiles/data/datasources/entrepreneur_profiles_remote_datasource.dart';
7 import '../entrepreneur_profiles/data/repositories/entrepreneur_profile_repository_impl.dart';
8 import '../entrepreneur_profiles/domain/usecases/get_map_points.dart';
9 import '../entrepreneur_location/data/datasources/entrepreneur_location_remote_datasource.dart';
10 import '../sales_zones/data/datasources/sales_zones_remote_datasource.dart';
11 import '../sales_zones/data/repositories/sales_zone_repository_impl.dart';
12 import '../sales_zones/domain/usecases/get_sales_zones.dart';
13 import 'presentation/blocs/map_cubit.dart';
14 import 'presentation/pages/map_page.dart';
15
16 You, last month | 1 author (You)
17 class MapRoutes {
18   static const String base = '/map';
19
20   static Route onGenerateRoute(RouteSettings settings) {
21     switch (settings.name) {
22       case base:
23         final firestore = FirebaseFirestore.instance;
24         final profilesRepo = EntrepreneurProfileRepositoryImpl(
25           profiles: EntrepreneurProfilesRemoteDataSource(firestore),
26           locations: EntrepreneurLocationRemoteDataSource(firestore),
27           favorites: EntrepreneurFavoritesDataSource(firestore),
28         );
29         final zonesRepo = SalesZoneRepositoryImpl(
30           salesZonesRemoteDataSource(firestore),
31           salesZoneRepositoryImpl,
32         );
33         return MaterialPageRoute(
34           builder:
35             () => BlocProvider(
36               create:
37                 () => MapCubit(
38                   getPoints: getEntrepreneurMapPoints(profilesRepo),
39                   getZones: getSalesZones(zonesRepo),
40                 ).load(), // MapCubit
41               child: const EntrepreneursMapPage(),
42             ), // BlocProvider
43         ); // MaterialPageRoute
44   }
45 }
```

En *map_routes.dart* se define el enrutamiento que soporta explorar puntos de venta con filtros y navegación a detalle dentro de Mapa de puntos de venta cercanos,

Figura 178. Código de *map_page.dart*: buscador y estado vacío del mapa.

```
lib > modules > features > entrepreneur_map > presentation > pages > map_page.dart > EntrepreneursMapPageState > build
11 You, last month | 1 author (You)
12 class EntrepreneursMapPage extends StatelessWidget {
13   const EntrepreneursMapPage({super.key});
14
15   @override
16   State<EntrepreneursMapPage> createState() => _EntrepreneursMapPageState();
17 }
18
19 You, 3 weeks ago | 1 author (You)
20 class _EntrepreneursMapPageState extends State<EntrepreneursMapPage> {
21   final _searchCtrl = TextEditingController();
22   List<MapMarker> _lastMarkersSource = const [];
23   List<MapMarker> _markers = const [];
24
25   @override
26   void dispose() {
27     _searchCtrl.dispose();
28   }
29
30   @override
31   Widget build(BuildContext context) {
32     return BlocConsumer<MapCubit, MapState> { // BlocConsumer
33       // ...
34     }
35
36     MapMarker _toMarker(EntrepreneurProfile p) {
37       final geo = p.location.geoPoint!;
38       return MapMarker(
39         id: p.uid,
40         latitude: geo.latitude,
41         longitude: geo.longitude,
42         label: p.name,
43       );
44     }
45
46     void _onMarkerTap(String id) {
47       for (final p in _lastMarkersSource) {
48         if (p.uid == id) {
49           context.read<MapCubit>().select(p);
50           return;
51         }
52       }
53     }
54
55     // ...
56   }
57 }
```

Este componente en *map_page.dart* configura la pantalla asociada a Mapa de puntos de venta cercanos para explorar puntos de venta con filtros y navegación a detalle.

Figura 179. Código de *map_cubit.dart*: definición de *MapState* y filtros, *load* y manejo de filtros/selección/cámara.

```
lib > modules > features > entrepreneur_map > presentation > bloc > map_cubit.dart > MapCubit

You, last month | author (You)
1 import 'package:flutter_bloc/flutter_bloc.dart';
2
3 import '../domain/entities/entrepreneur_profile.dart';
4 import '../domain/entities/usecases/get_map_points.dart';
5 import '../domain/entities/sales_zone.dart';
6 import '../domain/entities/get_sales_zones.dart';
7
8 You, last month | author (You)
9 class MapFilters {
10   final String? zoneId;
11   final List<String>? tags;
12   final String? query;
13
14   const MapFilters({this.zoneId, this.tags = const [], this.query = ""});
15
16   MapFilters copyWith({String? zoneId, List<String>? tags, String? query}) {
17     return MapFilters(
18       zoneId: zoneId ?? this.zoneId,
19       tags: tags ?? this.tags,
20       query: query ?? this.query,
21     );
22   }
23
24   MapFilters clear() => const MapFilters();
25 }
26
27 You, last month | author (You)
28 class MapState {
29   final bool loading;
30   final String? error;
31   final List<EntrepreneurProfile>? points;
32   final List<EntrepreneurProfile>? withoutGeo;
33   final List<SalesZone>? zones;
34   final List<String>? tagOptions;
35   final MapFilters? filters;
36   final List<EntrepreneurProfile>? selected;
37   final double? cameraLat;
38   final double? cameraLong;
39   final double? zoom;
40
41   const MapState({
42     this.loading = false,
43     this.error,
44     this.points = const [],
45     this.withoutGeo = const [],
46     this.zones = const [],
47     this.tagOptions = const [],
48     this.filters = const MapFilters(),
49     this.selected,
50     this.cameraLat = 0,
51     this.cameraLong = 0,
52     this.zoom = 11,
53   });
54 }
```

Las figuras 180 y 181 completan la historia desde el caso de uso *get_map_points.dart* y el repositorio que articula perfiles y ubicaciones. Así, la exploración geográfica no se limita a la representación visual, sino que se sostiene en la consulta integrada de los puntos disponibles.

Figura 180. Código de *get_map_points.dart*: filtros por *zona/tags/query*.

```
lib > modules > features > entrepreneur_profiles > domain > usecases > get_map_points.dart > GetEntrepreneurMapPoints

You, last month | author (You)
1 import '../entities/entrepreneur_profile.dart';
2 import '../repositories/entrepreneur_profile_repository.dart';
3
4 You, last month | author (You)
5 class GetEntrepreneurMapPoints {
6   final EntrepreneurProfileRepository repository;
7   GetEntrepreneurMapPoints(this.repository);
8
9   Future<List<EntrepreneurProfile>> call({
10     String? zoneId,
11     List<String>? tags,
12     String? query,
13   }) async {
14     final list = await repository.fetchPublicProfiles();
15     return list.where((p) {
16       if (p.location?.geoPoint == null) return false;
17       final matchesZone =
18         (zoneId == null || zoneId.isEmpty)
19           ? true
20           : p.location?.zoneId == zoneId;
21       final matchesTags =
22         (tags == null || tags.isEmpty)
23           ? true
24           : p.tags.any((t) => tags.contains(t));
25       final matchesQuery =
26         (query == null || query.trim().isEmpty)
27           ? true
28           : (p.name.toLowerCase().contains(query.toLowerCase()) ||
29             (p.zoneText ?? '').toLowerCase().contains(query.toLowerCase()));
30       return matchesZone && matchesTags && matchesQuery;
31     }).toList();
32   }
33 }
```

Figura 181. Código de *entrepreneur_profile_repository_impl.dart*

```

18 > module > features > entrepreneur_profile > data > repositories > entrepreneur_profile_repository_impl.dart > % Implement entrepreneur profile repository impl
1 import 'package:entrepreneur_profile/data/datasources/entrepreneur_location_remote_datasource.dart';
2 import 'package:entrepreneur_profile/data/models/entrepreneur_location_model.dart';
3 import 'package:entrepreneur_profile/domain/entities/entrepreneur_profile.dart';
4 import 'package:entrepreneur_profile/domain/repositories/entrepreneur_profile_repository.dart';
5 import 'package:entrepreneur_profile/data/datasources/entrepreneur_favorites_datasource.dart';
6 import 'package:entrepreneur_profile/data/datasources/entrepreneur_profiles_remote_datasource.dart';
7 import 'package:entrepreneur_profile/data/models/entrepreneur_profile_model.dart';
8
9 You set month | author (this)
10 class EntrepreneurProfileRepositoryImpl
11   implements EntrepreneurProfileRepository {
12     final EntrepreneurProfilesRemoteDataSource profiles;
13     final EntrepreneurLocationRemoteDataSource locations;
14     final EntrepreneurFavoritesDataSource favorites;
15
16     EntrepreneurProfileRepositoryImpl({
17       required this.profiles,
18     }) {}
19
20 @override
21 Future<List<EntrepreneurProfile>> fetchPublicProfiles(
22   String viewerId,
23 ) async {
24   // You, last month + feat: Implement entrepreneur profiles feature w.
25   final base = await profiles.fetchPublic();
26   final ids = base.map(p => p.id).toList();
27   final locs = await locations.fetchMany(ids);
28   final favs =
29     viewerId != null && viewerId.isNotEmpty
30       ? await favorites.fetchFavorites(viewerId)
31       : <String>[];
32
33   return base.map(p => {
34     final loc = locs[p.id];
35     return merge(p, loc, favs.contains(p.id));
36   }).toList();
37 }
38
39 @override
40 Future<EntrepreneurProfile> fetchDetail(
41   String profileId,
42 ) async {
43   //
44   //
45   //
46   //
47   //
48   //
49   //
50   //
51   //
52   //
53   //
54   //
55   //
56   //
57   //
58   //
59   //
60   //
61   //
62   //
63   //
64   //
65   //
66   //
67   //
68   //
69   //
70   //
71   //
72   //
73   //
74   //
75   //
76   //
77   //
78   //
79   //
80   //
81   //
82   //
83   //
84   //
85   //
86   //
87   //
88   //
89   //
90   //
91   //
92   //
93   //
94   //
95   //
96   //
97   //
98   //
99   //
100  //
101  //
102  //
103  //
104  //
105  //
106  //
107  //
108  //
109  //
110  //
111  //
112  //
113  //
114  //
115  //
116  //
117  //
118  //
119  //
120  //
121  //
122  //
123  //
124  //
125  //
126  //
127  //
128  //
129  //
130  //
131  //
132  //
133  //
134  //
135  //
136  //
137  //
138  //
139  //
140  //
141  //
142  //
143  //
144  //
145  //
146  //
147  //
148  //
149  //
150  //
151  //
152  //
153  //
154  //
155  //
156  //
157  //
158  //
159  //
160  //
161  //
162  //
163  //
164  //
165  //
166  //
167  //
168  //
169  //
170  //
171  //
172  //
173  //
174  //
175  //
176  //
177  //
178  //
179  //
180  //
181  //
182  //
183  //
184  //
185  //
186  //
187  //
188  //
189  //
190  //
191  //
192  //
193  //
194  //
195  //
196  //
197  //
198  //
199  //
200  //
201  //
202  //
203  //
204  //
205  //
206  //
207  //
208  //
209  //
210  //
211  //
212  //
213  //
214  //
215  //
216  //
217  //
218  //
219  //
220  //
221  //
222  //
223  //
224  //
225  //
226  //
227  //
228  //
229  //
230  //
231  //
232  //
233  //
234  //
235  //
236  //
237  //
238  //
239  //
240  //
241  //
242  //
243  //
244  //
245  //
246  //
247  //
248  //
249  //
250  //
251  //
252  //
253  //
254  //
255  //
256  //
257  //
258  //
259  //
260  //
261  //
262  //
263  //
264  //
265  //
266  //
267  //
268  //
269  //
270  //
271  //
272  //
273  //
274  //
275  //
276  //
277  //
278  //
279  //
280  //
281  //
282  //
283  //
284  //
285  //
286  //
287  //
288  //
289  //
290  //
291  //
292  //
293  //
294  //
295  //
296  //
297  //
298  //
299  //
300  //
301  //
302  //
303  //
304  //
305  //
306  //
307  //
308  //
309  //
310  //
311  //
312  //
313  //
314  //
315  //
316  //
317  //
318  //
319  //
320  //
321  //
322  //
323  //
324  //
325  //
326  //
327  //
328  //
329  //
330  //
331  //
332  //
333  //
334  //
335  //
336  //
337  //
338  //
339  //
340  //
341  //
342  //
343  //
344  //
345  //
346  //
347  //
348  //
349  //
350  //
351  //
352  //
353  //
354  //
355  //
356  //
357  //
358  //
359  //
360  //
361  //
362  //
363  //
364  //
365  //
366  //
367  //
368  //
369  //
370  //
371  //
372  //
373  //
374  //
375  //
376  //
377  //
378  //
379  //
380  //
381  //
382  //
383  //
384  //
385  //
386  //
387  //
388  //
389  //
390  //
391  //
392  //
393  //
394  //
395  //
396  //
397  //
398  //
399  //
400  //
401  //
402  //
403  //
404  //
405  //
406  //
407  //
408  //
409  //
410  //
411  //
412  //
413  //
414  //
415  //
416  //
417  //
418  //
419  //
420  //
421  //
422  //
423  //
424  //
425  //
426  //
427  //
428  //
429  //
430  //
431  //
432  //
433  //
434  //
435  //
436  //
437  //
438  //
439  //
440  //
441  //
442  //
443  //
444  //
445  //
446  //
447  //
448  //
449  //
450  //
451  //
452  //
453  //
454  //
455  //
456  //
457  //
458  //
459  //
460  //
461  //
462  //
463  //
464  //
465  //
466  //
467  //
468  //
469  //
470  //
471  //
472  //
473  //
474  //
475  //
476  //
477  //
478  //
479  //
480  //
481  //
482  //
483  //
484  //
485  //
486  //
487  //
488  //
489  //
490  //
491  //
492  //
493  //
494  //
495  //
496  //
497  //
498  //
499  //
500  //
501  //
502  //
503  //
504  //
505  //
506  //
507  //
508  //
509  //
510  //
511  //
512  //
513  //
514  //
515  //
516  //
517  //
518  //
519  //
520  //
521  //
522  //
523  //
524  //
525  //
526  //
527  //
528  //
529  //
530  //
531  //
532  //
533  //
534  //
535  //
536  //
537  //
538  //
539  //
540  //
541  //
542  //
543  //
544  //
545  //
546  //
547  //
548  //
549  //
550  //
551  //
552  //
553  //
554  //
555  //
556  //
557  //
558  //
559  //
560  //
561  //
562  //
563  //
564  //
565  //
566  //
567  //
568  //
569  //
570  //
571  //
572  //
573  //
574  //
575  //
576  //
577  //
578  //
579  //
580  //
581  //
582  //
583  //
584  //
585  //
586  //
587  //
588  //
589  //
590  //
591  //
592  //
593  //
594  //
595  //
596  //
597  //
598  //
599  //
600  //
601  //
602  //
603  //
604  //
605  //
606  //
607  //
608  //
609  //
610  //
611  //
612  //
613  //
614  //
615  //
616  //
617  //
618  //
619  //
620  //
621  //
622  //
623  //
624  //
625  //
626  //
627  //
628  //
629  //
630  //
631  //
632  //
633  //
634  //
635  //
636  //
637  //
638  //
639  //
640  //
641  //
642  //
643  //
644  //
645  //
646  //
647  //
648  //
649  //
650  //
651  //
652  //
653  //
654  //
655  //
656  //
657  //
658  //
659  //
660  //
661  //
662  //
663  //
664  //
665  //
666  //
667  //
668  //
669  //
670  //
671  //
672  //
673  //
674  //
675  //
676  //
677  //
678  //
679  //
680  //
681  //
682  //
683  //
684  //
685  //
686  //
687  //
688  //
689  //
690  //
691  //
692  //
693  //
694  //
695  //
696  //
697  //
698  //
699  //
700  //
701  //
702  //
703  //
704  //
705  //
706  //
707  //
708  //
709  //
710  //
711  //
712  //
713  //
714  //
715  //
716  //
717  //
718  //
719  //
720  //
721  //
722  //
723  //
724  //
725  //
726  //
727  //
728  //
729  //
730  //
731  //
732  //
733  //
734  //
735  //
736  //
737  //
738  //
739  //
740  //
741  //
742  //
743  //
744  //
745  //
746  //
747  //
748  //
749  //
750  //
751  //
752  //
753  //
754  //
755  //
756  //
757  //
758  //
759  //
760  //
761  //
762  //
763  //
764  //
765  //
766  //
767  //
768  //
769  //
770  //
771  //
772  //
773  //
774  //
775  //
776  //
777  //
778  //
779  //
780  //
781  //
782  //
783  //
784  //
785  //
786  //
787  //
788  //
789  //
790  //
791  //
792  //
793  //
794  //
795  //
796  //
797  //
798  //
799  //
800  //
801  //
802  //
803  //
804  //
805  //
806  //
807  //
808  //
809  //
810  //
811  //
812  //
813  //
814  //
815  //
816  //
817  //
818  //
819  //
820  //
821  //
822  //
823  //
824  //
825  //
826  //
827  //
828  //
829  //
830  //
831  //
832  //
833  //
834  //
835  //
836  //
837  //
838  //
839  //
840  //
841  //
842  //
843  //
844  //
845  //
846  //
847  //
848  //
849  //
850  //
851  //
852  //
853  //
854  //
855  //
856  //
857  //
858  //
859  //
860  //
861  //
862  //
863  //
864  //
865  //
866  //
867  //
868  //
869  //
870  //
871  //
872  //
873  //
874  //
875  //
876  //
877  //
878  //
879  //
880  //
881  //
882  //
883  //
884  //
885  //
886  //
887  //
888  //
889  //
890  //
891  //
892  //
893  //
894  //
895  //
896  //
897  //
898  //
899  //
900  //
901  //
902  //
903  //
904  //
905  //
906  //
907  //
908  //
909  //
910  //
911  //
912  //
913  //
914  //
915  //
916  //
917  //
918  //
919  //
920  //
921  //
922  //
923  //
924  //
925  //
926  //
927  //
928  //
929  //
930  //
931  //
932  //
933  //
934  //
935  //
936  //
937  //
938  //
939  //
940  //
941  //
942  //
943  //
944  //
945  //
946  //
947  //
948  //
949  //
950  //
951  //
952  //
953  //
954  //
955  //
956  //
957  //
958  //
959  //
960  //
961  //
962  //
963  //
964  //
965  //
966  //
967  //
968  //
969  //
970  //
971  //
972  //
973  //
974  //
975  //
976  //
977  //
978  //
979  //
980  //
981  //
982  //
983  //
984  //
985  //
986  //
987  //
988  //
989  //
990  //
991  //
992  //
993  //
994  //
995  //
996  //
997  //
998  //
999  //
1000 //

```

4.3.4.1.10. Historia de usuario 27: Ver perfiles de emprendedoras

Tal como se muestra en el anexo 4, la consulta de perfiles de emprendedoras se construye a partir del listado general, el detalle individual y la marcación de favoritos. Las figuras 182, 183 y 184 permiten seguir este recorrido y muestran cómo la información pública de las emprendedoras se vuelve navegable dentro de la aplicación.

Figura 182. Interfaz móvil del listado de emprendedoras

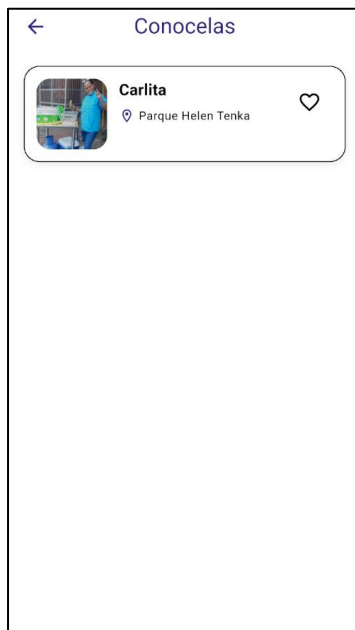
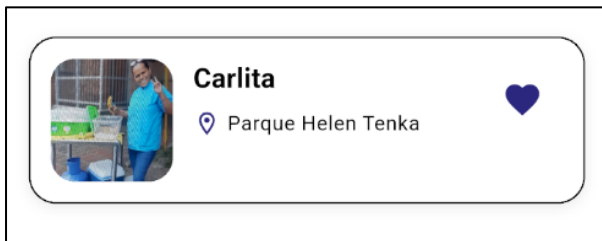


Figura 183. Interfaz móvil del detalle de emprendedora



Figura 184. Interfaz móvil de favorito aplicado en el listado



El desarrollo hecho para las interfaces mostradas se presentadas en *entrepreneurs_page.dart*, *entrepreneurs_list_cubit.dart* y *entrepreneur_detail_cubit.dart*, visibles en las figuras 185, 186 y 187. Este bloque organiza la carga de listados, la actualización de favoritos y la consulta del detalle sin recargar cada pantalla con lógica dispersa.

Figura 185. Código de *entrepreneurs_page.dart*: estados de UI (carga/vacío/error).

```

12 > modules > features > entrepreneur_profiles > presentation > pages > entrepreneurs_page.dart > EntrepreneursPage > build
13
14 class EntrepreneursPage extends StatelessWidget {
15   Widget build(BuildContext context) {
16     return BlocConsumer<EntrepreneursListCubit, EntrepreneursListState>(
17       // You, last month + feat: Implement entrepreneur profiles feature w...
18       listenerWhen: (p, c) => p.info != c.info || p.error != c.error,
19       listener: (context, state) {
20         if (state.info != null) AppSnackBar.showToast(context, state.info!);
21         if (state.error != null) AppSnackBar.showToast(context, state.error!);
22       },
23       builder: (context, state) {
24         return Scaffold(
25           appBar: AppBar(title: const Text("Conoceas")),
26           body:
27             state.loading
28             ? const Center(child: CircularProgressIndicator())
29             : _buildContent(context, state),
30         ); // Scaffold
31       }, // BlocConsumer
32     );
33   }
34
35   Widget _buildContent(BuildContext context, EntrepreneursListState state) {
36     if (state.error != null && state.items.isEmpty) {
37       return ErrorRetry(
38         message: state.error!,
39         onRetry: () => context.read<EntrepreneursListCubit>().load(),
40       );
41     }
42     if (state.items.isEmpty) {
43       return const EmptyStateView(
44         icon: Icons.group_2_outlined,
45         message: 'Aun no hay emprendedoras publicadas',
46       );
47     }
48   }

```

Figura 186. Código de *entrepreneurs_list_cubit.dart*: *load* y *toggleFavorite*.

```
lib > modules > features > entrepreneur_profiles > presentation > blocs > entrepreneurs_list_cubit.dart > % EntrepreneursListCubit
You, last month | 1 author (You)
1 import 'package:flutter_bloc/flutter_bloc.dart';
2
3 import '../core/utils/errors/firebase_errors.dart';
4 import '../domain/entities/entrepreneur_profile.dart';
5 import '../domain/usecases/get_entrepreneurs.dart';
6 import '../domain/usecases/toggle_favorite_entrepreneur.dart';
7
8 You, last month | 1 author (You)
9 class EntrepreneursListState {
10   final bool loading;
11   final List<EntrepreneurProfile> items;
12   final String? error;
13   final String? info;
14
15   const EntrepreneursListState({
16     this.loading = false,
17   });
18
19   EntrepreneursListState copyWith({
20     bool? loading,
21     List<EntrepreneurProfile>? items,
22     String? error,
23     String? info,
24   }) {
25     return EntrepreneursListState(
26       loading: loading ?? this.loading,
27       items: items ?? this.items,
28       error: error ?? this.error,
29       info: info ?? this.info,
30     );
31   }
32 }
33
34 You, last month | 1 author (You)
35 class EntrepreneursListCubit extends Cubit<EntrepreneursListState> {
36   final GetEntrepreneurs _getEntrepreneurs;
37   final ToggleFavoriteEntrepreneur _toggleFavorite;
38   final String? viewerId;
```

Figura 187. Código de *entrepreneur_detail_cubit.dart*: carga del perfil y *toggleFavorite*.

```
lib > modules > features > entrepreneur_profiles > presentation > blocs > entrepreneur_detail_cubit.dart > % EntrepreneurDetailCubit
34 class EntrepreneurDetailCubit extends Cubit<EntrepreneurDetailState> {
35
36   Future<void> load() async {
37     You, last month | 1 author (You)
38     emit(state.copyWith(loading: true, error: null, info: null));
39     try {
40       final p = await _getDetail(entrepreneurId: viewerId);
41       if (p == null) {
42         emit(state.copyWith(loading: false, error: 'Perfil no disponible'));
43         return;
44       }
45       emit(state.copyWith(loading: false, profile: p));
46     } catch (e) {
47       emit(
48         state.copyWith(
49           loading: false,
50           error: FirebaseErrorMapper.friendly(
51             e,
52             generic: 'No se pudo cargar el perfil',
53           ),
54         );
55     }
56   }
57
58   Future<void> toggleFavorite() async {
59     final current = state.profile;
60     if (current == null) return;
61     if (viewerId == null || viewerId.isEmpty) {
62       emit(state.copyWith(info: 'Inicia sesión para guardar favoritos'));
63       return;
64     }
65     final next = current.copyWith(isFavorite: !current.isFavorite);
66     emit(state.copyWith(profile: next, info: null, error: null));
67     try {
68       await _toggleFavorite(
69         entrepreneurId: current.id,
70         viewerId: viewerId,
71         isFavorite: next.isFavorite,
72       );
73     } catch (e) {
74       emit(
75         state.copyWith(
76           loading: false,
77           error: FirebaseErrorMapper.friendly(
78             e,
79             generic: 'No se pudo actualizar el favorito',
80           ),
81         );
82     }
83   }
84 }
```

El respaldo funcional se completa con *get_entrepreneurs.dart*, *get_entrepreneur_detail.dart*, *toggle_favorite_entrepreneur.dart* y *entrepreneur_favorites_datasource.dart*, presentados en las figuras 188, 189, 190 y 191. De esta manera, la consulta y la interacción con favoritos quedan sostenidas por operaciones diferenciadas de dominio y datos.

Figura 188. Código de *get_entrepreneurs.dart*: caso de uso listado.

```
lib > modules > features > entrepreneur_profiles > domain > usecases > get_entrepreneurs.dart > ...
You, last month | 1 author (You)
1 import '../entities/entrepreneur_profile.dart';
2 import '../repositories/entrepreneur_profile_repository.dart';
3
4 You, last month | 1 author (You)
5 class GetEntrepreneurs {
6   final EntrepreneurProfileRepository repository;
7   GetEntrepreneurs(this.repository);
8
9   Future<List<EntrepreneurProfile>> call({String? viewerId}) {
10     return repository.fetchPublicProfiles(viewerId: viewerId);
11   }
12 }
```

Figura 189. Código de *get_entrepreneur_detail.dart*: orquestación del detalle por UID.

```
lib > modules > features > entrepreneur_profiles > domain > usecases > get_entrepreneur_detail.dart > GetEntrepreneurDetail
You, last month | 1 author (You)
1 import '../entities/entrepreneur_profile.dart';
2 import '../repositories/entrepreneur_profile_repository.dart';
3
4 class GetEntrepreneurDetail {
5   final EntrepreneurProfileRepository repository;
6   GetEntrepreneurDetail(this.repository);
7
8   Future<EntrepreneurProfile> call(
9     String entrepreneurId, {
10    String? viewerUid,
11  }) {
12    return repository.fetchDetail(entrepreneurId, viewerUid: viewerUid);
13  }
14 }
```

Figura 190. Código de *toggle_favorite_entrepreneur.dart*: encapsulación del marcado/desmarcado de favorito.

```
lib > modules > features > entrepreneur_profiles > domain > usecases > toggle_favorite_entrepreneur.dart > ToggleFavoriteEntrepreneur
You, last month | 1 author (You)
1 import '../repositories/entrepreneur_profile_repository.dart';
2
3 class ToggleFavoriteEntrepreneur {
4   final EntrepreneurProfileRepository repository;
5   ToggleFavoriteEntrepreneur(this.repository);
6
7   Future<bool> call(
8     required String entrepreneurId,
9     required String userId,
10    required bool favorite,
11  ) {
12    return repository.setFavorite(
13      entrepreneurId: entrepreneurId,
14      userId: userId,
15      favorite: favorite,
16    );
17  }
18 }
```

La implementación en *toggle_favorite_entrepreneur.dart* respalda consultar y gestionar datos de cuenta desde la App en Ver perfiles de emprendedoras mediante una pieza de soporte.

Figura 191. Código de *entrepreneur_favorites_datasource.dart*

```
lib > modules > features > entrepreneur_profiles > data > datasources > entrepreneur_favorites_datasource.dart > EntrepreneurFavoritesDataSource
You, last month | 1 author (You)
1 import 'package:cloud_firestore/cloud_firestore.dart';
2
3 import '../core/config/collections.dart';
4
5 class EntrepreneurFavoritesDataSource {
6   final FirebaseFirestore firestore;
7   EntrepreneurFavoritesDataSource(this.firestore);
8
9   CollectionReference<Map<String, dynamic>> _col(String userId) {
10     return firestore
11       .collection(Col.users)
12       .doc(userId)
13       .collection(Col.favoritesEntrepreneurs);
14   }
15
16   Future<Set<String>> fetchFavorites(String userId) async {
17     final snap = await _col(userId).get();
18     return snap.docs.map((d) => d.id).toSet();
19   }
20
21   Future<void> setFavorite({
22     required String userId,
23     required String entrepreneurId,
24     required bool favorite,
25   }) async {
26     final ref = _col(userId).doc(entrepreneurId);
27     if (favorite) {
28       await ref.set({
29         'createdAt': FieldValue.serverTimestamp(),
30       }, SetOptions(merge: true));
31     } else {
32       await ref.delete();
33     }
34   }
35 }
```

4.3.4.1.11. Historia de usuario 28: Testimonios o mensajes motivacionales

El anexo 4 incorpora los testimonios o mensajes motivacionales como un contenido de acompañamiento visible en el inicio y ampliable en una vista de detalle. Las figuras 192 y 193 muestran el acceso desde el carrusel principal y la lectura extendida de cada historia, integrando este recurso al entorno general de la aplicación.

Figura 192. Interfaz móvil del carrusel de historias

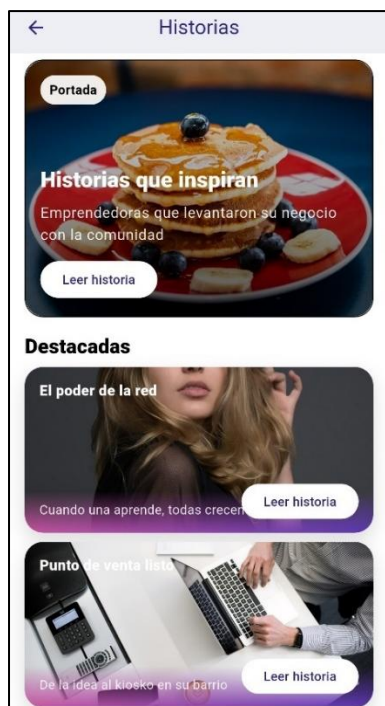


Figura 193. Interfaz móvil del detalle de una historia



La construcción del carrusel y el control de su comportamiento se apoyan en *stories_carousel.dart* y *stories_cubit.dart*, visibles en las figuras 194 y 195. En este bloque se articula la carga del contenido y el desplazamiento entre elementos sin romper la continuidad del inicio.

Figura 194. Código de *stories_carousel.dart*: *PageView* y navegación a detalle.

```
lib > modules > features > stories > presentation > widgets > stories_carousel.dart > _StoriesCarouselState > build
17 class _StoriesCarouselState extends State<StoriesCarousel> {
27   Widget build(BuildContext context) {
54     return Column(
55       mainAxisAlignment: MainAxisAlignment.min,
56       children: [
57         SizedBox(
58           height: 140,
59           child: PageView.builder(
60             controller: _controller,
61             itemCount: stories.length,
62             onPageChanged:
63               (index) =>
64                 context.read<StoriesCubit>().setPage(index),
65             itemBuilder: (context, index) {
66               final story = stories[index];
67               return Padding(
68                 padding: const EdgeInsets.symmetric(horizontal: 4),
69                 child: EditorialBanner(
70                   title: story.title,
71                   subtitle: story.subtitle,
72                   imageUrl: story.imageUrl,
73                   layout:
74                     index.isEven
75                       ? EditorialBannerLayout.imageLeft
76                       : EditorialBannerLayout.imageRight,
77                   onPressed: () => widget.onOpenStory(story),
78                 ), // EditorialBanner
79               ); // Padding

```

Figura 195. Código de *stories_cubit.dart*: *load* y *setPage*.

```
lib > modules > features > stories > presentation > blocs > stories_cubit.dart > StoriesCubit
4 class StoriesCubit {
5   final bool loading;
6   final List<Story> stories;
7   final String? error;
8   final int page;
9
10  const StoriesCubit({
11    this.loading = false,
12    this.stories = const [],
13    this.error,
14    this.page = 0,
15  });
16
17  StoriesCubit copyWith({
18    bool? loading,
19    List<Story>? stories,
20    String? error,
21    int? page,
22  }) {
23    return StoriesCubit(
24      loading: loading ?? this.loading,
25      stories: stories ?? this.stories,
26      error: error,
27      page: page ?? this.page,
28    );
29  }
30
31  You last month | author (bad
32
33  You last month | author (bad
34  class StoriesCubit extends Cubit<StoriesCubit> {
35    final StoriesCubit getStories;
36    StoriesCubit(this.getStories) : super(const StoriesCubit());
37
38    Future<void> load() async {
39      emit(state.copyWith(loading: true, error: null));
40      try {
41        final items = await getStories();
42        emit(state.copyWith(loading: false, stories: items, page: 0));
43      } catch (e) {
44        emit(state.copyWith(loading: false, error: e.toString()));
45      }
46    }
47
48    void setPage(int page) {

```

Las figuras 196, 197 y 198 completan la historia con el caso de uso *GetStories*, el repositorio correspondiente y el *cubit* del detalle. Con ello, la sección de historias mantiene un soporte técnico que diferencia la carga del contenido, su almacenamiento y la apertura de cada testimonio.

Figura 196. Código de *get_stories.dart*: caso de uso *GetStories*.

```
lib > modules > features > stories > domain > usecases > get_stories.dart > ...
You, last month | 1 author (You)
1 import '../entities/story.dart';
2 import '../repositories/stories_repository.dart';
3
4 class GetStories {
5   final StoriesRepository repository;
6   GetStories(this.repository);
7
8   Future<List<Story>> call() => repository.fetchStories();
9 }
10
```

Figura 197. Código de *stories_repository_impl.dart*: combinación remota/local.

```
lib > modules > features > stories > data > repositories > stories_repository_impl.dart > StoriesRepositoryImpl
You, last month | 1 author (You)
1 import '../entities/story.dart';
2 import '../domain/repositories/stories_repository.dart';
3 import '../datasources/stories_local_datasource.dart';
4 import '../datasources/stories_remote_datasource.dart';
5
6 class StoriesRepositoryImpl implements StoriesRepository {
7   final StoriesRemoteDataSource remote;
8   final StoriesLocalDataSource local;
9
10  StoriesRepositoryImpl({required this.remote, required this.local});
11
12  @override
13  Future<List<Story>> fetchStories() async {
14    final localStories = await local.fetchStories();
15    if (localStories.isNotEmpty) return _sort(localStories);
16
17    final remoteStories = await remote.fetchStories();
18    if (remoteStories.isNotEmpty) return _sort(remoteStories);
19
20    return const [];
21  }
22
23  @override
24  Future<Story> fetchById(String id) async {
25    final localStories = await fetchStories();
26    try {
27      return localStories.firstWhere((s) => s.id == id);
28    } catch (_) {}
29
30    final remoteStory = await remote.fetchById(id);
31    return remoteStory;
32  }
}
```

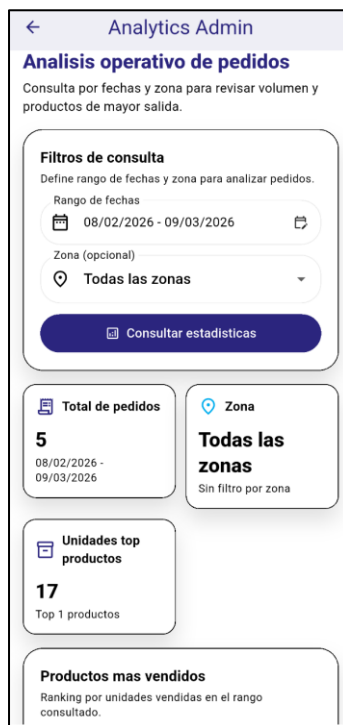
Figura 198. Código de *story_detail_cubit.dart*

```
lib > modules > features > stories > presentation > blocs > story_detail_cubit.dart > StoryDetailCubit
26 class StoryDetailCubit extends Cubit<StoryDetailState> {
27
28   Future<void> load() async {
29     emit(state.copyWith(loading: true, error: null));
30     try {
31       final story = await getDetail(id);
32       emit(state.copyWith(loading: false, story: story));
33     } catch (e) {
34       emit(state.copyWith(loading: false, error: e.toString()));
35     }
36   }
37 }
38
```

4.3.4.1.12. Historia de usuario 29: Visualización de estadísticas (sin exportar)

Tal como se desarrolla en el anexo 4, la visualización de estadísticas sin exportación se integra al panel administrativo como una síntesis operativa del estado general del sistema. La figura 199 sitúa esta vista y muestra cómo los indicadores se incorporan al entorno de administración.

Figura 199. Interfaz móvil del *Dashboard Admin* (*AdminDashboardPage*), vista general del encabezado “Panel administrativo” y el bloque principal de indicadores.



La composición del panel y la carga reactiva del resumen se apoyan en *admin_dashboard_page.dart* y *admin_dashboard_cubit.dart*, presentados en las figuras 200 y 201. Este bloque organiza la lectura de métricas dentro de una vista concentrada y estable.

Figura 200. Código de *admin_dashboard_page.dart*: KPIs, chips y accesos rápidos del dashboard.

```

14 class AdminDashboardPage extends StatelessWidget {
15   const AdminDashboardPage({super.key});
16
17   @override
18   Widget build(BuildContext context) {
19     return BlocBuilder<AdminDashboardCubit, AdminDashboardState>(() {
20       // BlocBuilder ...
21     });
22   }
23
24   You, 2 weeks ago | author (No)
25
26   class _DashboardContent extends StatelessWidget {
27     final AdminDashboardSummary summary;
28     const _DashboardContent({required this.summary});
29
30     String _dbg(String value, String label) {
31       return debugCorruptText(value, label: label);
32     }
33
34     @override
35     Widget build(BuildContext context) {
36       final statusChips =
37         summary.statusCounts.entries
38         .map((e) => AdminSummaryChipData(_statusLabel(e.key), e.value))
39         .toList();
40
41       final kpiItems = [
42         AdminKpiData(...),
43         if (summary.topProducts.isNotEmpty)
44           AdminKpiData(...),
45         AdminKpiData(...),
46         AdminKpiData(...),
47       ];
48
49       final quickActions = [
50         ...
51       ];
52
53       return RefreshIndicator( // RefreshIndicator ...
54     );
55   }

```


4.3.4.1.13. Historia de usuario 30: Registro de ubicación por la emprendedora

Tal como se recoge en el anexo 4, el registro de ubicación por la emprendedora se estructura como un formulario que permite describir el punto de venta, gestionar su imagen y reconocer estados de carga o ubicación previa. Las figuras 204, 205 y 206 muestran estas variantes de la interfaz y consolidan la captura de este dato dentro del flujo de la emprendedora.

Figura 204. Interfaz móvil del formulario de ubicación de la emprendedora



← Registrar ubicación

Ubicación de venta
Comparte el barrio o sector donde vendes para que puedan encontrarte. Esta ubicación también se usará para prellenar tu solicitud al proyecto.

Barrio / sector
Barrio central

Descripción breve
Puesto de empanadas

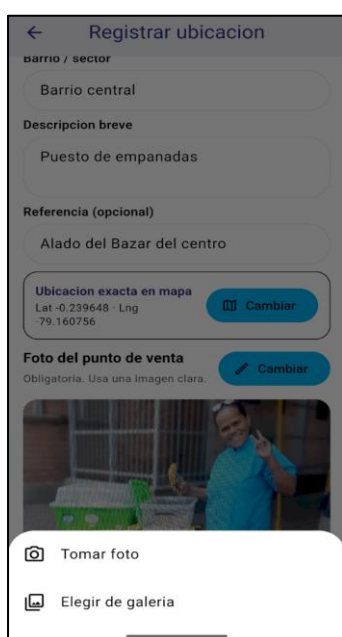
Referencia (opcional)
Alado del Bazar del centro

Ubicación exacta en mapa
Lat -0.239648 · Lng -79.160756 [Cambiar](#)

Foto del punto de venta
Obligatoria. Usa una imagen clara. [Cambiar](#)



Figura 205. Interfaz móvil de estado de carga en ubicación



← Registrar ubicación

Barrio / sector
Barrio central

Descripción breve
Puesto de empanadas

Referencia (opcional)
Alado del Bazar del centro

Ubicación exacta en mapa
Lat -0.239648 · Lng -79.160756 [Cambiar](#)

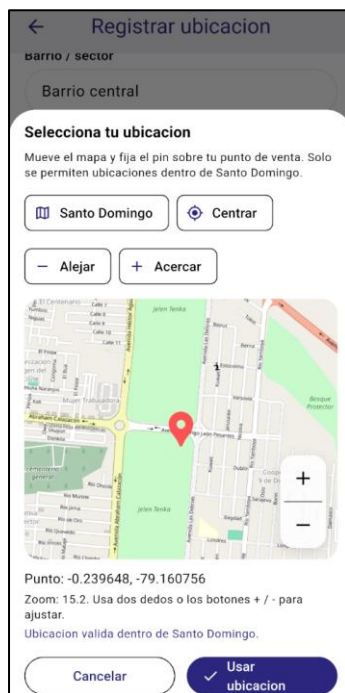
Foto del punto de venta
Obligatoria. Usa una imagen clara. [Cambiar](#)



 Tomar foto

 Elegir de galería

Figura 206. Interfaz móvil en *EntrepreneurLocationPage*, confirmación “Selecciona tu ubicación” al detectar una ubicación previa.



El control inmediato del formulario se concentra en *entrepreneur_location_cubit.dart*, visible en la figura 207. Desde este componente se validan campos, se detectan cambios pendientes y se mantiene ordenado el proceso antes del guardado.

Figura 207. Código de *entrepreneur_location_cubit.dart*: setters y control dirty/fieldErrors.

```
lib > modules > features > entrepreneur_location > presentation > blocs > entrepreneur_location_cubit.dart > EntrepreneurLocationCubit
129 class EntrepreneurLocationCubit extends Cubit<EntrepreneurLocationState> {
189   void setAddress(String value) => emit(
190     state.copyWith(
191       addressText: value,
192       dirty: true,
193       errorMessage: null,
194       saveSuccess: false,
195       fieldErrors: {...state.fieldErrors, 'address': null},
196     ),
197   );
198
199   void setDescription(String value) => emit(
200     state.copyWith(
201       description: value,
202       dirty: true,
203       errorMessage: null,
204       saveSuccess: false,
205       fieldErrors: {...state.fieldErrors, 'description': null},
206     ),
207   );
```

La obtención y persistencia de la ubicación se completan con los casos de uso *get_entrepreneur_location.dart* y *save_entrepreneur_location.dart*, junto con el repositorio y la fuente remota mostrados en las figuras 208, 209, 210 y 211. De este modo, la historia

conserva una separación clara entre la captura en pantalla y la administración de los datos asociados.

Figura 208. Código del caso de uso de *get_entrepreneur_location.dart*

```
lib > modules > features > entrepreneur_location > domain > usecases > get_entrepreneur_location.dart > ...
You, last month | 1 author (You)
1 import '../entities/entrepreneur_location.dart';
2 import '../repositories/entrepreneur_location_repository.dart';
3
You, last month | 1 author (You)
4 class GetEntrepreneurLocation {
5   final EntrepreneurLocationRepository repository;
6   GetEntrepreneurLocation(this.repository);
7
8   Future<EntrepreneurLocation?> call(String uid) {
9     return repository.fetch(uid);
10  }
11 }
```

Figura 209. Código del caso de uso de *save_entrepreneur_location.dart*

```
lib > modules > features > entrepreneur_location > domain > usecases > save_entrepreneur_location.dart > ...
You, last month | 1 author (You)
1 import '../entities/entrepreneur_location.dart';
2 import '../repositories/entrepreneur_location_repository.dart';
3
You, last month | 1 author (You)
4 class SaveEntrepreneurLocation {
5   final EntrepreneurLocationRepository repository;
6   SaveEntrepreneurLocation(this.repository);
7
8   Future<EntrepreneurLocation> call({
9     required String uid,
10    required EntrepreneurLocation location,
11  }) {
12    return repository.save(uid: uid, location: location);
13  }
14 }
```

Figura 210. Código de *entrepreneur_location_repository_impl.dart*

```
lib > modules > features > entrepreneur_location > data > repositories > entrepreneur_location_repository_impl.dart > EntrepreneurLocationRepositoryImpl
10 class EntrepreneurLocationRepositoryImpl
11   implements EntrepreneurLocationRepository {
12   final EntrepreneurLocationRemoteDataSource remote;
13   final LocationMediaUploadDataSource media;
14
15   EntrepreneurLocationRepositoryImpl(
16     required this.remote,
17     required this.media,
18   );
19
20   @override
21   Future<EntrepreneurLocation?> fetch(String uid) {
22     return remote.fetchByUid(uid);
23   }
24
25   @override
26   Future<EntrepreneurLocation> save({
27     required String uid,
28     required EntrepreneurLocation location,
29   }) {
30     final model = _toModel(location, uid: uid);
31     return remote.upsert(uid: uid, payload: model);
32   }
33
34   @override
35   Future<UploadedFile> uploadPhoto({
36     required String uid,
37     required Uint8List bytes,
38     required String contentType,
39   }) {
40     return media.uploadLocationPhoto(
41       bytes: bytes,
42       uid: uid,
43       contentType: contentType,
44     );
45   }
46
47   EntrepreneurLocationModel _toModel(
48     EntrepreneurLocation location, {
49     required String uid,
```

Figura 211. Código de *entrepreneur_location_remote_datasource.dart*: *doc(uid).get()*.

```
lib > modules > features > entrepreneur_location > data > datasources > entrepreneur_location_remote_datasource.dart >
You, last month | 1 author (You)
1 import 'package:cloud_firestore/cloud_firestore.dart';
2
3 import '../../../../../core/config/collections.dart';
4 import '../models/entrepreneur_location_model.dart';
5
6 You, last month | 1 author (You)
7 class EntrepreneurLocationRemoteDataSource {
8   final FirebaseFirestore firestore;
9
10  EntrepreneurLocationRemoteDataSource(this.firestore);
11
12  Future<EntrepreneurLocationModel?> fetchByUid(String uid) async {
13    final ref = firestore.collection(Col.entrepreneurLocations).doc(uid);
14    final doc = await ref.get();
15    if (!doc.exists) return null;
16    return EntrepreneurLocationModel.fromDoc(doc);
17  }
```

4.3.4.1.14. Historia Técnica 1: Seguridad y privacidad de aplicación Móvil

Las funciones *isSignedIn()*, *userRole()*, *isAdmin()* e *isOwner()* definen la seguridad de acceso a los datos mediante la verificación de si la persona ha iniciado sesión, si es propietaria del registro y si posee permisos administrativos. Además, en el bloque `match /users/{userId}`, la actualización del perfil mantiene protegido el campo `role`, lo que evita que la propia usuaria modifique libremente permisos sensibles dentro del sistema. Como se observa en la figura 212.

Figura 212. Lógica de control de acceso por autenticación, propiedad y rol.

```
firestore.rules
2 service cloud_firestore {
3   match /databases/{db}/documents {
4     match /users/{userId} {
5       allow update: if isSignedIn() && (
6         (request.auth.uid == userId
7           || request.resource.data.email == resource.data.email
8           || ('email' in request.auth.token)
9           && request.resource.data.email == request.auth.token.email)
10        || ('emailLowercase' in request.resource.data)
11        || request.resource.data.emailLowercase == resource.data.emailLowercase
12        || ('email' in request.auth.token)
13        && request.resource.data.emailLowercase == request.auth.token.email)
14      )
15      && (
16        (('role' in resource.data) && !('role' in request.resource.data))
17        || (
18          ('role' in resource.data)
19          && ('role' in request.resource.data)
20          && request.resource.data.role == resource.data.role
21        )
22      )
23      && ownerProjectStatusMutationAllowed()
24      && (
25        !('projectEntrepreneurReviewedAt' in resource.data)
26        && !('projectEntrepreneurReviewedAt' in request.resource.data)
27      )
28    }
29  }
30 }
```

La clase *AppCheckInitializer* activa un mecanismo de verificación que comprueba que las solicitudes provengan de una instalación legítima de la *app* según la plataforma utilizada. Esta medida añade una capa de protección previa antes de consumir servicios

remotos, reduciendo el riesgo de uso indebido desde clientes no autorizados. Como se observa en la figura 213.

Figura 213. Lógica de verificación de integridad de la aplicación con *App Check*.

```
lib > modules > core > services > appcheck > app_check_initializer.dart > % AppCheckInitializer
8 class AppCheckInitializer {
9   static Future<bool> activate() async {
17   } else if (io.Platform.isAndroid) {
18     await FirebaseAppCheck.instance.activate(
19       androidProvider:
20         kDebugMode
21           ? AndroidProvider.debug
22           : AndroidProvider.playIntegrity,
23     );
24   } else if (io.Platform.isIOS || io.Platform.isMacOS) {
25     await FirebaseAppCheck.instance.activate(
26       appleProvider:
27         kDebugMode ? AppleProvider.debug : AppleProvider.deviceCheck,
28     );
29   }
30
31   if (kDebugMode && ff.FeatureFlags.appCheckPrintDebugSecretOnBoot) {
32     try {
33       final token = await FirebaseAppCheck.instance.getToken(true);
34       debugPrint('AppCheck debug token: ${token?.substring(0, 10)}...');
35     } catch (e) {
36       debugPrint('AppCheck getToken(true) failed: $e');
37     }
38   }
39   return true;
40 } catch (e) {
41   if (kDebugMode) debugPrint('AppCheck.activate failed: $e');
42   return false;
43 }
```

La función *rateLimit()* regula el número de veces que una operación puede ser intentada durante un periodo específico. La solicitud se bloquea de manera temporal si el número de intentos excede el límite establecido, lo cual contribuye a evitar abusos y la sobrecarga en las funciones delicadas de la aplicación. Como se visualiza en la figura 214.

Figura 214. Lógica de limitación de intentos en funciones sensibles.

```
functions > src > utils > rateLimits > ...
15 export async function rateLimit(
16   ...
40   const key =
41     normalizedSuffix.length > 0 ? `${baseKey}:${normalizedSuffix}` : baseKey;
42
43   const now = Date.now();
44   const ref = db.collection(collection).doc(key);
45   const snap = await ref.get();
46
47   let attempts: number[] = [];
48   if (snap.exists) {
49     const data = snap.data() || {};
50     attempts = (data.attempts as number[] | undefined) ?? [];
51     attempts = attempts.filter((t) => now - t < windowMs);
52   }
53
54   if (attempts.length >= limit) {
55     throw new HttpsError("resource-exhausted", "Too many attempts. Try again later.");
56   }
57
58   // Write only in commit mode.
59   if (mode === "commit") {
60     attempts.push(now);
61     await ref.set({ attempts }, { merge: true });
62   }
63 }
```

La función *requireAdmin(uid)* vuelve a comprobar en el servidor si el perfil de la usuaria corresponde realmente a una administradora o super administradora antes de permitir cambios en configuraciones importantes. De esta forma, la aplicación no confía únicamente en la sesión iniciada, sino que refuerza la autorización en operaciones administrativas sensibles. Como se observa en la figura 215.

Figura 215. Lógica de validación de rol administrativo en funciones críticas.

```

functions > src > functions > catalogSettings.ts > ...
10 async function requireAdmin(uid: string): Promise<void> {
11   if (!uid) throw new HttpsError("unauthenticated", "Auth required");
12   const profile = await db.collection("users").doc(uid).get();
13   if (!profile.exists) {
14     throw new HttpsError("permission-denied", "Admin profile required");
15   }
16   const role = ((profile.data()?.role ?? "") as string).toLowerCase();
17   const isAdmin = role === "admin" || role === "super";
18   if (!isAdmin) {
19     throw new HttpsError("permission-denied", "Admin role required");
20   }
21 }
22
23
24 export const getCatalogSettings = onCall({
25   {
26     region: "us-central1",
27     enforceAppCheck: true,
28     cors: true,
29     invoker: "public",
30   },
31   async (req) => {
32     try {
33       await ratelimit(req.app?.token, {
34         mode: "peek",
35         limit: 40,
36         collection: "config_reads",
37       });
38       const uid = req.auth?.uid || "";
39       await requireAdmin(uid);

```

La función *uploadProfileImage* realiza la carga de imágenes de perfil desde el servidor y no directamente con credenciales expuestas en el dispositivo móvil. Antes de aceptar el archivo, verifica que exista una sesión activa y que la imagen cumpla condiciones de formato y tamaño, fortaleciendo así la protección de los archivos personales y del servicio de almacenamiento. Como se observa en la figura 216.

Figura 216. Lógica de carga segura de imágenes de perfil.

```

functions > src > functions > cloudinaryUploads > uploadProfileImage > onCall() callback
23 export const uploadProfileImage = onCall({
24   {
25     region: "us-central1",
26     cors: true,
27     enforceAppCheck: true,
28     invoker: "public",
29     secrets: [CLD_CLOUD_NAME, CLD_API_KEY, CLD_API_SECRET],
30     timeoutSeconds: 30,
31     memory: "256MiB",
32   },
33   async (req) => {
34     try {
35       await ratelimit(req.app?.token, {
36         mode: "commit",
37         limit: 20,
38         collection: "cid_upload_attempts",
39       });
40
41       const uid = req.auth?.uid || "";
42       if (!uid) throw new HttpsError("unauthenticated", "Auth required");
43
44       const bytesBase64 = (req.data?.bytesBase64 ?? "").toString();
45       const contentType = (req.data?.contentType ?? "").toString();
46       const fileName = (req.data?.fileName ?? "avatar.jpg").toString();
47
48       if (!bytesBase64) throw new HttpsError("invalid-argument", "bytesBase64 required");
49       if (!/image\/(jpeg|png)$/i.test(contentType)) {
50         throw new HttpsError("invalid-argument", "Only JPEG/PNG allowed");
51       }
52
53       const buf = Buffer.from(bytesBase64, "base64");
54       if (buf.length) throw new HttpsError("invalid-argument", "Invalid base64");
55       if (buf.length > 5 * 1024 * 1024) {
56         throw new HttpsError("failed-precondition", "Image too large (>5MB)");
57       }

```

La función *rebuildPublicEntrepreneur* construye una versión pública del perfil de cada emprendedora usando solo la información necesaria para mostrarla en listados o mapas. Esta separación evita exponer directamente todos los datos internos de la usuaria y contribuye a una mejor protección de su privacidad dentro de la aplicación. Como se ve en la figura 217.

Figura 217. Lógica de construcción de perfiles públicos con datos controlados.

```

functions > src > functions > publicEntrepreneur > ...
129 export async function rebuildPublicEntrepreneur(uid: string): Promise<void> {
130
131   const profileIsActive = profileData?.["isActive"] !== false;
132   const profileIsVisible = profileData?.["isVisible"] !== false;
133   const visibleInProfiles =
134     approvalStatus === "approved" && profileIsActive && profileIsVisible;
135   const visibleInMap =
136     visibleInProfiles &&
137     hasLocation &&
138     hasGeo &&
139     coverageRegion == ACTIVE_REGION.id;
140
141   const payload: Record<string, unknown> = {
142     uid: normalizedUid,
143     name:
144       firstNonEmpty([
145         trimOrNull(profileData?.["name"]),
146         trimOrNull(userData?.["fullName"]),
147         trimOrNull(userData?.["name"]),
148         trimOrNull(userData?.["displayName"]),
149         trimOrNull(applicant?.["fullName"]),
150         trimOrNull(userData?.["businessName"]),
151         trimOrNull(applicant?.["businessName"]),
152       ]) ?? "Emprendedora",
153     approvalStatus,
154     visibleInProfiles,
155     visibleInMap,
156     hasLocation,
157     hasGeo,
158     coverageRegion,
159     ...profileData?.["businessData"],
160   };
161
162   await firestore.collection("publicEntrepreneurs").doc(normalizedUid).set(
163     payload, { merge: true }
164   );
165 }

```

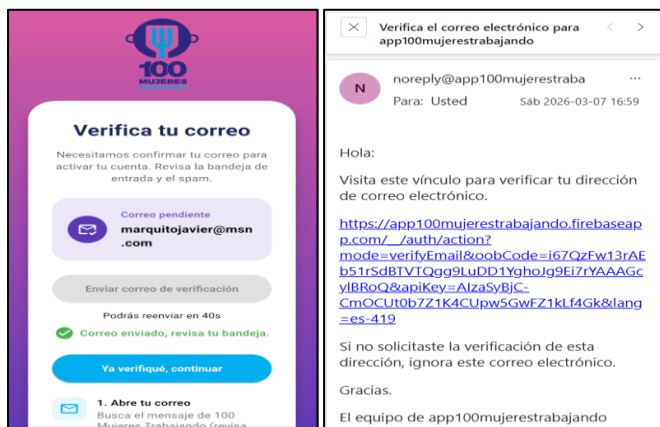
La clase *EmailVerificationCubit* gestiona la verificación de cuenta por correo electrónico. Allí se controla el envío del mensaje de verificación, la comprobación posterior del estado *emailVerified* y, de manera complementaria, un período temporal de gracia para permitir el acceso limitado mientras la persona usuaria completa la confirmación de su cuenta. De esta forma, la aplicación refuerza la seguridad del acceso sin interrumpir completamente la continuidad de uso. Como se observa en la figura 218 y figura 219.

Figura 218. Lógica de verificación de cuenta por correo electrónico.

```

218 class EmailVerificationCubit extends Cubit<EmailVerificationState> {
219
220   Future<void> refreshStatus() async {
221     emit(state.copyWith(checking: true, error: null));
222     try {
223       await _reloadUser();
224       await _syncCurrentUserEmailState();
225       final verified = _auth.currentUser?.emailVerified ?? false;
226       DateTime? deadline = state.graceDeadline;
227       bool graceUsed = state.graceUsed;
228       String? errorMessage;
229       final requiresVerification = requiresEmailVerification(state.email);
230       if (requiresVerification) {
231         emit(
232           state.copyWith(
233             checking: false,
234             verified: true,
235             graceDeadline: null,
236             graceUsed: false,
237             error: null,
238           ),
239         );
240       }
241       return;
242     } catch (e) {
243       emit(
244         state.copyWith(
245           checking: false,
246           verified: false,
247           graceDeadline: null,
248           graceUsed: false,
249           error: e.toString(),
250         ),
251       );
252     }
253   }
254
255   if (verified && state.userId != null) {
256     await _graceStorage.clearDeadline(state.userId);
257     await _graceStorage.clearGraceUsage(state.userId);
258     await _updateGrace(deadline: null, used: false);
259     deadline = null;
260     graceUsed = false;
261   } else if (!verified) {
262     // ...
263   }
264 }

```

Figura 219. Interfaz móvil de verificación del correo mediante *link*.

4.3.4.2. Sprint 2 – Gráfico de trabajo pendiente

Fue posible gestionar las actividades del segundo conjunto de historias a través de *Jira Software*. El avance de las tareas del equipo se observó mediante el gráfico del trabajo pendiente, considerando los puntos de historia que se finalizaron en ciertos días. El gráfico del segundo *sprint* se muestra en la figura 220.

Figura 220. Gráfica de trabajo restante del 2do *Sprint* en *Jira*.



4.3.4.3. Sprint 2 - Revisión (*Sprint Review*)

Después de finalizar el segundo *sprint*, el equipo llevó a cabo la reunión de revisión en presencia de la gerente de la compañía "100 Mujeres", con el mismo objetivo que tenía la del primer *sprint*. Esta fase se enfocó en finalizar los módulos de administración de pagos, notificaciones y comunicación, consulta y trazabilidad de pedidos, planificación y logística de entregas, gestión territorial, y comunidad y visibilidad de emprendedoras, integrando componentes de mapa, perfiles y contenidos informativos para fortalecer la experiencia de seguimiento y participación en la plataforma.

Para el seguimiento de las tareas fue gestionado nuevamente a través de *Jira*, y se vigiló el progreso, por medio del gráfico de trabajo restante del *sprint* 2. Confirmando que todas las funcionalidades presentadas cumplieran con los criterios de aceptación.

4.3.4.4. Sprint 2 - Retrospectiva (*Sprint Retrospective*)

Se tuvieron en cuenta tres preguntas para la retrospectiva del segundo *sprint*, las cuales permitirán obtener una visión clara de cómo se desarrolló el *sprint*. Como se muestra en la tabla 22

Tabla 22. Sprint 2 retrospectiva

Aspectos exitosos del sprint	Mejoras para aplicar	Errores en el sprint
Se consolidaron funciones clave para operación y experiencia, con verificación de pagos, notificaciones, consulta por fecha y gestión de entregas y zonas.	Conviene reforzar la integración entre módulos nuevos, cerrar reglas y permisos pendientes y pulir la claridad de filtros y pasos en flujos críticos.	Se detectaron permisos o reglas incompletas en componentes, y ajustes de integración y detalles de usabilidad que afectaron algunos recorridos.

4.3.5. Sprint 3

4.3.5.1. Sprint 3 – Planificación (*Sprint Planning*)

Las historias de usuario que se han considerado para el desarrollo del tercer *sprint backlog*, las cuales tienen una probabilidad alta de ser desarrolladas, están detalladas en la tabla 23. También se llevaron a cabo las tareas de cada historia de usuario que se ejecutan en este *sprint*, para lograr el acceso a la aplicación *web*.

Tabla 23 Sprint Backlog 3

Sprint Backlog					
Sprint 3					
Objetivo: Implementar las funciones básicas para el aplicativo <i>web</i> , tales como <i>EmprendedoraLogin</i> , <i>AdminLogin</i> , <i>MenúPrincipal</i> , <i>Register</i> , <i>MiPerfil</i> .					
Historia	Est	Categoría	Tarea	Est	Estado
H1 [Web] – Menú Principal – Llamado a la Acción (<i>app</i>)	5	Desarrollo	Crear menú de acceso mediante comprobación de <i>Backend</i>	2	Completado
		Desarrollo	Crear llamado a la acción desde el <i>FrontEnd</i> (<i>app.jsx</i>)	1.5	Completado
		Diseño	Diseño de para la continuidad del objetivo (<i>app.css</i> , <i>styles.js</i>)	0.5	Completado
		Pruebas	Pruebas de corroboración y acceso	1	Completado
H2 [Web] – Registro de Emprendedoras (<i>register</i>)	3	Desarrollo	Crear registro con lógica para validación (<i>register.jsx</i>)	1	Completado
		Diseño	Crear interfaz y diseño de registro (<i>styles.js</i>)	1	Completado
		Pruebas	Registro con registros mediante cédula o correo	1	Completado
H3 [Web] - Login de Emprendedoras (<i>EmprendedoraLogin</i>)	4	Desarrollo	Codificar <i>Frontend</i> de Emprendedoras (<i>EmprendedoraLogin.jsx</i>)	1.5	Completado
		Diseño	Crear interfaz y diseño <i>login</i> de emprendedoras (<i>styles.js</i>)	0.5	Completado
		Desarrollo	Integrar validación mediante <i>Backend</i> y <i>BDD</i> (<i>mysql</i>)	2	Completado

Figura 222. Componente *handleMouseMove.jsx*

```

> App.jsx > ...
4 const Home = () => {
8   const handleMouseMove = (e) => {
13 };
14
15 return (
16   <div style={homeStyles.homeContainer}>
17     {}
18     <nav style={homeStyles.navBar}>
19       <img src={logo} alt="Logo" style={homeStyles.logo} />
20       <div style={homeStyles.navLinks}>
21         <button
22           style={homeStyles.navButtonEmprendedora}
23           onClick={() => navigate('/emprendedora-login')}
24         >
25           Soy Emprendedora
26         </button>
27         <button
28           style={homeStyles.navButtonAdmin}
29           onClick={() => navigate('/admin-login')}
30         >
31           Soy Administradora
32         </button>
33       </div>
34     </nav>

```

Para lo realizado en la figura 223, requirió la creación de la interfaz por medio del uso de estilos CSS simples y con el uso de una función llamada *handleMouseMove*, para que la imagen del fondo se mueva conforme se mueve el *Mouse* encima con el logo de la Empresa 100 Mujeres. Y brindando una experiencia de llamada a la acción mediante 2 opciones, para la ventana de *AdminLogin.jsx* o *EmprendedoraLogin.js* respectivamente. Y también dando la tercera alternativa que si es nueva en el proyecto proceda al registro para llenar el formulario tal y como se muestra en las capturas de los códigos anteriores.

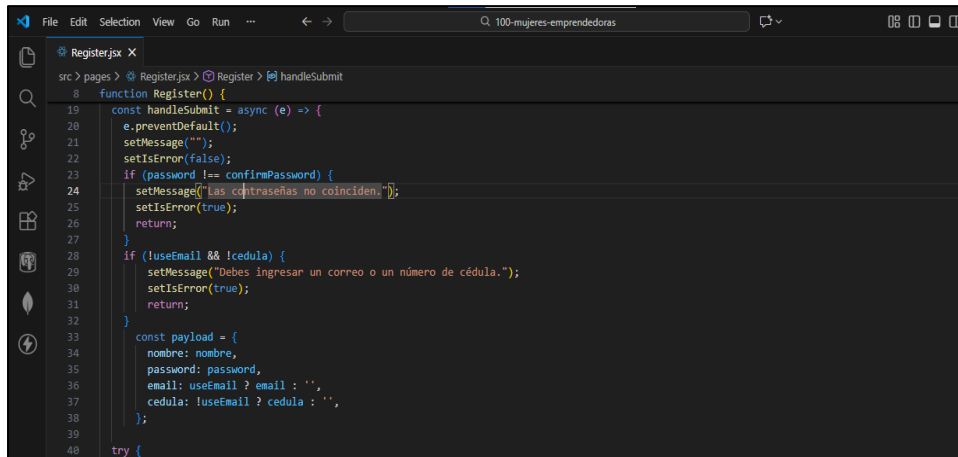
Figura 223. Interfaz de *App.jsx* de vista *web*

4.3.5.1.2. Historia de Usuario 2: Registro de Emprendedoras

Así como se muestra en el anexo 4, dentro del código se dejó adelantado el trabajo en el cual se sincroniza el *backend* con el *frontend*, creando así la importación de componentes tales como *useNavigate*, *React*, y también *axios*. De los cuales generan el punto de partida para entablar con el *backend* mediante *Node.js* recibiendo como parámetro

post el campo a su elección tal y como se indica en la figura 224, se brinda la opción de ingresar mediante correo personal o cédula y así lograr culminar el proceso de registro de las emprendedoras.

Figura 224. Código de *register.jsx*



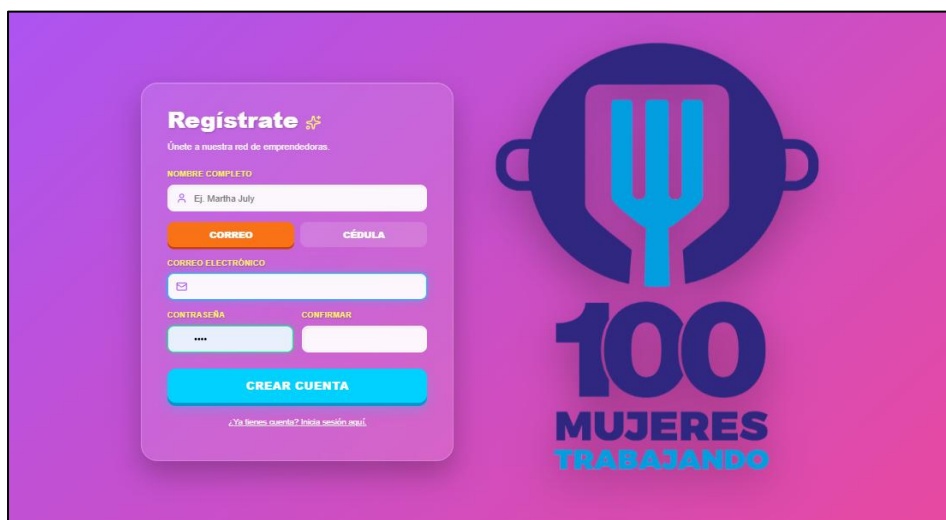
```

8 function Register() {
19   const handleSubmit = async (e) => {
20     e.preventDefault();
21     setMessage("");
22     setIsError(false);
23     if (password !== confirmPassword) {
24       setMessage("Las contraseñas no coinciden.");
25       setIsError(true);
26       return;
27     }
28     if (!useEmail && !cedula) {
29       setMessage("Debes ingresar un correo o un número de cédula.");
30       setIsError(true);
31       return;
32     }
33     const payload = {
34       nombre: nombre,
35       password: password,
36       email: useEmail ? email : "",
37       cedula: useEmail ? cedula : "",
38     };
39     try {

```

Para el desarrollo de la plantilla web de registro tal y como se ve en la siguiente figura 225, se empleó varios recursos de CSS, tomando en cuenta que un gran número de emprendedoras no cuentan con correo electrónico activo, se optó por dejar como opcional y dándole la alternativa de que puedan ingresar con la cédula como parámetro de autenticación principal, ajustando la función `!useEmail && !cedula` para determinar si es con cualquiera de las opciones existentes dentro de la variable `async handleSubmit`. para comparar si las contraseñas coinciden entre sí o para recibir como parámetro la cédula o el correo electrónico.

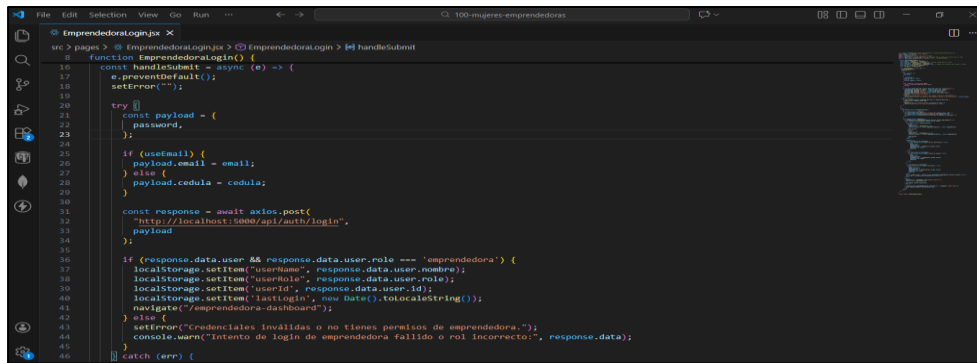
Figura 225. Ventana *Register.jsx*



4.3.5.1.3. Historia de Usuario 3: Login de Emprendedora

Tal como se muestra en el anexo 4, en esta sección del código para empezar se define una función llamada *EmprendedoraLogin* la cual se caracteriza por la definición de todos los campos que se irán agregando en el formulario, seguido a eso se crea el componente *handleSubmit* en el cual ingresan todos los datos y se redirige al servidor para la decisión de eso según corresponda, después se utiliza la estructura visual (UI) cuyo componente es utilizado por JSX, mediante el contenedor principal *fullPageContainer* con la aplicación de dos paneles y el degradado correspondiente a la paleta de colores establecida, tal y como se indica en la figura de *Register.jsx*. Se ve, en la figura 226 y 227.

Figura 226. Código de *EmprenderoraLogin.jsx*



```

function EmprenderoraLogin() {
  const handleSubmit = async (e) => {
    e.preventDefault();
    setError("");

    try {
      const payload = {
        password,
      };

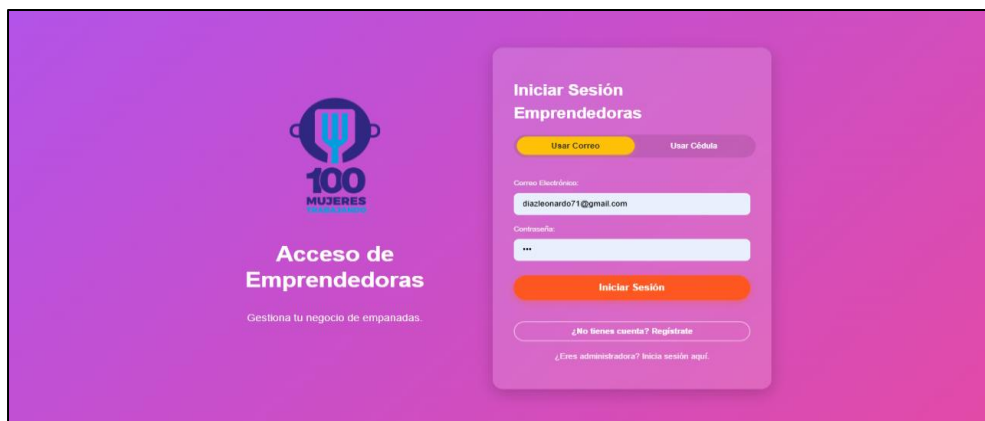
      if (useEmail) {
        payload.email = email;
      } else {
        payload.cedula = cedula;
      }

      const response = await axios.post(
        "http://localhost:5000/api/auth/login",
        payload
      );

      if (response.data.user && response.data.user.role === 'emprendedora') {
        localStorage.setItem('userName', response.data.user.nombre);
        localStorage.setItem('userRole', response.data.user.role);
        localStorage.setItem('userId', response.data.user.id);
        localStorage.setItem('lastLogin', new Date().toLocaleString());
        navigate('/emprendedora-dashboard');
      } else {
        setError('Credenciales inválidas o no tienes permisos de emprendedora');
        console.warn('Intento de login de emprendedora fallido o rol incorrecto', response.data);
      }
    } catch (err) {
  
```

Seguido a eso se construyó la ventana de *EmprenderoraLogin.jsx* tal y como la muestra la figura *Ventana de EmprenderoraLogin.jsx*, cuyos componentes *useState* y *useNavigate* se usaron como método de ingreso del contenido de los campos y para saltar de distintas paginas a otras respectivamente. También se utilizó el componente *toggleContainer* para poder distinguir entre la cédula o correo, según sea el caso.

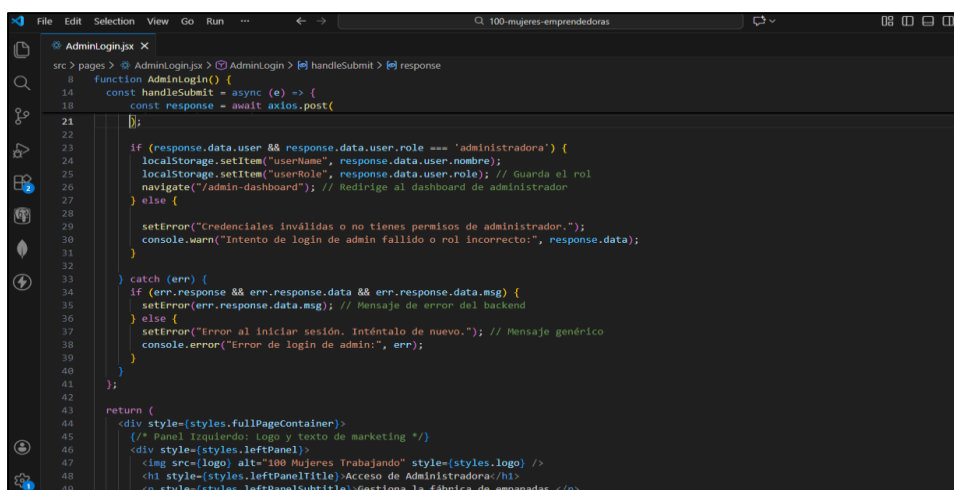
Figura 227. Ventana de *EmprenderoraLogin.jsx*



4.3.5.1.4. Historia de Usuario 4: Login Administradora

Tal como se muestra en el anexo 4. En la figura 228 y 229, *AdminLogin.jsx* en la aplicación *web* crea una función llamada *AdminLogin* la cual se caracteriza por la definición de los campos que se irán agregando en el formulario, sumado a eso se construye el componente *handleSubmit* en el cual ingresan todos los datos y se envía al servidor para corroborar que se trate de las credenciales del administrador y que no sea una emprendedora ya que la información es confidencial, después se utiliza la estructura visual (UI) cuyo componente es utilizado por JSX, mediante el contenedor *fullPageContainer* con la aplicación paneles derecho e izquierdo y el degradado orientado a la paleta de colores anteriormente establecida, similar a la figura del código de *EmprenderoraLogin.jsx*.

Figura 228. Código de *AdminLogin.jsx*

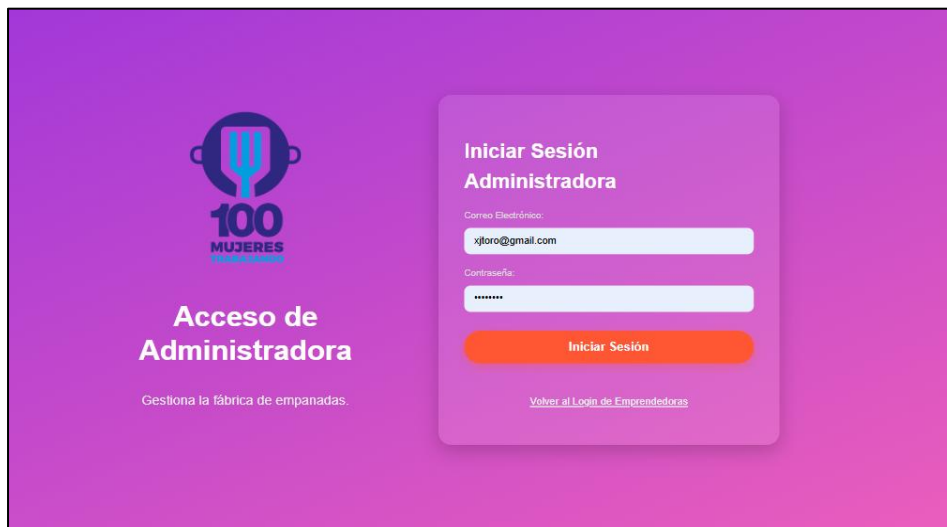


```

src > pages > AdminLogin.jsx > AdminLogin > handleSubmit > response
8  function AdminLogin() {
14  const handleSubmit = async (e) => {
18  const response = await axios.post(
21  );
22
23  if (response.data.user && response.data.user.role === 'administradora') {
24    localStorage.setItem("userName", response.data.user.nombre);
25    localStorage.setItem("userRole", response.data.user.role); // Guarda el rol
26    navigate("/admin-dashboard"); // Redirige al dashboard de administrador
27  } else {
28    setError("Credenciales inválidas o no tienes permisos de administrador.");
29    console.warn("Intento de login de admin fallido o rol incorrecto:", response.data);
30  }
31
32  } catch (err) {
33    if (err.response && err.response.data && err.response.data.msg) {
34      setError(err.response.data.msg); // Mensaje de error del backend
35    } else {
36      setError("Error al iniciar sesión. Inténtalo de nuevo."); // Mensaje genérico
37      console.error("Error de login de admin:", err);
38    }
39  }
40  };
41
42  return (
43    <div style={styles.fullPageContainer}>
44      <div style={styles.leftPanel}>
45        <img src={logo} alt="100 Mujeres Trabajando" style={styles.logo} />
46        <h1 style={styles.leftPanelTitle}>Acceso de Administradora:</h1>
47        <p style={styles.leftPanelSubtitle}>Gestiona la fábrica de empanadas.</p>

```

A continuación, se desarrolló la ventana de *AdminLogin.jsx*, cuyos componentes *useState* y *useNavigate* se usaron como método de ingreso no sin antes pasar por el siguiente condicional *response.data.user && response.data.user.role === 'administradora'* y se utilizó para diferenciar entre los roles según la *BDD mysql*, a tal punto que sin esa condición se tiene inseguridad en cuanto a la información. Tal y como se mencionó en la figura del *EmprenderoraLogin.jsx*, mientras allá verifica las credenciales asignadas por la administradora, acá se verifica si el rol es el correcto con el campo *userRole*.

Figura 229. Ventana *AdminLogin.jsx*

4.3.5.1.5. Historia de Usuario 5: Perfil de Emprendedora

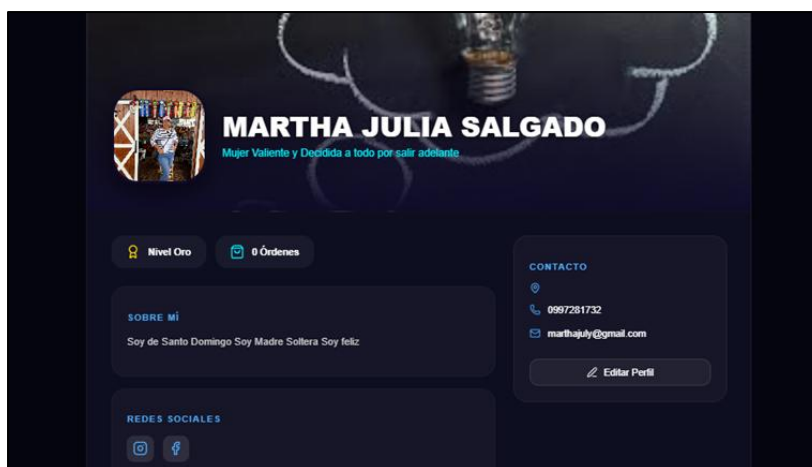
Tal como se muestra en el anexo 4. En esta sección se empleó componentes tales como el *hook* *useState* y la función *useEffect* la cual se utilizó para la dinámica de la interfaz y también para traer la información desde el servidor para el posterior a eso validar la misma mediante el uso de datos no obligatorios respectivamente, tales como *setIsEditing(true)* y *setIsMandatory(true)*, caso contrario procede a bloquear la navegación, mientras no esté completa la información. Y la función principal para la persistencia de los datos es *handleSaveClick* utilizó *axios.put* para el envío de datos hacia el *backend*, y también *react-toastify* a manera de *FeedBack* para indicar alertas visuales de autenticación y error. Esto se refleja en las figuras 230 y 231.

Figura 230. Código de *MiPerfil.jsx*

```
src > components > MiPerfil.jsx > modernStyles > glassCard
76 function MiPerfil({ userId }) {
85   useEffect(() => {
86     const fetchPerfil = async () => {
87       };
88     if (userId) fetchPerfil();
89   }, [userId]);
90
91   const handleSaveClick = async () => {
92     if (!formData.descripcion || !formData.ciudad || !formData.telefono) {
93       toast.error("Todos los campos marcados son obligatorios.");
94       return;
95     }
96     setLoading(true);
97     try {
98       const updateData = new FormData();
99       updateData.append('descripcion', formData.descripcion);
100      updateData.append('ciudad', formData.ciudad);
101      updateData.append('telefono', formData.telefono);
102      if (fotoPerfilFile) updateData.append('fotoPerfil', fotoPerfilFile);
103
104      const response = await axios.put('http://localhost:5000/api/emprendedoras/${userId}', updateData);
105
106      setPerfil(response.data);
107      setIsEditing(false);
108      setIsMandatory(false);
109      toast.success('Perfil actualizado con éxito!');
110    } catch (err) {
111      toast.error('Error al guardar. ');
112    } finally {
113      setLoading(false);
114    }
115  }
116 }
```

Y para el desarrollo del diseño (UI/UX) de la interfaz se empleó el efecto `backdropFilter: 'blur(20px)'` para dar el efecto de cristal esmerilado, más que todo para darle un toque de perfil profesional a cada emprendedora, sumado a eso el uso del componente `useEffect` con `axios` que gestiona el ciclo de vida de la información mediante *React Hooks*. También se utilizó la variable `isMandatory` para persuadir a la emprendedora a llenar toda la información que se requiere y como para finalizar de utilizó *React Toastify* para notificar al usuario que la actualización se logró sin novedades.

Figura 231. Ventana de *MiPerfil.jsx*



4.3.5.2. Sprint 3 – Gráfico de trabajo pendiente

Como se visualiza en la figura 232, podemos observar ahora en el gráfico del *sprint* 3 mediante el uso de la herramienta *Jira Software* que la pendiente avanza gradualmente conforme se llevó al cabo el desarrollo del aplicativo *Web*. Poniendo en evidencia el tiempo en el cual se inició y se logró su culminación, dejando claro los días o meses en los que vas se tuvo avances significativos.

Figura 232. Gráfica del 3er Sprint en Jira.



4.3.5.3. Sprint 3 - Revisión (*Sprint Review*)

Una vez finalizado el *sprint* 3 se procedió a la revisión en compañía de la administradora del proyecto, con la finalidad de evidenciar los requerimientos y requisitos, que se establecieron en su momento, dejando en claro el proceso paso a paso, observado en la gráfica y en la práctica, empezando desde la pantalla de inicio con la llamada de acción, seguido a eso el *login* de emprendedora, *login* de administración, registro de emprendedora y el perfil de emprendimiento.

Mediante el uso de la herramienta *Jira* se manejó y se observó el proceso a cabalidad, y con la gráfica se logra apreciar de mejor manera, quedando así abierto el espacio para el último *sprint*.

4.3.5.4. Sprint 3 - Retrospectiva (*Sprint Retrospective*)

Durante la finalización de la fase del *sprint* 3 se logró una interacción efectiva, lo cual generó pautas descriptivas en la cual resalta la siguiente información de la tabla 24.

Tabla 24. Sprint 3 retrospectiva

Aspectos exitosos del <i>sprint</i>	Mejoras para aplicar	Errores en el <i>sprint</i>
Se consolidó el ingreso efectivo mediante la categoría, y su funcionalidad, sea para emprendedora o administradora, creando así una diferenciación lógica de prioridades e información.	Mejoramiento de alertas de mensajes cuando un usuario ingresa a su cuenta mediante seguridad de patrones e interconexión tipo red social del perfil de cada emprendedora,	Corregir las dimensiones de la foto de perfil que suben los emprendedores, ya que al no tener un agende de medición, las fotos pierden las medidas originales y se tiene cierta distorsión de esta.

4.3.6. Sprint 4

4.3.6.1. Sprint 4 – Planificación (*Sprint Planning*)

Las tablas 25 y 26 presentan en detalle las historias técnicas e historias de usuario que se consideraron para el desarrollo del cuarto *sprint backlog*, todas ellas con una alta probabilidad de ser desarrolladas. De manera similar, se realizaron varias tareas para cada historia de usuario que conformaban el *sprint*.

Tabla 25. Sprint Backlog 4

Sprint Backlog					
Sprint 4					
Objetivo: Implementar funciones básicas para el aplicativo web, tales como <i>RealizarPedido</i> , <i>RegistrarPago</i> , <i>HistorialPedidos</i> , <i>AdminDashboard</i> , <i>EmprendedoraDashboard</i> , <i>AdminResumen</i> , <i>AdminGestionPedidos</i> , <i>AdminGestiónProductos</i> , y <i>AdminGestiónUsuarios</i> .					
Historia	Est	Categoría	Tarea	Est	Estado
H6 [Web] – Realizar Pedido	5	Desarrollo	Seleccionar menú mediante calculo al instante del total (<i>RealizarPedido.jsx</i>)	2	Completado
		Diseño	Diseño de carrito de compras y tarjetas de productos (<i>styles.js</i> y <i>RealizarPedido.jsx</i>)	1.5	Completado
		Pruebas	Pruebas de envío y verificación del <i>Backend</i>	1.5	Completado
H7 [Web] – Registro de pagos	4	Desarrollo	Ingresar el pago y establecer la ruta y fecha del envío (<i>RegistrarPago.jsx</i>)	2	Completado
		Diseño	Corroboración visual de pago, estado del mismo y estilos (<i>styles.js</i>)	2	Completado
H8 [Web] – Historial de Pedidos	3	Desarrollo	Lista y consulta de pedidos mediante filtro <i>userId</i> (<i>HistorialPedidos.jsx</i>)	1.5	Completado
		Diseño	Tablas de pedidos con sus estados	1.5	Completado
H9 [Web] – Administrador <i>Dashboard</i>	5	Desarrollo	Configurar panel principal con acceso limitado y navegación parcial (<i>AdminDashboard.jsx</i>)	2	Completado
		Diseño	<i>Layout</i> administrativo con temática oscura y <i>sidebar</i> .	3	Completado
H10 [Web] – <i>Admin</i> Gestión Pedidos	5	Desarrollo	Crear sistema de cambio de estados de pedido y validación de pagos. (<i>AdminGestionPedidos.jsx</i>)	3	Completado
		Pruebas	Simulación de flujo desde el pedido a la aceptación de pago.	2	Completado
H11 [Web] – <i>Admin</i> Gestión Productos	4	Desarrollo	<i>CRUD</i> es decir nombre, precio, <i>stock</i> . (<i>AdminGestionProductos.jsx</i>)	2.5	Completado
		Diseño	Edición de productos y añadidura de variedades	1.5	Completado
H12 [Web] – <i>Admin</i> Gestión Usuarios	3	Desarrollo	Lista de Emprendedoras y control de cuentas. (<i>AdminGestiónUsuarios.jsx</i>)	2	Completado
		Diseño	Buscador de usuarios y filtros por zonas.	1	Completado

Tabla 26. Sprint Backlog Técnico Web

Historia	Est	Categoría	Tarea	Est	Estado
HT-2: Seguridad y Privacidad de la Aplicación Web	8	Desarrollo	Implementar validación de autenticación y control de roles en <i>login</i> de administrador	2	Completado
		Desarrollo	Implementar proceso de <i>hash</i> seguro de contraseñas en el registro	1	Completado

Desarrollo	Implementar control de acceso a módulos administrativos mediante validación de rol	2	Completado
Desarrollo	Implementar limpieza segura de sesión al cerrar sesión	1	Completado
Desarrollo	Implementar validación de tipo de archivo en carga de comprobantes de pago	1	Completado
Desarrollo	Implementar validación de estado activo de cuenta durante autenticación	1	Completado

Una vez que llegó a su finalización así el *sprint* 3, se llevó al cabo la realización del *sprint* 4, tomando en cuenta el grado de dificultad que se presentó en anterior, del mismo modo se utilizó la herramienta *Jira* para darle el seguimiento correcto, tal como se mostró en la figura 232 indicada anteriormente

4.3.6.1.1. Historia de Usuario 6: Realizar Pedido

Como se evidencia en el anexo 4, para el código se usa la importación de componentes tales como *useNavigate*, *React*, y también *axios*. De los cuales generan el punto de partida para entablar con el *backend* mediante *Node.js* recibiendo como parámetro *post* el campo a su elección tal y como se indica en la figura 233 y 234, brindando la opción seleccionar el pedido a su elección libre. Añadiendo como factor fundamental *floatingCart* para ser empleado como carrito de compras de forma dinámica y con un botón de pago.

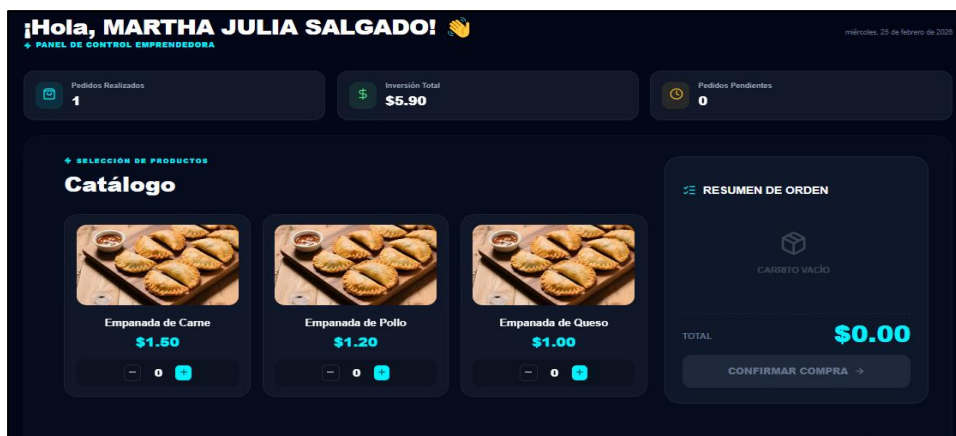
Figura 233. Código de *RealizarPedido.jsx*

```

114 function RealizarPedido({ userId }) {
115   useEffect(() => {
116     const fetchProductos = async () => {
117       const response = await axios.get('http://localhost:5000/api/productos');
118       setProductos(response.data.map(p => ({
119         ...p,
120         precio: parseFloat(p.precio) || 0,
121         imagenDisplay: empanadaImagenUnica
122       })));
123     };
124     fetchProductos();
125   }, []);
126
127   useEffect(() => {
128     let total = 0;
129     Object.keys(pedidoActual).forEach(id => {
130       const p = productos.find(item => item.id === parseInt(id));
131       if (p) total += p.precio * pedidoActual[id];
132     });
133     setTotalPedido(total);
134   }, [pedidoActual, productos]);
135
136   const cambiarCantidad = (id, delta) => {
137     setPedidoActual(prev => {
138       const nuevaCant = (prev[id] || 0) + delta;
139       const nuevo = { ...prev };
140       if (nuevaCant > 0) nuevo[id] = nuevaCant;
141       else delete nuevo[id];
142       return nuevo;
143     });
144   };
145 }
  
```

Con un diseño futurista y a la vanguardia se abre paso la ventana de Registro de Pedidos mediante el uso del carrito de compras *backdropFilter: 'blur(25px)'* cuya selección de colores le da ese tono de mejorías. Y respecto usuario *XU*, se integró las siguientes opciones tales como *react-toastify* y contenido de manera general en rejillas aceptables llamadas *display: grid* que organiza las opciones para que exista simetría.

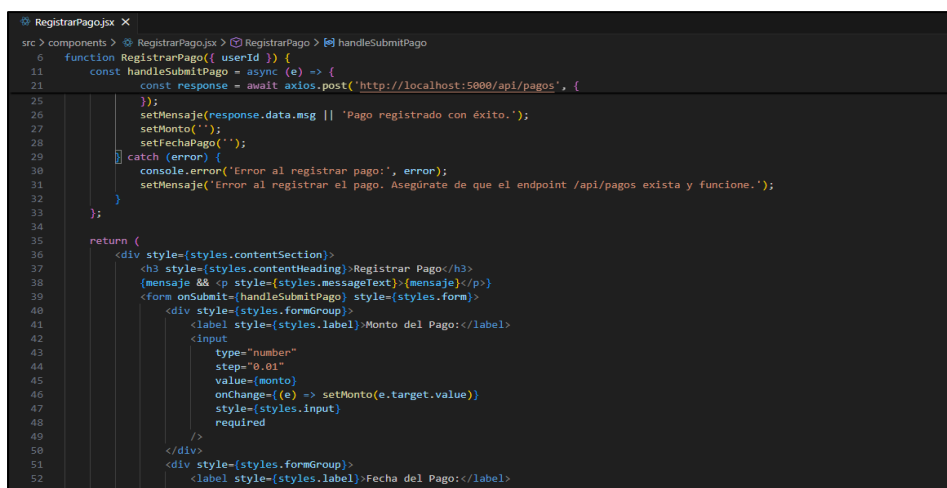
Figura 234. Ventana de *RegistrarPedido.jsx*



4.3.6.1.2. Historia de Usuario 7: Comprobante & Ruta

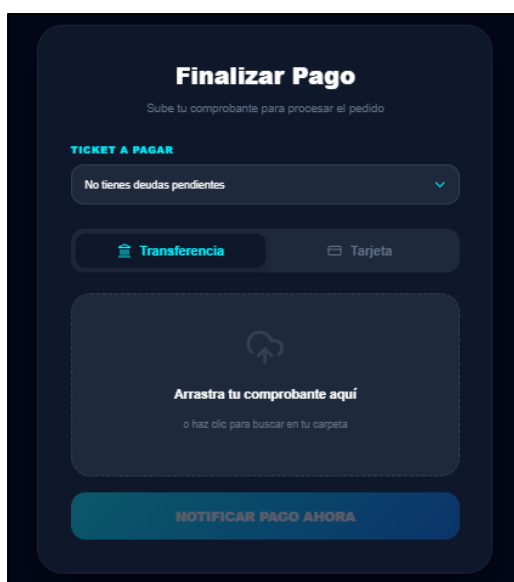
De acuerdo con lo presentado en el anexo 4, para el desarrollo de esta sección se utilizó los estados del *React useState* que maneja la interactividad del formulario, mediante 3 parámetros básicos como el monto, fecha de pago y el mensaje, gestionando el *feedback*, indicando mensajes de éxito o de error. Lo más relevante es la función *handleSubmitPago*, la cual valida la información que ingresa y utiliza aquella que ya se encuentra en el *backend* mediante el fragmento *axios.post*. En la figura 235 y 236, se detallan lo realizado.

Figura 235. Ventana de Registro de pagos



En la estructura del código se estableció un parámetro importante en el diseño, tales como *styles.contentSection*, cuya función es ser un contenedor que unifica el fondo y los márgenes principales, también se utilizaron botones como en nuestro caso *styles.actionButton*, para redirigir la entrada de datos o de envío de parámetros. Se tiene previsto añadir un buscador de comprobantes mediante la fecha y el monto, para tener un control detallado con los pagos de las emprendedoras. Y así darle seguimiento, y también brinda la opción para pagar mediante tarjeta de débito o crédito, pero sería a futuro. Ya que de momento gran porcentaje de emprendedoras cancelan mediante transferencia o depósitos directo a la cuenta de la empresa.

Figura 236. Ventana de Registro de Pagos



4.3.6.1.3. Historia de Usuario 8: Historial de Pedidos

Según lo expuesto en el anexo 4, para el desarrollo del módulo o segmento de historial de usuario, fue fundamental en el código abarcar todo lo relacionado a los pedidos en tiempo real, mediante el uso de *useEffect*, se logró una actualización constante y eso se logró con el parámetro *setInterval* mediante un tiempo establecido de cada 8 segundos. Así mismo la *BDD* también se vio actualizada, esto gracias a la comunicación constante con el *backend*, también se implementó el uso de una función de un mensaje personalizado llamado *mostrarMensaje100Mujeres* que se importó para los módulos en que se vio la necesidad de usarlo.

También se usó *FormData* para implementar el envío del comprobante de pago de cada pedido de la emprendedora, y se usó dinamismo de datos mediante el uso de *detalles.forEach*, lo cual sirvió para ayudar en la creación del *ticket* de pedido, mediante la función *generarTicket* con la librería *jsPDF*. Tal como se observa en la figura 237 y 238.

Figura 237. Código de *HistorialPedidos.jsx* – Parte 1

```
src > components > HistorialPedidos.jsx > HistorialPedidos
1 import React, { useState, useEffect } from 'react';
2 import axios from 'axios';
3 import { emprendedoraStyles as styles } from '../styles/styles';
4
5 function HistorialPedidos({ userId }) {
6   const [historial, setHistorial] = useState([]);
7   const [mensaje, setMensaje] = useState("");
8
9   useEffect(() => {
10     const fetchHistorial = async () => {
11       if (!userId) {
12         setMensaje('No se pudo cargar el historial: ID de usuario no disponible.');
```

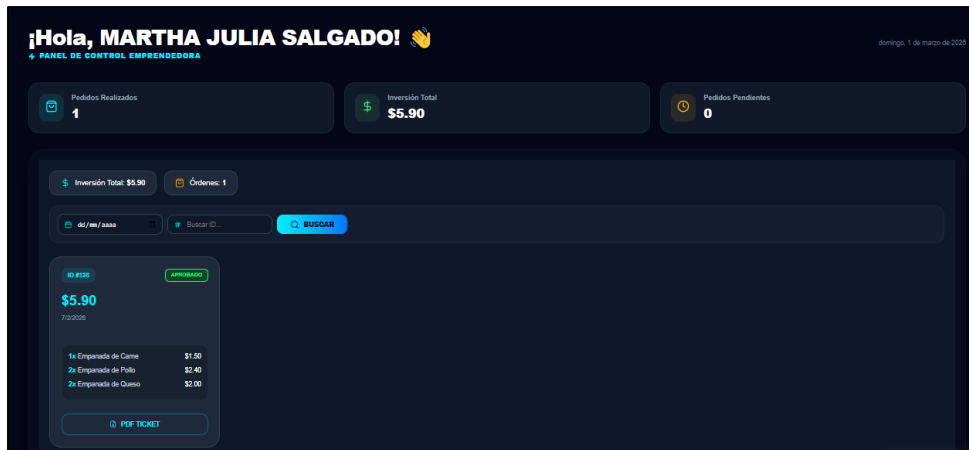
Figura 238 Código de *HistorialPedidos.jsx* – Parte 2

```
src > components > HistorialPedidos.jsx > ...
10 function HistorialPedidos({ userId }) {
11   ...
12   const manejarEnvioComprobante = async () => {
13     ...
14   };
15
16   const generarTicket = (pedido) => {
17     const doc = new jsPDF({ unit: 'mm', format: [80, 150] });
18     doc.setFillColor(15, 23, 42);
19     doc.rect(0, 0, 80, 20, 'F');
20     doc.setTextColor(0, 242, 254);
21     doc.setFontSize(14);
22     doc.setFont("helvetica", "bold");
23     doc.text("100 MUJERES", 40, 10, { align: "center" });
24     doc.setFontSize(8);
25     doc.text("TRABAJANDO", 40, 15, { align: "center" });
26
27     doc.setTextColor(40, 40, 40);
28     doc.setFontSize(10);
29     doc.text("TICKET DE COMPRA #${pedido.id}", 10, 30);
30     doc.setFontSize(8);
31     doc.text("Fecha: ${new Date(pedido.fecha_pedido).toLocaleString()}", 10, 35);
32     doc.line(10, 38, 70, 38);
33
34     let y = 45;
35     doc.setFont("helvetica", "bold");
36     doc.text("Producto", 10, y);
37     doc.text("Cant.", 45, y);
38     doc.text("Total", 60, y);
39     doc.setFont("helvetica", "normal");
40
41     pedido.detalles?.forEach((item) => {
42       y += 5;
43       const subtotal = item.cantidad * Number(item.precio || 0);
```

Para la elaboración del diseño del historial de pedidos, se utilizaron contrastes en azul cian con estética en modo oscuro, también implementó un modal de notificación de pagos para subir el pago de manera sencilla y rápida, brindando así una experiencia en muy amigable, así mismo en la parte superior se estableció tarjetas informativas que se actualizan en tiempo real con los parámetros “Pedidos Realizados”, la “Inversión Total” acumulada y los “Pedidos Pendientes”. Esto se visualiza en la figura 239.

También se implementó indicadores dinámicos con un conteo de órdenes e inversión total con la finalidad de mantener un control financiero por parte del emprendedor. Y por último las tarjetas de cada pedido, tiene un resumen visual, desde la fecha, el detalle de cada compra, número de *ticket*, estado de este, ya sea pendiente o aprobado. Y por último la opción de *PDF TICKET* que permite la descarga de cada tarjeta de pedido.

Figura 239. Ventana de Historial Pedidos



4.3.6.1.4. Historia de Usuario 9: Administrador Dashboard

En función de lo presentado en el anexo 4, para el desarrollo en el código de *AdminResume.jsx* se utilizó 3 tipos de gráficos para el *backend*, *AreaChart*, *BarChart* y *PieChart*, lo cual transformó datos numéricos en gráficos visuales y estratégicos, en este caso se optó por que el resumen se actualice cada segundo mediante el uso de *setInterval*, y también se utilizó *recentActivities*, con la finalidad de implementar un monitor de actividades en tiempo real, con el objetivo de verificar al instante, pagos, pedidos y montos con el efecto de pulso, que indica que el sistema está respirando información a cada momento.

Se utilizó un componente de animación numérica llamado *AnimatedNumber* mediante *window.requestAnimationFrame* para el incremento de los valores. En el *backend* utiliza promesas recurrentes tales como *summary*, *product-sales*, *ranking*, *history* que disparan múltiples peticiones, desde el *frontend* hasta el servidor *backend*. En detalle, esto se visualiza en las figuras 240, 241 y 242.

Figura 240. Código de *AdminResumen.jsx* - *AnimatedNumber*

```

src > components > AdminResumen.jsx > AdminResumen
9  const AnimatedNumber = ({ value, isCurrency = false }) => {
13  useEffect(() => {
21      const step = (timestamp) => {
22          if (!startTimestamp) startTimestamp = timestamp;
23          const progress = Math.min((timestamp - startTimestamp) / duration, 1);
24          const easeOut = 1 - Math.pow(1 - progress, 3);
25          const current = easeOut * end;
26          setDisplayValue(current);
27
28          if (progress < 1) {
29              window.requestAnimationFrame(step);
30          } else {
31              prevValue.current = end;
32          }
33      };
34      window.requestAnimationFrame(step);
35  }, [value]);
36
37  return (
38      <span>
39          {isCurrency ? '$' : ''}
40          {displayValue.toLocaleString('en-US', {
41              minimumFractionDigits: isCurrency ? 2 : 0,
42              maximumFractionDigits: isCurrency ? 2 : 0
43          })}
44      </span>
45  );
46  };
47
48  const COLORS = ['#00f2fe', '#00C49F', '#FFBB28', '#FF8042', '#8884d8'];
49

```

Figura 241. Código de *AdminResumen.jsx* - *AdminResumen*

```

src > components > AdminResumen.jsx > AdminResumen
50  const AdminResumen = () => {
52      const [timeFilter, setTimeFilter] = useState('month');
53      const [loading, setLoading] = useState(true);
54      const [productSales, setProductSales] = useState([]);
55      const [history, setHistory] = useState([]);
56      const [recentActivities, setRecentActivities] = useState([]);
57      const [ranking, setRanking] = useState([]);
58      const [stats, setStats] = useState({ totalRecaudado: 0, totalPendiente: 0, emprendedorasActivas: 0 });
59
60      const fetchRecentActivity = async () => {
61          try {
62              const response = await axios.get('http://localhost:5000/api/admin/recent-activity');
63              setRecentActivities(Array.isArray(response.data) ? response.data : []);
64          } catch (error) {
65              console.error("Error al cargar actividad:", error);
66              setRecentActivities([]);
67          }
68      };
69
70      const fetchDashboardData = async () => {
71          try {
72              const config = { params: { period: timeFilter } };
73              const [summaryRes, productRes, rankingRes, historyRes] = await Promise.all([
74                  axios.get('http://localhost:5000/api/admin/summary/current-month-stats', config),
75                  axios.get('http://localhost:5000/api/admin/summary/product-sales'),
76                  axios.get('http://localhost:5000/api/admin/summary/top-entrepreneurs'),
77                  axios.get('http://localhost:5000/api/admin/summary/monthly-history')
78              ]);
79
80              const s = summaryRes.data;
81              setStats({
82                  totalRecaudado: Number(s?.totalRecaudado || 0),
83                  totalPendiente: Number(s?.totalPendiente || 0),
84                  emprendedorasActivas: Number(s?.emprendedorasActivas || 0)
85              });
86
87              setProductSales(Array.isArray(productRes.data) ? productRes.data.map(item => ({
88                  name: item.name,
89                  value: parseFloat(item.value || 0)
90              })) : []);
91
92              setRanking(Array.isArray(rankingRes.data) ? rankingRes.data : []);
93
94              const mesesReferencia = ['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', 'Nov', 'Dic'];
95              const dataMap = {};
96
97              if (Array.isArray(historyRes.data)) {
98                  historyRes.data.forEach(item => {
99                      dataMap[item.mes] = parseFloat(item.total || 0);
100                  });
101              }
102          }
103      };
104

```

Figura 242. Código de *AdminResumen.jsx* - *fetchDashboardData*

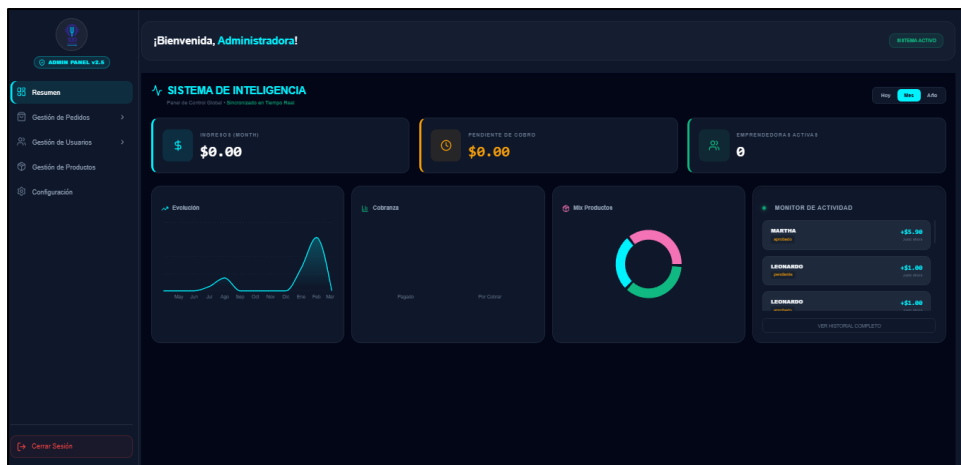
```

src > components > AdminResumen.jsx > AdminResumen
50  const AdminResumen = () => {
70      const fetchDashboardData = async () => {
72          const config = { params: { period: timeFilter } };
73          const [summaryRes, productRes, rankingRes, historyRes] = await Promise.all([
74              axios.get('http://localhost:5000/api/admin/summary/current-month-stats', config),
75              axios.get('http://localhost:5000/api/admin/summary/product-sales'),
76              axios.get('http://localhost:5000/api/admin/summary/top-entrepreneurs'),
77              axios.get('http://localhost:5000/api/admin/summary/monthly-history')
78          ]);
79
80          const s = summaryRes.data;
81          setStats({
82              totalRecaudado: Number(s?.totalRecaudado || 0),
83              totalPendiente: Number(s?.totalPendiente || 0),
84              emprendedorasActivas: Number(s?.emprendedorasActivas || 0)
85          });
86
87          setProductSales(Array.isArray(productRes.data) ? productRes.data.map(item => ({
88              name: item.name,
89              value: parseFloat(item.value || 0)
90          })) : []);
91
92          setRanking(Array.isArray(rankingRes.data) ? rankingRes.data : []);
93
94          const mesesReferencia = ['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', 'Nov', 'Dic'];
95          const dataMap = {};
96
97          if (Array.isArray(historyRes.data)) {
98              historyRes.data.forEach(item => {
99                  dataMap[item.mes] = parseFloat(item.total || 0);
100              });
101          }
102      };
103

```

Así la interfaz utilizó un estilo *Cyber-Dark* con fondos en #020617 y elementos en #0f172a, también se usó efectos de cristal *glassmorphism* y bordes neón, y con la visualización del gráfico de evolución, gráfico de cobranza y gráfico *mix* de productos como se mencionó respectivamente en el código de *AdminResumen.jsx*. La implementación del monitor en tiempo real fue pieza fundamental de la comunicación cada segundo entre el *backend* y *frontend*, ya que muestra una lista de las transacciones más recientes, detalles de la transacción y botón de historial. Y por último los indicadores de desempeño mediante métricas de Ingresos, pendientes de cobro, y emprendedoras activas, que al mismo tiempo pueden ser filtrados por día, mes y año. Ta como se observa en la figura 243.

Figura 243. Ventana de AdminDashboard.jsx



4.3.6.1.5. Historia de Usuario 10. Administrador Gestión Pedidos

Para la realización del código se utilizó *useAdmin* para centralizar el estado global de los pedidos con el uso de *allOrders* y a la vez se implementó *despacharPedido*, *aprobarPago* como función de mutación. También de usó *lucide-react* como librería externa, y *xlsx* para generación de reportes y *sweetalert2*, que gestiona la visualización de comprobantes. Para recalcular la lista de pedidos a cada momento utiliza *useMemo* en base a los criterios de búsqueda como fecha, pedido y para filtrar estado mediante la segmentación de categorías. Por último, se utilizó *verComprobante* para verificar ingreso de imágenes, siendo que si no se despliega como modal visual y para la exportación de datos de reporte general se usó *exportToExcel* y *getEstadoStyle* para la lógica visual que permite que los *badges* cambien su estética. Tal como se muestra en la figura 244, 245 y 246

Figura 244. Código de *GestionAdminPedidos.jsx* - *getEstadoStyle*

```

src > components > AdminGestionPedidos.jsx > AdminGestionPedidos > exportarResumenProduccion
10 const getEstadoStyle = (estado) => {
11   switch (estado) {
12     case 'aprobado': return { color: '#4ade88', bg: 'rgba(34, 197, 94, 0.15)', border: '#22c55e' };
13     case 'en_validacion': return { color: '#fbbf24', bg: 'rgba(245, 158, 11, 0.15)', border: '#f59e0b' };
14     case 'pendiente': return { color: '#f87171', bg: 'rgba(239, 68, 68, 0.15)', border: '#ef4444' };
15     default: return { color: '#94a3b8', bg: 'rgba(148, 163, 184, 0.15)', border: '#1e293b' };
16   }
17 };
18
19 const AdminGestionPedidos = () => {
20   const { allOrders, loading, despacharPedido, aprobarPago } = useAdmin();
21   const [searchTerm, setSearchTerm] = useState('');
22   const [filterEstado, setFilterEstado] = useState('Todos');
23
24   const pedidosFiltrados = useMemo(() => {
25     if (!allOrders) return [];
26     return allOrders.filter(pedido => {
27       const nombre = (pedido.nombre_empresa || pedido.empresa || '').toLowerCase();
28       const id = pedido.id.toString();
29       const fecha = new Date(pedido.fecha_pedido).toLocaleDateString();
30       const busqueda = searchTerm.toLowerCase();
31
32       const coincideBusqueda = nombre.includes(busqueda) || id.includes(busqueda) || fecha.includes(busqueda);
33       const coincideEstado = filterEstado === 'Todos' || pedido.estado_pago === filterEstado;
34
35       return coincideBusqueda && coincideEstado;
36     });
37   }, [searchTerm, filterEstado, allOrders]);
38
39   const verComprobante = (url) => {
40     if (!url || url === 'null' || url === '') {
41       Swal.fire({

```

Figura 245. Código de *GestionAdminPedidos.jsx* - *verComprobante*

```

src > components > AdminGestionPedidos.jsx > AdminGestionPedidos > exportarResumenProduccion
19 const AdminGestionPedidos = () => {
39   const verComprobante = (url) => {
40     if (!url || url === 'null' || url === '') {
41       Swal.fire({
42         title: 'Sin archivo',
43         text: 'No hay comprobante disponible en el servidor.',
44         icon: 'info',
45         background: 'rgba(255, 255, 255, 0.1)',
46         color: 'white',
47       });
48       return;
49     }
50     const fullUrl = `http://localhost:5000${url}`;
51     Swal.fire({
52       title: 'Comprobante de Pago',
53       imageUrl: fullUrl,
54       imageAlt: 'Comprobante',
55       background: 'rgba(255, 255, 255, 0.1)',
56       color: 'white',
57       confirmButtonColor: '#00f2fe',
58       imageWidth: 400,
59       imageHeight: 'auto'
60     });
61   };
62
63   const exportToExcel = () => {
64     const dataToExport = pedidosFiltrados.map(p => ({
65       ID: `${p.id}`,
66       Empresa: p.nombre_empresa || p.empresa,
67       Fecha: new Date(p.fecha_pedido).toLocaleDateString(),
68       Total: Number(p.total).toFixed(2),
69       Estado_Pago: p.estado_pago || 'PENDIENTE',

```

Figura 246. Código de *GestionAdminPedidos.jsx* - *exportarResumenProduccion*

```

src > components > AdminGestionPedidos.jsx > AdminGestionPedidos > exportarResumenProduccion
19 const AdminGestionPedidos = () => {
78   const exportarResumenProduccion = () => {
79     const pedidosParaProducir = pedidosFiltrados.filter(p =>
80       p.estado_pago === 'aprobado' && p.estado !== 'despachado'
81     );
82
83     if (pedidosParaProducir.length === 0) {
84       Swal.fire('Sin Pedidos', 'No hay pedidos pendientes de despacho.', 'info');
85       return;
86     }
87
88     const consolidado = {};
89     const detalleReparto = [];
90
91     pedidosParaProducir.forEach(p => {
92       if (!p.detalle_items) return;
93       const productos = p.detalle_items.split(', ');
94       productos.forEach(prodStr => {
95         const match = prodStr.match(/^(.+)(\d+)(.*)$/);
96         if (match) {
97           const cantidad = parseInt(match[2]);
98           const nombre = match[1].trim();
99           consolidado[nombre] = (consolidado[nombre] || 0) + cantidad;
100           detalleReparto.push({
101             'TICKET #': `${p.id}`,
102             'EMPRESA': (p.nombre_empresa || 'N/A').toUpperCase(),
103             'PRODUCTO': nombre.toUpperCase(),
104             'CANTIDAD': cantidad,
105             'FECHA': new Date(p.fecha_pedido).toLocaleDateString(),
106             'ESTADO_CARGA': 'Pendiente'
107           });
108         }

```

Para el desarrollo de la interfaz del módulo de *GestionAdminPedidos.jsx* se continuó con la estética *Cyber-Dark* ya que facilita la lectura de valores mediante el uso de colores neón, seguido a eso en la parte superior se encuentran resumen de producción para descargar el detalle para la preparación de los pedidos y exportar que genera un reporte general, ambos en *Excel*, y sumado a eso el filtrado de los pedidos, mediante 3 categorías, aprobado, en validación y pendiente. Cabe mencionar que mientras los estados se actualizan la columna de acciones se va actualizando poco a poco. Mejorando la presentación y que la información fluye a simple vista. Como se observa en la figura 247.

Figura 247. Ventana de *GestionAdminPedidos.jsx*

ID	EMPRESA/EMPRESA	FECHA	TOTAL	ESTADO PEDIDO	ACCIONES
#128	MARTHA JULIA SALGADO	7/2/2028	\$5.00	pendiente	
#127	Leonardo	7/2/2028	\$1.00	pendiente	
#126	Leonardo	7/2/2028	\$1.00	pendiente	
#125	Juanita Almazán	7/2/2028	\$2.00	pendiente	
#124	Juanita Almazán	7/2/2028	\$4.00	pendiente	
#123	Juanita Almazán	7/2/2028	\$4.00	pendiente	
#122	Juanita Almazán	7/2/2028	\$4.00	pendiente	
#121	Juanita Almazán	7/2/2028	\$1.00	pendiente	

4.3.6.1.6. Historia de Usuario 11. Administrador Gestión Productos

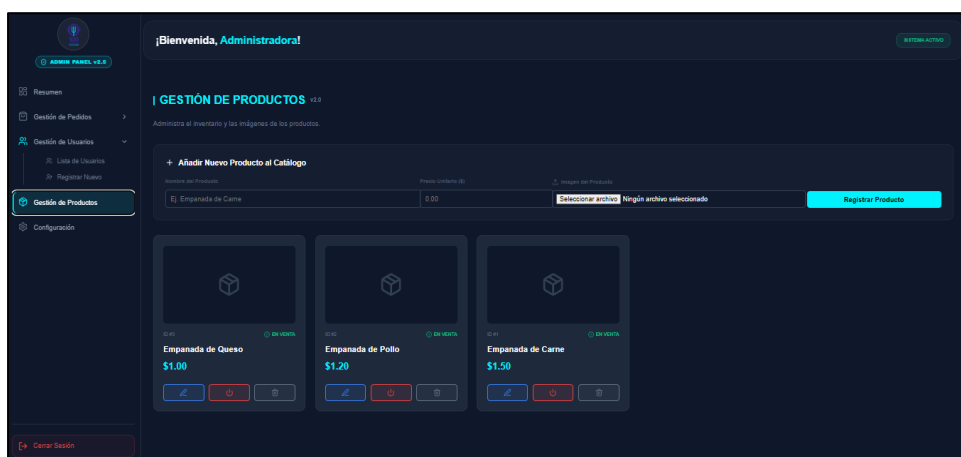
De acuerdo con lo presentado en el anexo 4, para la construcción del módulo se utilizó la función *fetchProducts*, la misma que sincroniza el estado con la base de datos *MySQL*, también se usó la función *handleImageChange* ya que captura el archivo tipo file y lo almacena. Ahora la función *toggleStatus* permite manipular un producto sin borrarlo. Ahora la función *resetFormLimpia* utiliza los campos y reestablece el input manualmente mediante el *DOM*. Y todo esto a la par del uso de *usestate*, y *useEffect*, que ayudan al manejo de lista de productos y también para el disparo de la función dentro del componente *fetchProducts*. Cabe mencionar que la manera más esencial para el envío de archivos mediante el uso de la función *multipart/form-data* es menester para reagrupar los conjuntos datos que representan campos del formulario. Tal como se muestra en la figura 248 y 249.

Figura 248. Código de *GestionAdminProductos.jsx*

```

AdminGestionProductos.jsx X
src > components > AdminGestionProductos.jsx > AdminGestionProductos > handleAddProduct
5 function AdminGestionProductos() {
27   useEffect(() => {
28     fetchProducts();
29   }, []);
30
31   const handleAddProduct = async (e) => {
32     e.preventDefault();
33     try {
34       await axios.post('http://localhost:5000/api/admin/productos', newProduct);
35       setNewProduct({ nombre: '', precio: '', activo: true });
36       fetchProducts();
37       setError('');
38     } catch (err) {
39       console.error('Error al añadir producto:', err);
40       setError('Error al añadir el producto.');
```

El diseño del componente de gestión de administración de productos consiste en un diseño *Glassmorphism* totalmente alineado con las imágenes anteriores, con tonos neon, y fondos oscuros. También dispone de *CSS Grid* con *repeat(auto-fill, minmax(280px, 1fr))* que ajustan las tarjetas automáticamente al ancho del monitor, sumado a eso tenemos un *Feedback* mediante el uso de *badges* dinamizados que varían dependiendo del estado. En cuanto al *backend* podemos observar que tenemos el uso de *API REST*, mediante el uso de *GET*, *POST*, *PUT* y *DELETE*, y esto abre la posibilidad de crear más productos a futuro ya que ese es el objetivo del crecimiento de las emprendedoras mediante la diversificación.

Figura 249. Ventana de *GestionAdminProductos.jsx*

4.3.6.1.7. Historia de Usuario 12 Administrador Gestión Usuarios

De acuerdo con lo presentado en el anexo 4, para la implementación del código se usó el filtrado multinivel mediante un *useEffect* que sincroniza dos estados: *searchTerm* y

activeFilter, esto permite búsquedas combinadas, para buscar a usuarios con secuencias de caracteres combinados ya sea por ingreso o por selección de categorías respectivamente. En *Toggle Status* en lugar de borrar usuarios, se implementa un eliminado lógico mediante la función *toggleUserStatus*, mismo que cambia el valor del campo en la base de datos de 0 a 1 o viceversa, permitiendo inhabilitar cuentas, pero manteniendo el historial. El segmento *editUser* controla la aparición de una interfaz superpuesta, ya que permite una edición pre-cargada inmediata sin navegar a otra página, también integra *react-toastify* para notificar al administrador cualquier actualización, y por último el sistema distingue estrictamente entre emprendedora y administradora. Tal como se visualiza, en la figura 250 y 251.

Figura 250. Código de GestionAdminUsuarios.jsx - AdminGestionUsuarios

```
src > components > @ AdminGestionUsuarios.jsx > AdminGestionUsuarios > fetchUsers
8 function AdminGestionUsuarios() {
14   const fetchUsers = async () => {
27   };
28
29   useEffect(() => {
30     fetchUsers();
31   }, []);
32
33   const handleUpdateUser = async (e) => {
34     e.preventDefault();
35     try {
36       await axios.put('http://localhost:5000/api/admin/users/${editUser.id}', editUser);
37       setEditUser(null);
38       fetchUsers();
39       toast.success('Usuario actualizado con éxito.');
```

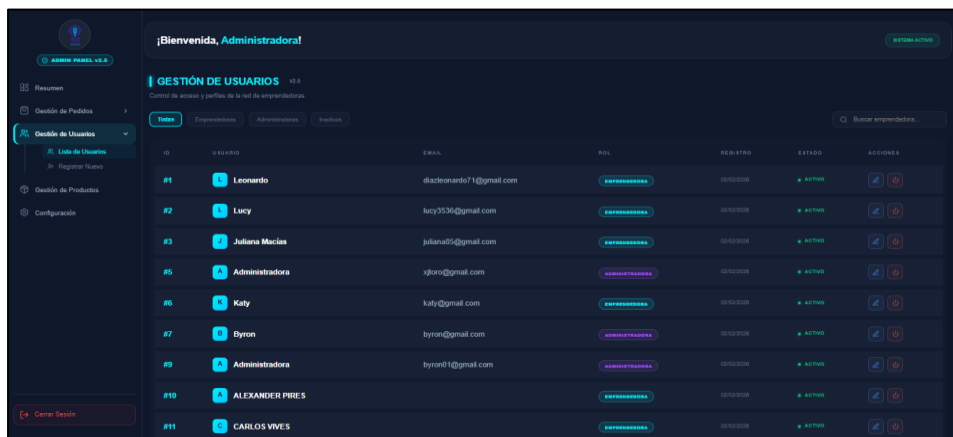
Figura 251. Código de GestionAdminUsuarios.jsx - useEffet y handleUpdateUser

```
src > components > @ AdminGestionUsuarios.jsx > AdminGestionUsuarios > filteredUsers.map() callback
7 function AdminGestionUsuarios() {
35   useEffect(() => {
36     let result = users;
37
38     if (activeFilter === 'Emprendedoras') {
39       result = result.filter(u => u rol === 'emprendedora');
40     } else if (activeFilter === 'Administradoras') {
41       result = result.filter(u => u rol === 'administradora');
42     } else if (activeFilter === 'Inactivos') {
43       result = result.filter(u => u activo === 0);
44     }
45
46     if (searchTerm) {
47       result = result.filter(u =>
48         u nombre.toLowerCase().includes(searchTerm.toLowerCase()) ||
49         u email.toLowerCase().includes(searchTerm.toLowerCase())
50       );
51     }
52
53     setFilteredUsers(result);
54   }, [searchTerm, activeFilter, users]);
55
56   const handleUpdateUser = async (e) => {
57     e.preventDefault();
58     try {
59       await axios.put('http://localhost:5000/api/admin/users/${editUser.id}', editUser);
60       setEditUser(null);
61       fetchUsers();
62       toast.success('Perfil actualizado correctamente.');
```

El diseño mantiene la línea *Cyber-Dark*, pero optimizado para la lectura de datos densos, la idea es mantener el mismo patrón en todos los módulos, los usuarios parecen

estar en una tarjeta dentro de la tabla creada, facilitando la lectura horizontal. También se implementó avatares automáticos con la inicial del nombre con un degradado *cian #00f2fe*, los roles están manejados con colores distintos, tales como Púrpura para Administradora y *Cian* para Emprendedora. Y por último se usó un efecto de *Glassmorphism* con el uso de *backdrop-filter: blur(8px)* en el *overlay* se crea una sensación profunda en el fondo al momento de editar cualquiera de las opciones. Como se observa en la figura 252.

Figura 252. Ventana de *GestionAdminUsuarios.jsx*



4.3.6.1.8. Historia Técnica 2: Seguridad y privacidad de la aplicación web

La autenticación de la aplicación web valida la identidad del usuario al iniciar sesión en *AdminLogin.jsx* y confirma el rol que este tiene antes de otorgarle acceso a los módulos protegidos. Evita accesos no autorizados y centraliza el control de acceso dentro del sistema. Como se muestra en la figura 253.

Figura 253. Validación de autenticación y control de roles en el acceso del administrador.

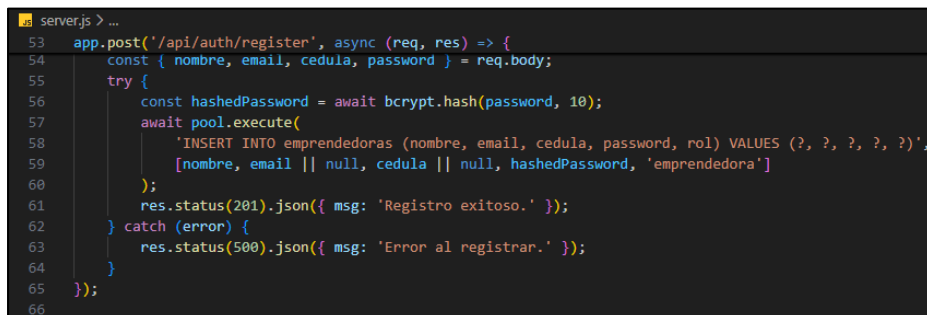
```

8  function AdminLogin() {
14     const handleSubmit = async (e) => {
18         const response = await axios.post(
20             { email, password }
21         );
22
23         if (response.data.user && response.data.user.role === 'administradora') {
24             localStorage.setItem("userName", response.data.user.nombre);
25             localStorage.setItem("userRole", response.data.user.role); // Guarda el rol
26             navigate("/admin-dashboard"); // Redirige al dashboard de administrador
27         } else {
28             setError("Credenciales inválidas o no tienes permisos de administrador.");
29             console.warn("Intento de login de admin fallido o rol incorrecto:", response.data);
30         }
31     }
32
33     } catch (err) {
34         if (err.response && err.response.data && err.response.data.msg) {
35             setError(err.response.data.msg); // Mensaje de error del backend
36         } else {
37             setError("Error al iniciar sesión. Inténtalo de nuevo."); // Mensaje genérico
38             console.error("Error de login de admin:", err);
39         }
40     }

```

Las contraseñas no se administran en texto plano; antes de ser guardadas en la base de datos, son sometidas a un proceso seguro de *hash*. Esto aumenta la seguridad de los datos personales del usuario y reduce el riesgo de que la información almacenada sea expuesta o a la que se acceda sin autorización. Como ilustra la figura 254.

Figura 254. Proceso de hash seguro de contraseñas en el registro en server.js



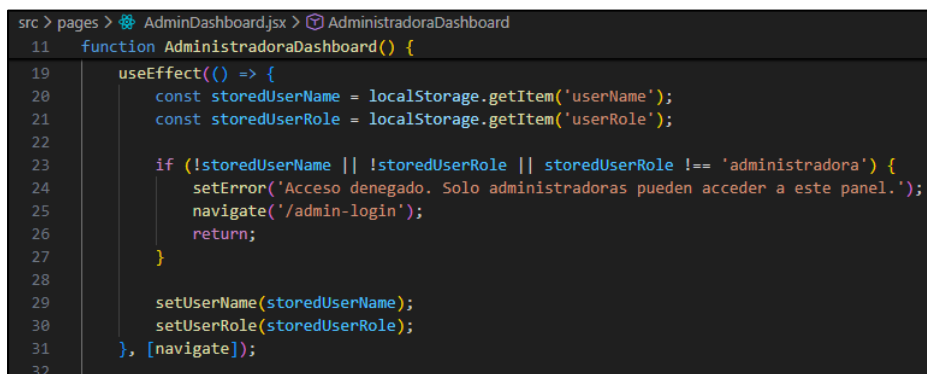
```

53 app.post('/api/auth/register', async (req, res) => {
54   const { nombre, email, cedula, password } = req.body;
55   try {
56     const hashedPassword = await bcrypt.hash(password, 10);
57     await pool.execute(
58       'INSERT INTO emprendedoras (nombre, email, cedula, password, rol) VALUES (?, ?, ?, ?, ?)',
59       [nombre, email || null, cedula || null, hashedPassword, 'emprendedora']
60     );
61     res.status(201).json({ msg: 'Registro exitoso.' });
62   } catch (error) {
63     res.status(500).json({ msg: 'Error al registrar.' });
64   }
65 });
66

```

Para habilitar la vista del panel de administración, se realiza una validación previa en la aplicación. Este control comprueba si la sesión existe y cuál es el rol del usuario, lo que previene que los usuarios no autorizados tengan acceso a funciones exclusivas de los administradores; así si no se logra la validación, la aplicación envía automáticamente al usuario a la página de inicio de sesión. Esto se muestra en la figura 255.

Figura 255. Control de acceso a módulos administrativos mediante validación de rol



```

src > pages > AdminDashboard.jsx > AdministradoraDashboard
11 function AdministradoraDashboard() {
12
19   useEffect(() => {
20     const storedUserName = localStorage.getItem('userName');
21     const storedUserRole = localStorage.getItem('userRole');
22
23     if (!storedUserName || !storedUserRole || storedUserRole !== 'administradora') {
24       setError('Acceso denegado. Solo administradoras pueden acceder a este panel.');
```

Se administra la sesión en la aplicación almacenando los datos del usuario en el navegador y limpiándolos de manera segura después de que se cierra sesión. Este procedimiento garantiza que el identificador del usuario, la información personal y el rol no persistan tras la sesión del usuario, reduciendo de esta manera el riesgo de acceso no permitido desde ese mismo aparato. En aplicaciones de uso compartido, es especialmente relevante cerrar la sesión. Como se representa en la figura 256.

Figura 256. Limpieza de la sesión tras cerrar sesión de *SidebarEmprendedora.jsx*

```

src > components > SidebarEmprendedora.jsx > SidebarEmprendedora
11  const SidebarEmprendedora = ({
12
13      const confirmLogout = () => {
14          setShowLogoutConfirm(false);
15          setIsExiting(true);
16
17          setTimeout(() => {
18              localStorage.clear();
19              sessionStorage.clear();
20              if (setIsLoggingOut) {
21                  setIsLoggingOut(true);
22              } else {
23                  window.location.href = '/';
24              }
25          }, 2500);
26      };

```

La validación de archivos sensibles en operaciones críticas como el registro de comprobantes de pago implementa controles que restringen el tipo de contenido que el usuario puede cargar. Este mecanismo previene la inyección de archivos maliciosos o de tipos no esperados, limitando la entrada únicamente a formatos de imagen. Aunque la validación ocurre en el cliente, esta representa una primera línea de validación hacia la integridad de las operaciones transaccionales. Como se observa en la figura 257.

Figura 257. Validación de tipo de archivo en carga de comprobantes de pago.

```

src > components > RegistrarPago.jsx > RegistrarPago > onDrop
9  const RegistrarPago = ({ userId, setVista }) => {
39  // 2. MANEJO DE ARCHIVOS (DRAG & DROP)
40  const handleFile = (file) => {
41      if (file && file.type.startsWith('image/')) {
42          setComprobante(file);
43          setPreviewUrl(URL.createObjectURL(file));
44      } else {
45          alert("Por favor sube una imagen válida (JPG, PNG).");
46      }
47  };

```

La verificación de estados de usuario antes de permitir operaciones críticas asegura que solo cuentas activas puedan completar transacciones o acceder a funcionalidades protegidas. Este control en el servidor previamente al envío de datos de sesión al cliente previene el acceso de cuentas deshabilitadas, suspendidas o inactivas que podrían haberse registrado anteriormente pero que no deberían de tener acceso. Lo anterior descrito, se muestra en la figura 258.

Figura 258. Validación de estado activo de cuenta durante autenticación en *server.js*

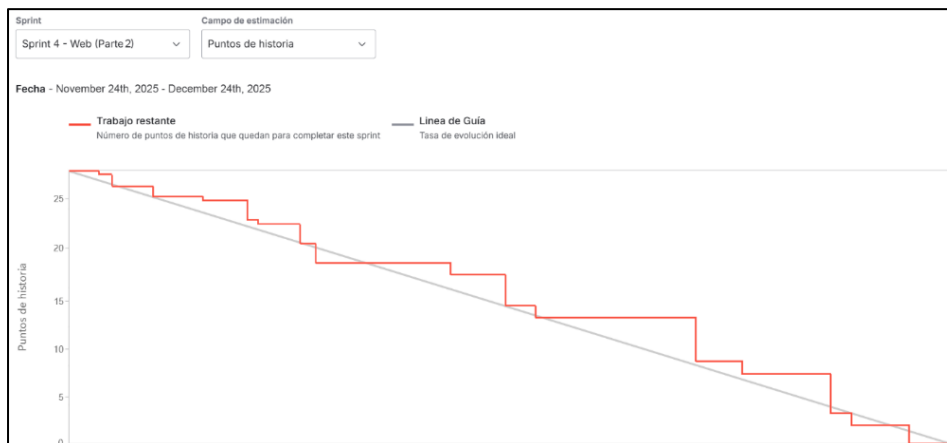
```

server.js > app.get('/api/emprendedoras/id') callback
67 app.post('/api/auth/login', async (req, res) => {
68   const { email, cedula, password } = req.body;
69   try {
70     const [rows] = await pool.execute('SELECT * FROM emprendedoras WHERE email = ? OR cedula = ?',
71     [email || null, cedula || null]);
72     if (rows.length === 0) return res.status(400).json({ msg: 'Usuario no encontrado.' });
73     const user = rows[0];
74     const isMatch = await bcrypt.compare(password, user.password);
75     if (!isMatch) return res.status(400).json({ msg: 'Contraseña incorrecta.' });
76     if (user.activo === 0) return res.status(403).json({ msg: 'Cuenta inactiva.' });
77
78     res.json({ user: { id: user.id, nombre: user.nombre, email: user.email, role: user.rol } });
79   } catch (error) { res.status(500).json({ msg: 'Error en login.' }); }
80 });
81

```

4.3.6.2. Sprint 4 – Gráfico de trabajo pendiente

En este último *sprint* se evidenció el uso de la herramienta *Jira*, la cual permite identificar como va cambiando el gráfico gradualmente, tal como se aprecia en la figura 259, dejando explícito la pendiente con la tendencia marcada en los segmentos incrustados en las fechas a nivel cronológico. Mostrando que curva en esa fecha se dedicó mayor concentración de avance, pero aun así logrando avanzar y culminar el *sprint* 4.

Figura 259. Gráfica del 4er *Sprint* en *Jira*.

4.3.6.3. Sprint 4 - Revisión (*Sprint Review*)

Después de culminado el *sprint* 4 se procede a la revisión con la administradora del proyecto 100 Mujeres Trabando, con la finalidad de evidenciar lo establecido que se solicitó en su momento, marcando un antes y un después, comenzando desde la opción para realizar pedido, registro de pagos, historial de pedidos, *dashboard* de administración, administración de gestión de pedidos, administración de gestión de productos, etc.

Mediante el uso de la herramienta *Jira* se analizó y corroboró que se cumplió el objetivo principal y a su vez observando el gráfico se aprecia con énfasis aquellos días en los que se implementó más trabajo de desarrollo en contraste con otras fechas.

4.3.6.4. Sprint 4 - Retrospectiva (*Sprint Retrospective*)

Una vez lograda la culminación de la última fase del *sprint* 4, para su retrospectiva se consideraron tres preguntas, con las cuales se obtendrá una perspectiva clara de cómo fue realizado el *sprint*. Como puede verse en la tabla 27.

Tabla 27. Sprint 4 retrospectiva

Aspectos exitosos del <i>sprint</i>	Mejoras para aplicar	Errores en el <i>sprint</i>
Se logró la funcionalidad para el envío de los pedidos, lo cual generó cambios de estados en tiempo real teniendo control total del sistema y facilitando la revisión del administrador, y también la apertura de la descarga del pedido total para que el encargado tenga una idea de la cantidad de materia prima del día a día.	Una de las mejoras a aplicar sería la de notificaciones y validación automática mediante el registro de los pagos a transferencia directa o recaudación, de tal forma que se obtenga mejor fluidez y eficacia.	Se logró descubrir que, al filtrar productos por categoría, se puede lograr mejor resultado ya que no muestra, ya que se pueden generar datos que permitan facilitar a la toma de decisiones y también se generalizó el tiempo de espera en todos los aspectos.

4.4. Resultado general

El resultado general se basó en la implementación del sistema, así como en la validación de la propuesta y la validación de la hipótesis.

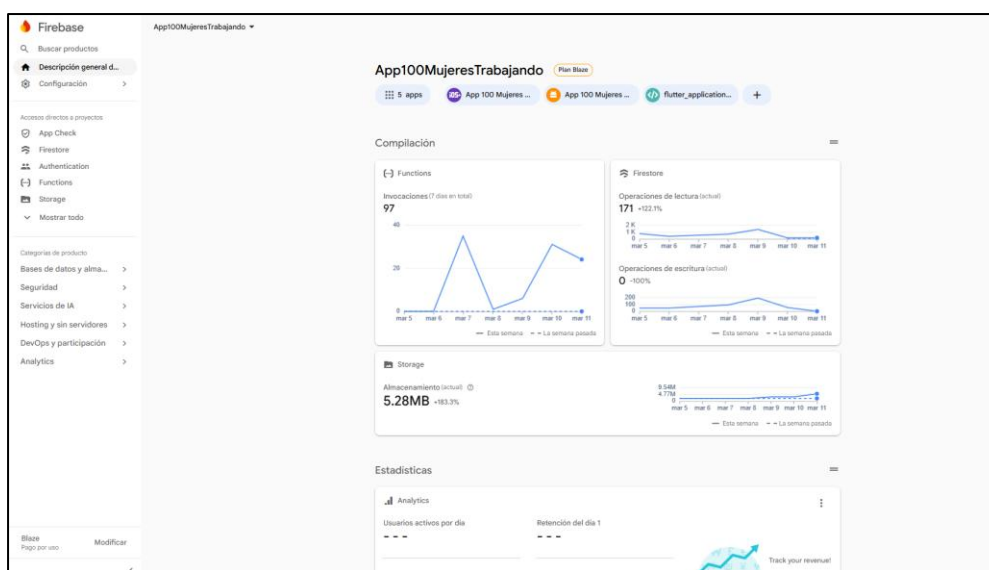
4.4.1. Implementación del sistema Web y Móvil

Completado el *sprint* 4, se procedió a realizar la implementación del sistema, desplegando la aplicación web y generando el instalador móvil para su entrega formal, como se evidencia en el anexo 6. La implementación del sistema se realizó mediante una arquitectura basada en servicios en la nube, utilizando *Flutter* para el desarrollo del cliente de teléfonos, *React* para los navegadores web y *Firebase* como plataforma principal de *backend* y base de datos. Esta etapa se enfocó en la configuración del sistema para su operación en producción, con fin de consolidar que permitiera la comunicación entre la aplicación móvil y web, con su base de datos y servicios complementarios.

4.4.1.1. Configuración del proyecto en *Firebase*

Como primer paso, se configuró el proyecto productivo en *Firebase*, registrando la aplicación móvil y vinculándola con los servicios necesarios para su operación. En este proceso se integraron los archivos de configuración generados por *Firebase* para *Android* y *iOS*, además de la inicialización centralizada desde *Flutter* mediante el archivo de opciones correspondiente. Esta etapa permitió establecer la conexión entre la aplicación y el proyecto en la nube, tal como se observa en la figura 260.

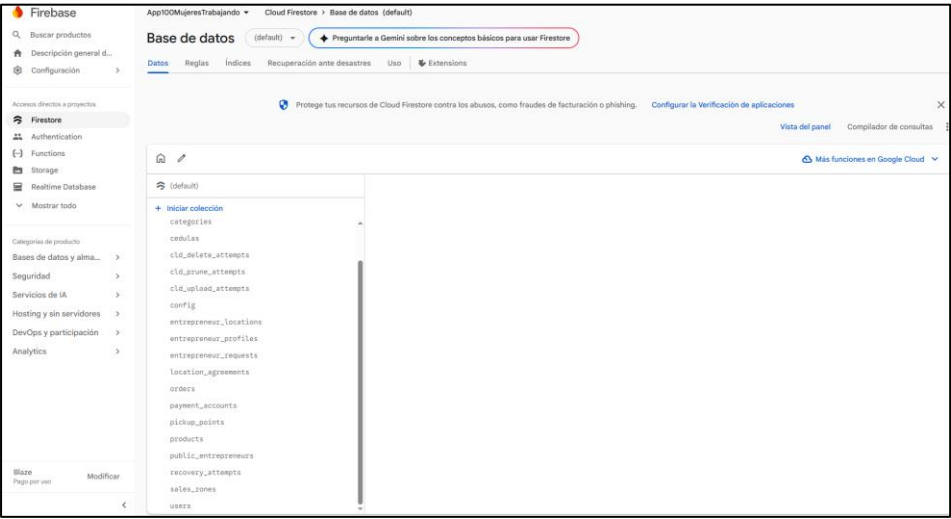
Figura 260. Pantalla principal del proyecto en *Firebase Console*



4.4.1.2. Configuración de la base de datos en *Cloud Firestore*

La base de datos se implementó mediante *Cloud Firestore*, aprovechando su modelo *NoSQL* y su integración nativa con aplicaciones móviles en tiempo real. En esta herramienta se definieron las colecciones y subcolecciones requeridas por el sistema, tales como usuarios, pedidos, productos, categorías, notificaciones y otros módulos administrativos. De igual manera, se configuraron las reglas de seguridad y los índices necesarios para controlar el acceso a la información y optimizar las consultas más frecuentes. Esta configuración permitió disponer de una base de datos escalable, segura y alineada con la estructura funcional de la aplicación, como se observa en la figura 261.

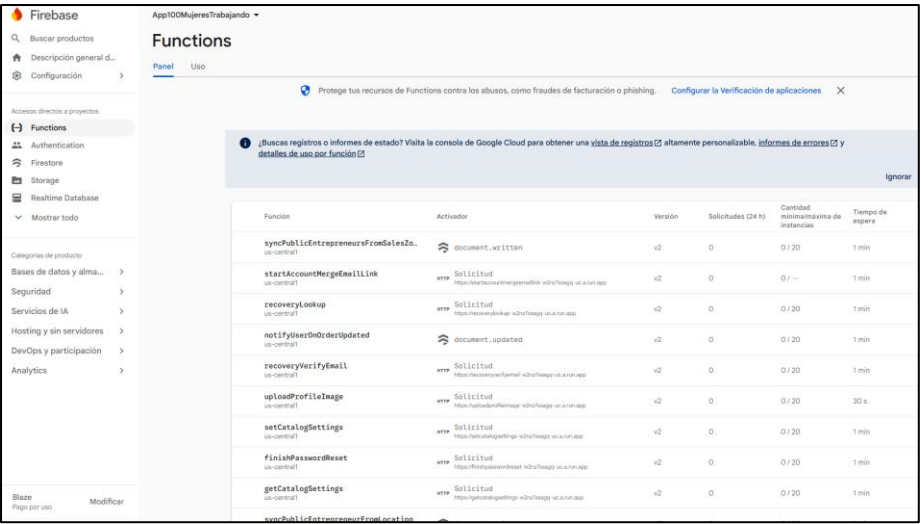
Figura 261. *Firebase Console en Firestore Database > Data*



4.4.1.3. Despliegue del *Backend* en *Firebase Cloud Functions*

La lógica de negocio del lado servidor fue desplegada en *Firebase Cloud Functions*, permitiendo ejecutar procesos que no debían residir directamente en el cliente móvil. En este entorno se publicaron funciones orientadas a tareas como recuperación de acceso, sincronización de datos, gestión de configuraciones, notificaciones automáticas y otros procesos internos del sistema. Asimismo, se configuró el proceso de compilación y despliegue del código fuente desde el módulo de funciones, lo que facilitó mantener separado el *frontend* móvil de la lógica *backend*. Esta estrategia permitió mejorar la seguridad, centralizar operaciones críticas y reducir la dependencia de procesos ejecutados localmente en el dispositivo, como se observa en la figura 262.

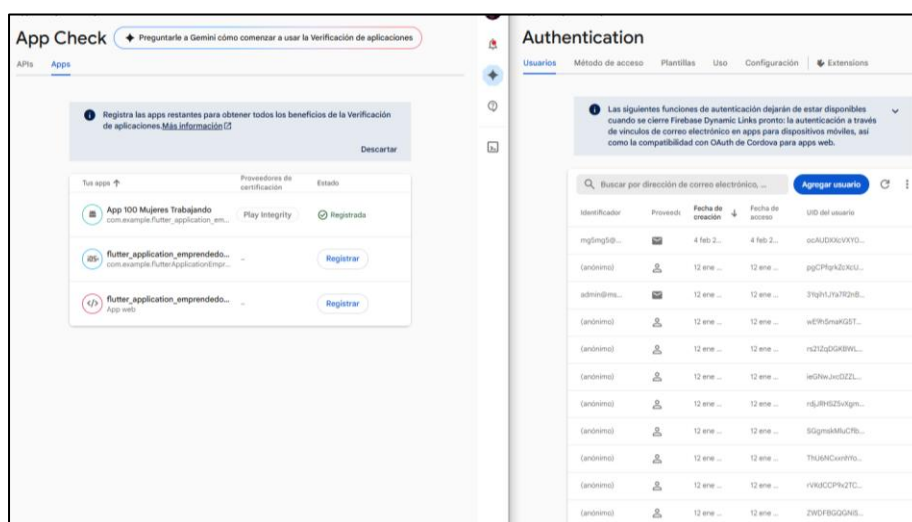
Figura 262. Despliegues de *backend* en *Functions Firebase*.



4.4.1.4. Integración de autenticación y seguridad

Para la gestión de acceso de usuarias y administradoras, se integró *Firebase Authentication* como servicio de autenticación principal. A través de este componente se habilitaron procesos como registro, inicio de sesión, verificación de correo y recuperación de cuenta. Adicionalmente, se incorporó *Firebase App Check* como mecanismo de protección para restringir el consumo de recursos *backend* únicamente a clientes válidos, fortaleciendo la seguridad del sistema frente a accesos no autorizados o automatizados. Con esta integración se aseguró que tanto la identidad de las usuarias como el acceso a los servicios del *backend* estuvieran protegidos bajo un mismo esquema de seguridad, tal como se observa en la figura 263.

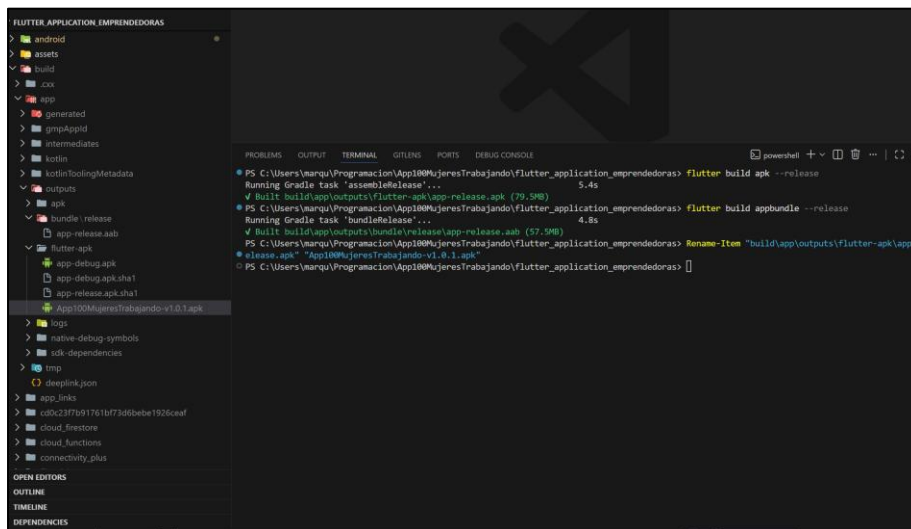
Figura 263. *Authentication* y *App Check*.



4.4.1.5. Generación del instalador de la aplicación móvil

Tras esto, se procedió a la generación del instalador de la aplicación móvil en modo *release*, con el propósito de obtener una versión lista para distribución y pruebas en dispositivos reales. Para ello fue necesario configurar la firma de la aplicación mediante un *keystore* en *Android* y establecer los parámetros requeridos para la compilación final del paquete. Una vez completado este proceso, se obtuvo el archivo instalador correspondiente, apto para instalación manual y validación funcional fuera del entorno de desarrollo. Esto permitió disponer de una versión ejecutable del sistema sin requerir todavía su publicación en tiendas oficiales, como se observa en la figura 264.

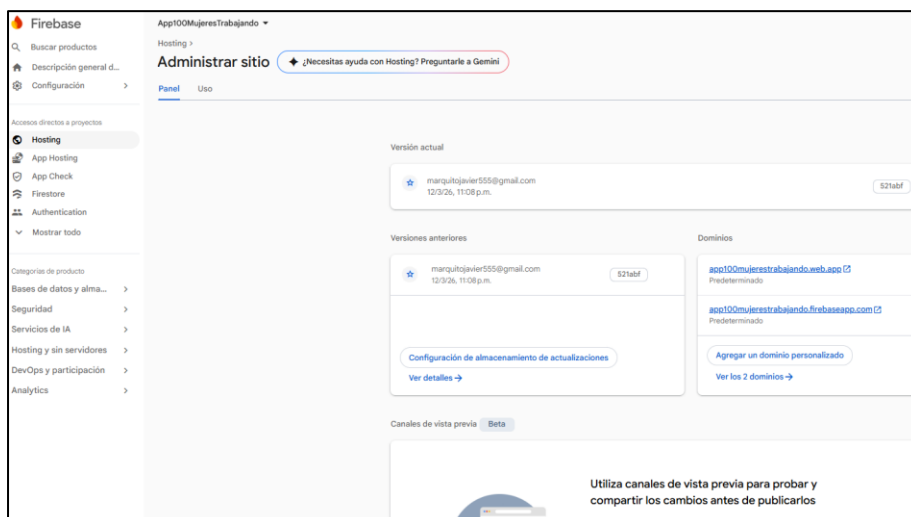
Figura 264. Archivo *APK* generado para instalar y compartir.



4.4.1.6. Despliegue del *Frontend Web*

Por último, el *frontend web* se desplegó en *Firebase Hosting*, utilizando la configuración del proyecto ya preparada. Para ello, se generó primero la compilación de producción del sistema, obteniendo los archivos estáticos optimizados en la carpeta de salida correspondiente; posteriormente, se autenticó la cuenta del proyecto en *Firebase CLI*, seleccionando el proyecto compartido y se ejecutando el despliegue exclusivo del hosting, sin modificar las reglas ni los índices de *Firestore* que continúan siendo compartidos con la aplicación móvil. Como resultado, la interfaz *web* quedó publicada en línea bajo la infraestructura de *Firebase Hosting*, con su versión liberada y administrable desde la consola del proyecto, tal como se observa en la figura 265.

Figura 265. Despliegue del *frontend web* en *Firebase Hosting*



En última instancia, la aplicación fue desplegada a producción desde el 23/12/2025. El *PreTest* se realizó el 21/12/2025, y después de entorno a unos dos meses, fue realizado el *PostTest* el 25/02/2026.

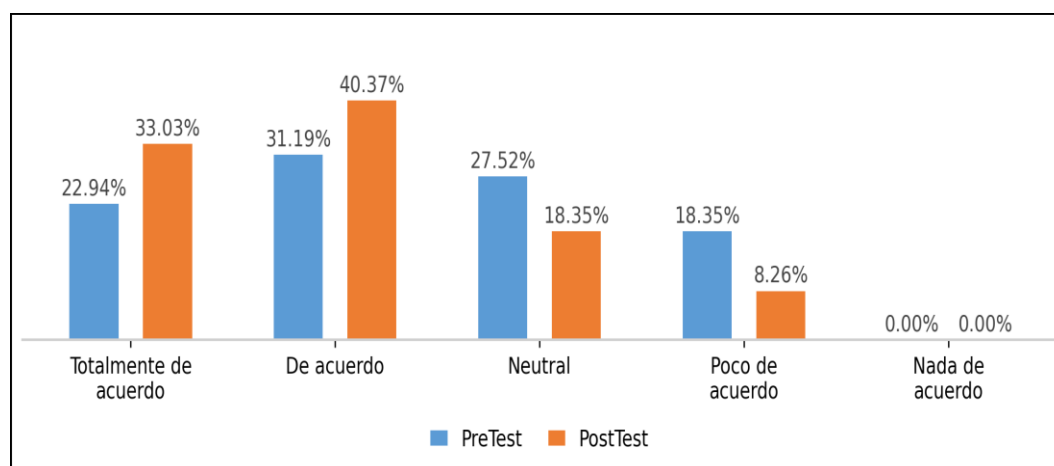
4.4.2. Validación de la propuesta de intervención

Mediante la implementación del instrumento de recolección de datos en una muestra compuesta por 109 mujeres emprendedoras, se evidenció un alto nivel de satisfacción en relación con la gestión de pedidos. En el Anexo 3 se detallan las preguntas que conformaron el instrumento de encuesta.

Se llevó a cabo un análisis de los resultados los cuales se consiguieron al aplicar la encuesta en *pretest* y *posttest*, en el cual se compararon de un grupo total de 27, se compararon 6 preguntas. Esto hizo posible determinar hasta qué punto optimizó la gestión de pedidos a través de las aplicaciones *web* y *móvil*. Los resultados mostraron una gran tasa de aceptación y un impacto positivo en los procedimientos administrados por medio de la aplicación.

Pregunta 1. ¿Te da confianza compartir tus datos en una App?

Figura 266. Confianza para compartir datos.

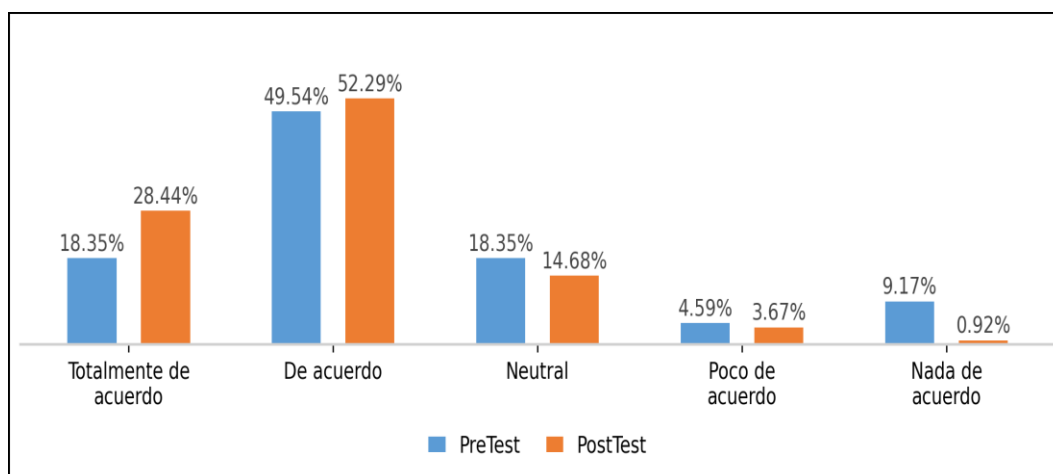


Análisis e interpretación: La figura 266 muestra un cambio relevante hacia mayor confianza, “totalmente de acuerdo” sube de 22.94% a 33.03%, y “de acuerdo” de 31.19% a 40.37%. A la par, “neutral” baja de 27.52% a 18.35% y “poco de acuerdo” de 18.35% a 8.26%.

El descenso del desacuerdo sugiere que la experiencia de uso reduce temores iniciales en torno al registro, manejo de información y flujo del pedido. El cambio más visible se presenta en la reducción de “poco de acuerdo”, reduciéndose en un 10.09%, y en el aumento de “totalmente de acuerdo”, el cual incremento un 10.09%. En este tipo de sistemas, la confianza suele consolidarse a partir de la práctica y de la claridad con que la App solicita y gestiona los datos.

Pregunta 2. ¿Cómo te sientes al usar tecnología para tu negocio, como páginas Web?

Figura 267. Actitud hacia el uso de tecnología en el emprendimiento.

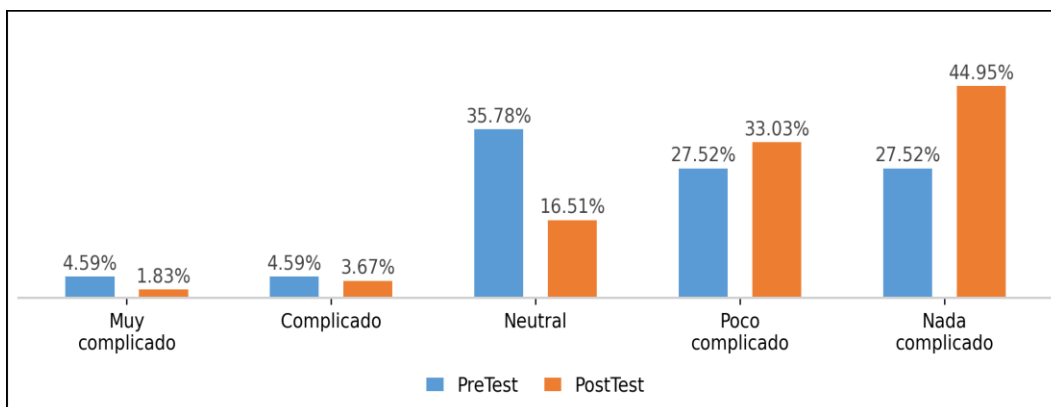


Análisis e interpretación: En la figura 267, “totalmente de acuerdo” sube de 18.35% a 28.44% y “de acuerdo” de 49.54% a 52.29%, “neutral” baja de 18.35% a 14.68%, “poco de acuerdo” se mantiene bajo (4.59% a 3.67%) y “nada de acuerdo” cae de 9.17% a 0.92%.

Más que medir una función específica, esta pregunta refleja la disposición hacia herramientas digitales aplicadas al negocio. El cambio más perceptible se da en el aumento de “totalmente de acuerdo”, incrementando 10.09%, y en la reducción de “nada de acuerdo”, reduciéndose en un 8.25%, lo que sugiere una actitud más favorable frente al uso de tecnología tras la experiencia con la App.

Pregunta 3. ¿Qué tan difícil es para ti enviar y recibir información digitalmente?

Figura 268. Complicación para enviar y recibir información digitalmente.

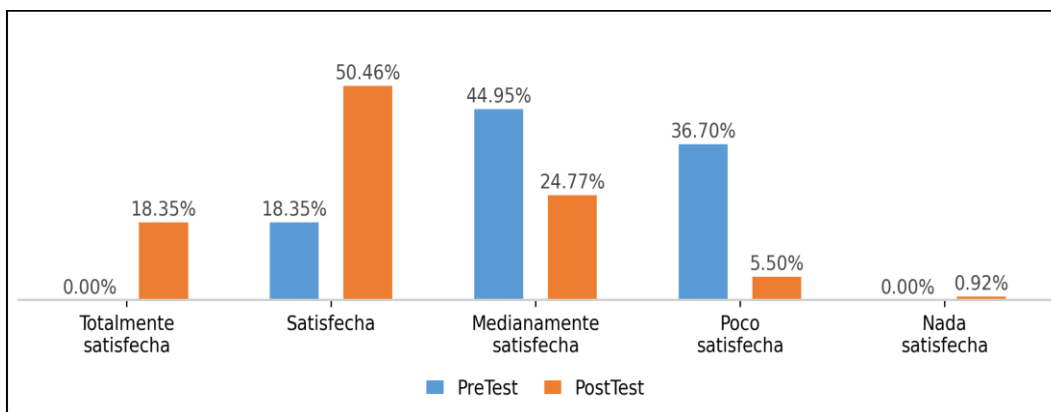


Análisis e interpretación: En la figura 268 se aprecia un desplazamiento hacia menor dificultad, “nada complicado” sube de 27.52% a 44.95% y “poco complicado” de 27.52% a 33.03%. En contraste, “neutral” baja de 35.78% a 16.51%, las categorías de mayor complicación se mantienen bajas y disminuyen ligeramente (“muy complicado” de 4.59% a 1.83%, y “complicado” de 4.59% a 3.67%).

Este cambio puede asociarse a la práctica repetida de tareas digitales vinculadas al pedido. La variación más marcada se ubica en la reducción de “neutral”, reduciendo el valor en un 19.27%, y en el aumento de “nada complicado”, al incrementar un 17.43%, lo que evidencia una menor percepción de dificultad. Aun así, la presencia reducida en “complicado” indica que una parte del grupo todavía requiere apoyo simple y constante.

Pregunta 4. ¿Qué tan satisfecha te sientes con la forma actual de pedir tus pedidos?

Figura 269. Satisfacción con la forma actual de pedir.

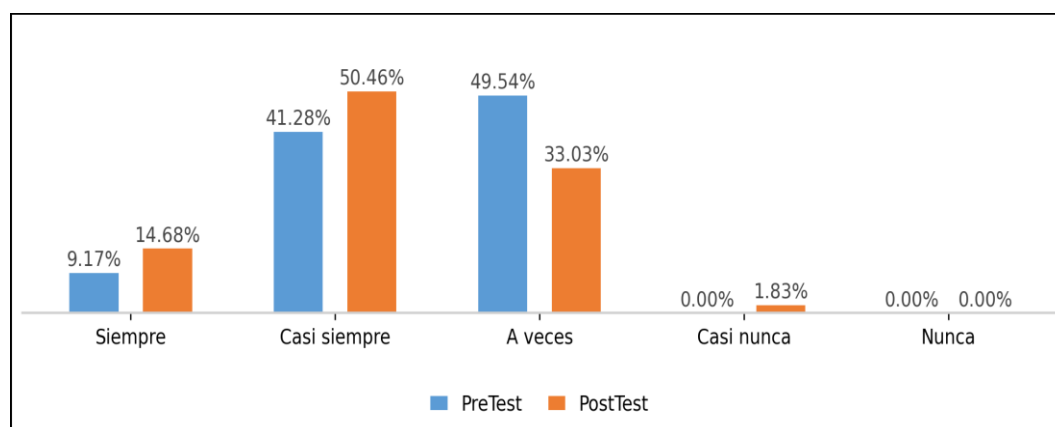


Análisis e interpretación: En la figura 269 evidencia el cambio más notorio hacia satisfacción. En el *pretest* predominaban “medianamente satisfecha” (44.95%) y “poco satisfecha” (36.70%). En el *posttest*, “satisfecha” aumenta de 18.35% a 50.46% y aparece “totalmente satisfecha” con 18.35% (anteriormente en 0.00%). Al mismo tiempo, “poco satisfecha” baja a 5.50% y “nada satisfecha” aparece en 0.92% (antes 0.00%).

El resultado sugiere que el proceso de pedido se percibe mejor organizado y más llevadero tras la implementación. El cambio más notorio ocurre con “satisfecha”, incrementando un +32.11%, mientras “poco satisfecha” se reduce un 31.20%. Además, la aparición de “totalmente satisfecha” con 18.35% refuerza la mejora observada, aunque la presencia mínima de “nada satisfecha” muestra que aún existen aspectos puntuales por perfeccionar.

Pregunta 5. ¿Con qué frecuencia realizan pedidos actualmente?

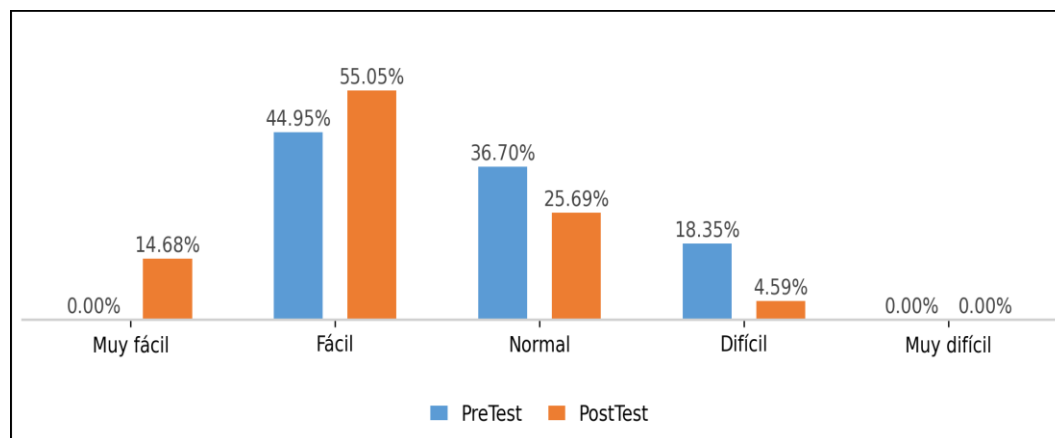
Figura 270. Frecuencia de realización de pedidos.



Análisis e interpretación: En la figura 270, “siempre” sube de 9.17% a 14.68% y “casi siempre” de 41.28% a 50.46%, “a veces” baja de 49.54% a 33.03%, aparece “casi nunca” con 1.83%. Este indicador es sensible al período de observación y a la demanda real, sin embargo, el desplazamiento de las respuestas sugiere una mayor regularidad en la realización de pedidos. La variación más visible se presenta en la reducción de “a veces”, reduciéndose un 16.51% en comparación, y en el aumento de “casi siempre”, en cambio incrementa un 9.18%, lo que resulta compatible con un proceso más simple y controlado.

Pregunta 6. ¿Qué tan fácil te resulta gestionar sus pedidos?

Figura 271. Facilidad para gestionar pedidos.



Análisis e interpretación: En la figura 271, “muy fácil” pasa de 0.00% a 14.68% y “fácil” sube de 44.95% a 55.05%, “normal” baja de 36.70% a 25.69% y “difícil” cae de 18.35% a 4.59%. Este comportamiento se alinea directamente con la variable de estudio, ya que refleja una mejora en la gestión del pedido. La variación más marcada se observa en el aumento de “muy fácil”, al incrementar un 14.68%, y en la reducción de “difícil”, reduciendo un 13.76%. Esto sugiere menos fricción en tareas como registrar, revisar, organizar o dar seguimiento, aunque el grupo que permanece en “normal” indica que todavía existe margen de simplificación.

4.4.3. Validación de la Hipótesis

La tabla 28 presenta la codificación de los escenarios utilizada para validar la hipótesis, estipular los códigos para los escenarios de *pretest* y *posttest*, así como las respuestas de las preguntas establecidas en el instrumento de recolección de datos.

Tabla 28. Recodificación de Escenarios.

Recodificación	Escenarios
0	Sin Aplicación <i>Web</i> y Móvil
1	Con Aplicación <i>Web</i> y Móvil

A partir de esta codificación, se construyó una base analítica estandarizada, documentada en el anexo 12 que incluyó ítems en escala de *Likert*, respuestas no estructuradas y mediciones en emprendedoras. Con la matriz depurada, se efectuó el

análisis estadístico en *IBM SPSS*, tal como se muestra en las figuras 272 y 273 del estudio, y se aplicó regresión logística binaria para contrastar los efectos observados.

Figura 272. Flujo de análisis estadístico en *SPSS* (IBM Corporation, 2011)

ESCAMEROS	¿Quitarías el robot a tu hijo/a?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?
1	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
2	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
3	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
4	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
5	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
6	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
7	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
8	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
9	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
10	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
11	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
12	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
13	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
14	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
15	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
16	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
17	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
18	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
19	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
20	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
21	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
22	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
23	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
24	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
25	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
26	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
27	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
28	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
29	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
30	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
31	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
32	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
33	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
34	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
35	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3

Figura 273. Flujo de análisis estadístico en *SPSS* (IBM Corporation, 2011)

ESCAMEROS	¿Quitarías el robot a tu hijo/a?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?	¿Quitarías el robot a tu hijo/a si fuera un niño/a?	¿Quitarías el robot a tu hijo/a si fuera una niña?
1	PRE	De acuerdo	Mediamente n.	De acuerdo	Normal	De acuerdo	Normal	De acuerdo	Poco de acuerdo	Necesario	Segura	De acuerdo	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
2	PRE	Neutral	Mediamente n.	De acuerdo	Normal	De acuerdo	Normal	De acuerdo	Poco de acuerdo	Necesario	Segura	De acuerdo	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
3	PRE	Totalmente de ac.	Necesario	Totalmente de ac.	Fácil	Totalmente de ac.	Fácil	De acuerdo	Poco de acuerdo	Muy necesario	Totalmente segura	Neutral	Complicado	Poco sa	De acuerdo	Complicado	Poco sa
4	PRE	Neutral	Mediamente n.	De acuerdo	Fácil	De acuerdo	Fácil	De acuerdo	Poco de acuerdo	Mediamente n.	Segura	De acuerdo	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
5	PRE	Totalmente de ac.	Muy necesario	Totalmente de ac.	Muy fácil	Totalmente de ac.	Fácil	Totalmente de ac.	De acuerdo	Necesario	Totalmente segura	De acuerdo	Muy complicado	Mediamente	De acuerdo	Muy complicado	Mediamente
6	PRE	De acuerdo	Necesario	Neutral	Fácil	De acuerdo	Fácil	De acuerdo	Poco de acuerdo	Necesario	Segura	De acuerdo	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
7	PRE	De acuerdo	Mediamente n.	De acuerdo	Fácil	De acuerdo	Fácil	De acuerdo	Poco de acuerdo	Necesario	Segura	De acuerdo	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
8	PRE	Poco de acuerdo	Poco necesario	Totalmente de ac.	Normal	Totalmente de ac.	Difícil	De acuerdo	De acuerdo	Poco necesario	Totalmente segura	Poco de acuerdo	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
9	PRE	Totalmente de ac.	Poco necesario	Totalmente de ac.	Normal	Totalmente de ac.	Muy fácil	Totalmente de ac.	De acuerdo	Necesario	Totalmente segura	Totalmente de ac.	Neutral	Poco sa	De acuerdo	Complicado	Poco sa
10	PRE	De acuerdo	Muy necesario	Totalmente de ac.	Normal	Totalmente de ac.	Fácil	Totalmente de ac.	Totalmente de ac.	Muy necesario	Totalmente segura	Nada de acuerdo	Muy complicado	Mediamente	De acuerdo	Muy complicado	Mediamente
11	PRE	De acuerdo	Poco necesario	Poco de acuerdo	Fácil	De acuerdo	Fácil	De acuerdo	De acuerdo	Muy necesario	Segura	De acuerdo	Complicado	Mediamente	De acuerdo	Complicado	Mediamente
12	PRE	Neutral	Mediamente n.	De acuerdo	Normal	De acuerdo	Fácil	De acuerdo	Poco de acuerdo	Necesario	Segura	De acuerdo	Complicado	Mediamente	De acuerdo	Complicado	Mediamente
13	PRE	De acuerdo	Innecesario	Totalmente de ac.	Muy fácil	Totalmente de ac.	Muy fácil	Totalmente de ac.	De acuerdo	Necesario	Totalmente segura	Totalmente de ac.	Muy complicado	Mediamente	De acuerdo	Muy complicado	Mediamente
14	PRE	Neutral	Necesario	De acuerdo	Fácil	De acuerdo	Fácil	De acuerdo	Totalmente de ac.	Poco necesario	Segura	De acuerdo	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
15	PRE	De acuerdo	Necesario	De acuerdo	Fácil	De acuerdo	Fácil	De acuerdo	De acuerdo	Necesario	Segura	De acuerdo	Complicado	Mediamente	De acuerdo	Complicado	Mediamente
16	PRE	Totalmente de ac.	Innecesario	Totalmente de ac.	Fácil	Totalmente de ac.	Muy fácil	Totalmente de ac.	De acuerdo	Necesario	Totalmente segura	Totalmente de ac.	Muy complicado	Poco sa	De acuerdo	Muy complicado	Poco sa
17	PRE	De acuerdo	Necesario	Totalmente de ac.	Fácil	Totalmente de ac.	Fácil	De acuerdo	Totalmente de ac.	Mediamente n.	Segura	De acuerdo	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
18	PRE	Neutral	Mediamente n.	De acuerdo	Fácil	De acuerdo	Fácil	De acuerdo	De acuerdo	Necesario	Poco segura	De acuerdo	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
19	PRE	Poco de acuerdo	Innecesario	Totalmente de ac.	Fácil	Totalmente de ac.	Muy fácil	Totalmente de ac.	Totalmente de ac.	Necesario	Segura	Totalmente de ac.	Muy complicado	Poco sa	De acuerdo	Muy complicado	Poco sa
20	PRE	De acuerdo	Muy necesario	De acuerdo	Fácil	De acuerdo	Fácil	De acuerdo	De acuerdo	Necesario	Segura	De acuerdo	Muy complicado	Poco sa	De acuerdo	Muy complicado	Poco sa
21	PRE	De acuerdo	Poco necesario	De acuerdo	Fácil	De acuerdo	Normal	De acuerdo	De acuerdo	Poco necesario	Totalmente segura	De acuerdo	Nada de acuerdo	Poco sa	De acuerdo	Nada de acuerdo	Poco sa
22	PRE	Totalmente de ac.	Poco necesario	Totalmente de ac.	Fácil	Totalmente de ac.	Muy fácil	Totalmente de ac.	Totalmente de ac.	Poco necesario	Totalmente segura	De acuerdo	Complicado	Mediamente	De acuerdo	Complicado	Mediamente
23	PRE	De acuerdo	Poco necesario	Totalmente de ac.	Fácil	De acuerdo	Normal	Totalmente de ac.	Poco de acuerdo	Poco necesario	Totalmente segura	De acuerdo	Muy complicado	Mediamente	De acuerdo	Muy complicado	Mediamente
24	PRE	De acuerdo	Innecesario	Totalmente de ac.	Normal	Totalmente de ac.	Normal	De acuerdo	Mediamente n.	Totalmente segura	De acuerdo	Complicado	Poco sa	De acuerdo	Complicado	Poco sa	De acuerdo
25	PRE	De acuerdo	Mediamente n.	De acuerdo	Fácil	De acuerdo	Muy fácil	De acuerdo	De acuerdo	Necesario	Moderadamente	Nada de acuerdo	Neutral	Sa	De acuerdo	Neutral	Sa
26	PRE	Totalmente de ac.	Mediamente n.	Neutral	Fácil	De acuerdo	Fácil	De acuerdo	Totalmente de ac.	Necesario	Segura	Nada de acuerdo	Complicado	Poco sa	De acuerdo	Complicado	Poco sa
27	PRE	De acuerdo	Necesario	Totalmente de ac.	Fácil	Totalmente de ac.	Muy fácil	De acuerdo	Poco de acuerdo	Muy necesario	Segura	Totalmente de ac.	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
28	PRE	Neutral	Necesario	Totalmente de ac.	Normal	De acuerdo	Fácil	De acuerdo	Totalmente de ac.	Poco necesario	Segura	De acuerdo	Muy complicado	Mediamente	De acuerdo	Muy complicado	Mediamente
29	PRE	De acuerdo	Mediamente n.	Totalmente de ac.	Muy fácil	Totalmente de ac.	Fácil	Totalmente de ac.	De acuerdo	Necesario	Segura	De acuerdo	Neutral	Sa	De acuerdo	Neutral	Sa
30	PRE	Neutral	Poco necesario	De acuerdo	Fácil	De acuerdo	Fácil	De acuerdo	Totalmente de ac.	Muy necesario	Totalmente segura	De acuerdo	Neutral	Mediamente	De acuerdo	Neutral	Mediamente
31	PRE	Totalmente de ac.	Innecesario	Totalmente de ac.	Fácil	De acuerdo	Muy fácil	Totalmente de ac.	De acuerdo	Necesario	Segura	Totalmente de ac.	Muy complicado	Poco sa	De acuerdo	Muy complicado	Poco sa
32	PRE	Totalmente de ac.	Necesario	Totalmente de ac.	Normal	Totalmente de ac.	Fácil	Totalmente de ac.	Totalmente de ac.	Mediamente n.	Totalmente segura	De acuerdo	Complicado	Mediamente	De acuerdo	Complicado	Mediamente
33	PRE	De acuerdo	Necesario	Neutral	Muy fácil	De acuerdo	Fácil	De acuerdo	De acuerdo	Necesario	Poco segura	De acuerdo	Muy complicado	Poco sa	De acuerdo	Muy complicado	Poco sa
34	PRE	Neutral	Necesario	Totalmente de ac.	Normal	De acuerdo	Difícil	Totalmente de ac.	De acuerdo	Necesario	Segura	De acuerdo	Complicado	Sa	De acuerdo	Complicado	Sa
35	PRE	Totalmente de ac.	Innecesario	Totalmente de ac.	Fácil	De acuerdo	Fácil	De acuerdo	Poco de acuerdo	Poco necesario	Segura	Poco de acuerdo	Muy complicado	Poco sa	De acuerdo	Muy complicado	Poco sa

Se empleó la "regresión logística binaria" con un grado de libertad de 1 ($gl=1$) para analizar los datos del instrumento y detectar puntos específicos en la "gestión de emprendimientos". Como se muestra en la tabla 29, se examinaron elementos tales como: la complicación para enviar y recibir información digitalmente, la satisfacción con la forma actual de pedir, la frecuencia de realización de pedidos, y la facilidad para gestionar los pedidos.

Tabla 29. Estudio cruzado en base a la aplicación *web* y móvil

Recodificación	Puntuación	GI	Sig.
¿Te da confianza compartir tus datos en una App?	8.526	1	.004
¿Cómo te sientes al usar tecnología para tu negocio, como páginas <i>Web</i> ?	8.961	1	.003
¿Qué tan difícil es para ti enviar y recibir información digitalmente?	11.131	1	.001
¿Qué tan satisfecha te sientes con la forma actual de pedir tus pedidos?	62.085	1	.001
¿Con qué frecuencia realizan pedidos actualmente?	3.891	1	.049
¿Qué tan fácil te resulta gestionar sus pedidos?	24.685	1	.001

Todas las preguntas evaluadas y los datos reflejados en la tabla 29, evidenciaron un nivel de significancia estadística menor a 0.05 ($p < 0.05$), lo que permite rechazar la hipótesis nula, en conclusión, la implementación de la Aplicación *Web* y Móvil incide en la gestión de emprendimiento en la empresa 100 Mujeres del cantón Santo Domingo.

5. DISCUSIÓN

El desarrollo de esta investigación, se apoyó en el diagnóstico y en la aplicación del *pretest*, los cuales permitieron determinar los procesos considerados en el primer objetivo específico. Por medio del diagnóstico, se identificó que la gestión de los emprendimientos de “100 Mujeres” presentaba limitaciones en la actualización de productos y servicios, un seguimiento manual de pedidos y restricciones en la inscripción de emprendedoras. Por otra parte, se llevó a cabo una reunión en donde la administradora está acorde con la implementación de una estructura ordenada para realizar pedidos, registrar productos y dar seguimiento a las solicitudes, ya que discierne en mejorar esos procesos entre las emprendedoras del proyecto y la administración de gestión de emprendimientos. Además, se complementa con Menezes (2023), quien destaca que la identificación de requerimientos debe partir del entendimiento de los procesos reales de la organización (p. 178).

Adicionalmente, a partir del estudio cruzado realiza la encuesta a 109 emprendedoras del proyecto, aquellos resultados manifiestan un incremento del 50.46% con la satisfacción en la manera actual de realizar los pedidos. Asimismo, se evidenció un aumento del 55.05% en cuanto a la facilidad para gestionar pedidos, lo cual representa el cambio más importante generado por la aplicación. Otros resultados relevantes incluyen incremento del 22.94% en la necesidad de capacitación ante cambios en la manera de realizar pedidos, un aumento 31.19% en cuanto a la aceptación de usar la aplicación en otro dispositivo. Además, un incremento del 20.18% en la necesidad de que sea en varios dispositivos a la vez, evidenciando así el impacto positivo en los procesos de gestión y en la optimización de las actividades desarrolladas. Este resultado coincide con Wu et al. (2024), quienes sostienen que el aprovechamiento de las tecnologías digitales para responder a las necesidades de los consumidores puede generar numerosas oportunidades de emprendimiento (p. 3).

Por otra parte, en el segundo objetivo específico se seleccionaron las herramientas principales que sirvieron para el desarrollo del sistema, las cuales fueron *React* con *Vite*

para la *web*, *Node.js* para la lógica del sistema, *Flutter* para la aplicación móvil y *Firebase* para los servicios integrados. Además, para la organización interna del proyecto se emplearon *Clean Architecture* y *BLoC*, con el fin de mantener una estructura modular, sincronizada y adaptable a futuros cambios. Esta decisión fue pertinente debido a que el sistema debía atender distintos perfiles de usuarios, gestionar pedidos y comprobantes en tiempo real, y sostener su funcionamiento sin rehacer la base del sistema. Esta estructura se relaciona con lo expuesto por Cerny et al. (2024), quienes afirman que una arquitectura efectiva permite realizar modificaciones con mayor facilidad y contribuye a evitar el deterioro de la calidad del sistema (p. 279). En la empresa 100 Mujeres, la arquitectura elegida no solo favoreció la mantenibilidad técnica, sino también la posibilidad de fortalecer la gestión de emprendimientos con una operación estable y viable.

Finalmente, para el tercer objetivo específico, se desarrolló e implementó una aplicación *web* y móvil funcional, la cual contó con módulos de registro, catálogo, pedidos, carga de comprobantes, historial, panel administrativo y cambio de estados en tiempo real. Asimismo, para la gestión del proyecto se empleó la metodología *Scrum*, que permitió organizar el desarrollo en revisiones e incorporaciones progresivas de funciones de acuerdo con las necesidades operativas identificadas. Este resultado coincide con Durmic (2024), quien señala que el enfoque ágil permite alinear el negocio y la tecnología mediante la integración de representantes del área de negocio y del personal técnico en un mismo equipo de proyecto (p. 2956). De este modo, la implementación no representó únicamente la entrega de un *software*, sino también una mejora en el seguimiento, la coordinación y la autogestión de los emprendimientos vinculados a la organización.

6. CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

Se concluye que el diagnóstico, las encuestas realizadas a las emprendedoras y la reunión de planificación del *sprint* permitieron identificar los procesos actuales vinculados a la gestión de emprendimientos. A partir de ello, se evidenciaron limitaciones en la actualización de productos y servicios, el seguimiento manual de pedidos y las restricciones en la inscripción de emprendedoras. Estos resultados permitieron reconocer el flujo de actividades y determinar puntos claves para la propuesta, orientados al registro de productos, la organización de pedidos y el seguimiento de solicitudes, fortaleciendo así la gestión de emprendimientos.

La evaluación de las tecnologías y herramientas también permitió determinar las más apropiadas para la creación y ejecución de la propuesta. Se eligieron *React* con *Vite* para la *web*, *Node.js* para la lógica del sistema, *Firebase* para los servicios integrados y *Flutter* para la aplicación móvil por ser herramientas rápidas de implementar, escalables y versátiles. Además, el uso de *Clean Architecture* y *BLoC* permitió mantener una estructura modular, sincronizada y adaptable a futuros cambios, facilitando la centralización de la información, la mantenibilidad técnica y la estabilidad del sistema.

De igual manera, se desarrolló e implementó una aplicación *web* y móvil funcional, conformada por módulos de registro, catálogo, pedidos, carga de comprobantes, historial, panel administrativo y cambio de estados en tiempo real. La metodología *Scrum* facilitó la flexibilidad requerida durante el desarrollo, permitiendo organizar las tareas, atender incidencias e incorporar mejoras de forma progresiva conforme a las necesidades operativas identificadas. Esto favoreció una planificación ordenada y una ejecución continua de las actividades previstas.

Finalmente, el análisis comparativo entre los resultados del *pretest* y *posttest* permitió evidenciar que la aplicación *Web* y *Móvil* tuvo un impacto significativo en la gestión de emprendimientos. En este sentido, la propuesta no solo permitió fortalecer el control y

seguimiento de los procesos, sino también mejorar la coordinación, la autogestión y la validación de la hipótesis planteada por medio del análisis estadístico aplicado.

6.2. Recomendaciones

Se recomienda realizar evaluaciones periódicas a emprendedoras, clientes y administradoras, por medio de encuestas cortas dentro de la *App* y entrevistas semiestructuradas, con el fin de identificar mejoras en la gestión de pedidos, la visualización del catálogo y el seguimiento por estados, incorporando funcionalidades de acuerdo con las necesidades operativas identificadas.

Asimismo, se propone mantener un proceso continuo de retroalimentación mediante reuniones programadas, para revisar el flujo real de trabajo: recepción de pedidos, validación de comprobantes, cambios de estado, coordinación de entregas y control de inconsistencias. Esto permitirá ajustar reglas de negocio, mensajes al usuario y priorización de mejoras en función de problemas reales detectados en operación.

Se sugiere se evalúe el rendimiento y costos de la solución hecha en *React* con *Vite* para la *web*, *Node.js* para la lógica del sistema, y *Flutter* para la aplicación móvil, dado el uso de los servicios integrados de *Firebase*, tomando en cuenta la documentación oficial y las buenas prácticas para tener reglas adecuadas, índices y con fin de tener consultas y lecturas correctas a la base de datos *Firestore*. Además, se recomienda implementar monitoreo, *logs* y métricas, para detectar fallos recurrentes y sostener su funcionamiento.

En cuanto a la estructura modular, sincronizada y adaptable a futuros cambios, se recomienda continuar aplicando *Clean Architecture* y *BLoC*, con el fin de mantener la mantenibilidad técnica, la centralización de la información y la estabilidad del sistema. También es recomendable reforzar la seguridad mediante reglas de acceso y validaciones en la capa de datos, garantizando consistencia entre el pedido y su evidencia asociada.

Además, se recomienda mantener la metodología *Scrum*, con retrospectivas al cierre de cada *sprint*, para organizar el desarrollo en revisiones e incorporaciones

progresivas de funciones, identificar cuellos de botella y ajustar la planificación y ejecución de tareas conforme surgían nuevos requerimientos.

Finalmente, se recomienda revisar y depurar periódicamente los instrumentos de recolección de datos, encuestas y guías de entrevista, idealmente con apoyo de docentes o expertos en emprendimientos y transformación digital, con el objetivo de mejorar la claridad de las preguntas y obtener evidencia más precisa sobre el impacto de la aplicación *web* y *móvil* en la gestión de emprendimientos.

7. REFERENCIAS

- Alarcón, C. A., y Hernández, V. H. (2024). Modelos de gestión de talento humano. Análisis crítico. *Revista Científica ECOCIENCIA*, 11(1), 1-17.
<https://doi.org/10.21855/ecociencia.111.847>
- Alessandria, S. K., y Kayfitz, B. (2021). *Flutter Cookbook: Over 100 Proven Techniques and Solutions for App Development with Flutter 2.2 and Dart*.
- Allen, G., y Owens, M. (2011). *The Definitive Guide to SQLite*. Apress.
- Altadill-Izura, P. X. (2022). *REACT PRÁCTICO: DESDE CERO A DESARROLLOS WEB AVANZADOS*. Anaya Multimedia.
- Álvarez-Parodys, E., Cantillo-Sarmiento, A., y Parody-Muñoz, A. (2023). Tendencias investigativas en la gestión de operaciones logísticas en el campo de la ingeniería industrial en América Latina. *Ingeniare*, 19(35), 11-27. <https://doi.org/10.18041/1909-2458/ingeniare.35.12440>
- Amador-Martínez, M., Hernández, N. I., Azcarate, L. A., Valenzuela, J. C., y Andrade, K. A. (2023). Empoderamiento de mujeres emprendedoras, actoras en la economía del Valle del Cauca, Colombia. *LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades*, 4(6). <https://doi.org/10.56712/latam.v4i6.1428>
- Anwar, Z. (6 de 4 de 2025). *Permits and licenses: Overview, definition, and example*. Cobrief. <https://www.cobrief.app/resources/legal-glossary/permits-and-licenses-overview-definition-and-example/>
- Arbulú-Ballesteros, M. A., Ramos-Farroñán, E. V., García-Juárez, H. D., Ganoza-Ubillús, L. M., Arbulú-Castillo, J. C., Saavedra-Torres, I., . . . Martínez-Vargas, A. M. (2024). Fostering Sustainable Development Through Women's Entrepreneurship: Psychosocial Factors and Attitudinal Mediation in Colombian Higher Education. *Sustainability*, 16, 10562. <https://doi.org/10.3390/su162310562>

- Ariani, A. F., Putra, A. B., y Sugata, T. L. (2025). Implementation of Firebase in the development of Android-based queue reservation and treatment record applications. *JSil | Jurnal Sistem Informasi*, 12(1), 63-71. <https://doi.org/10.30656/jsii.v11i1.6936>
- Arias-Gonzales, J. L., y Covinos-Gallardo, M. (2021). *Diseño y metodología de la investigación [Pdf]*. Enfoques Consulting E.I.R.L. https://gc.scalahed.com/recursos/files/r161r/w26022w/Arias_S2.pdf
- Arias-Orezano, J. F., Reyna-Barreto, B. D., y Mamani-Apaza, G. (2021). Repercusión de arquitectura limpia y la norma ISO/IEC 25010 en la mantenibilidad de aplicativos Android. *TecnoLógicas*, 24(52), e2104. <https://doi.org/10.22430/22565337.2104>
- Armijos, J. L., Morocho, R. F., Redrován, F. F., y Torres, D. A. (2021). Estado del arte: métricas del desarrollo de software móvil. *3C Tecnología. Glosas de innovación aplicadas a la pyme*, 10(3), 17-37. <https://doi.org/10.17993/3ctecno/2021.v10n3e39.17-37>
- Asamblea Constituyente. (2008). *Constitución de la República del Ecuador*. Montecristi, Ecuador. https://www.defensa.gob.ec/wp-content/uploads/downloads/2021/02/Constitucion-de-la-Republica-del-Ecuador_act_ene-2021.pdf
- Asamblea Nacional del Ecuador. (2020). Ley Orgánica de Emprendimiento e Innovación (Ley s/n). Suplemento del Registro Oficial No. 151 (28 de febrero de 2020). https://www.gob.ec/sites/default/files/regulations/2020-03/Documento_LEY-ORGANICA-EMPREDIMIENTO-INNOVACION.pdf
- Bin-Uzayr, S. (2023). *Mastering Visual Studio Code: A Beginner's Guide* (1 ed.). Boca Raton: CRC Press. <https://doi.org/10.1201/9781003311973>
- Biørn-Hansen, A., Rieger, C., Grønli, T., Majchrzak, T. A., y Ghinea, G. (2020). An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*, 25(4), 2997-3040. <https://doi.org/10.1007/s10664-020-09827-6>

- Bocangel-Weydert, G. A., Chuquiyauri-Saldivar, E. S., Bocangel-Marín, G. A., Piñán-García, J. H., Ramírez-Reyes, G., García-Bonilla, H. W., y Gomez-Meza, L. J. (2024). *Desarrollo de Soluciones Web - JavaScript - Node.js - Express - MySQL*. Atena Editora. <https://doi.org/10.22533/at.ed.077242009>
- Bueno-Tacuri, A. E., y Jácome-Ortega, M. J. (2021). Gestión de operaciones para la mejora continua en Organizaciones. *Revista Arbitrada Interdisciplinaria Koinonía*, 6(12), 334. <https://doi.org/10.35381/r.k.v6i12.1292>
- Calle-Herencia, C. A. (2022). La transformación digital y su importancia en las pymes. *Iberoamerican Business Journal*, 5(2), 64-81. <https://doi.org/10.22451/5817.ibj2022.vol5.2.11059>
- Camburn, B., Viswanathan, V., Linsey, J., Anderson, D., Jensen, D., Crawford, R., . . . Wood, K. (2017). Design prototyping methods: state of the art in strategies, techniques, and guidelines. *Design Science*, 3. <https://doi.org/10.1017/dsj.2017.10>
- Carriel, E. E., Bowen, J. R., Holguín, P. A., López, J. M., Vera, M. I., Amores, R. E., . . . Sotomayor, C. A. (2017). *Gestión de emprendedores en el Ecuador*. <https://doi.org/10.21855/librosecotec.48>
- Carvajal-Camperos, M., Almodóvar-Martínez, P., y Vassolo, R. (2021). Análisis del concepto y alcance de las alianzas estratégicas: un en-foque longitudinal (1972-2020). *Revista Venezolana De Gerencia*, 26(Número Especial 5), 290-314. <https://doi.org/10.52080/rvgluz.26.e5.20>
- Casado, E. R. (2021). *Más que diseño de experiencia (UX): Life-centered design para productos y servicios*. ESIC Editorial.
- Celi-Párraga, R. J., Boné-Andrade, M. F., y Mora-Olivero, A. P. (2023). *Programación Web del Frontend al Backend*. Editorial Grupo AEA.
- Cerny, T., Chy, M. S., Abdelfattah, A. S., Soldani, J., y Bogner, J. (2024). On maintainability and microservice dependencies: How do changes propagate? En *Proceedings of the 14th International Conference on Cloud Computing and Services Science (CLOSER 2024)* (pp. 277-286). SciTePress. <https://doi.org/10.5220/0012725200003711>

- Chanchlani, A. S., y Gune, S. H. (1 de 2025). Comprehensive Review on Various Web Technology Tools. *Grenze International Journal of Engineering and Technology*, 11(1), 4121-4124.
- Chauhan, C., Dakoliya, M., y Sharma, R. K. (2023). Use of Fibonacci sequence in project estimation. *Indian Journal of Science and Technology*, 16(33), 2649-2652.
<https://doi.org/10.17485/IJST/v16i33.1534>
- Chen, G., Naveed, R. T., Singh, A., Waris, M., Waseem, W., y Muneer, S. (2025). The effect of women's development on the relationship between the social impact of green microfinance institutions and poverty in Pakistan. *Discover Sustainability*, 6(1).
<https://doi.org/10.1007/s43621-025-00790-0>
- Cohn, M. (2006). *Agile Estimating and Planning*. Prentice-Hall PTR.
- Comisión Económica para América Latina y el Caribe (CEPAL). (2022). *La sociedad del cuidado: horizonte para una recuperación sostenible con igualdad de género (LC/CRM.15/3)*. <https://www.cepal.org/es/publicaciones/48363-la-sociedad-cuidado-horizonte-recuperacion-sostenible-igualdad-genero>
- Cornell Law School. (2021). *business license*. LII / Legal Information Institute.
https://www.law.cornell.edu/wex/business_license
- Delgado-Olivera, L. d., y Díaz-Alonso, L. M. (2021). Modelos de Desarrollo de Software. *Revista Cubana de Ciencias Informáticas*, 15(1), 37-51.
<https://www.redalyc.org/articulo.oa?id=378366538003>
- Díaz, J., y Perez, L. (2024). Design of event management software for entrepreneurs applying heuristic principles and user experience. En Proceedings of the 22nd LACCEI International Multi-Conference for Engineering, Education and Technology (LACCEI 2024). <https://doi.org/10.18687/LACCEI2024.1.1.1280>
- Durmic, N. (2024). Adopting agile with Scrum by outsourced project teams: A case of Bosnia and Herzegovina. *TEM Journal*, 13(4), 2955-2968. <https://doi.org/10.18421/TEM134-30>

Escudero-Sánchez, C. L., y Cortez-Suárez, L. A. (2018). *Técnicas y métodos cualitativos para la investigación científica*. UTMACH.

<https://repositorio.utmachala.edu.ec/bitstream/48000/12501/1/Tecnicas-y-MetodoscualitativosParaInvestigacionCientifica.pdf>

Flores-Bañuelos, M. d., Hernández-Mejía, S., y García-Santillán, A. (2023). Relación entre el marketing digital, las habilidades en internet y las características del negocio: Un estudio en mujeres emprendedoras. *Contaduría y Administración*, 69(2), e448.

<https://doi.org/10.22201/fca.24488410e.2024.5081>

Gallardo-García, M. G. (2025). *Investigación cuantitativa: fundamentos teóricos y prácticos*. Universidad Juárez del Estado de Durango.

https://juridicas.ujed.mx/static/docs/invest_cuant.pdf

Gancedo, M. R., y Ibáñez-Maraña, J. L. (2021). *CLAVES y RETOS para desarrollar UN PROYECTO COMÚN RURAL EMPRENDIMIENTO SOCIAL en tiempos de Covid*. Social Soluciones S. Coop. de Iniciativa Social.

<https://www.socialsoluciones.coop/publicaciones/manual%20claves%20y%20retos%20POEDA%202021%20ISBN.pdf>

Gauchat, J. D. (2017). *El gran libro de HTML5, CSS3 y JavaScript*. Marcombo.

Gillfillan, I. (2003). *La biblia de MySQL*. Anaya Multimedia.

Gobierno Autónomo Descentralizado Municipal de Santo Domingo. (27 de 1 de 2021).

Ordenanza de estímulos tributarios para atraer nuevas inversiones y emprendimientos en el área de turismo, construcción, industria, comercio u otras actividades productivas, culturales, educativas y deportivas en el cantón Santo Domingo. Ordenanza Municipal N.º M-014-WEA.

Godoy-Rosero, P. C., Alava-Cedeño, M. D., Añapa-Tapuyo, W. J., y Quiñonez-Francis, H. S. (2022). Aplicaciones móviles en modelos de microemprendimientos: Métricas y fortalezas en el comercio ecuatoriano. *Revista Tecnológica Ciencia y Educación Edwards Deming*, 6(1), 50–63. e-ISSN: 2576-0971.

<https://doi.org/10.37957/rfd.v6i1.89>

- Godoy-Rosero, P., Chamorro-Quiñónez, J. G., Castillo-Ruano, G., y Robles-Quiñónez, G. D. (2021). Gestión de emprendimientos en modelos de negocio basados en aplicaciones digitales y teletrabajo. *Polo del Conocimiento*, 6(11), 337-353.
<https://doi.org/10.23857/pc.v6i11.3267>
- Google. (2024). *Android Security Paper 2024 (Versión 1.0)*. Android Open Source Project.
<https://source.android.com/security/reports/AndroidSecurityWhitePaper-2024.pdf>
- Guerrero-Velástegui, C. A., Godoy-Calvachi, X. A., Castro-Cruz, L. D., Torres-Pérez, J. M., y Terán-Guerrero, F. N. (2023). *Gestión Empresarial: Concepción Legal bajo enfoque del proceso administrativo*. Apuntes desde la perspectiva académica.
<https://doi.org/10.55813/egaea.l.2022.43>
- Gülcüoğlu, E., Üstün, A. B., y Seyhan, N. (2021). Comparison of Flutter and React Native platforms. <https://doi.org/10.34231/iuyd.888243>
- Hernandez, S. J. (2011). *INTRODUCCION a LA ADMINISTRACIÓN: Teoría general administrativa: origen, evolución y vanguardia* (5.ª ed.). Mc Graw-Hill Interamericana Editores.
- Hernández-Bejarano, M., y Baquero-Rey, L. E. (2020). *Fundamentos de Programación Web*. Editorial Universidad ECCI. <https://doi.org/10.18180/LIBROECCI.ISBN.978-958-8817-40-8>
- Hernández-Sampieri, R., y Mendoza-Torres, C. P. (2018). *Metodología de la investigación: las rutas cuantitativa, cualitativa y mixta* (1.ª ed.). McGraw-Hill.
- Hervás, R. N. (2023). *Gestión de bases de datos*. Ra-Ma Editorial.
- Hoyos, G. (2024). *Dominando React, una guía introductoria para desarrolladores* (1.ª ed.). DOXA EDITION.
- Kocaoglu, B., y Acar, A. Z. (2016). Process development in customer order information systems to gain competitive advantage: a SME case study. *International Journal Of Logistics Systems And Management*, 23(2), 209.
<https://doi.org/10.1504/ijlsm.2016.073968>

- Kuzmiakova, A. (2023). *Creation and Management of Database Systems*. Arcler Press.
<https://research.ebsco.com/linkprocessor/plink?id=142838a0-6cdb-35d4-aa5a-7e50f843d172>
- Lemahieu, W., Vanden-Broucke, S., y Baesens, B. (2018). *Principles of Database Management*. <https://doi.org/10.1017/9781316888773>
- Lira-Camargo, J., y Lira-Camargo, L. G. (2021). *Informática Básica para Todos: Iniciando con la tecnología (1.ª ed. digital)*.
- Llerena-Izquierdo, J. (1 de 2023). *Guía de aprendizaje de Programación*. Quito: Universidad Politécnica Salesiana.
- López-de-Sosoaga-López-de-Robles, A., Ugalde-Gorostiza, A. I., Rodríguez-Miñambres, P., y Rico-Martínez, A. (2015). La enseñanza por proyectos: una metodología necesaria para los futuros docentes. *Opción*, 31(1), 395-413.
<http://www.redalyc.org/articulo.oa?id=31043005022>
- Mamani, J. H., Gamarra, J. E., y Lucana, E. C. (2024). *PROYECTO DE TESIS: Guía práctica para investigación cuantitativa*. En Editora Científica Digital eBooks.
<https://doi.org/10.37885/978-65-5360-556-5>
- Marcó, F., Loguzzo, H. A., y Fedi, J. L. (2016). *Introducción a la gestión y administración en las organizaciones (2.ª ed.)*. Universidad Nacional Arturo Jauretche.
<https://rid.unaj.edu.ar/items/7bc98821-4687-4a4b-9038-5b953ba16164>
- Márquez-Coca, W. A., Valenzuela-Chicaiza, C. V., Acosta-Jaramillo, C. A., Gaón-Rojas, N. M., y Chimarro-Amaguaña, J. D. (2023). *esarrollo de aplicaciones web con tecnologías modernas*. Religación Press.
<https://doi.org/10.46652/ReligacionPress.108>
- Mashapure, R., Nyagadza, B., Muzondo, P., Mutanda, B., Mthombeni, A., Tapera, J., y Hamunakwadi, P. (2025). Bridging the digital technology gender gap: Challenges faced by women entrepreneurs in Mashonaland West Zimbabwe. *Journal of Asian and African Studies*. *Advance online publication*.
<https://doi.org/10.1177/00219096251369519>

- Medina-Romero, M. Á., Hurtado-Tiza, D. R., Muñoz-Murillo, J. P., Ochoa-Cervantez, D. O., y Izundegui-Ordóñez, G. (7 de 2023). *Método mixto de investigación: Cuantitativo y cualitativo* (1.ª ed.). Puno: Instituto Universitario de Innovación Ciencia y Tecnología Inudi Perú S.A.C. <https://doi.org/10.35622/inudi.b.105>
- Medina-Romero, M. Á., Rojas-León, C. R., Bustamante-Hoces, W., Loaiza-Carrasco, R. M., Martel-Carranza, C. P., y Castillo-Acobo, R. Y. (2 de 2023). *Metodología de la investigación: Técnicas e instrumentos de investigación* (1.ª ed.). Puno: Instituto Universitario de Innovación Ciencia y Tecnología Inudi Perú S.A.C. <https://doi.org/10.35622/inudi.b.080>
- Menezes, T. (2023). A review to find elicitation methods for business process automation software. *Software*, 2(2), 177-196. <https://doi.org/10.3390/software2020008>
- Molina-Ríos, J. R., y Pedreira-Souto, M. L. (2019). "SWIRL", *METODOLOGÍA PARA EL DISEÑO Y DESARROLLO DE APLICACIONES WEB*. 3ciencias, Editorial Área de Innovación y Desarrollo, S.L. <https://doi.org/10.17993/IngyTec.2019.55>
- Moore, K. (2025). *Mastering Flutter: Learn to develop Flutter apps for iOS, Android, desktop and web (English Edition)*. BPB Publications.
- Moreno-Pérez, J. C. (2018). *Entornos de desarrollo*. EDITORIAL SÍNTESIS, S. A.
- Napoli, M. L. (2019). *Beginning flutter: A Hands On Guide to App Development*. John Wiley & Sons.
- Odeniran, Q., Wimmer, H., y Rebman, C. M. (2023). Node.js or PHP? Determining the better website server backend scripting language. *Issues in Information Systems*, 24(1), 328-341. https://doi.org/10.48009/1_iis_2023_128
- Organización Internacional del Trabajo [OIT]. (2015). *Políticas para la formalización de las micro y pequeñas empresas en América Latina: Experiencias, avances y desafíos*. Organización Internacional del Trabajo. https://oiss.org/wp-content/uploads/2018/11/formalizacion_en_america_latina.pdf
- Organización Internacional del Trabajo [OIT]. (2021). *Promoción del emprendimiento sostenible a través de incubadoras y aceleradores de empresas y centros de*

innovación.

https://www.ilo.org/sites/default/files/wcmsp5/groups/public/%40ed_emp/%40emp_ent/%40ifp_seed/documents/publication/wcms_830263.pdf

Organización Naciones Unidas [ONU]. (2023). *Informe de los Objetivos de Desarrollo Sostenible (Jennifer Ross, Ed.)*. https://unstats.un.org/sdgs/report/2023/The-Sustainable-Development-Goals-Report-2023_Spanish.pdf

Paredes-Guerrero, R. B., Tapia-Segura, S. G., y Silva-Vimos, M. E. (2021). Análisis administrativo, legal y operativo, para la creación de emprendimientos de alimentos y bebidas. *Polo del Conocimiento: Revista Científico - Profesional*, 6(9), 21-44. <https://doi.org/10.23857/pc.v6i9.2998>

Payne, R. (2019). *Beginning App Development with Flutter*. En Apress eBooks. <https://doi.org/10.1007/978-1-4842-5181-2>

Pérez-Flores, A. (2024). Respuesta carta editor “Población y muestra”. *International Journal Of Interdisciplinary Dentistry*, 17(2), 67. <https://doi.org/10.4067/s2452-55882024000200067>

Presidencia de la República del Ecuador. (4 de 8 de 2020). *Reglamento General a la Ley Orgánica de Emprendimiento e Innovación*. Decreto Ejecutivo 1113; Registro Oficial Suplemento 260 de 04 de agosto de 2020.

Pressman, R. S., y Maxim, B. R. (2019). *Software Engineering: A Practitioners Approach*.

Rahman, A., y Muktadir, M. (2021). SPSS: An Imperative Quantitative Data Analysis Tool for Social Science Research. *International Journal Of Research And Innovation In Social Science*, 05(10), 300-302. <https://doi.org/10.47772/ijriss.2021.51012>

Randhawa, T. S. (2022). *Mobile applications: Design, Development and Optimization*. En Springer eBooks. <https://doi.org/10.1007/978-3-030-02391-1>

Richards, M., y Ford, N. (2020). *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media, Inc. <https://mrce.in/ebooks/Software-Fundamentals%20of%20Software%20Architecture.pdf>

Rivadeneira-Pacheco, J. L., De-la-Hoz-Suárez, A. I., y Barrera-Argüello, M. V. (2020).

Análisis general del SPSS y su utilidad en la estadística. *E-IDEA JOURNAL OF BUSINESS SCIENCES*.

<https://core.ac.uk/works/4655377/?t=26107462bfb1b48ae6bcb7b3d3b3a439-4655377>

Rivera-Barberán, G. R., Zurita-Castro, C., Jara-Nivelo, J. J., y Avilés-Almeida, P. A. (2024).

Emprendimiento Una alternativa de negocios (1.^a ed.). E-Idea Omwin S. A.

<https://estudioidea.org/wp-content/uploads/2025/01/Libro-de-emprendimiento-definitivo-B5-CON-PORTADA.pdf>

Rojas-López, M. D., Rincón-López, C., y Mesa-León, S. (2014). Alianzas estratégicas:

alternativas generadoras de valor. *Universidad & Empresa*, 16(27), 281-302.

<https://doi.org/10.12804/rev.univ.empresa.27.2014.10>

Sahni, B. P. (2022). Overview on estimation in agile methodology. *Journal of Artificial*

Intelligence, Machine Learning and Data Science, 1(1), 511-514.

<https://doi.org/10.51219/JAIMLD/bhupinder-paul/137>

Salto-Viteri, H., y Franco-Bayas, M. (2019). La transición del manejo de bases de datos

entre el modelo SQL al NoSQL en la enseñanza de carreras tecnológicas. *Journal of*

Science and Research, E-ISSN: 2528-8083. <https://doi.org/10.5281/zenodo.3598497>

Sánchez-Riofrío, A.-M., Faytong-Haro, M., Roa, O., y Valderrama-Álvarez, J. (26 de 8 de

2024). Enhancing Women's Entrepreneurship in Rural Ecuador: A Study Protocol

Aligned with the Sustainable Development Goals. *Economies*, 12, 225.

<https://doi.org/10.3390/economies12090225>

Santana, Y. M., y Deodato, T. C. (2024). Cultura organizacional en la gestión de talento

humano de las Pymes, cantón Santa Elena 2024. *JOURNAL BUSINESS SCIENCE*,

5(1), 161-175. <https://doi.org/10.56124/jbs.v5i1.011>

Sarrab, M., Al-Shihi, H., y Safia, N. (2021). *Handbook of Mobile Application Development: A*

Guide to Selecting the Right Engineering and Quality Features. Bentham Science

Publishers Pte. Ltd.

- Sathiya-Priya, M., y Vijay-Ananth, S. (2025). A COMPARATIVE STUDY TOWARDS ANDROID OS AND iOS. *International Journal Of Research In Technology And Innovation*, 10(3), b68-b78. <https://www.ijrti.org/viewpaperforall?paper=IJRTI2503110>
- Schwaber, K., y Sutherland, J. (2020). *La Guía Definitiva de Scrum: Las Reglas del Juego*. SCRUMstudy. (2022). *Guía de los fundamentos de Scrum (Guía del SBOK®) – Cuarta edición*. SCRUMstudy. <https://scrumstudy.com.mx/wp-content/uploads/2025/01/SCRUMstudy-SBOK-Guide-4th-Edition-Spanish.pdf>
- Semma, A. B., Ali, M., Saerozi, M., Mansur, M., y Kusri. (2023). Cloud computing: Google Firebase Firestore optimization analysis. *Indonesian Journal of Electrical Engineering and Computer Science*, 29(3), 1719-1728. <https://doi.org/10.11591/ijeecs.v29.i3.pp1719-1728>
- Silva, N. C., Samayani, D. A., Alejandro, J. P., Carnero, M. Z., y Sulla-Torres, J. A. (2022). Development of a mobile application for the registration and attendance control of university students based on Machine Learning. En Proceedings Of The 2nd LACCEI International Multiconference On Entrepreneurship, Innovation And Regional Development (LEIRD 2022): “Exponential Technologies And Global Challenges: Moving Toward A New Culture Of Entrepreneurship And Innovation For Sustainable Development”. <https://doi.org/10.18687/leird2022.1.1.8>
- Skrzypiec, S., y Plechawska-Wójcik, M. (2023). Comparative analysis of Angular and React frameworks. *Journal of Computer Sciences Institute*, 28, 256-263. <https://doi.org/10.35784/jcsi.3724>
- Slinger, E., Pather, S., y du Plessis, M. (2024). Determinants of mobile application adoption among micro-entrepreneurs. *South African Journal of Information Management*, 26(1), a1731. <https://doi.org/10.4102/sajim.v26i1.1731>
- Solis-Galván, J. A., Vázquez-Reyes, S., Garza-Veloz, I., Velasco-Elizondo, P., Mauricio-Gonzalez, A., y de-la-Luz-Martinez-Fierro, M. (2022). Development of an auxiliary platform (Mentali) for the primary screening of anxiety and depression in young

- adults. *International Journal of Environmental Research and Public Health*, 19, 14033. <https://doi.org/10.3390/ijerph192114033>
- Soto-Fernández, S. L., Reyna-Ángeles, O., Hernández-Ángeles, C., García-Pérez, L. S., y Cruz-Hernández, I. Y. (6 de 2021). Análisis comparativo de bases de datos relacionales y no relacionales, para tener como referencia el tiempo de respuesta en la optimización de consultas. En *Memorias del Congreso Internacional de Investigación Academia Journals Oaxaca 2021* (Vol. 13, pp. 401-404). Academia Journals.
- Tarrillo-Saldaña, O. T., Mejía-Huamán, J. M., Dávila-Mego, J. S., Chilón-Camacho, W. M., Pintado-Castillo, C. A., Tapia-Idrogo, C. E., y Velez-Escobar, S. B. (2024). *Metodología de la investigación una mirada Global Ejemplos prácticos. En CID - Centro de Investigación y Desarrollo eBooks*. Centro de Investigación y Desarrollo. https://doi.org/10.37811/cli_w1078
- Thomas, P., Delia, L., Corbalan, L., Cáseres, G., Fernandez-Sosa, J., Tesone, F., . . . Pesado, P. (2018). Tendencias en el desarrollo de aplicaciones para dispositivos móviles. En *XX Workshop de Investigadores en Ciencias de la Computación* (pp. 588-592).
- Titmus, M. (2021). *Cloud Native Go: Building Reliable Services in Unreliable Environments*. O'Reilly Media.
- Tomás, J., Carbonell, V., Bataller, J., y Lloret, J. (2019). *Firebase: trabajar en la nube*. Marcombo; Alfaomega.
- Torres, C. A., y Cardona, P. F. (2023). Ecosistemas de emprendimiento: hacia una reflexión práctica y conceptual. *Revista Universidad y Empresa*, 24(43), 1-38. <https://doi.org/10.12804/revistas.urosario.edu.co/empresa/a.12044>
- Ujianti, N. M., Antaguna, I. N., Renaya, I. N., Maharani, N. P., Indrawan, I. A., y Sastradi, I. K. (2024). Legal Certainty of Business Licensing Integrated Electronically in Investment Licensing. *International Journal Of Social Science And Human Research*, 07(12). <https://doi.org/10.47191/ijsshr/v7-i12-13>

- Usca-Paguay, E. J. (5 de 2023). *Desarrollo de Aplicaciones Android 1* (1 ed.). Quito: Red Internacional de Investigación en Ciencias Sociales y Humanidades “R2ICS”.
- Valenzuela-Velasco, C. V., Benalcázar-Dalfo, G. B., y Delgado-Saeteros, Z. D. (2024). Gestión de inventarios en organizaciones de emprendimiento: Una aproximación teórica. *Prohominum*, 6(2), 193-204. <https://doi.org/10.47606/acven/ph0242>
- Velásquez, S. M., Monsalve-Sossa, D. E., Zapata, M. E., Gómez-Adasme, M. E., y Ríos, J. P. (2019). Pruebas a aplicaciones móviles: avances y retos. *Lámpsakos*(21), 39-50. <https://doi.org/10.21501/21454086.2983>
- Wu, K., Zhu, M., y Qu, Y. (2024). The impact of digital technology on entrepreneurship—Evidence from China General Social Survey. *PLOS ONE*, 19(9), e0310188. <https://doi.org/10.1371/journal.pone.0310188>
- Yunhee, J. (2024). *An analysis of the impact of UX/UI improvements and web accessibility enhancements on user experience and conversion rates in e-commerce platforms*. <https://doi.org/10.13140/RG.2.2.33671.71846>

8. ANEXOS

Anexo 1: Carta de asignación, registro de recursos y cronograma



Pontificia Universidad Católica del Ecuador
Santo Domingo

DIRECCIÓN DE INVESTIGACIÓN
VINCULACIÓN E INNOVACIÓN

Santo Domingo, 5 de noviembre del 2025

MSc.
Ximena Toro Arévalo
Gerente de la empresa 100 MUJERES S.A.S. B.I.C.
Presente. –

Reciba un atento saludo de quienes formamos parte de la Pontificia Universidad Católica del Ecuador, Sede Santo Domingo.

Por medio del presente informo a usted que el estudiante SILVA AVILA MARCO JAVIER con C.C. 172246608 y la estudiante DIAZ BARREIRO LEONARDO ANDRÉS con C.C. 0803589175 son estudiantes del OCTAVO nivel, asignatura INTEGRACIÓN CURRICULAR, Escuela de SISTEMAS. Actualmente se encuentran elaborando su Trabajo de Titulación denominado: **APLICACIÓN WEB Y MÓVIL PARA LA GESTIÓN DE EMPRENDIMIENTOS EN LA EMPRESA 100 MUJERES S.A.S. B.I.C. DEL CANTÓN SANTO DOMINGO**.
PERIODO 2025-2026, por tal motivo, solicito muy comedidamente se le permita desarrollar en su Institución las siguientes actividades:

ACTIVIDADES:

- Solicitud de información referente al manejo de procesos de gestión de emprendimientos.
- Aplicación de las encuestas dirigidas a las emprendedoras del negocio.
- Aplicación de la entrevista a la gerente de la empresa 100 Mujeres Trabajando SASBIC.
- Entrega de aplicación y documentación del sistema desarrollado.

Los estudiantes deberán mantener la disciplina durante su estancia en la Institución, así como acogerse a las normativas necesarias para su desempeño.

Es necesario aclarar que los resultados de la investigación serán difundidos, mismos que previamente deberán ser comunicados a su persona a partir de lo cual le emitirá una carta que avale el impacto generado.

De antemano le agradezco su amable atención y ayuda en formar profesionales idóneos para servir a la sociedad.

Cordialmente,


 MSc. Mikel Ugando Peña
 RESPONSABLE DE INVESTIGACIÓN FORMATIVA PUCESD
 eduvia@pucesd.edu.ec
 (02)3702868 Ext. 204

Dirección: Vía a Guayaquil Km. 2
 Código postal: 20205. Teléfono: (09) 49526425
 Santo Domingo, Ecuador. <https://pucesd.edu.ec/>





REGISTRO DE RECURSOS DEL PROYECTO			
GASTOS			
Recursos	Cantidad	Valor unitario (USD)	Valor total USD
Humano			
Estudiantes (apoyo/honorarios de desarrollo)	2	\$350.00	\$700.00
Tecnología			
Computadora personal (Alienware m16 R2) – uso del equipo	6	\$55.56	\$333.36
Computadora compañero – uso del equipo (gama media)	6	\$22.22	\$133.32
Firestore (consumo mensual – Firestore/Storage/Auth)	2	\$5.00	\$10.00
Hosting (Frontend – App web)	2	\$5.00	\$10.00
Hosting (Backend/API)	2	\$5.00	\$10.00
Hosting (Base de Datos / MySQL)	2	\$5.00	\$10.00
Dominio (anual)	1	\$12.00	\$12.00
Materiales			
Copias / impresiones (anexos y evidencias)	80	\$0.25	\$20.00
Empastado / anillado	2	\$6.00	\$12.00
Papelería (carpetas, hojas, etc.)	1	\$5.00	\$5.00
Servicios			
Transporte (reuniones/validación/pruebas)	6	\$3.50	\$21.00
Internet (mes)	2	\$20.00	\$40.00
Energía eléctrica (mes)	2	\$10.00	\$20.00
TOTAL GASTOS			\$1,336.68
INGRESOS			
Recursos	Cantidad	Valor unitario (USD)	Valor total USD
Fuente de ingresos			
Recursos propios	1	\$1,336.68	\$1,336.68
TOTAL INGRESOS			\$1,336.68
TOTAL - BALANCE (Ingresos - Gastos)			\$0.00

PRODUCT BACKLOG DEL TRABAJO DE TITULACIÓN DE GRADO															
N°	Actividades	Estado	2025						2026						
			Producto Mínimo Viable I = 8° nivel						Producto Mínimo Viable II = 9° nivel						
			Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre	Enero	Febrero	Marzo
ÉPICA 1	Desarrollo de la propuesta	HECHO		😊											
ÉPICA 2	Ejecución del Proyecto	HECHO													😊
ÉPICA 3	Revisión literaria	HECHO				😊									
ÉPICA 4	Metodología de la investigación	HECHO				😊									
ÉPICA 5	Resultados	HECHO													😊
ÉPICA 5.1	Del primer objetivo específico	HECHO					😊								
ÉPICA 5.2	Del segundo objetivo específico	HECHO						😊							
ÉPICA 5.3	Del tercer objetivo específico	HECHO									😊				
ÉPICA 6	Desarrollo de la propuesta de intervención (Gestión adaptativa con incremento iterativo "Scrum")	HECHO													
ÉPICA 6.1	Prototipo de diseño operativo, llamado solución en punta	HECHO													
ÉPICA 6.2	Sprint 1	HECHO													
ÉPICA 6.3	Sprint 2	HECHO													
ÉPICA 6.4	Sprint 3	HECHO													
ÉPICA 6.5	Sprint 4	HECHO													
ÉPICA 7	Conclusiones y Recomendaciones	HECHO													😊
ÉPICA 8	Informe Final del Trabajo de Titulación de Grado	HECHO													😊
ÉPICA 8.1	Correcciones de Lectores	HECHO													😊
ÉPICA 8.2	Anexos	HECHO													😊
ÉPICA 9	Disertación de Grado	HECHO													😊

Nota : PMV=Producto mínimo viable; 😊 = Relaciona la semana con "Done"

Anexo 2: Carta de impacto y consentimiento informado



Santo domingo, 19 de enero de 2026

Mg. Yullio Cano de la Cruz
Director de Postgrados
Pontificia Universidad Católica del Ecuador Sede Santo Domingo
Presente. -

De mi consideración:

Reciba un cordial saludo y deseos de éxitos en sus delicadas funciones.

Por medio del presente, pongo en su conocimiento que el Trabajo de Titulación denominado **Aplicación Web y Móvil para la Gestión de Emprendimientos en la Empresa 100 Mujeres S.A.S. B.I.C, del Cantón Santo Domingo**; elaborado por los estudiantes, **Marco Silva y Leonardo Díaz**, estudiantes de Ingeniería en Tecnologías de la Información ha significado un gran aporte y ha favorecido al fortalecimiento organizacional de nuestra empresa, generando un impacto en las emprendedoras, en los clientes y en la comunidad del cantón Santo Domingo.

En nombre de todas las mujeres que trabajamos día a día por fortalecer nuestra provincia y el país agradecemos y felicitamos este tipo de iniciativas y vinculación con la comunidad, pues son el fiel reflejo del verdadero papel que debe jugar la academia al servicio de la sociedad.

Sin otro particular le reitero mi más alta consideración por la atención dada a la presente, me suscribo de usted.

Atentamente,



MSc. Ximena Toro
Gerente de 100 Mujeres Trabajando



SANTO DOMINGO
DIRECCIÓN DE INVESTIGACIÓN
VINCULACIÓN E INNOVACIÓN

Consentimiento informado

La empresa/organización 100 MUJERES S.A.S. B.I.C. ubicado en Santo Domingo, libre y voluntariamente participa en el proyecto de Trabajo de Titulación de Grado de la Pontificia Universidad Católica del Ecuador Sede Santo Domingo, con el título **"Aplicación Web y Móvil para la Gestión De Emprendimientos en la Empresa 100 Mujeres S.A.S. B.I.C, del Cantón Santo Domingo"**, elaborado por Marco Silva y Leonardo Díaz, estudiantes de la carrera de Ingeniería de Tecnologías de la Información.

Luego de firmar este documento certifico lo siguiente:

- Recibimos una copia de este documento de consentimiento informado.
- Estamos de acuerdo en que los datos recopilados, fotografías y resultados de este proyecto de Trabajo de Titulación de Grado se publiquen en artículos académicos, conferencias, en páginas web institucionales y en otros medios de comunicación.
- No esperamos recibir beneficios o pago por la participación.

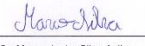
Y a los efectos que procedan, firmamos el presente consentimiento informado.
Santo Domingo, 11 de diciembre del 2025.

Firma de la Gerente de 100 Mujeres Trabajando



MSc. Ximena Toro

Firma de los autores del Trabajo de Titulación de Grado:



Sr. Marco Javier Silva Avila



Sr. Leonardo Andrés Díaz Barreiro

Dirección: Vía a Chone Km. 2
Código postal: 230202 - Teléfono: (593-09) 2834225
Santo Domingo, Ecuador - <http://pucesd.edu.ec/>

30 años
fe, razón y corazón

Anexo 3: Validación de instrumentos de recolección de datos



SANTO DOMINGO

Santo Domingo, 03 de julio de 2025

Estimado Mg. Luis Ulloa,

De mi consideración:

El motivo del presente es que le hemos elegido a usted para redactar la solicitud de revisión y validación de los instrumentos de recolección de datos.

A continuación, encontrará la entrevista y encuesta que contienen las preguntas que permitirán recolectar información, de acuerdo con el trabajo de titulación de grado **"APLICACIÓN MÓVIL Y WEB PARA LA GESTIÓN DE EMPRENDIMIENTOS EN LA EMPRESA "100 MUJERES S.A.S. B.I.C", DEL CANTON SANTO DOMINGO"**, dirigida a las emprendedoras del proyecto social 100 Mujeres Trabajando.

Para la validación de los instrumentos se adjunta la operacionalización de variables, con la finalidad de que se visualice la relación de las preguntas con las categorías e indicadores. Además, se encuentran divididos los instrumentos en dos partes, la primera corresponde a la entrevista (preguntas de fondo verde) para las dos variables) y la segunda a la encuesta (preguntas de la variable independiente de fondo turquoise y las preguntas de la variable dependiente fondo gris).

Gracias por su valiosa colaboración en este trabajo de titulación de grado.


Marco Javier Silva Avila
mjsilva@pucesd.edu.ec


Leonardo Andrés Díaz Barreiro
ladiaz@pucesd.edu.ec

SANTO DOMINGO

Variable independiente: Aplicación Móvil	Dimensiones	Indicadores	Preguntas	Herramienta
Según Banchawe (2022) una aplicación móvil es un software que normalmente se ejecuta en dispositivos móviles personales, tal como teléfonos inteligentes o tabletas.	Plataforma de ejecución	Android	¿Qué celular tiene? Android o iOS?	Encuesta a Emprendedoras
		iOS	¿Le gustaría usar la aplicación en otro dispositivo?	Encuesta a Emprendedoras
Consecuentemente, para su funcionamiento se compone de una interfaz gráfica de usuario, almacenar datos de comunicación, tanto entre dispositivos como con servidores empresariales a través de redes adecuadas. Con este sentido, se tienen frameworks de interfaz gráfica, los cuales proporcionan librerías, con contenedores de interacción móvil para la interacción de usuario. En otras palabras, las aplicaciones móviles proporcionan necesidades	Entorno de desarrollo móvil	Multiparadigma	¿Qué tan necesario cree que es usar la App en varios dispositivos?	Encuesta a Emprendedoras
		Flutter	¿Esta usted de acuerdo con que la App funciones igual en cualquier dispositivo?	Encuesta a Emprendedoras
Arquitectura de la aplicación	Librerías y paquetes externos (Pub.dev)	Dart	¿Qué tan buena es la experiencia de usuario en su aplicación?	Reunión de planificación del sprint con la Administradora
		Clean Architecture	¿Qué tan importante considera que sea una aplicación tenga una estructura ordenada y clara?	Encuesta a emprendedoras
Gestión de estado de la aplicación	Estructura modular del código		¿Le gustaría que el sistema le guíe paso a paso para hacer pedidos o registros?	Encuesta a emprendedoras
	Navegación entre pantallas		¿Qué tan fácil le resulta el uso de una aplicación móvil?	Encuesta a emprendedoras
	Gestión de estado de la aplicación		¿Le gustaría ver en tiempo real si tu pedido está aceptado o en camino?	Encuesta a Emprendedoras

Variable dependiente: Gestión de emprendimientos

[illegible]

básicas para la interacción	Diseño funcional del	Diseño de interfaz de
-----------------------------	----------------------	-----------------------

	Sistema móvil	Sistema de escritorio
El almacenamiento de datos y el soporte para la comunicación de datos. En apoyo a múltiples sistemas móviles, surgen aplicaciones móviles	Diseño de experiencia de usuario (UX)	Encuesta a emprendedores
Habitualmente, y bajo otro enfoque que únicamente rescató un aspecto del desarrollo, se han enfrentado los desafíos que se tradujeron en versiones nativas de las plataformas compatibles (fig. 4, 57, 113, 222).	Capacidades del dispositivo móvil	Encuesta a emprendedoras
	Firebase	Encuesta a emprendedoras
	SQL lite	Encuesta a emprendedoras
	APIs	Encuesta a emprendedoras
	Seguridad de datos y autenticación	Encuesta a emprendedoras

Preguntas

Objetivo:

Objetivo:
Implementar una aplicación web y móvil que permita optimizar la gestión de emprendimientos del proyecto 100 Mujeres Trabajando del cantón Santo Domingo.

Baremog

Claridad: Se refiere si la pregunta está comprendida por los destinatarios.

Pertinencia: Se refiere si la pregunta corresponde con lo que se quiere indagar

Las preguntas en cuanto a su claridad y pertinencia se encuentran bajo la escala valorativa Likert del 1 al 5 (donde 1 es el menor valor y 5 el mayor). Podrá añadir una formulación alternativa y observación, en caso de que considere necesario

Marque con una cruz (X) el tramo del baremo que exprese mejor su juicio "Claridad" y "Pertinencia" sobre los ítems propuestos:

Reunión de planificación del sprint con la administradora de la empresa

Variable independiente: Aplicación Móvil (**Categoría:** Entorno de desarrollo móvil)

1. ¿Qué funciones considera más importantes para garantizar los pedidos y seguimiento de estos?

CLARIDAD					PERTINENCIA					FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1	2	3	4	5	1	2	3	4	5		

Variable independiente: Aplicación Móvil (**Categoría:** Almacenamiento y servicios backend)

2. ¿Qué medidas considera necesarias para validar los comprobantes de pago subidos por las emprendedoras y garantizar la seguridad de esta información?

CLARIDAD					PERTINENCIA					FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1	2	3	4	5	1	2	3	4	5		

Variable independiente 2: Aplicación Web

Conceptualización	Indicadores	Preguntas	Encuentra a E
En palabras de Jerson Hidalgo Crespo (2020), una política que opera es una solución informática que opera sobre la arquitectura web, utilizando sendos y navegadores para resolver problemas o satisfacer necesidades específicas de los usuarios.	División de programación PIP JavaScript CSS React Herramientas de desarrollo Código Visual Studio Code	¿Qué tan organizada es la programación de la arquitectura web? ¿Cómo se ve una solución informática para un negocio, como página Web? ¿Considera que el desarrollo de una aplicación web, de escritorio o una herramienta para administrar una información, es una experiencia agradable? ¿Considera que la aplicación de una herramienta para administrar una información, le permite hacer más?	Reunión de planificación del sprint con la Administración Encuentra a E emprendedora
Más, sin embargo, según la Universidad ECU (2022), una aplicación web es una arquitectura web, utilizando sendos y navegadores para proporcionar funcionalidades que resuelven problemas o satisfacer necesidades específicas de los usuarios.	Analisis de Datos Relacional: MySQL No Relacional: Metadatos Tecnologías Diseño de Software	¿Qué tan eficaz es para E enviar y recibir información digitalmente? ¿Qué tan organizada es la información de sus emprendimientos? ¿Cómo se ve la información de sus emprendimientos? ¿Con qué frecuencia organiza la información de su emprendimiento? ¿Qué tan E se prepara para enviar la información de E?	Encuentra a E emprendedora Reunión de planificación del sprint con la Administración Reunión de planificación del sprint con la Administración Encuentra a E emprendedora
	Metadatos Tecnologías Diseño de Software		Reunión de planificación del sprint con la Administración Encuentra a E emprendedora

Variable independiente: Aplicación Móvil (**Categoría:** Diseño funcional del sistema móvil)

3. Considerando a emprendedoras con diferentes niveles de educación, ¿Qué criterios de diseño considera se han de tomar en cuenta?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Web (**Categoría:** Lenguaje de programación)

4. ¿Cuán tan familiarizado se encuentra con el uso de aplicaciones Web?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Web (**Categoría:** Herramientas de Desarrollo)

5. ¿Considera que al desarrollo de una aplicación web, de acuerdo a los requerimientos, puede fortalecer la gestión de operaciones y logística en su empresa?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Web (**Categoría:** Software de Programación)

6. ¿Considera que la aplicación de nuevas tecnologías puede facilitar la gestión de talento humano?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Web (**Categoría:** Base de Datos)

7. ¿Esta de acuerdo en almacenar toda la información de sus emprendedoras productos en una aplicación tecnológica?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Web (**Categoría:** Metodología de Desarrollo de Software)

8. ¿Cómo guarda la información de las emprendedoras?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Web (**Categoría:** Metodología de Desarrollo de Software)

9. ¿Cuál es el proceso para registrar la información de las emprendedoras?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de Emprendimientos (**Categoría:** Gestión de Operaciones y Logística)

10. ¿Qué medidas de control de calidad considera importantes para asegurar la entrega de pedidos?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de Emprendimientos (**Categoría:** Gestión de Operaciones y Logística)

11. ¿Qué cambios cree usted que se deben realizar para mejorar la gestión de pedidos?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de Emprendimientos (**Categoría:** Gestión de Operaciones y Logística)

12. ¿La empresa tiene planes de incrementar la oferta de productos?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de Emprendimientos (**Categoría:** Gestión de Talento Humano)

13. ¿Pueden las emprendedoras vender otros productos además de la oferta de la empresa?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de Emprendimientos (**Categoría:** Gestión Legal y Aspectos Regulatorios)

14. ¿Cómo está constituida legalmente la empresa?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de Emprendimientos (**Categoría:** Gestión de Operaciones y Logística)

15. ¿Qué licencias o permisos son los necesarios para garantizar el trabajo de las emprendedoras?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de Emprendimientos (**Categoría:** Gestión de Redes y Ecosistemas Emprendedores)

16. ¿Cómo evalúa la empresa la coordinación y apoyo entre emprendedoras?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de Emprendimientos (**Categoría:** Gestión de Redes y Ecosistemas Emprendedores)

17. ¿Se da algún tipo de apoyo inicial a las emprendedoras?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de Emprendimientos (**Categoría:** Gestión de Redes y Ecosistemas Emprendedores)

18. ¿Cómo se realiza la distribución de las emprendedoras para asignarles las zonas de venta?

CLARIDAD	PERTINENCIA	FORMULACIÓN ALTERNATIVA	OBSERVACIÓN
1 2 3 4 5	1 2 3 4 5		

Encuesta dirigida a los clientes

Tema del Trabajo de Titulación de grado: Aplicación Móvil y Web para la Gestión de Emprendimientos en la Empresa "100 Mujeres S.A.S. B.I.C.", del cantón Santo Domingo.

Objetivo: Recopilar la información necesaria para validar la propuesta de optimización de la gestión de emprendimientos, específicamente en el proceso de pedidos del producto *bandejas de empanadas*.

Instrucciones al público objetivo: La encuesta está dirigida a las emprendedoras de la Empresa "100 MUJERES S.A.S. B.I.C." en base a la información obtenida, permitirá conocer el manejo de nuevas tendencias tecnológicas en el manejo de procesos de promoción.

PREGUNTAS

Variable independiente: Aplicación Móvil (**Categoría:** Plataforma de ejecución)

1. ¿Qué celular tienes para usar la App?

- A. Android
B. iOS

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Plataforma de ejecución)

2. ¿Te gustaría usar la aplicación en otro dispositivo?

- A. Totalmente de acuerdo
B. De acuerdo
C. Neutral
D. Poco de acuerdo
E. Nada de acuerdo

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Plataforma de ejecución)

3. ¿Qué tan necesario crees que es usar la App en varios dispositivos?

- A. Muy necesario
B. Necesario
C. Medianamente necesario
D. Poco necesario
E. Innecesario

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Entorno de desarrollo móvil)

4. ¿Está usted de acuerdo con que la App funcione igual en cualquier dispositivo?

- A. Totalmente de acuerdo
B. De acuerdo
C. Neutral
D. Poco de acuerdo
E. Nada de acuerdo

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Arquitectura de la aplicación)

5. ¿Qué tan importante consideras que una aplicación tenga una estructura ordenada y clara?

- A. Muy importante
B. Importante
C. Medianamente importante
D. Poco importante
E. Nada importante

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Arquitectura de la aplicación)

6. ¿Te gustaría que el sistema te guíe paso a paso para hacer pedidos o registros?

- A. Totalmente de acuerdo
B. De acuerdo
C. Neutral
D. Poco de acuerdo
E. Nada de acuerdo

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Arquitectura de la aplicación)

7. ¿Qué tan fácil te resulta el uso de una aplicación móvil?

- A. Muy fácil
B. Fácil
C. Normal
D. Difícil
E. Muy difícil

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Arquitectura de la aplicación)

8. ¿Te gustaría ver en tiempo real si tu pedido está aceptado o en camino?

- A. Totalmente de acuerdo
B. De acuerdo
C. Neutral
D. Poco de acuerdo
E. Nada de acuerdo

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Diseño funcional del sistema móvil)

9. ¿Qué tan satisfecha estás con el diseño, cumple con lo que esperabas de la aplicación?

- A. Totalmente satisfecha
B. Satisfecha
C. Medianamente satisfecha
D. Poco satisfecha
E. Nada satisfecha

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Diseño funcional del sistema móvil)

10. ¿Te resulta fácil usar una App para hacer tus pedidos?

- A. Muy fácil
B. Fácil
C. Normal
D. Difícil
E. Muy difícil

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Diseño de experiencia de usuario - UX)

11. ¿Qué tan fácil te resulta entender los iconos e imágenes en una aplicación?

- A. Muy fácil
B. Fácil
C. Normal
D. Difícil
E. Muy difícil

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Diseño de experiencia de usuario - UX)

12. ¿Qué tanto estás de acuerdo, en que una aplicación pensada para emprendedoras considere el uso de letras grandes y colores claros?

- A. Totalmente de acuerdo
B. De acuerdo
C. Neutral
D. Poco de acuerdo
E. Nada de acuerdo

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Diseño de experiencia de usuario - UX)

13. ¿Te gustaría que puedas calcular el total de tu pedido automáticamente?

- A. Totalmente de acuerdo
B. De acuerdo
C. Neutral
D. Poco de acuerdo
E. Nada de acuerdo

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Capacidades del dispositivo móvil)

14. ¿Te gustaría que tus clientes puedan ver en un mapa dónde estás vendiendo hoy?

- A. Totalmente de acuerdo
B. De acuerdo
C. Neutral
D. Poco de acuerdo
E. Nada de acuerdo

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Capacidades del dispositivo móvil)

15. ¿Te gustaría recibir notificaciones cuando tu pedido esté listo?

- A. Totalmente de acuerdo
B. De acuerdo
C. Neutral
D. Poco de acuerdo
E. Nada de acuerdo

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Almacenamiento y servicios backend)

16. ¿Te da confianza compartir tus datos en una App?

- A. Totalmente de acuerdo
B. De acuerdo
C. Neutral
D. Poco de acuerdo
E. Nada de acuerdo

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Almacenamiento y servicios backend)

17. ¿Qué tan necesario consideras poder usar la App sin internet?

- A. Muy necesario
B. Necesario
C. Medianamente necesario
D. Poco necesario
E. Innecesario

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Móvil (**Categoría:** Almacenamiento y servicios backend)

18. ¿Qué tan segura te sentirías al compartir tu comprobante de pago en la App?

- A. Totalmente segura
B. Segura
C. Moderadamente segura
D. Poco segura
E. Nada segura

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Web (**Categoría:** Lenguaje de programación)

19. ¿Cómo te sientes al usar tecnología para tu negocio, como páginas Web?

- A. Totalmente de acuerdo
B. De acuerdo
C. Neutral
D. Poco de acuerdo
E. Nada de acuerdo

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Web (**Categoría:** Software de Programación)

20. ¿Qué tan difícil es para ti enviar y recibir información digitalmente?

- A. Muy complicado
B. Complicado
C. Neutral
D. Poco complicado
E. Nada complicado

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable independiente: Aplicación Web (**Categoría:** Metodología de Desarrollo de Software)

21. ¿Con qué frecuencia organiza la información de su emprendimiento?

- A. Muy frecuentemente
B. Frecuentemente
C. Ocasionalmente
D. Raramente
E. Nunca

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de emprendimientos (**Categoría:** Gestión de Operaciones y Logística)

22. ¿Qué tan satisfecha te sientes con la forma actual de pedir sus pedidos?

- A. Totalmente satisfecha
B. Satisfecha
C. Medianamente satisfecha
D. Poco satisfecha
E. Nada satisfecha

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de emprendimientos (**Categoría:** Gestión de Operaciones y Logística)

23. ¿Con qué frecuencia realizan pedidos actualmente?

- A. Siempre
B. Casi siempre
C. A veces
D. Casi nunca
E. Nunca

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Variable dependiente: Gestión de emprendimientos (**Categoría:** Gestión de Operaciones y Logística)

24. ¿Qué tan fácil te resulta gestionar sus pedidos?

- A. Totalmente satisfecha
B. Satisfecha
C. Medianamente satisfecha
D. Poco satisfecha
E. Nada satisfecha

Relevancia	Claridad	Formulación alternativa:	Observación:
1 2 3 4 5	1 2 3 4 5		

Pontificia Universidad Católica del Ecuador		SANTO DOMINGO									
Variable dependiente: Gestión de emprendimientos (Categoría: Gestión del Talento Humano) 25. ¿Con qué datos se siente más cómoda registrándose? A. Solo correo B. Solo número C. Ambos D. Otro E. Prefiero no compartir <table border="1"> <thead> <tr> <th>Relevancia</th> <th>Claridad</th> <th>Formulación alternativa:</th> <th>Observación:</th> </tr> </thead> <tbody> <tr> <td>1 2 3 4 5</td> <td>1 2 3 4 5</td> <td></td> <td></td> </tr> </tbody> </table>				Relevancia	Claridad	Formulación alternativa:	Observación:	1 2 3 4 5	1 2 3 4 5		
Relevancia	Claridad	Formulación alternativa:	Observación:								
1 2 3 4 5	1 2 3 4 5										
Variable dependiente: Gestión de emprendimientos (Categoría: Gestión del Talento Humano) 26. ¿Qué tan necesario sería capacitarte si cambia la forma de realizar pedidos? A. Muy necesario B. Necesario C. Medianamente necesario D. Poco necesario E. Innecesario <table border="1"> <thead> <tr> <th>Relevancia</th> <th>Claridad</th> <th>Formulación alternativa:</th> <th>Observación:</th> </tr> </thead> <tbody> <tr> <td>1 2 3 4 5</td> <td>1 2 3 4 5</td> <td></td> <td></td> </tr> </tbody> </table>				Relevancia	Claridad	Formulación alternativa:	Observación:	1 2 3 4 5	1 2 3 4 5		
Relevancia	Claridad	Formulación alternativa:	Observación:								
1 2 3 4 5	1 2 3 4 5										
Variable dependiente: Gestión de emprendimientos (Categoría: Gestión Legal y Aspectos Regulatorios) 27. ¿Qué tan difícil consideras conseguir los permisos para vender? A. Muy difícil B. Difícil C. Normal D. Fácil E. Muy fácil <table border="1"> <thead> <tr> <th>Relevancia</th> <th>Claridad</th> <th>Formulación alternativa:</th> <th>Observación:</th> </tr> </thead> <tbody> <tr> <td>1 2 3 4 5</td> <td>1 2 3 4 5</td> <td></td> <td></td> </tr> </tbody> </table>				Relevancia	Claridad	Formulación alternativa:	Observación:	1 2 3 4 5	1 2 3 4 5		
Relevancia	Claridad	Formulación alternativa:	Observación:								
1 2 3 4 5	1 2 3 4 5										
Datos generales Edad: A. 18-25 B. 26-30 C. 30-40 D. 40 o más Género: A. Femenino B. Masculino Discapacidad: A. Auditiva B. Visual C. Física D. Otro E. Ninguna											
Pontificia Universidad Católica del Ecuador SANTO DOMINGO Seréis mis testigos ¡GRACIAS POR SU COLABORACIÓN! Una vez finalizada su validación, puede realizar comentarios, sugerencias o la aprobación, además, es pertinente que agregue sus datos personales. Comentarios de validación: Se ha revisado lo solicitado, y los instrumentos son pertinentes para su aplicación. Datos informativos del experto Nombres y Apellidos: Luis Javier Ulloa Meneses Profesión y cargo: Profesor Tiempo Completo (Titular Agregado I) Título universitario: Magister en Big Data y Data Science Email: ljuilloa@pucesd.edu.ec Fecha y hora de validación: 14/07/2025 – 20h00  Firma											
Pontificia Universidad Católica del Ecuador SANTO DOMINGO Seréis mis testigos ¡GRACIAS POR SU COLABORACIÓN! Una vez finalizada su validación, puede realizar comentarios, sugerencias o la aprobación, además, es pertinente que agregue sus datos personales. Comentarios de validación: La encuesta y entrevista está bien diseñada. Sin embargo, tiene varias preguntas erróneas, puesto que no debería formar parte de la encuesta ya que se dirigen a consumidores finales. Datos informativos del experto Nombres y Apellidos: NELSON CARRIÓN BOSQUEZ Profesión y cargo: VICEDECANO DE INVESTIGACIÓN FACE UNIVERSIDAD CATÓLICA DEL NORTE DE CHILE. Título universitario: DOCTOR EN ADMINISTRACIÓN ESTRATÉGICA DE EMPRESAS Email: nelson.carrión@ucn.cl Fecha y hora de validación: 11 de agosto 2025											

Anexo 4. Historias de usuario

HISTORIAS DE USUARIO – APP MÓVIL

Historia de Usuario	
Número: 1	Usuario: Usuario no autenticado
Nombre historia: Pantalla de Bienvenida	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada:
Programador responsable:	
Descripción: Como usuario no autenticado quiero ver una pantalla de bienvenida con información básica del proyecto y botones para registrarme o iniciar sesión, para entender de inmediato qué hace la App y decidir cómo continuar.	
Escenario de prueba: - Dado que abro la App, cuando carga la pantalla de bienvenida, entonces veo el logo, nombre, frase breve y tres botones: Registrarse, Iniciar sesión, Continuar como visitante. - Dado que toco Registrarse, cuando navego, entonces se abre la pantalla de selección de rol. - Dado que toco Iniciar sesión, cuando navego, entonces se abre la pantalla de Login. - Dado que toco Continuar como visitante, cuando navego, entonces entro a la pantalla principal con acceso limitado (si aplica).	

Historia de Usuario	
Número: 3	Usuario: usuario (Cliente o Emprendedora)
Nombre historia: Selección de Rol	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada:
Programador responsable:	
Descripción: Como usuario (Cliente o Emprendedora), quiero poder elegir si me registro como cliente o como emprendedora para que la app se configure correctamente según mi rol.	
Escenario de prueba: - Dado que llego a la pantalla de selección de rol, cuando elijo Cliente, entonces me lleva al formulario de registro de cliente. - Dado que elijo Emprendedora, entonces me lleva al formulario de registro de emprendedora. - Si no selecciono nada y presiono continuar, entonces se muestra un mensaje de validación.	

Historia de Usuario	
Número: 5	Usuario: Cliente o emprendedora
Nombre historia: Ingreso de Usuario	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como cliente o emprendedora Quiero iniciar sesión con mi número de cédula y contraseña Para acceder a mi cuenta y realizar pedidos.	
Escenario de prueba: - Dado que ingreso mis credenciales correctas Cuando presiono 'Iniciar sesión', Entonces accedo al sistema. - Dado que ingreso datos incorrectos Cuando presiono 'Iniciar sesión' Entonces se muestra un mensaje de error.	

Historia de Usuario	
Número: 7	Usuario: Emprendedora/Cliente/Administradora
Nombre historia: Dashboard Principal y Navegación por Rol	
Prioridad en negocio: Muy Alta	Riesgo en desarrollo: Moderado
Puntos estimados: 3	Iteración asignada:
Programador responsable:	
Descripción: Como usuario de la App (Cliente, Emprendedora o Admin), quiero acceder a un Dashboard Principal que muestre de forma clara las opciones comunes y principales según mi rol, para orientarme fácilmente, visualizar lo más relevante para mí y acceder solo a las funciones que me corresponden.	
Escenario de prueba: Dado que accedo después de iniciar sesión o registrarme, cuando se carga el Dashboard, entonces visualizo el logo, mi perfil, y un conjunto de opciones ordenadas según mi rol (por ejemplo: "Hacer pedido" si soy cliente, "Ver pedidos" si soy admin). Dado que navego por las secciones desde el Dashboard, cuando selecciono una opción (como "Mapa" o "Pedidos"), entonces se me redirige correctamente a la vista correspondiente, manteniendo el diseño coherente y accesible. Dado que soy un usuario con permisos específicos (ej. administrador), cuando ingreso, entonces no puedo acceder a módulos que no corresponden a mi rol (por ejemplo: realizar pedidos si soy solo gestor).	

Historia de Usuario	
Número: 2	Usuario: Usuario de la aplicación
Nombre historia: Configuración de accesibilidad y preferencias de visualización	
Prioridad en negocio: Media	Riesgo en desarrollo: Medio
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como usuario de la aplicación quiero poder ajustar el tamaño de texto y el tema de color (claro, oscuro o automático) para mejorar la legibilidad y usar la app de forma cómoda según mis necesidades.	
Escenario de prueba: - Dado que accedo a la pantalla de configuración de accesibilidad, Cuando deslizo el control de tamaño de texto, Entonces la vista previa y el contenido de la app se ajustan dinámicamente. - Dado que elijo un modo de tema (claro, oscuro o automático), Cuando guardo mi elección, Entonces la aplicación se muestra con el esquema de colores correspondiente y mantiene contraste legible. - Dado que estoy en cualquier pantalla de la app, Cuando presiono el ícono de configuración, Entonces puedo acceder rápidamente al diálogo de Accesibilidad sin salir del flujo principal. - Dado que cierro sesión y vuelvo a entrar, Cuando la app inicia, Entonces mantiene la preferencia guardada para tamaño de texto y tema.	

Historia de Usuario	
Número: 4	Usuario: Cliente o Emprendedora
Nombre historia: Registro de Usuario	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como cliente o emprendedora Quiero registrarme con mis datos personales Para poder gestionar mi emprendimiento desde la aplicación.	
Escenario de prueba: Dado el registro de mis datos (nombre, apellidos, dirección, teléfono, cédula o RUC, contraseña) Cuando presiono 'Registrarse' Entonces se muestra un mensaje y mi cuenta se guarda correctamente. Dado que los datos son incompletos o erróneos Cuando intento registrarme, Entonces se muestran los errores para corregirlos.	

Historia de Usuario	
Número: 6	Usuario: administradora
Nombre historia: Solicitud de Acceso para Administradoras	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 3	Iteración asignada:
Programador responsable:	
Descripción: Como posible administradora, quiero solicitar acceso especial al sistema mediante un formulario de verificación, para que la empresa pueda validar mis credenciales y concederme permisos administrativos si corresponde.	
Escenario de prueba: - Dado que accedo a la opción "¿Eres administradora?" desde la pantalla de inicio o login, cuando selecciono esa opción, entonces se me solicita ingrese el código de invitación para administrador y formulario especial para solicitar acceso administrativo. - Dado el ingreso de un código válido, cuando presiono verificar entonces se muestra un formulario para crear la cuenta administradora. - Dado el ingreso de un código inválido cuando presiono verificar entonces se muestra un mensaje indicando la causa. - Dado que completo el formulario con mis datos personales (nombre, cédula/RUC, cargo, motivo) y documentos opcionales, cuando presiono "Enviar solicitud", entonces el sistema guarda la solicitud y muestra un mensaje de confirmación. - Dado que dejo campos obligatorios vacíos o ingreso datos inválidos, cuando intento enviar el formulario, entonces se muestra una advertencia indicando qué campos deben corregirse. - Dado que ya envié una solicitud anteriormente, cuando vuelvo a ingresar, entonces el sistema me informa que mi solicitud está en revisión o ya fue aprobada/rechazada. - Dado que mi solicitud fue aprobada por un administrador interno, cuando intento ingresar al sistema con mi cuenta registrada, entonces tengo acceso al panel administrativo en lugar del panel común.	

Historia de Usuario	
Número: 8	Usuario: usuario autenticado
Nombre historia: Visualización del Perfil de Usuario	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 3	Iteración asignada:
Programador responsable:	
Descripción: Como usuario autenticado, quiero ver mis datos personales registrados, para verificar que mi información esté correcta y actualizada.	
Escenario de prueba: - Dado que visualizo mis datos personales registrados, cuando accedo a la sección de perfil desde el dashboard, entonces se muestra mi nombre, cédula, rol, número de teléfono y foto de perfil si está disponible. - Dado que visualizo si mi perfil está completo, cuando alguno de los datos está ausente (como foto o dirección), entonces se muestra un mensaje o ícono que sugiere completarlo.	

Historia de Usuario	
Número: 9	Usuario: Usuario autenticado
Nombre historia: Cerrar Sesión del Usuario	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Puntos estimados: 3	Iteración asignada:
Programador responsable:	
Descripción: Como usuario autenticado (Cliente, Emprendedora o Administradora), Quiero poder cerrar sesión de forma clara y segura desde la sección de Perfil o Ajustes de Cuenta, Para proteger mis datos personales y evitar accesos no autorizados desde dispositivos compartidos.	
Escenario de prueba: Dado que estoy autenticado dentro de la app, Cuando ingreso a mi menú de Perfil o Ajustes de Cuenta, Entonces puedo ver la opción "Cerrar sesión" claramente disponible. Dado que decido cerrar sesión, Cuando toco el botón "Cerrar sesión", Entonces se muestra un mensaje de confirmación para validar mi decisión ("¿Estás seguro de que quieres cerrar sesión?"). Dado que confirmo cerrar sesión, Cuando acepto la acción, Entonces la App invalida mi sesión actual, elimina mis credenciales locales y me redirige a la Pantalla de Bienvenida o Login. Dado que he cerrado sesión correctamente, Cuando intento volver a la pantalla anterior usando el botón Atrás del dispositivo, Entonces la App NO debe permitirme regresar a la sesión cerrada.	

Historia de Usuario	
Número: 10	Usuario: Cualquiera.
Nombre historia: Recuperación de Contraseña	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada:
Programador responsable:	
Descripción: Como usuario (Cliente, Emprendedora o Admin), quiero recuperar mi contraseña si la olvido, para poder volver a ingresar a la App sin complicaciones y sin depender de soporte técnico	
Escenario de prueba: Dado que olvidé mi contraseña y quiero recuperarla, cundo ingreso a la opción "¿Olvidaste tu contraseña?" en la pantalla de login, entonces se me solicita seleccionar e ingresar mi correo electrónico o cédula para encontrar la cuenta. Dado el intento de recuperación cundo busco mi cuenta con mi documento (Cedula, RUC, o Pasaporte) o correo ingresado entonces se me solicita verificar mediante un medio que permita recuperar la cuenta y mostrar los campos de cambio de contraseña. Dado que he verificado mi intento de recuperación, cundo ingreso una nueva contraseña válida y confirmo, entonces se actualiza la contraseña y puedo iniciar sesión normalmente entonces se solicitará ingresar una nueva contraseña valida.	

Historia de Usuario	
Número: 12	Usuario: Administrador
Nombre historia: Gestión de productos del catálogo (presentaciones y variantes).	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como administradora, quiero crear y editar productos del catálogo central, asignándolos a una categoría y definiendo su presentación (unidad, cantidad y precio base), para ampliar y mantener actualizada la oferta de productos que se muestra a los clientes y emprendedoras en la app.	
Escenario de prueba: 1. Creación de nuevo producto del catálogo Dado que estoy en el módulo de catálogo, Cuando presiono el botón "+ Nuevo producto", Entonces se abre un formulario donde puedo seleccionar una categoría, ingresar el nombre del producto, tipo de unidad, cantidad por unidad, precio base, estado (visible/oculto) y, opcionalmente, variantes (sabores, tipos de masa) e imagen. Dado que completo los campos obligatorios, cundo presiono "Guardar producto", Entonces el producto se guarda correctamente en el catálogo central y aparece en la lista de productos de la categoría seleccionada. 2. Validación de datos del producto Dado que intento guardar el producto con campos obligatorios vacíos o datos inválidos (por ejemplo, precio en blanco o menor o igual a cero), Cuando presiono "Guardar producto", Entonces se muestran mensajes de error junto a los campos correspondientes indicando qué debo corregir. 3. Edición y visibilidad de un producto Dado que un producto ya existe en el catálogo, Cuando lo selecciono y abro la opción "Editar", Entonces puedo modificar su nombre, presentación, precio base, variantes y estado (visible/oculto). Dado que marco un producto como "Oculto", Cuando guardo los cambios, Entonces el producto deja de mostrarse en las pantallas de pedido y en las vistas públicas, pero se mantiene internamente.	

Historia de Usuario	
Número: 11	Usuario: usuario autenticado
Nombre historia: Edición de Perfil de Usuario	
Prioridad en negocio: Alta	Riesgo en desarrollo: Moderado
Puntos estimados: 1	Iteración asignada:
Programador responsable:	
Descripción: Como usuario autenticado (Cliente, Emprendedora o Admin), quiero editar mis datos de perfil (nombre, teléfono, foto de perfil, correo, entre otros) para mantener mi información actualizada y correcta dentro de la app.	
Escenario de prueba: Dado que quiero editar mis datos de perfil, cundo ingreso a la sección "Perfil" y presiono "Editar", entonces se muestran los campos modificables con su contenido actual. Dado que modifico uno o más datos del perfil, cundo presiono "Guardar cambios", entonces los datos se actualizan correctamente y se refleja la nueva información. Dado que intento guardar el perfil con datos incompletos o inválidos, cundo confirmo, entonces se muestra un mensaje de error indicando qué debe corregirse.	

Historia de Usuario	
Número: 13	Usuario: Emprendedora / Cliente
Nombre historia: Realizar pedido de bandejas productos	
Prioridad en negocio: Muy Alta	Riesgo en desarrollo: Moderado
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como emprendedora/cliente Quiero visualizar y seleccionar tipo y cantidad de bandejas de la oferta de productos Para realizar un pedido desde mi cuenta.	
Escenario de prueba: Dado que selecciono tipos y cantidades de bandejas y completo los campos requeridos, cuando confirmo el pedido, entonces se guarda el pedido con estado inicial y createdAt/updatedAt, y se muestra confirmación (pantalla/animación) con número/D y estado. Dado que el pedido fue creado correctamente, cuando navego a Pedidos > Activos, entonces el pedido aparece en la lista y, si es el más reciente, se muestra de forma destacada como "pedido en curso" (sin duplicarse en la lista). Dado que elijo "Retiro" y selecciono un punto de retiro, además ingreso nombre y teléfono, cuando confirmo, entonces el sistema registra pickupPointId/pickupPoint + recipientName + recipientPhone como obligatorios del pedido. Dado que elijo "Entrega" e ingreso dirección, nombre y teléfono, cuando confirmo, entonces el sistema registra deliveryAddress + recipientName + recipientPhone como obligatorios del pedido. Dado que falta al menos un campo requerido, cuando intento confirmar, entonces el sistema no permite confirmar y muestra validaciones en los campos. Dado que selecciono más de 5 bandejas, cuando confirmo, entonces el precio unitario cambia a \$1 y el total se recalcula correctamente.	

Historia de Usuario	
Número: 14	Usuario: Emprendedora / Cliente
Nombre historia: Adjuntar comprobante de pago	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 3	Iteración asignada:
Programador responsable:	
Descripción: Como cliente o emprendedor, quiero subir el comprobante de pago para confirmar que ya realicé el pago del pedido.	
Escenario de prueba: Dado que el pedido fue creado, cuando subo el comprobante dentro de 1 hora, entonces el pedido queda con paymentStatus = "en_revision" (o "pending_validation" según el modelo) y proofUrl asociado, y en Pedidos > Activos se refleja visualmente el estado (chip/badge). Dado que no se sube el comprobante en el tiempo indicado, cuando se consulte el estado en Pedidos, entonces el pedido aparece como pendiente de comprobante / no procesado (sin cambiar automáticamente a "pagado"), y para administración se identifica como pendiente (con indicador de vencimiento si aplica). Dado que ya existe un comprobante asociado al pedido, cuando ingreso al detalle del pedido, entonces puedo visualizar el comprobante (thumbnail y visor fullscreen) y su estado de revisión.	

Historia de Usuario	
Número: 15	Usuario: Emprendedora / Cliente
Nombre historia: Ampliación del pedido: tipos de entrega	
Prioridad en negocio: Alta	Riesgo en desarrollo: Moderado
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como cliente o emprendedora, quiero elegir entre retiro o entrega para definir cómo recibiré mi pedido.	
Escenario de prueba: • Dado que selecciono la opción "Retiro", cuando realizo un pedido, entonces se habilita la selección de punto de retiro y el pedido queda con deliveryMethod = pickup. • Dado que selecciono "Entrega", cuando confirmo mi dirección, entonces el pedido queda con deliveryMethod = delivery y se registra deliveryAddress y (si aplica) deliveryCost. • Dado que no selecciono ninguna opción, cuando intento confirmar, entonces se muestra advertencia para completarlo. • Dado un pedido ya creado, cuando lo visualizo en Pedidos (lista), entonces se muestra un badge "Retiro/Entrega" y en el detalle se muestran los datos correspondientes.	

Historia de Usuario	
Número: 17	Usuario: Administrador
Nombre historia: Gestión de pedidos por parte del administrador	
Prioridad en negocio: Muy Alta	Riesgo en desarrollo: Alto
Puntos estimados: 8	Iteración asignada:
Programador responsable:	
Descripción: Como administrador, quiero recibir avisos de nuevos pedidos y contar con una bandeja operativa de pedidos (lista + detalle) para dar seguimiento, cambiar estados y coordinar acciones del pedido.	
Escenario de prueba: • Dado un nuevo pedido, cuando ingreso a Admin > Pedidos, entonces lo veo en la lista con indicadores de estado operativo, método de entrega y estado de pago, y puedo abrir su detalle. • Dado que abro el pedido, cuando visualizo el detalle, entonces se muestra información del cliente, productos, entrega/retiro y comprobante (si existe). • Dado que el pedido fue revisado, cuando se cambia el estado (recibido/preparación/listo/entregado/cancelado), entonces se guarda updatedAt y queda registrado el nuevo estado para que el cliente lo vea reflejado en su sección Pedidos. • Dado que notificaciones push no estén implementadas aún, cuando ocurra un cambio de estado, entonces el sistema deja el evento listo para notificación (placeholder/log) sin bloquear el flujo operativo.	

Historia de Usuario	
Número: 19	Usuario: Cliente o Administrador
Nombre historia: Notificaciones push	
Prioridad en negocio: Muy Alta	Riesgo en desarrollo: Alto
Puntos estimados: 8	Iteración asignada:
Programador responsable:	
Descripción: Como cliente, emprendedora o administradora, quiero recibir notificaciones sobre mis pedidos o novedades para estar informado en tiempo real, especialmente ante cambios de estado, verificación de pago y agendamientos.	
Escenario de prueba: • Dado un cambio de estado en un pedido, cuando ocurre, entonces se prepara/envía una notificación al usuario correspondiente (si no hay push aún, se deja placeholder/log y no se bloquea el flujo). • Dado un nuevo evento o comunicado, cuando se publica, entonces se notifica a la audiencia correspondiente (cliente/emprendedora/admin).	

Historia de Usuario	
Número: 21	Usuario: Administrador
Nombre historia: Consulta de pedidos por fecha (administrador)	
Prioridad en negocio: Moderada	Riesgo en desarrollo: Bajo
Puntos estimados: 3	Iteración asignada:
Programador responsable:	
Descripción: Como administrador, quiero ingresar un rango de fecha para consultar los pedidos recibidos, su cantidad y detalles, desde Admin > Pedidos.	
Escenario de prueba: • Dado que ingreso una fecha de inicio y fin, cuando consulto, entonces se muestran todos los pedidos recibidos en ese rango, pudiendo abrir el detalle de cualquiera.	

Historia de Usuario	
Número: 16	Usuario: Cliente / Emprendedora
Nombre historia: Selección de punto de retiro	
Prioridad en negocio: Alta	Riesgo en desarrollo: Moderado
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como cliente o emprendedora, quiero ver dónde puedo retirar mi pedido para recogerlo en el lugar correcto.	
Escenario de prueba: • Dado un pedido con retiro, cuando selecciono punto, entonces se guarda la ubicación (pickupPointId y datos del punto). • Dado el acceso al historial o activos, cuando consulto un pedido, entonces veo la dirección y foto del punto de retiro (si existe), y si existe un mapsUrl/ubicación, entonces puedo abrir "Cómo llegar/Ver ruta".	

Historia de Usuario	
Número: 18	Usuario: Administrador
Nombre historia: Verificación de pago (admin)	
Prioridad en negocio: Alta	Riesgo en desarrollo: Moderado
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como administrador, quiero verificar el estado de pago de cada pedido para asegurar que solo se procesen los pedidos pagados.	
Escenario de prueba: • Dado un pedido con paymentStatus = "en_revision/pending_validation", cuando reviso el comprobante, entonces puedo marcarlo como "pagado" y registrar paidAt/paidBy. • Dado un pedido con paymentStatus = "pendiente_comprobante", cuando consulto, entonces el pedido aparece como pendiente/no procesado y puede mostrar indicador de vencimiento (solo visual) si aplica. • Dado que el comprobante es inválido (si se contempla), cuando lo rechazo, entonces el estado vuelve a pendiente_comprobante (y el comprobante no se borra automáticamente salvo regla de negocio explícita).	

Historia de Usuario	
Número: 20	Usuario: Clientes / Emprendedora
Nombre historia: Consulta de pedidos por fecha (cliente)	
Prioridad en negocio: Moderada	Riesgo en desarrollo: Bajo
Puntos estimados: 3	Iteración asignada:
Programador responsable:	
Descripción: Como cliente o emprendedora, quiero ver mis pedidos anteriores filtrados por fecha para consultar mis pedidos realizados en el pasado.	
Escenario de prueba: • Dado que accedo a Pedidos > Historial, cuando filtro por un período (p. ej., últimos 7/15/30 días, 3/6 meses) o un rango equivalente, entonces se muestran los pedidos realizados en ese periodo. • Dado que aplico filtros (período/estado como entregado o cancelado si aplica), cuando confirmo "Aplicar", entonces la lista se actualiza y puedo "Limpiar filtros" para volver al estado inicial.	

Historia de Usuario	
Número: 22	Usuario: Administrador
Nombre historia: Agendamiento de entregas	
Prioridad en negocio: Alta	Riesgo en desarrollo: Moderado
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como administrador, quiero asignar fecha (y datos asociados) para organizar la logística, y que la emprendedora/cliente lo visualice claramente en su app en una sección de pedidos programados.	
Escenario de prueba: • Dado un pedido confirmado, cuando agendo fecha (scheduledAt) y datos necesarios (punto/dirección/notas si aplica), entonces la emprendedora/cliente lo ve en su App en Pedidos > Programados con fecha/hora visible. • Dado un cambio en la programación, cuando lo actualizo, entonces se refleja en la App del usuario y queda listo para notificación (push si existe, placeholder/log si aún no).	

Historia de Usuario	
Número: 23	Usuario: Administrador
Nombre historia: Gestión de zonas de venta	
Prioridad en negocio: Alta	Riesgo en desarrollo: Moderado
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como administrador, quiero crear y asignar zonas de venta para cada emprendedora para organizar puntos de retiro y distribución.	
Escenario de prueba: • Dado que accedo al módulo de zonas, cuando creo una nueva zona con nombre, descripción y ubicación geográfica, entonces se guarda correctamente y queda disponible para asignación. • Dado que tengo varias emprendedoras registradas, cuando asigno una o varias a una zona específica, entonces sus perfiles muestran la zona correspondiente. • Dado que actualizo o elimino una zona, cuando guardo los cambios, entonces se reflejan automáticamente en los perfiles y en el mapa de puntos de venta.	

Historia de Usuario	
Número: 25	Usuario: Administradora
Nombre historia: Registro de acuerdos privados de ubicación	
Prioridad en negocio: Moderada	Riesgo en desarrollo: Moderado
Puntos estimados: 3	Iteración asignada:
Programador responsable:	
Descripción: Como administradora quiero registrar y actualizar acuerdos privados de ubicación (consentimientos de empresas para colocar un puesto en su ingreso) para mantener un mínimo respaldo interno del lugar y empresa asociada.	
Escenario de prueba: • Dado que creo un nuevo acuerdo con {empresa, punto de ubicación, tipo de acuerdo: verbal/escrito, persona de contacto, vigencia opcional}, Cuando guardo, Entonces el acuerdo queda visible en el panel de acuerdos con estado "activo". • Dado un acuerdo existente, Cuando adjunto evidencia opcional (foto del puesto o del acceso, PDF, nota interna), Entonces se guarda como anexo sin bloquear el flujo operativo. • Dado un acuerdo con fecha de vigencia (opcional), Cuando la fecha está vencida, Entonces el sistema lo marca como "vencido" y permite renovarlo o archivarlo. • Dado un acuerdo duplicado (misma empresa + mismo punto), Cuando intento crearlo, Entonces se alerta de posible duplicado y se sugiere actualizar el existente. • Dado que elimino o archivo un acuerdo, Cuando confirmo, Entonces el registro queda inactivo, conservando historial (quién, cuándo).	

Historia de Usuario	
Número: 27	Usuario: Cliente
Nombre historia: Ver perfiles de emprendedoras	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 3	Iteración asignada:
Programador responsable:	
Descripción: Como usuario, quiero conocer a las emprendedoras y ver su información de contacto para apoyarlas con más confianza y poder comunicarme con ellas fácilmente.	
Escenario de prueba: • Escenario 1: Visualización del perfil completo ○ Dado que accedo a la sección "Conócelas", Cuando selecciono una, Entonces veo su nombre, frase, foto, ubicación, y una lista de productos ofrecidos del proyecto o suyos. • Escenario 2: Funcionalidad de favoritos y persistencia ○ Dado que marco una como favorita, Cuando regreso, Entonces aparece resaltada en el mapa o lista. • Escenario 3: Interacción con la información de contacto ○ Dado que estoy viendo el perfil detallado de una emprendedora, Cuando visualizo la sección de contacto, Entonces puedo ver claramente su número de teléfono. ○ Y (Opcional, si se implementa) al tocar el número se abre el marcador del teléfono, o existe un botón para copiar el número al portapapeles, o un botón para abrir un chat de WhatsApp con ese número.	

Historia de Usuario	
Número: 24	Usuario: Emprendedora
Nombre historia: Solicitud de nueva emprendedora del proyecto	
Prioridad en negocio: Alta	Riesgo en desarrollo: Moderado
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como emprendedora quiero realizar una solicitud, cargando mis datos personales, documentos legales y fotos de mi punto de venta para ser parte del proyecto.	
Escenario de prueba: • Dada mi solicitud, tras el ingreso mis datos y documentos cuando presiono 'Registrar' entonces se guarda mi solicitud. • Dada mi solicitud, si falta un dato requerido cuando intento registrarme entonces se muestra error para completarlo.	

Historia de Usuario	
Número: 26	Usuario: Cliente
Nombre historia: Mapa de puntos de venta cercanos	
Prioridad en negocio: Muy Alta	Riesgo en desarrollo: Alto
Puntos estimados: 8	Iteración asignada:
Programador responsable:	
Descripción: Como usuario, Quiero visualizar en un mapa <i>referencial</i> los puntos de venta de las emprendedoras y explorar por <i>zona y tipo de producto</i> , Para ubicar de forma sencilla los puestos y conocer información básica de cada emprendedora desde la App.	
Escenario de prueba: 1. Cargar mapa y puntos • Dado que accedo a la sección "Mapa", Cuando se carga la vista, Entonces se muestra un mapa referencial con los pins de puntos de venta disponibles y un contador de resultados. 2. Filtrar por zona • Dado que selecciono una zona en el filtro, Cuando aplico el filtro, Entonces el mapa y el listado asociado muestran solo los puntos de venta correspondientes a esa zona. 3. Filtrar por tipo de producto • Dado que elijo un tipo de producto (p. ej., <i>Empanadas, Bebidas</i>), Cuando aplico el filtro, Entonces se actualizan los pins y el listado para incluir únicamente los puestos que ofrecen ese tipo. 4. Buscar por nombre o barrio (opcional si hay buscador) • Dado que ingreso un término de búsqueda (nombre de emprendedora o barrio), Cuando confirmo la búsqueda, Entonces se filtran los puntos a los que coinciden con el término. 5. Ver detalle resumido del punto • Dado que toco un pin en el mapa o un elemento del listado, Cuando se despliega la ficha resumida, Entonces visualizo nombre, foto, zona/barrio, tipo(s) de producto y horarios si están disponibles, con el CTA "Ver perfil/puesto". 6. Abrir perfil desde el mapa • Dado que estoy viendo la ficha resumida de un punto, Cuando toco "Ver perfil/puesto", Entonces se abre la pantalla del perfil de la emprendedora con su información ampliada. 7. Sin resultados • Dado que tengo filtros activos, Cuando no existe ningún punto que cumpla los criterios, Entonces se muestra un estado vacío con el mensaje "No encontramos puntos con esos criterios" y la opción "Limpiar filtros". 8. Recuperar estado entre vistas • Dado que regreso del perfil al mapa, Cuando vuelve a mostrarse la sección, Entonces se mantienen los filtros aplicados y el nivel de zoom/posición previos.	

Historia de Usuario	
Número: 28	Usuario: Emprendedora, Cliente o Visitante
Nombre historia: Testimonios o mensajes motivacionales	
Prioridad en negocio: Baja	Riesgo en desarrollo: Bajo
Puntos estimados: 2	Iteración asignada:
Programador responsable:	
Descripción: Como emprendedora, cliente o visitante que usa la app, quiero ver un banner interactivo con frases e historias breves, para conocer el impacto social del proyecto.	
Escenario de prueba: 1. Visualización inicial y contenido Dado que quiero ver un banner interactivo para conocer el impacto social, Cuando navego a la pantalla de inicio de la app, Entonces el banner se muestra de forma destacada y visualizo mensajes de impacto social que incluyen frases breves y la foto de una emprendedora. 2. Escenario 2: Navegación entre diferentes historias (Manual) Dado que quiero ver las diferentes emprendedoras y sus historias, Cuando deslizo el dedo (swipe) hacia la izquierda o toco las flechas de navegación, Entonces el banner avanza para mostrar la siguiente historia y los detalles de una emprendedora diferente. 3. Escenario 3: Ver detalles de una historia específica. Dado que quiero conectar y conocer más sobre una emprendedora específica, Cuando toco (tap) en la imagen o el nombre de la emprendedora que se muestra en el banner, Entonces se abre una nueva pantalla o una ventana modal (pop-up) que muestra la historia completa con más detalles sobre su proyecto e impacto.	

Historia de Usuario	
Número: 29	Usuario: Administrador
Nombre historia: Visualización de estadísticas (sin exportar)	
Prioridad en negocio: Moderada	Riesgo en desarrollo: Alto
Puntos estimados: 5	Iteración asignada:
Programador responsable:	
Descripción: Como administrador, quiero ver estadísticas por fechas y productos para analizar la actividad de la plataforma. Escenario de prueba: • Dado un rango de fechas seleccionado, cuando consulto, entonces veo la cantidad total de pedidos y tipos más vendidos. • Dado que filtro por zona, cuando visualizo, entonces veo los pedidos realizados por esa ubicación.	

Historia de Usuario	
Número: 30	Usuario: Emprendedora
Nombre historia: Registro de ubicación por la emprendedora	
Prioridad en negocio: Moderada	Riesgo en desarrollo: Alto
Puntos estimados: 8	Iteración asignada:
Programador responsable:	
Descripción: Como emprendedora autenticada, quiero registrar manualmente la dirección o punto donde vendo (barrio, parque, sector), junto con una foto y descripción breve, para que las personas puedan ubicarme más fácilmente desde la aplicación. Escenario de prueba: • Dado que estoy en el formulario de registro de ubicación, cuando ingreso manualmente el barrio, parque o sector donde vendo y adjunto una foto con descripción breve, entonces se guarda correctamente mi ubicación y se muestra una notificación de éxito. • Dado que dejen un campo obligatorio vacío (como la descripción o dirección), cuando intento guardar, entonces se muestra un mensaje de error indicando qué falta completar. • Dado que subo una imagen no válida (formato incorrecto o muy pesada), cuando confirmo el envío, entonces se muestra una alerta indicando que el archivo no es aceptado. • Dado que ya he registrado una ubicación antes, cuando intento registrar una nueva, entonces se me pregunta si deseo reemplazar la ubicación anterior. • Dado que actualizo mi punto de venta manualmente, cuando confirmo los nuevos datos, entonces el sistema reemplaza los datos anteriores y actualiza el mapa y perfil público.	

HISTORIAS DE USUARIO – APP WEB

Historia de Usuario	
Número: 1	Usuario: Administrador, Emprendedoras, Usuarios
Nombre historia: Menú Principal – Llamado a la Acción	
Prioridad en negocio: 100	Riesgo en desarrollo: ALTO
Puntos estimados: 5	Sprint: 3
Programador responsable: Leonardo Díaz	
Descripción: Como usuario visitante (cliente, emprendedora o administradora) Quiero visualizar la pantalla de inicio con botones de acción claros (Call to Action) Para identificar rápidamente cómo registrarme o iniciar sesión en el aplicativo web según el rol que corresponda. Escenario de prueba: Dado que ingreso a la URL del aplicativo Cuando carga la página de inicio Entonces visualizo el logo, una frase de bienvenida y los botones destacados de "Registrarse" e "Iniciar Sesión". Dado que el servidor no carga los estilos correctamente Cuando se visualiza la página Entonces los botones deben mantener su funcionalidad básica de enlace.	
Historia de Usuario	
Número: 3	Usuario: Emprendedoras
Nombre historia: Login de Emprendedoras	
Prioridad en negocio: 100	Riesgo en desarrollo: ALTO
Puntos estimados: 4	Sprint: 3
Programador responsable: Leonardo Díaz	
Descripción: Como Emprendedoras Quiero ingresar las credenciales Para acceder a la plataforma. Escenario de prueba: Dado el ingreso de credenciales Cuando presione el botón "Ingresar" Entonces me redirige a la interfaz de emprendedora. Dado el ingreso de credenciales erróneas Cuando presione el botón "Ingresar" Entonces se muestra un mensaje de advertencias indicando error de autenticación.	
Historia de Usuario	
Número: 4	Usuario: Administradora
Nombre historia: Login para Administradora	
Prioridad en negocio: 90	Riesgo en desarrollo: ALTA
Puntos estimados: 5	Sprint: 3
Programador responsable: Leonardo Díaz	
Descripción: Como Administradora Quiero ingresar las credenciales Para acceder a la plataforma. Escenario de prueba: Dado el ingreso de credenciales Cuando presione el botón "Ingresar" Entonces me redirige a la interfaz de administradora. Dado el ingreso de credenciales erróneas Cuando presione el botón "Ingresar" Entonces se muestra un mensaje de advertencias indicando error de autenticación.	
Historia de Usuario	
Número: 6	Usuario: Emprendedora
Nombre historia: Realizar Pedido	
Prioridad en negocio: 95	Riesgo en desarrollo: ALTO
Puntos estimados: 5	Sprint: 4
Programador responsable: Leonardo Díaz	
Descripción: Como emprendedora Quiero seleccionar productos del catálogo y confirmar mi pedido Para iniciar el proceso de compra de suministros desde el aplicativo web. Escenario de prueba: Dado que he agregado productos al carrito Cuando presiono "Confirmar Pedido" Entonces el sistema genera una orden con un ID único y cambia el estado a "Pendiente de Pago". Dado que intento confirmar un pedido con el carrito vacío Cuando presiono el botón de acción Entonces el sistema mantiene el botón deshabilitado o muestra un mensaje de "Carrito vacío". Dado que el pedido se registra correctamente Cuando el proceso finaliza Entonces el carrito de compras se limpia automáticamente y redirige a la sección de pagos.	
Historia de Usuario	
Número: 5	Usuario: Emprendedora
Nombre historia: Perfil de Emprendedoras	
Prioridad en negocio: 85	Riesgo en desarrollo: ALTA
Puntos estimados: 4	Sprint: 3
Programador responsable: Leonardo Díaz	
Descripción: Como emprendedora Quiero actualizar mis datos de contacto y mi foto de perfil Para mantener mi información actualizada ante la administración y los clientes. Escenario de prueba: Dado que modifico mi número de teléfono o nombre en el formulario Cuando presiono "Guardar Cambios" Entonces el sistema actualiza la base de datos y muestra los nuevos datos en pantalla. Dado que intento guardar cambios dejando el campo "Nombre" o "Correo" vacío Cuando presiono guardar Entonces el sistema muestra un mensaje de error indicando que son campos requeridos.	

Historia de Usuario		Historia de Usuario	
Número: 7	Usuario: Emprendedora	Número: 8	Usuario: Emprendedora
Nombre historia: Registro de pagos		Nombre historia: Historial de pedidos	
Prioridad en negocio: 95	Riesgo en desarrollo: ALTO	Prioridad en negocio: 90	Riesgo en desarrollo: ALTO
Puntos estimados: 4	Sprint: 4	Puntos estimados: 3	Sprint: 4
Programador responsable: Leonardo Díaz		Programador responsable: Leonardo Díaz	
Descripción: Como emprendedora Quiero adjuntar el comprobante de mi transferencia bancaria Para que la administradora valide mi pago y proceda con el despacho de mis productos.		Descripción: Como emprendedora Quiero visualizar el historial de mis pedidos realizados Para hacer seguimiento al estado de mis compras y mantener un control de mis registros.	
Escenario de prueba: Dado que seleccione un archivo de imagen válido (.jpg/.png) Cuando presiono "Enviar Pago" Entonces el sistema guarda el comprobante y cambia el estado del pedido a "En Validación". Dado que intento guardar el registro sin haber seleccionado ningún archivo Cuando presiono el botón de envío Entonces el sistema muestra un mensaje de error: "Debe adjuntar un comprobante". Dado que intento subir un archivo con formato no permitido (ej. .pdf o .txt) Cuando seleccione el archivo Entonces el sistema bloquea la acción y muestra una alerta de formato inválido.		Escenario de prueba: Dado que ingreso al módulo de "Mis Pedidos" Cuando la página carga Entonces visualizo una lista cronológica de mis órdenes con su ID, fecha, monto total y estado actual. Dado que hago clic en un pedido específico de la lista Cuando el sistema presiona la solicitud Entonces se despliega el detalle completo con los productos adquiridos y sus cantidades. Dado que el servidor de base de datos no responde Cuando intento cargar el historial Entonces se muestra una alerta indicando: "Error al obtener el historial, intente más tarde".	

Historia de Usuario		Historia de Usuario	
Número: 9	Usuario: Administrador	Número: 10	Usuario: Administrador
Nombre historia: Administrador Dashboard		Nombre historia: Admin Gestión Pedidos	
Prioridad en negocio: 90	Riesgo en desarrollo: MEDIO	Prioridad en negocio: 95	Riesgo en desarrollo: ALTO
Puntos estimados: 5	Sprint: 4	Puntos estimados: 5	Sprint: 4
Programador responsable: Leonardo Díaz		Programador responsable: Leonardo Díaz	
Descripción: Como Administradora Quiero visualizar métricas clave (ventas, pedidos y usuarios) Para tomar decisiones basadas en el rendimiento real del negocio.		Descripción: Como Administradora Quiero revisar los comprobantes de pago y actualizar el estado de las órdenes Para gestionar el flujo de ventas y asegurar que los pedidos pagados sean despachados.	
Escenario de prueba: Dado que el Dashboard carga Cuando el sistema consulta la base de datos Entonces visualizo tarjetas con el total de ventas del mes, pedidos pendientes y cantidad de emprendedoras activas. Dado que existen nuevos pedidos Cuando visualizo las gráficas de rendimiento Entonces estas reflejan los datos actualizados de forma visual y numérica. Dado que no hay actividad en el periodo seleccionado Cuando carga el Dashboard Entonces las métricas muestran el valor "0" o un mensaje de "Sin datos disponibles".		Escenario de prueba: Dado que visualizo un comprobante de pago válido Cuando presiono el botón "Aprobar Pago" Entonces el sistema actualiza el estado del pedido a "Aprobado" y se descuenta el stock del inventario. Dado que el comprobante adjunto es ilegible o incorrecto Cuando presiono el botón "Rechazar Pago" Entonces el sistema solicita un motivo de rechazo y regresa el pedido al estado "Pendiente de Pago". Dado que el servidor no responde al intentar cambiar un estado Cuando confirmo la acción Entonces el sistema muestra una alerta de error y mantiene el estado original del pedido.	

Historia de Usuario		Historia de Usuario	
Número: 11	Usuario: Administrador	Número: 12	Usuario: Administrador
Nombre historia: Admin Gestión Productos		Nombre historia: Visualización del catálogo sin autenticación	
Prioridad en negocio: 100	Riesgo en desarrollo: ALTO	Prioridad en negocio: 90	Riesgo en desarrollo: MEDIO
Puntos estimados: 4	Sprint: 4	Puntos estimados: 3	Sprint: 4
Programador responsable: Leonardo Díaz		Programador responsable: Leonardo Díaz	
Descripción: Como Administradora Quiero gestionar el stock y las categorías de los productos Para asegurar que la oferta en el aplicativo sea correcta y esté actualizada.		Descripción: Como Administradora Quiero gestionar los roles y el acceso de las emprendedoras Para garantizar la seguridad del sistema y el control de perfiles en el aplicativo web.	
Escenario de prueba: Dado que actualizo el stock de un producto a cero Cuando guardo los cambios Entonces el sistema le asigna automáticamente la etiqueta "Agotado" y deshabilita la compra. Dado que modifiqué la categoría de un producto Cuando el sistema guarda el registro Entonces el producto se muestra correctamente bajo el nuevo filtro en la vista de cliente. Dado que intento guardar una edición borrando el nombre o el precio Cuando confirmo Entonces el sistema bloquea la acción y marca los campos en rojo.		Escenario de prueba: Dado que detecto una cuenta con actividad inusual Cuando cambio su estado a "Suspendido" Entonces el sistema bloquea su inicio de sesión de manera inmediata. Dado que intento editar un perfil borrando el correo electrónico Cuando presiono guardar Entonces el sistema impide la acción y notifica que el campo es requerido. Dado que intento cambiar mi propio rol de "Admin" a "Emprendedora" Cuando el sistema valida Entonces muestra una advertencia para evitar que el sistema se quede sin administradores. Dado que utilice el filtro de búsqueda por correo electrónico, cuando ingreso el dato, entonces la tabla muestra únicamente la fila correspondiente al usuario consultado.	

Historia de técnica		Historia de técnica	
Título: Seguridad y privacidad de la aplicación móvil	Estimación: 8	Título: Funciones básicas y panel administrativo del aplicativo web	Estimación: 29
Historia relacionada: HT1-Seguridad y privacidad de la aplicación móvil		Historia relacionada: Sprint 4- Implementar funciones básicas del aplicativo web y administración	
Descripción: Se enfoca en implementar medidas de seguridad y privacidad en la aplicación móvil, protegiendo perfiles, permisos, archivos e interacciones con servicios remotos. Incluye reglas de acceso en Firestore.rules según autenticación, propiedad y rol, verificación de integridad con AppCheckInitializer antes del consumo de servicios; limitación de intentos en funciones sensibles con rateLimit(); validación de privilegios administrativos con requireAdmin(uid); carga segura de imágenes de perfil; separación entre datos públicos e internos; y verificación de cuenta por correo con periodo de gracia.		Descripción: Se enfoca en implementar las funciones básicas del aplicativo web para el flujo comercial y la administración del sistema. Incluye la selección de menú y el cálculo inmediato del total en RealizarPedido.jsx, registro y confirmación visual de pagos en RegistrarPago.jsx, consulta del historial de pedidos por usuario, panel administrativo con acceso restringido, gestión de pedidos con cambio de estados y validación de pagos, CRUD de productos y control de usuarios con búsquedas y filtros. Todo ello responde al objetivo funcional definido para el Sprint 4.	
Criterio de aceptación: • Proteger la actualización de perfiles y permisos con reglas de Firestore basadas en autenticación, propiedad y rol. • Verificar la integridad de la app con AppCheckInitializer antes de consumir servicios remotos. • Restringir funciones sensibles con rateLimit() para evitar abuso y saturación. • Validar el rol administrativo con requireAdmin(uid) antes de cambios de configuración. • Permitir uploadProfileImage solo a usuarios autenticados y con validación de formato y tamaño. • Reconstruir perfiles públicos con rebuildPublicEntrepreneur separando datos públicos de los internos. • Gestionar verificación de correo y periodo de gracia con EmailVerificationCubit.		Criterio de aceptación: • Permitir seleccionar productos y calcular el total al instante en RealizarPedido.jsx. • Mostrar carrito y tarjetas de productos con una interfaz consistente y usable. • Registrar pagos con ruta y fecha de envío en RegistrarPago.jsx y reflejar su estado visual. • Consultar historial de pedidos filtrado por userID y visualizar estados en tablas. • Restringir el AdminDashboard a accesos autorizados y ofrecer navegación base del módulo administrativo. • Gestionar pedidos desde el panel admin con cambios de estado y validación de pagos. • Administrar productos con operaciones CRUD de nombre, precio, stock y variedades. • Administrar usuarios mediante listado de emprendedoras, control de cuentas, buscador y filtros por zonas.	
Beneficio de justificación: Garantiza que la aplicación móvil proteja la información del usuario, reduzca el riesgo de abuso de servicios y mantenga separados los datos públicos de los internos. También fortalece la confianza en la plataforma al validar identidad, integridad del cliente y privilegios antes de ejecutar operaciones sensibles.		Beneficio de justificación: Consolida el flujo principal de compra y operación del aplicativo web en un solo sprint: comprar, registrar pagos, consultar pedidos y administrar la plataforma. Esto mejora la usabilidad para clientes y emprendedoras, y brinda al administrador herramientas directas para controlar pedidos, catálogo y cuentas.	
Impacto en el producto o proyecto: Sin estas medidas, perfiles y permisos podrían modificarse sin control, funciones críticas quedarían expuestas a abusos o cambios no autorizados, y la carga de archivos o la publicación de datos podría comprometer privacidad e integridad. Además, el sistema sería más vulnerable a automatizaciones maliciosas, cuentas no verificadas y accesos indebidos.		Impacto en el producto o proyecto: Sin estas funcionalidades, el aplicativo web no tendría un flujo operativo completo de pedido a pago ni capacidad de seguimiento y administración. La falta de paneles y módulos de gestión limitaría el control del negocio, retrasaría la atención de pedidos y dificultaría el mantenimiento de productos, usuarios y estados de pago.	
Dependencias: • Firebase Authentication para identidad y verificación por correo. • Cloud Firestore y archivo firestore.rules. • Firebase App Check / AppCheckInitializer. • Cloud Functions con utilidades rateLimit() y requireAdmin(uid). • Cloud Storage y flujo uploadProfileImage para imágenes de perfil. • Lógica rebuildPublicEntrepreneur y gestor EmailVerificationCubit en Flutter.		Dependencias: • React / componentes JSX: RealizarPedido.jsx, RegistrarPago.jsx, HistorialPedidos.jsx y AdminDashboard.jsx. • Módulos administrativos AdminGestionPedidos.jsx, AdminGestionProductos.jsx y AdminGestionUsuarios.jsx. • Archivo styles.js para estilos, tarjetas, tablas y layout administrativo. • Backend o API para envío de pedidos, registro y validación de pagos. • Base de datos con entidades de pedidos, pagos, productos, usuarios y zonas. • Sistema de autenticación y autorización para acceso administrativo. • Caso de prueba para flujo pedido → pago → validación.	
Tareas o sub tareas: • Definir reglas de acceso por autenticación, propiedad y rol en firestore.rules. • Implementar verificación de integridad con AppCheckInitializer. • Incorporar limitación de intentos mediante rateLimit(). • Validar rol administrativo con requireAdmin(uid). • Asegurar carga protegida de imágenes con uploadProfileImage. • Construir perfiles públicos controlados con rebuildPublicEntrepreneur. • Gestionar verificación de cuenta por correo y periodo de gracia con EmailVerificationCubit.		Tareas o sub tareas: • [Desarrollo] Seleccionar menú mediante cálculo al instante del total (RealizarPedido.jsx). • [Diseño] Diseñar carrito de compras y tarjetas de productos (styles.js y RealizarPedido.jsx). • [Pruebas] Ejecutar pruebas de envío y verificación del backend. • [Desarrollo] Ingresar el pago y establecer ruta y fecha del envío (RegistrarPago.jsx). • [Diseño] Corroborar visualmente el pago, su estado y estilos (styles.js). • [Desarrollo] Listar y consultar pedidos mediante filtro userID (HistorialPedidos.jsx). • [Diseño] Construir tablas de pedidos con sus estados. • [Desarrollo] Configurar panel principal con acceso limitado y navegación parcial (AdminDashboard.jsx). • [Diseño] Definir layout administrativo con temática oscura y sidebar. • [Desarrollo] Crear sistema de cambio de estados de pedido y validación de pagos (AdminGestionPedidos.jsx). • [Pruebas] Simular el flujo desde el pedido hasta la aceptación del pago. • [Desarrollo] Implementar CRUD de nombre, precio y stock (AdminGestionProductos.jsx). • [Diseño] Permitir edición de productos y añadidura de variedades.	

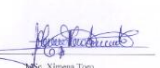
Anexo 5. Pruebas de aceptación

Pontificia Universidad Católica del Ecuador		PUCE					
CASO DE PRUEBA 01 Fecha: 03/02/2026							
Nombre caso de prueba: Pantalla de Bienvenida (acceso inicial)		Sprint: 1					
Módulo/sección a evaluar: Inicio / Bienvenida		Historia de usuario asociada: 1					
Técnica de prueba: Caja Negra ☑ Caja Blanca ☐		Tipo: Prueba de Aceptación					
Descripción: - Escenario 1 (Éxito): Dado que abro la App, cuando carga la pantalla de bienvenida, entonces veo el logo, nombre, frase breve y tres botones: Registrarse, Iniciar sesión, Continuar como visitante. - Escenario 2 (Éxito): Dado que toco Registrarse, cuando navego, entonces se abre la pantalla de selección de rol. - Escenario 3 (Éxito): Dado que toco Iniciar sesión, cuando navego, entonces se abre la pantalla de Login. - Escenario 4 (Éxito): Dado que toco Continuar como visitante, cuando navego, entonces entro a la pantalla principal con acceso limitado (si aplica). - Escenario 5 (Negativo - campos obligatorios): Dado que estoy en el flujo de Pantalla de Bienvenida, cuando dejo campos obligatorios vacíos o incompletos e intento continuar, entonces se muestra un mensaje de validación y no se completa la acción. - Escenario 6 (Negativo - datos inválidos/duplicados/permisos): Dado que estoy en el flujo de Pantalla de Bienvenida, cuando ingreso datos inválidos/incorrectos o no tengo permisos según mi rol e intento continuar, entonces se muestra un mensaje de error y no se completa la acción.							
2) Pre-condiciones - Tener acceso a Internet. - Contar con la aplicación instalada y operativa. - No tener una sesión activa iniciada. - Contar con la cuenta requerida (según la IU).							
Pasos y condiciones de ejecución - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Pantalla de Bienvenida. - Ejecutar el/los escenario(s) según corresponda. - Verificar el resultado mostrado por el sistema.							
Resultado esperado - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba		<table border="1"> <thead> <tr> <th>Éxito</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>		Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
MS. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
CASO DE PRUEBA 02 Fecha: 03/02/2026							
Nombre caso de prueba: Configuración de accesibilidad y preferencias de visualización		Sprint: 1					
Módulo/sección a evaluar: Configuración de accesibilidad y preferencias de visualización		Historia de usuario asociada: 2					
Técnica de prueba: Caja Negra ☑ Caja Blanca ☐		Tipo: Prueba de Aceptación					
Descripción: Como usuario de la aplicación quiero poder ajustar el tamaño de texto y el tema de color (claro, oscuro o automático) para mejorar la legibilidad y usar la app de forma cómoda según mis necesidades. - Escenario 1 (Éxito): Dado que accedo a la pantalla de configuración de accesibilidad, Cuando deslizo el control de tamaño de texto, Entonces la vista previa y el contenido de la app se ajustan dinámicamente. - Escenario 2 (Éxito): Dado que elijo un modo de tema (claro, oscuro o automático), Cuando guardo mi elección, Entonces la aplicación se muestra con el esquema de colores correspondiente y mantiene contraste legible. - Escenario 3 (Éxito): Dado que estoy en cualquier pantalla de la app, Cuando presiono el ícono de configuración, Entonces puedo acceder rápidamente al diálogo de Accesibilidad sin salir del flujo principal. - Escenario 4 (Éxito): Dado que cierro sesión y vuelvo a entrar, Cuando la app inicia, Entonces mantiene la preferencia guardada para tamaño de texto y tema.							
Pre-condiciones - Tener acceso a Internet. - Contar con la aplicación instalada y operativa.							
Pasos y condiciones de ejecución - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Configuración de accesibilidad y preferencias de visualización. - Ejecutar el/los escenario(s) según corresponda. - Verificar el resultado mostrado por el sistema.							
Resultado esperado - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba		<table border="1"> <thead> <tr> <th>Éxito</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>		Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
MS. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
CASO DE PRUEBA 03 Fecha: 03/02/2026							
Nombre caso de prueba: Selección de Rol		Sprint: 1					
Módulo/sección a evaluar: Selección de Rol		Historia de usuario asociada: 3					
Técnica de prueba: Caja Negra ☑ Caja Blanca ☐		Tipo: Prueba de Aceptación					
Descripción: Como usuario (Cliente o Emprendedor), quiero poder elegir si me registro como cliente o como emprendedor para que la app se configure correctamente según mi rol. - Escenario 1 (Éxito): Dado que llego a la pantalla de selección de rol, cuando elijo Cliente, entonces me lleva al formulario de registro de cliente. - Escenario 2 (Fallo): Dado que elijo Emprendedor, entonces me lleva al formulario de registro de emprendedor. - Escenario 3 (Negativo - campos obligatorios): Si no selecciono nada y presiono continuar, entonces se muestra un mensaje de validación. - Escenario 4 (Negativo - datos inválidos/duplicados/permisos): Dado que estoy en el flujo de Selección de Rol, cuando ingreso datos inválidos/incorrectos o no tengo permisos según mi rol e intento continuar, entonces se muestra un mensaje de error y no se completa la acción.							
Pre-condiciones - Tener acceso a Internet. - Contar con la aplicación instalada y operativa. - Contar con la cuenta requerida (según la IU).							
Pasos y condiciones de ejecución - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Selección de Rol. - Ejecutar el/los escenario(s) según corresponda. - Verificar el resultado mostrado por el sistema.							
Resultado esperado - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba		<table border="1"> <thead> <tr> <th>Éxito</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>		Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
MS. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
CASO DE PRUEBA 04 Fecha: 03/02/2026							
Nombre caso de prueba: Registro de Usuario		Sprint: 1					
Módulo/sección a evaluar: Registro de Usuario		Historia de usuario asociada: 4					
Técnica de prueba: Caja Negra ☑ Caja Blanca ☐		Tipo: Prueba de Aceptación					
Descripción: Como cliente o emprendedor Quiero registrarme con mis datos personales Para poder gestionar mi emprendimiento desde la aplicación. - Escenario 1 (Fallo): Dado el registro de mis datos (nombre, apellidos, dirección, teléfono, cédula o RUC, contraseña) Cuando presiono 'Registrarme' Entonces se muestra un mensaje y mi cuenta se guarda correctamente. - Escenario 2 (Negativo - campos obligatorios): Dado que los datos son incompletos o erróneos Cuando intento registrarme, Entonces se muestran los errores para corregirlos. - Escenario 3 (Negativo - datos inválidos/duplicados/permisos): Dado que estoy en el flujo de Registro de Usuario, cuando ingreso datos inválidos/incorrectos o no tengo permisos según mi rol e intento continuar, entonces se muestra un mensaje de error y no se completa la acción.							
Pre-condiciones - Tener acceso a Internet. - Contar con la aplicación instalada y operativa. - Contar con la cuenta requerida (según la IU).							
Pasos y condiciones de ejecución - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Registro de Usuario. - Ejecutar el/los escenario(s) según corresponda. - Verificar el resultado mostrado por el sistema.							
Resultado esperado - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba		<table border="1"> <thead> <tr> <th>Éxito</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>		Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
MS. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE	
CASA DE PRUEBA 05		Fecha: 03/02/2026	
Nombre caso de prueba: Ingreso de Usuario		Sprint: 1	
Módulo/sección a evaluar: Ingreso de Usuario		Historia de usuario asociada: 5	
Técnica de prueba: Caja Negra ☑ Caja Blanca ☐		Tipo: Prueba de Aceptación	
Descripción: Como cliente o emprendedor Quiero iniciar sesión con mi número de cédula y contraseña Para acceder a mi cuenta y realizar pedidos.			
- Escenario 1 (Éxito): Dado que ingreso mis credenciales correctas Cuando presiono "Iniciar sesión", Entonces acceso al sistema. - Escenario 2 (Negativo – datos inválidos/duplicados/permisos): Dado que ingreso datos incorrectos Cuando presiono "Iniciar sesión" Entonces se muestra un mensaje de error. - Escenario 3 (Negativo – campos obligatorios): Dado que estoy en el flujo de Ingreso de Usuario, cuando dejo campos obligatorios vacíos o incompletos e intento continuar, entonces se muestra un mensaje de validación y no se completa la acción.			
Pre-condiciones - Tener acceso a internet. - Contar con la aplicación instalada y operativa. - Contar con la cuenta rol requerido (según la HU).			
Pasos y condiciones de ejecución - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Ingreso de Usuario. - Ejecutar el/los escenario(s) según corresponda. - Verificar el resultado mostrado por el sistema.			
Resultado esperado - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.			
Estado de prueba		Exito	Fallo
Errores asociados:		SI	No
 MSc. Ximena Toro PRODUCT OWNER			

Pontificia Universidad Católica del Ecuador		PUCE	
CASA DE PRUEBA 06		Fecha: 03/02/2026	
Nombre caso de prueba: Selección de Acceso para Administradores		Sprint: 1	
Módulo/sección a evaluar: Selección de Acceso para Administradores		Historia de usuario asociada: 6	
Técnica de prueba: Caja Negra ☑ Caja Blanca ☐		Tipo: Prueba de Aceptación	
Descripción: Como puede administrador, quiero solicitar acceso especial al sistema mediante un formulario de verificación, para que la empresa pueda validar mis credenciales y concederme permisos administrativos si corresponde.			
- Escenario 1 (Éxito): Dado que accedo a la opción "Solicitar acceso especial" desde la pantalla de inicio o login, cuando selecciono esa opción, entonces se me solicita ingresar el código de verificación para administradores y formulario especial para solicitar acceso administrativo. - Escenario 2 (Éxito): Dado que ingreso de un código válido, cuando presiono "Verificar" entonces se muestra un formulario para arcar la cuenta administradores. - Escenario 3 (Negativo – datos inválidos/duplicados/permisos): Dado que ya ingresé un código inválido cuando presiono "Verificar" entonces se muestra un mensaje indicando lo mismo. - Escenario 4 (Éxito): Dado que completo el formulario con mis datos personales (nombre, edad, RUC, cargo, teléfono) y documento oportuno, cuando presiono "Enviar solicitud", entonces el sistema genera la solicitud y muestra un mensaje de confirmación. - Escenario 5 (Negativo – campos obligatorios): Dado que dejo campos obligatorios vacíos o ingreso datos inválidos, cuando intento enviar el formulario, entonces se muestra un mensaje indicando que campos deben completarse. - Escenario 6 (Negativo – datos inválidos/duplicados/permisos): Dado que ya envié una solicitud anteriormente, cuando vuelvo a ingresar, entonces el sistema me informa que mi solicitud está en revisión o ya ha sido aprobada/rechazada. - Escenario 7 (Éxito): Dado que mi solicitud fue aprobada por un administrador interno, cuando intento ingresar al sistema con mi cuenta registrada, entonces tengo acceso al panel administrativo en lugar del panel común.			
Pre-condiciones - Tener acceso a internet. - Contar con la aplicación instalada y operativa. - Contar con la cuenta rol requerido (según la HU).			
Pasos y condiciones de ejecución - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Selección de Acceso para Administradores. - Ejecutar el/los escenario(s) según corresponda. - Verificar el resultado mostrado por el sistema.			
Resultado esperado - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.			
Estado de prueba		Exito	Fallo
Errores asociados:		SI	No
 MSc. Ximena Toro PRODUCT OWNER			

Pontificia Universidad Católica del Ecuador		PUCE	
CASA DE PRUEBA 07		Fecha: 03/02/2026	
Nombre caso de prueba: Dashboard Principal y Navegación por Rol		Sprint: 1	
Módulo/sección a evaluar: Dashboard Principal y Navegación por Rol		Historia de usuario asociada: 7	
Técnica de prueba: Caja Negra ☑ Caja Blanca ☐		Tipo: Prueba de Aceptación	
Descripción: Como usuario de la App (Cliente, Emprendedor o Admin), quiero acceder a un Dashboard Principal que muestre de forma clara las opciones comunes y principales según mi rol, para orientarme fácilmente, visualizar lo más relevante para mí y acceder solo a las funciones que me corresponden.			
- Escenario 1 (Éxito): Dado que accedo después de iniciar sesión o registrarme, cuando se carga el Dashboard, entonces visualizo el logo, mi perfil, y un conjunto de opciones ordenadas según mi rol (por ejemplo: "Hacer pedido" si soy cliente, "Ver pedidos" si soy admin). - Escenario 2 (Éxito): Dado que navego por las secciones desde el Dashboard, cuando selecciono una opción (como "Mis" o "Pedidos"), entonces se me redirige correctamente a la vista correspondiente, manteniendo el diseño coherente y accesible. - Escenario 3 (Éxito): Dado que soy un usuario con permisos específicos (ej. administrador), cuando ingreso, entonces no puedo acceder a módulos que no corresponden a mi rol (por ejemplo: realizar pedidos si soy solo administrador).			
Pre-condiciones - Tener acceso a internet. - Contar con la aplicación instalada y operativa. - Contar con la cuenta rol requerido (según la HU).			
Pasos y condiciones de ejecución - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Dashboard Principal y Navegación por Rol. - Ejecutar el/los escenario(s) según corresponda. - Verificar el resultado mostrado por el sistema.			
Resultado esperado - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.			
Estado de prueba		Exito	Fallo
Errores asociados:		SI	No
 MSc. Ximena Toro PRODUCT OWNER			

Pontificia Universidad Católica del Ecuador		PUCE	
CASA DE PRUEBA 08		Fecha: 03/02/2026	
Nombre caso de prueba: Visualización del Perfil de Usuario		Sprint: 1	
Módulo/sección a evaluar: Visualización del Perfil de Usuario		Historia de usuario asociada: 8	
Técnica de prueba: Caja Negra ☑ Caja Blanca ☐		Tipo: Prueba de Aceptación	
Descripción: Como usuario autorizado, quiero ver mis datos personales registrados, para verificar que mi información esté correcta y actualizada.			
- Escenario 1 (Éxito): Dado que visualizo mis datos personales registrados, cuando accedo a la sección de perfil desde el dashboard, entonces se muestra mi nombre, cédula, rol, número de teléfono y foto de perfil si está disponible. - Escenario 2 (Éxito): Dado que visualizo si mi perfil está completo, cuando alguno de los datos está ausente (como: foto o dirección), entonces se muestra un mensaje o icono que sugiere completarlo. - Escenario 3 (Negativo – campos obligatorios): Dado que estoy en el flujo de Visualización del Perfil de Usuario, cuando dejo campos obligatorios vacíos o incompletos e intento continuar, entonces se muestra un mensaje de validación y no se completa la acción. - Escenario 4 (Negativo – datos inválidos/duplicados/permisos): Dado que estoy en el flujo de Visualización del Perfil de Usuario, cuando ingreso datos inválidos/incorrectos o no tengo permisos según mi rol e intento continuar, entonces se muestra un mensaje de error y no se completa la acción.			
Pre-condiciones - Tener acceso a internet. - Contar con la aplicación instalada y operativa. - Contar con la cuenta rol requerido (según la HU).			
Pasos y condiciones de ejecución - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Visualización del Perfil de Usuario. - Ejecutar el/los escenario(s) según corresponda. - Verificar el resultado mostrado por el sistema.			
Resultado esperado - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.			
Estado de prueba		Exito	Fallo
Errores asociados:		SI	No
 MSc. Ximena Toro PRODUCT OWNER			

Pontificia Universidad Católica del Ecuador		PUCE	
CASA DE PRUEBA 09		Fecha: 03/02/2026	
Nombre caso de prueba: Cerrar Sesión del Usuario		Sprint: 1	
Módulo/sección a evaluar: Cerrar Sesión del Usuario		Historia de usuario asociada: 9	
Técnica de prueba: Caja Negra ☑ Caja Blanca ☐		Tipo: Prueba de Aceptación	
Descripción: Como usuario autorizado (Cliente, Emprendedor o Administrador), Quiero poder cerrar sesión de forma clara y segura desde la sección de Perfil o Ajustes de Cuenta, Para proteger mis datos personales y evitar accesos no autorizados desde dispositivos compartidos.			
- Escenario 1 (Éxito): Dado que estoy autenticado dentro de la app, Cuando ingreso a mi menú de Perfil o Ajustes de Cuenta, Entonces puedo ver la opción "Cerrar sesión" claramente disponible. - Escenario 2 (Éxito): Dado que decido cerrar sesión, Cuando toco el botón "Cerrar sesión", Entonces se muestra un mensaje de confirmación para validar mi decisión ("¿Estás seguro de que quieres cerrar sesión?"). - Escenario 3 (Negativo – datos inválidos/duplicados/permisos): Dado que confirmo cerrar sesión, Cuando acepto la acción, Entonces la App invalida mi sesión actual, elimina mis credenciales locales y me redirige a la Pantalla de Bienvenida o Login. - Escenario 4 (Éxito): Dado que he cerrado sesión correctamente, Cuando intento volver a la pantalla anterior usando el botón Atrás del dispositivo, Entonces la App NO debe permitirme regresar a la sesión cerrada.			
Pre-condiciones - Tener acceso a internet. - Contar con la aplicación instalada y operativa. - Contar con la cuenta rol requerido (según la HU).			
Pasos y condiciones de ejecución - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Cerrar Sesión del Usuario. - Ejecutar el/los escenario(s) según corresponda. - Verificar el resultado mostrado por el sistema.			
Resultado esperado - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.			
Estado de prueba		Exito	Fallo
Errores asociados:		SI	No
 MSc. Ximena Toro PRODUCT OWNER			

Pontificia Universidad Católica del Ecuador		PUCE	
CASA DE PRUEBA 10		Fecha: 03/02/2026	
Nombre caso de prueba: Recuperación de Contraseña		Sprint: 1	
Módulo/sección a evaluar: Recuperación de Contraseña		Historia de usuario asociada: 10	
Técnica de prueba: Caja Negra ☑ Caja Blanca ☐		Tipo: Prueba de Aceptación	
Descripción: Como usuario (Cliente, Emprendedor o Admin), quiero recuperar mi contraseña si la olvidé, para poder volver a ingresar a la App sin complicaciones y sin depender de soporte técnico.			
- Escenario 1 (Éxito): Dado que olvidé mi contraseña y quiero recuperarla, cuando ingreso a la opción "¿Olvidaste tu contraseña?" en la pantalla de login, entonces se me solicita seleccionar e ingresar mi correo electrónico o cédula para encontrar la cuenta. - Escenario 2 (Éxito): Dado el intento de recuperación cuando busco mi cuenta con mi documento (Cédula, RUC, o Pasaporte) o correo ingresado entonces se me solicita verificar mediante un medio que permita recuperar la cuenta y mostrar los campos de cambio de contraseña. - Escenario 3 (Éxito): Dado que he verificado mi intento de recuperación, cuando ingreso una nueva contraseña válida y confirmo, entonces se actualiza la contraseña y puedo iniciar sesión nuevamente entonces se solicita ingresar una nueva contraseña válida.			
Pre-condiciones - Tener acceso a internet. - Contar con la aplicación instalada y operativa.			
Pasos y condiciones de ejecución - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Recuperación de Contraseña. - Ejecutar el/los escenario(s) según corresponda. - Verificar el resultado mostrado por el sistema.			
Resultado esperado - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.			
Estado de prueba		Exito	Fallo
Errores asociados:		SI	No
 MSc. Ximena Toro PRODUCT OWNER			

Pontificia Universidad Católica del Ecuador
Secretaría de Tecnología
Escuela de Ingeniería de Sistemas y Computación

PUCE

CASO DE PRUEBA 11 Fecha: 03/02/2026

Nombre caso de prueba: Edición de Perfil de Usuario **Sprint:** 1

Metodología a evaluar: Edición de Perfil de Usuario **Historia de usuario asociada:** 11

Técnica de prueba: Caja Negra ☑ Caja Blanca ☐ **Tipo:** Prueba de Aceptación

Descripción:
Como usuario autenticado (Cliente, Empleado o Administrador), quiero editar mis datos de perfil (nombre, teléfono, foto de perfil, correo, entre otros) para mantener mi información actualizada y correcta dentro de la app.

- Escenario 1 (Éxito): Dado que quiero editar mis datos de perfil, cuando ingreso a la sección "perfil" y presiono "editar", entonces se muestran los campos modificables con su contenido actual.
- Escenario 2 (Éxito): Dado que modifico uno o más datos del perfil, cuando presiono "Guardar cambios", entonces los datos se actualizan correctamente y se refleja la nueva información.
- Escenario 3 (Negativo - campos obligatorios): Dado que intento guardar el perfil con datos incompletos o inválidos, cuando confirmo, entonces se muestra un mensaje de error indicando qué debe corregirse.
- Escenario 4 (Negativo - datos inválidos/aplicados/permisos): Dado que estoy en el flujo de Edición de Perfil de Usuario, cuando ingreso datos inválidos/incorrectos o no tengo permisos según mi rol e intento continuar, entonces se muestra un mensaje de error y no se completa la acción.

Pre-condiciones

- Tener acceso a Internet.
- Contar con la aplicación instalada y operativa.
- Contar con la cuenta/rol requerido (según la IU).

Pasos y condiciones de ejecución

- Abrir la aplicación.
- Ingresar al módulo o flujo relacionado con Edición de Perfil de Usuario.
- Ejecutar el/los escenario(s) según corresponda.
- Verificar el resultado mostrado por el sistema.

Resultado esperado

- El sistema cumple el comportamiento definido en cada escenario.
- Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.

Estado de prueba

	Éxito	Fallo
Errores asociados:	SI	No

[Firma]
MSc. Ximena Toro
PRODUCT OWNER

Pontificia Universidad Católica del Ecuador
Secretaría de Tecnología
Escuela de Ingeniería de Sistemas y Computación

PUCE

CASO DE PRUEBA 12 Fecha: 03/02/2026

Nombre caso de prueba: Gestión de productos del catálogo **Sprint:** 1

Metodología a evaluar: Gestión de productos del catálogo **Historia de usuario asociada:** 12

Técnica de prueba: Caja Negra ☑ Caja Blanca ☐ **Tipo:** Prueba de Aceptación

Descripción:
Como administradores, quiero crear y editar productos del catálogo control, asignados a una categoría y definiendo su presentación (cantidad, cantidad y precio base), para ampliar y mantener actualizado la oferta de productos que se muestra a los clientes y emprendedores en la app.

- Escenario 1 (Éxito): Dado que estoy en el módulo de catálogo, cuando presiono el botón "Nuevo producto", entonces se abre un formulario donde puedo seleccionar una categoría, ingresar el nombre del producto, tipo de unidad, cantidad por unidad, precio base, canal (vólef/coche) y, opcionalmente, variantes (colores, tipos de masa e imagen).
- Escenario 2 (Negativo - campos obligatorios): Dado que completo los campos obligatorios, cuando presiono "Guardar producto", entonces el producto se guarda correctamente en el catálogo control y aparece en la lista de productos de la categoría seleccionada.
- Escenario 3 (Negativo - campos obligatorios): Dado que intento guardar el producto con campos obligatorios vacíos o datos inválidos (por ejemplo, precio en blanco o menor a igual a cero), cuando presiono "Guardar producto", entonces se muestra un mensaje de error justo a los campos correspondientes indicando qué debe corregirse.
- Escenario 4 (Éxito): Dado que un producto ya existe en el catálogo, cuando lo selecciono y abro la opción "editar", entonces puedo modificar su nombre, presentación, precio base, variantes y estado (vólef/coche).
- Escenario 5 (Éxito): Dado que quiero un producto como "Oleón", cuando guardo los cambios, entonces el producto deja de mostrarse en los resultados de pedidos y en las listas públicas, pero se mantiene internamente.
- Escenario 6 (Negativo - datos inválidos/aplicados/permisos): Dado que estoy en el flujo de Gestión de productos del catálogo (presentaciones y variantes), cuando ingreso datos inválidos/incorrectos o no tengo permisos según mi rol e intento continuar, entonces se muestra un mensaje de error y no se completa la acción.

Pre-condiciones

- Tener acceso a Internet.
- Contar con la aplicación instalada y operativa.
- Contar con la cuenta/rol requerido (según la IU).

Pasos y condiciones de ejecución

- Abrir la aplicación.
- Ingresar al módulo o flujo relacionado con: Gestión de productos del catálogo (presentaciones y variantes).
- Ejecutar el/los escenario(s) según corresponda.
- Verificar el resultado mostrado por el sistema.

Resultado esperado

- El sistema cumple el comportamiento definido en cada escenario.
- Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.

Estado de prueba

	Éxito	Fallo
Errores asociados:	SI	No

[Firma]
MSc. Ximena Toro
PRODUCT OWNER

Pontificia Universidad Católica del Ecuador
Secretaría de Tecnología
Escuela de Ingeniería de Sistemas y Computación

PUCE

CASO DE PRUEBA 13 Fecha: 03/02/2026

Nombre caso de prueba: Realizar pedido de bandejas productivas **Sprint:** 1

Metodología a evaluar: Realizar pedido de bandejas productivas **Historia de usuario asociada:** 13

Técnica de prueba: Caja Negra ☑ Caja Blanca ☐ **Tipo:** Prueba de Aceptación

Descripción:
Como emprendedor/Cliente quiero visualizar y seleccionar tipo y cantidad de bandejas de la oferta de productos para realizar un pedido desde mi cuenta.

- Escenario 1 (Éxito): Dado que selecciono tipo y cantidad de bandejas y completo los campos requeridos, cuando confirmo el pedido, entonces se genera el pedido con estado "Pendiente" y se muestra confirmación (para el administrador) con número ID y estado.
- Escenario 2 (Negativo - datos inválidos/aplicados/permisos): Dado que el pedido fue creado correctamente, cuando navego a Pedido > Activos, entonces el pedido aparece en la lista y, si es el más reciente, se muestra de forma detallada como "pedido en curso" (sólo visible en la IU).
- Escenario 3 (Negativo - campos obligatorios): Dado que el/los "Pedido" y selecciono un punto de retiro, además ingreso nombre y teléfono, cuando confirmo, entonces el sistema registra `pedidoPorRetiroPedidoPorRetiro` en `requestPhone` e `requestPhone` como obligatorio en el pedido.
- Escenario 4 (Negativo - campos obligatorios): Dado que el/los "Pedido" e ingreso dirección, nombre y teléfono, cuando confirmo, entonces el sistema registra `pedidoPorRetiroPedidoPorRetiro` e `requestPhone` como obligatorio en el pedido.
- Escenario 5 (Negativo - campos obligatorios): Dado que falta al menos un campo requerido, cuando intento confirmar, entonces el sistema no permite confirmar y muestra validaciones en los campos.
- Escenario 6 (Éxito): Dado que selecciono más de 5 bandejas, cuando confirmo, entonces el precio unitario cambia a \$1.00 y el total se recalcula automáticamente.

Pre-condiciones

- Tener acceso a Internet.
- Contar con la aplicación instalada y operativa.
- Contar con la cuenta/rol requerido (según la IU).

Pasos y condiciones de ejecución

- Abrir la aplicación.
- Ingresar al módulo o flujo relacionado con: Realizar pedido de bandejas productivas.
- Ejecutar el/los escenario(s) según corresponda.
- Verificar el resultado mostrado por el sistema.

Resultado esperado

- El sistema cumple el comportamiento definido en cada escenario.
- Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.

Estado de prueba

	Éxito	Fallo
Errores asociados:	SI	No

[Firma]
MSc. Ximena Toro
PRODUCT OWNER

Pontificia Universidad Católica del Ecuador
Secretaría de Tecnología
Escuela de Ingeniería de Sistemas y Computación

PUCE

CASO DE PRUEBA 14 Fecha: 03/02/2026

Nombre caso de prueba: Adjuntar comprobante de pago **Sprint:** 1

Metodología a evaluar: Adjuntar comprobante de pago **Historia de usuario asociada:** 14

Técnica de prueba: Caja Negra ☑ Caja Blanca ☐ **Tipo:** Prueba de Aceptación

Descripción:
Como cliente o emprendedor, quiero subir el comprobante de pago para confirmar que ya realicé el pago del pedido.

- Escenario 1 (Éxito): Dado que el pedido fue creado, cuando subo el comprobante dentro de 1 hora, entonces el pedido queda con `paymentStatus = "en_revisión"` ("pending_validation" según el modelo) y `proofUrl` asociado, y en Pedidos > Activos se refleja visualmente el estado (chip badge).
- Escenario 2 (Negativo - datos inválidos/aplicados/permisos): Dado que no se sube el comprobante en el tiempo indicado, cuando se consulta el estado en Pedidos, entonces el pedido aparece como pendiente de comprobante y no procesado (sin cambiar automáticamente a "pagado"), y para administración se identifica como pendiente (con indicador de vencimiento si aplica).
- Escenario 3 (Éxito): Dado que ya existe un comprobante asociado al pedido, cuando ingreso al detalle del pedido, entonces puedo visualizar el comprobante (thumbail y view full screen) y su estado de revisión.
- Escenario 4 (Negativo - campos obligatorios): Dado que estoy en el flujo de Adjuntar comprobante de pago, cuando elijo campos obligatorios vacíos o incompletos e intento continuar, entonces se muestra un mensaje de validación y no se completa la acción.

Pre-condiciones

- Tener acceso a Internet.
- Contar con la aplicación instalada y operativa.
- Contar con la cuenta/rol requerido (según la IU).

Pasos y condiciones de ejecución

- Abrir la aplicación.
- Ingresar al módulo o flujo relacionado con: Adjuntar comprobante de pago.
- Ejecutar el/los escenario(s) según corresponda.
- Verificar el resultado mostrado por el sistema.

Resultado esperado

- El sistema cumple el comportamiento definido en cada escenario.
- Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.

Estado de prueba

	Éxito	Fallo
Errores asociados:	SI	No

[Firma]
MSc. Ximena Toro
PRODUCT OWNER

Pontificia Universidad Católica del Ecuador
Secretaría de Tecnología
Escuela de Ingeniería de Sistemas y Computación

PUCE

CASO DE PRUEBA 15 Fecha: 03/02/2026

Nombre caso de prueba: Ampliación del pedido: tipos de entrega **Sprint:** 1

Metodología a evaluar: Ampliación del pedido: tipos de entrega **Historia de usuario asociada:** 15

Técnica de prueba: Caja Negra ☑ Caja Blanca ☐ **Tipo:** Prueba de Aceptación

Descripción:
Como cliente o emprendedor, quiero elegir entre retiro o entrega para definir cómo recibiré mi pedido.

- Escenario 1 (Éxito): Dado que selecciono la opción "Retiro", cuando realizo un pedido, entonces se actualiza la selección en punto de retiro y el pedido aparece con `deliveryMethod = "pickup"`.
- Escenario 2 (Éxito): Dado que selecciono "Entrega", cuando confirmo mi dirección, entonces el pedido queda con `deliveryMethod = "del_vary"` y se registra `deliveryAddress` y `lat` en el `deliveryCont`.
- Escenario 3 (Negativo - campos obligatorios): Dado que no se selecciona ninguna opción, cuando intento confirmar, entonces se muestra el mensaje de validación para completarlo.
- Escenario 4 (Éxito): Dado que un pedido ya existe, cuando lo visualizo en Pedidos (list), entonces se muestra un badge "Retiro/Entrega" y a su detalle se muestran los datos correspondientes.

Pre-condiciones

- Tener acceso a Internet.
- Contar con la aplicación instalada y operativa.
- Contar con la cuenta/rol requerido (según la IU).

Pasos y condiciones de ejecución

- Abrir la aplicación.
- Ingresar al módulo o flujo relacionado con: Ampliación del pedido: tipos de entrega.
- Ejecutar el/los escenario(s) según corresponda.
- Verificar el resultado mostrado por el sistema.

Resultado esperado

- El sistema cumple el comportamiento definido en cada escenario.
- Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.

Estado de prueba

	Éxito	Fallo
Errores asociados:	SI	No

[Firma]
MSc. Ximena Toro
PRODUCT OWNER

Pontificia Universidad Católica del Ecuador
Secretaría de Tecnología
Escuela de Ingeniería de Sistemas y Computación

PUCE

CASO DE PRUEBA 16 Fecha: 03/02/2026

Nombre caso de prueba: Selección de punto de retiro **Sprint:** 1

Metodología a evaluar: Selección de punto de retiro **Historia de usuario asociada:** 16

Técnica de prueba: Caja Negra ☑ Caja Blanca ☐ **Tipo:** Prueba de Aceptación

Descripción:
Como cliente o emprendedor, quiero ver cómo puedo retirar mi pedido para recogerlo en el lugar correcto.

- Escenario 1 (Éxito): Dado un pedido con retiro, cuando selecciono punto, entonces se guarda la ubicación (`pickUpPointId`) y datos del punto.
- Escenario 2 (Éxito): Dado el acceso al historial o activos, cuando consulto un pedido, entonces veo la dirección y foto del punto de retiro (o entrega), y si existe un maplet/ubicación, entonces puedo ver "Cómo llegar/Ver ruta".

Pre-condiciones

- Tener acceso a Internet.
- Contar con la aplicación instalada y operativa.
- Contar con la cuenta/rol requerido (según la IU).

Pasos y condiciones de ejecución

- Abrir la aplicación.
- Ingresar al módulo o flujo relacionado con: Selección de punto de retiro.
- Ejecutar el/los escenario(s) según corresponda.
- Verificar el resultado mostrado por el sistema.

Resultado esperado

- El sistema cumple el comportamiento definido en cada escenario.
- Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.

Estado de prueba

	Éxito	Fallo
Errores asociados:	SI	No

[Firma]
MSc. Ximena Toro
PRODUCT OWNER

Pontificia Universidad Católica del Ecuador		PUCE					
CASO DE PRUEBA 17 Fecha: 01/02/2026							
Nombre caso de prueba:	Validación de pagos por parte del administrador	Sprint:	2				
Medio/versión a evaluar:	Consultas de pedidos por fecha	Historia de usuario asociada:	17				
Técnica de prueba:	Caja Negra	Tipo:	Prueba de Aceptación				
Descripción: Como administrador, quiero recibir notificación de nuevos pedidos y contar con una función operativa de pedidos (crear, editar) para dar seguimiento, cambiar estados y coordinar acciones del pedido.							
Escenarios: - Escenario 1 (Exitoso): Dado un nuevo pedido, cuando ingreso a Pedidos, entonces lo veo en la lista con los campos de estado operativo, estado de entrega y estado de pago, y accedo al detalle. - Escenario 2 (Exitoso): Dado un nuevo pedido, cuando ingreso a Consultas de pedidos, entonces se muestra la información del cliente, productos, cantidades y descuentos (si aplica). - Escenario 3 (Exitoso): Dado que el pedido ha sido enviado, cuando se cambia su estado (pendiente/operativo/entregado/cancelado), entonces se guarda correctamente y queda registrado el nuevo estado para que el cliente lo ve reflejado en su sección Pedidos. - Escenario 4 (Exitoso): Dado un pedido en estado pendiente, cuando ingreso a la sección de pedidos, entonces se muestra el número del pedido, fecha de creación, monto de pago, estado de pago, estado de entrega, entonces al clickear el número veo una verificación (placeholder) de pago.							
Pre-condiciones: • Tener acceso a Internet. • Contar con la aplicación instalada y operativa. • Contar con la cuenta de usuario registrada (según la UI).							
Pasos y condiciones de ejecución: • Abrir la aplicación. • Ingresar al módulo o flujo relacionado con: Consultas de pedidos por parte del administrador. • Ejecutar el caso de prueba según corresponda. • Verificar el resultado mostrado por el sistema.							
Resultado esperado: • El sistema cumple el comportamiento definido en cada escenario. • Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba: <table border="1"> <thead> <tr> <th>Éxito</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>				Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
CASO DE PRUEBA 18 Fecha: 01/02/2026							
Nombre caso de prueba:	Validación de pagos (admin)	Sprint:	2				
Medio/versión a evaluar:	Validación de pagos (admin)	Historia de usuario asociada:	18				
Técnica de prueba:	Caja Negra	Tipo:	Prueba de Aceptación				
Descripción: Como administrador, quiero verificar el estado de pago de cada pedido para asegurar que solo se procesen los pedidos pagados.							
Escenarios: - Escenario 1 (Exitoso): Dado un pedido con pagos pendientes (no "pending_validation"), cuando reviso el comportamiento, entonces puedo marcarlo como "pagado" y registrar el pago (admin). - Escenario 2 (Negativo - datos inválidos/incorrectos/errores): Dado un pedido con pagos pendientes (no "pending_validation"), cuando reviso, entonces el pedido aparece como "pendiente" y puedo marcarlo como "pagado" (solo visual) si aplica. - Escenario 3 (Negativo - datos inválidos/incorrectos/errores): Dado que el comportamiento es inválido (si se completa), cuando lo reviso, entonces el estado vuelve a "pendiente" (comportamiento) y el comportamiento no se completa automáticamente (solo regla de negocio específica).							
Pre-condiciones: • Tener acceso a Internet. • Contar con la aplicación instalada y operativa. • Contar con la cuenta de usuario registrada (según la UI).							
Pasos y condiciones de ejecución: • Abrir la aplicación. • Ingresar al módulo o flujo relacionado con: Validación de pagos (admin). • Ejecutar el caso de prueba según corresponda. • Verificar el resultado mostrado por el sistema.							
Resultado esperado: • El sistema cumple el comportamiento definido en cada escenario. • Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba: <table border="1"> <thead> <tr> <th>Éxito</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>				Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

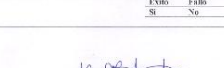
Pontificia Universidad Católica del Ecuador		PUCE					
CASO DE PRUEBA 19 Fecha: 01/02/2026							
Nombre caso de prueba:	Notificación de pago	Sprint:	2				
Medio/versión a evaluar:	Notificación de pago	Historia de usuario asociada:	19				
Técnica de prueba:	Caja Negra	Tipo:	Prueba de Aceptación				
Descripción: Como cliente, emprendedor o administrador, quiero recibir notificación sobre mis pedidos o novedades para estar informado en tiempo real, específicamente en los cambios de estado, verificación de pago y agendamiento.							
Escenarios: - Escenario 1 (Exitoso): Dado un cambio de estado en un pedido, cuando recibo, entonces se parametriza una notificación al usuario correspondiente (si no hay push not, se deja placeholder) y se notifica (si hay). - Escenario 2 (Exitoso): Dado un nuevo evento o comunicado, cuando se publica, entonces se notifica a la audiencia correspondiente (placeholder/admin).							
Pre-condiciones: • Tener acceso a Internet. • Contar con la aplicación instalada y operativa. • Contar con la cuenta de usuario registrada (según la UI).							
Pasos y condiciones de ejecución: • Abrir la aplicación. • Ingresar al módulo o flujo relacionado con: Notificación de pago. • Ejecutar el caso de prueba según corresponda. • Verificar el resultado mostrado por el sistema.							
Resultado esperado: • El sistema cumple el comportamiento definido en cada escenario. • Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba: <table border="1"> <thead> <tr> <th>Éxito</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>				Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
CASO DE PRUEBA 20 Fecha: 01/02/2026							
Nombre caso de prueba:	Consulta de pedidos por fecha	Sprint:	2				
Medio/versión a evaluar:	Consulta de pedidos por fecha	Historia de usuario asociada:	20				
Técnica de prueba:	Caja Negra	Tipo:	Prueba de Aceptación				
Descripción: Como cliente o emprendedor, quiero ver mis pedidos anteriores (filtrares por fecha) para consultar mis pedidos realizados en el pasado.							
Escenarios: - Escenario 1 (Exitoso): Dado que accedo a Pedidos > Historial, cuando filtro por un periodo (e.g., febrero 2025), entonces, se muestran los pedidos realizados en ese periodo. - Escenario 2 (Exitoso): Dado que accedo a Pedidos > Historial, cuando filtro por un periodo (e.g., febrero 2025), entonces, se muestran los pedidos realizados en ese periodo.							
Pre-condiciones: • Tener acceso a Internet. • Contar con la aplicación instalada y operativa. • Contar con la cuenta de usuario registrada (según la UI).							
Pasos y condiciones de ejecución: • Abrir la aplicación. • Ingresar al módulo o flujo relacionado con: Consulta de pedidos por fecha (cliente). • Ejecutar el caso de prueba según corresponda. • Verificar el resultado mostrado por el sistema.							
Resultado esperado: • El sistema cumple el comportamiento definido en cada escenario. • Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba: <table border="1"> <thead> <tr> <th>Éxito</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>				Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
CASO DE PRUEBA 21 Fecha: 01/02/2026							
Nombre caso de prueba:	Consulta de pedidos por fecha	Sprint:	2				
Medio/versión a evaluar:	Consulta de pedidos por fecha	Historia de usuario asociada:	21				
Técnica de prueba:	Caja Negra	Tipo:	Prueba de Aceptación				
Descripción: Como administrador, quiero ingresar un rango de fecha para consultar los pedidos recibidos, su cantidad y detalles, desde Admin > Pedidos.							
Escenarios: - Escenario 1 (Exitoso): Dado que ingreso una fecha de inicio y fin, cuando consulto, entonces se muestran todos los pedidos recibidos en ese rango, pagados, pendientes y cancelados.							
Pre-condiciones: • Tener acceso a Internet. • Contar con la aplicación instalada y operativa. • Contar con la cuenta de usuario registrada (según la UI).							
Pasos y condiciones de ejecución: • Abrir la aplicación. • Ingresar al módulo o flujo relacionado con: Consulta de pedidos por fecha (administrador). • Ejecutar el caso de prueba según corresponda. • Verificar el resultado mostrado por el sistema.							
Resultado esperado: • El sistema cumple el comportamiento definido en cada escenario. • Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba: <table border="1"> <thead> <tr> <th>Éxito</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>				Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
CASO DE PRUEBA 22 Fecha: 01/02/2026							
Nombre caso de prueba:	Agendamiento de entregas	Sprint:	2				
Medio/versión a evaluar:	Agendamiento de entregas	Historia de usuario asociada:	22				
Técnica de prueba:	Caja Negra	Tipo:	Prueba de Aceptación				
Descripción: Como administrador, quiero asignar fecha y hora de entrega a los pedidos, y que la información se visualice claramente en el app en una sección de pedidos programados.							
Escenarios: - Escenario 1 (Exitoso): Dado un pedido confirmado, cuando agendo fecha y hora de entrega y datos necesarios (puede/creo/actualiza si aplica), entonces la información se ve en la App en Pedidos > Programados con fecha y hora de entrega. - Escenario 2 (Exitoso): Dado un cambio de programación, cuando lo actualizo entonces se refleja en la App del usuario y queda listo para notificación (push si existe, placeholder si no lo es).							
Pre-condiciones: • Tener acceso a Internet. • Contar con la aplicación instalada y operativa. • Contar con la cuenta de usuario registrada (según la UI).							
Pasos y condiciones de ejecución: • Abrir la aplicación. • Ingresar al módulo o flujo relacionado con: Agendamiento de entregas. • Ejecutar el caso de prueba según corresponda. • Verificar el resultado mostrado por el sistema.							
Resultado esperado: • El sistema cumple el comportamiento definido en cada escenario. • Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba: <table border="1"> <thead> <tr> <th>Éxito</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>				Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad		Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad					
CASO DE PRUEBA 29 Fecha: 25/02/2026		Sprint: 2					
Nombre caso de prueba: Visualización de estadísticas (sin exportar)	Historia de usuario asociada: 29						
Medio/sección a evaluar: Visualización de estadísticas (sin exportar)	Tipos: Prueba de Aceptación						
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐							
Descripción: Como administrador, quiero ver estadísticas por fechas y productos para analizar la actividad de la plataforma.							
Pre-condiciones: - Tener acceso a Internet. - Contar con la aplicación instalada y operativa. - Contar con la cantidad requerida (según la UI).							
Pasos y condiciones de ejecución: - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Visualización de estadísticas (sin exportar). - Ejecutar el/los escenario/s según corresponda. - Verificar el resultado mostrado por el sistema.							
Resultado esperado: - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba		<table border="1"> <tr> <td>Éxito</td> <td>Fallo</td> </tr> <tr> <td>SI</td> <td>No</td> </tr> </table>		Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

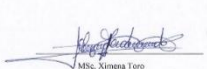
Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad		Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad					
CASO DE PRUEBA 30 Fecha: 25/02/2026		Sprint: 2					
Nombre caso de prueba: Registro de ubicación por la computadora	Historia de usuario asociada: 30						
Medio/sección a evaluar: Registro de ubicación por la computadora	Tipos: Prueba de Aceptación						
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐							
Descripción: Como emprendedor, quiero registrar automáticamente la dirección o punto donde voy (Inicio, paradas, visitas, punto de venta) y descripciones breves, para que las personas puedan utilizar más fácilmente desde la aplicación.							
Pre-condiciones: - Escenario 1 (Éxito): Dado que estoy en el formulario de registro de ubicación, cuando ingreso manualmente el barrio, punto de venta o sector donde voy y clicko en "Guardar", se muestra una notificación de éxito. - Escenario 2 (Negativo - campos obligatorios): Dado que dejo un campo obligatorio vacío (como la descripción o dirección), cuando intento guardar, entonces se muestra un mensaje de error indicando qué falta completar. - Escenario 3 (Negativo - datos invalidos/incorrectos): Dado que ingreso una imagen no válida (formato incorrecto o no soportada), cuando clicko en "Guardar", se muestra un mensaje de error indicando que el archivo no es soportado. - Escenario 4 (Éxito): Dado que ya he registrado una ubicación antes, cuando quiero registrar una nueva, se muestra un mensaje de "¿Desea reemplazar la ubicación anterior?" - Escenario 5 (Éxito): Dado que actualizo el número de verificación anterior, cuando confirmo los nuevos datos, entonces el sistema reemplaza los datos anteriores y actualiza el mapa y perfil público.							
Pre-condiciones: - Tener acceso a Internet. - Contar con la aplicación instalada y operativa. - Contar con la cantidad requerida (según la UI).							
Pasos y condiciones de ejecución: - Abrir la aplicación. - Ingresar al módulo o flujo relacionado con: Registro de ubicación por la computadora. - Verificar el resultado mostrado por el sistema.							
Resultado esperado: - El sistema cumple el comportamiento definido en cada escenario. - Ante validaciones/errores, se muestra el mensaje correspondiente y no se completa la acción.							
Estado de prueba		<table border="1"> <tr> <td>Éxito</td> <td>Fallo</td> </tr> <tr> <td>SI</td> <td>No</td> </tr> </table>		Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad		Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad					
CASO DE PRUEBA 31 Fecha: 25/02/2026		Sprint: 3					
Nombre caso de prueba: Menú Principal - Llamada a la acción	Historia de usuario asociada: 1						
Medio/sección a evaluar: Landing Page / Home	Tipos: Prueba de Aceptación						
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐							
Descripción: Como administrador o emprendedor quiero visualizar la pantalla de inicio con botones claros de registro y acceso Para conocer las opciones de la plataforma y navegar hacia mi sección correspondiente.							
Pre-condiciones: - Escenario 1 (Éxito): Dado que entro a la URL del sitio, cuando carga la página principal, entonces visualizo el logo, menú de navegación y los botones de llamada a la acción (CTA) destacados. - Escenario 2 (Éxito): Dado que hago clic en "Registrarse" o "Iniciar Sesión", cuando navego, entonces se abren las pantallas correspondientes. - Escenario 3 (Negativo): Dado que la página tiene elementos de carga, cuando intento interactuar antes de que carguen los scripts, entonces el sistema muestra la espera sin errores.							
Pre-condiciones: - Contar con acceso a Internet. - Tener un navegador web actualizado.							
Pasos y condiciones de ejecución: - Abrir la aplicación web en el navegador. - Verificar la visualización de los elementos principales (Logo, CTA, Menú). - Hacer clic en los botones de llamada a la acción.							
Resultado esperado: - La interfaz carga correctamente en menos de 3 segundos. - Los botones de llamada a la acción responden a los clicks correctos.							
Estado de prueba		<table border="1"> <tr> <td>Éxito</td> <td>Fallo</td> </tr> <tr> <td>SI</td> <td>No</td> </tr> </table>		Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad		Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad					
CASO DE PRUEBA 32 Fecha: 25/02/2026		Sprint: 3					
Nombre caso de prueba: Registro de Emprendedores	Historia de usuario asociada: 2						
Medio/sección a evaluar: Registro de Usuarios	Tipos: Prueba de Aceptación						
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐							
Descripción: Como emprendedor quiero registrarme con mis datos personales (nombre, correo y cédula y clave) Para poder crear una cuenta y gestionar mi emprendimiento desde la aplicación.							
Pre-condiciones: - Escenario 1 (Éxito): Dado que ingreso nombre, email válido y contraseña segura, cuando guardo el registro, entonces el sistema crea la cuenta y redirige al dashboard. - Escenario 2 (Negativo - campos obligatorios): Dado que dejo campos obligatorios vacíos, cuando intento guardar, entonces se muestran mensajes de error indicando qué falta completar. - Escenario 3 (Negativo - datos invalidos): Dado que ingreso un formato de email incorrecto, cuando intento guardar, entonces se bloquea la acción y se notifica el error.							
Pre-condiciones: - Formulario de registro accesible. - Opcional Correo electrónico o Cédula de identidad							
Pasos y condiciones de ejecución: - Navegar al registro. - Llenar el formulario. - Presionar "Registrar".							
Resultado esperado: - El sistema crea el usuario en la base de datos y muestra un mensaje de éxito (Toast).							
Estado de prueba		<table border="1"> <tr> <td>Éxito</td> <td>Fallo</td> </tr> <tr> <td>SI</td> <td>No</td> </tr> </table>		Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad		Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad					
CASO DE PRUEBA 33 Fecha: 25/02/2026		Sprint: 3					
Nombre caso de prueba: Login de Emprendedores	Historia de usuario asociada: 3						
Medio/sección a evaluar: Autenticación	Tipos: Prueba de Aceptación						
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐							
Descripción: Como emprendedor quiero iniciar sesión con mi correo y contraseña Para acceder a mis funciones privadas y gestionar mis productos y pedidos.							
Pre-condiciones: - Escenario 1 (Éxito): Dado que ingreso email y clave correctos, cuando presiono "Entrar", entonces accedo a mi panel personal y se genera la sesión activa. - Escenario 2 (Negativo): Dado que ingreso una contraseña incorrecta, cuando presiono "Entrar", entonces el sistema deniega el acceso y muestra un mensaje de "Credenciales incorrectas". - Escenario 3 (Negativo): Dado que ingreso un formato de correo inválido, cuando presiono entrar, entonces el sistema valida el campo antes de intentar la autenticación.							
Pre-condiciones: - Usuario registrado previamente en el sistema.							
Pasos y condiciones de ejecución: - Ir a la pantalla de Login. - Ingresar credenciales. - Validar el ingreso.							
Resultado esperado: - El usuario con credenciales válidas entra al sistema; de lo contrario, se bloquea con el mensaje de error respectivo.							
Estado de prueba		<table border="1"> <tr> <td>Éxito</td> <td>Fallo</td> </tr> <tr> <td>SI</td> <td>No</td> </tr> </table>		Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad		Sede Santa Domingo Escuela de Ingeniería, Infraestructura y Creatividad					
CASO DE PRUEBA 34 Fecha: 25/02/2026		Sprint: 3					
Nombre caso de prueba: Login para Administradores	Historia de usuario asociada: 4						
Medio/sección a evaluar: Autenticación Administrativa	Tipos: Prueba de Aceptación						
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐							
Descripción: Como Administrador quiero iniciar sesión con mis credenciales especiales Para acceder al panel de control y gestionar la red de emprendedores y productos.							
Pre-condiciones: - Escenario 1 (Éxito): Dado que ingreso credenciales de administrador, cuando el sistema valida el rol elevado, entonces soy redirigido al panel de gestión administrativa. - Escenario 2 (Negativo): Dado que una cuenta sin permisos de administrador intenta loguearse en esta sección, cuando el sistema valida, entonces deniega el acceso por falta de privilegios.							
Pre-condiciones: - Cuenta con rol "Administrador" configurada en la base de datos.							
Pasos y condiciones de ejecución: - Acceder al login administrativo. - Ingresar credenciales. - Verificar el redireccionamiento al panel exclusivo.							
Resultado esperado: - El sistema reconoce el rol y habilita las funciones de administración local.							
Estado de prueba		<table border="1"> <tr> <td>Éxito</td> <td>Fallo</td> </tr> <tr> <td>SI</td> <td>No</td> </tr> </table>		Éxito	Fallo	SI	No
Éxito	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santo Domingo Escuela de Historia, Infraestructura y Creatividad							
CASO DE PRUEBA 05		Fecha: 25/02/2026					
Nombre caso de prueba: Perfil de Empleado		Sprint: 3					
Medio de sección a evaluar: Perfil de Usuario		Historia de usuario asociada: 5					
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐		Tipo: Prueba de Aceptación					
Descripción: Como emprendedor Quiero actualizar mis datos de contacto y mi foto de perfil Para mantener mi información actualizada ante los clientes y la administración.							
- Escenario 1 (Exitoso): Dado que modifico mi número de contacto, cuando presiono "Guardar Cambios", entonces el sistema actualiza la información en la base de datos. - Escenario 2 (Exitoso): Dado que selecciono una nueva imagen de perfil, cuando subo el archivo, entonces la foto se actualiza visualmente en la aplicación. - Escenario 3 (Negativo): Dado que intento guardar el perfil dejando el nombre vacío, cuando confirmo, entonces el sistema impide el cambio y muestra un mensaje de error.							
Pre-condiciones							
• Sesión de emprendedor iniciada y operativa.							
Pasos y condiciones de ejecución							
• Entrar a "Mi Perfil". • Modificar datos o imagen. • Presionar "Guardar".							
Resultado esperado							
• Los cambios persisten en el sistema y se recibe una notificación de éxito tras la actualización.							
Estado de prueba		<table border="1"> <thead> <tr> <th>Exitos</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>		Exitos	Fallo	SI	No
Exitos	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santo Domingo Escuela de Historia, Infraestructura y Creatividad							
CASO DE PRUEBA 06		Fecha: 25/02/2026					
Nombre caso de prueba: Realizar Pedido		Sprint: 4					
Medio de sección a evaluar: Carrito de Compra (Checkout)		Historia de usuario asociada: 6					
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐		Tipo: Prueba de Aceptación					
Descripción: Como emprendedor Quiero seleccionar productos y confirmar mi pedido Para iniciar el proceso de compra de suministros para mi negocio.							
- Escenario 1 (Exitoso): Dado que he agregado productos al carrito, cuando confirmo el pedido, entonces el sistema genera un número de orden único y la registra en estado "Pendiente de Pago". - Escenario 2 (Negativo): Dado que intento confirmar un pedido con el carrito vacío, cuando presiono "Realizar Pedido", entonces el sistema muestra una alerta y bloquea la acción. - Escenario 3 (Negativo): Dado que un producto del carrito se queda sin stock mientras proceso la compra, cuando intento confirmar, entonces el sistema notifica el cambio y actualiza el total. - Escenario 4 (Exitoso): Dado que confirmo el pedido, cuando el proceso termina, entonces se limpia automáticamente el carrito de compra local.							
Pre-condiciones							
• Sesión de emprendedor iniciada. • Existencia de productos con stock disponible.							
Pasos y condiciones de ejecución							
• Navegar al catálogo. • Añadir 3 productos al carrito. • Ir al checkout y presionar "Confirmar Pedido".							
Resultado esperado							
• El sistema debe registrar la orden en la base de datos y mostrar a la pantalla de "Registro de Pago".							
Estado de prueba		<table border="1"> <thead> <tr> <th>Exitos</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>		Exitos	Fallo	SI	No
Exitos	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santo Domingo Escuela de Historia, Infraestructura y Creatividad							
CASO DE PRUEBA 07		Fecha: 25/02/2026					
Nombre caso de prueba: Registro de Pago		Sprint: 4					
Medio de sección a evaluar: Registro de Pago		Historia de usuario asociada: 7					
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐		Tipo: Prueba de Aceptación					
Descripción: Como emprendedor Quiero adjuntar el comprobante de mi transferencia bancaria Para que la administradora valide mi pago y despache mis productos.							
- Escenario 1 (Exitoso): Dado que selecciono un archivo de imagen válido, cuando guardo el registro, entonces el pedido cambia a estado "En Validación". - Escenario 2 (Negativo): Dado que subo un archivo .pdf o .docx, cuando el sistema valida, entonces muestra un error de "Formato de archivo no permitido". - Escenario 3 (Negativo): Dado que el archivo pesa más de 5MB, cuando intento subirlo, entonces se notifica que el archivo excede el tamaño máximo.							
Pre-condiciones							
• Tener un pedido en estado "Pendiente de Pago". • Tener el archivo de imagen del comprobante listo.							
Pasos y condiciones de ejecución							
• Ir a "Mis Pedidos". • Seleccionar "Subir Pago". • Cargar imagen y presionar "Enviar".							
Resultado esperado							
• El registro de pago se vincula al pedido y la administradora recibe la notificación visual en su panel.							
Estado de prueba		<table border="1"> <thead> <tr> <th>Exitos</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>		Exitos	Fallo	SI	No
Exitos	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santo Domingo Escuela de Historia, Infraestructura y Creatividad							
CASO DE PRUEBA 08		Fecha: 25/02/2026					
Nombre caso de prueba: Historial de Pedidos		Sprint: 4					
Medio de sección a evaluar: Historial de Pedidos		Historia de usuario asociada: 8					
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐		Tipo: Prueba de Aceptación					
Descripción: Como emprendedor Quiero visualizar el historial de mis pedidos realizados Para hacer seguimiento al estado de mis compras y descargar mis registros.							
- Escenario 1 (Exitoso): Dado que accedo al historial, cuando cargo la página, entonces veo la lista de órdenes con fecha, total y badge de estado (Pendiente/Aprobado). - Escenario 2 (Exitoso): Dado que clico un pedido específico, cuando hago clic en "Ver detalle", entonces visualizo el desglose de productos comprados.							
Pre-condiciones							
• Haber realizado al menos un pedido anteriormente.							
Pasos y condiciones de ejecución							
• Entrar al módulo "Mis Pedidos". • Verificar que los datos coincidan con las compras reales.							
Resultado esperado							
• La tabla muestra la información persistente de la base de datos de manera ordenada por fecha.							
Estado de prueba		<table border="1"> <thead> <tr> <th>Exitos</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>		Exitos	Fallo	SI	No
Exitos	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santo Domingo Escuela de Historia, Infraestructura y Creatividad							
CASO DE PRUEBA 09		Fecha: 25/02/2026					
Nombre caso de prueba: Administrador Dashboard		Sprint: 4					
Medio de sección a evaluar: Administrador Dashboard		Historia de usuario asociada: 9					
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐		Tipo: Prueba de Aceptación					
Descripción: Como Administradora Quiero visualizar métricas clave (ventas totales, usuarios nuevos, pedidos activos) Para tomar decisiones basadas en el rendimiento del sistema.							
- Escenario 1 (Exitoso): Dado que el dashboard carga, cuando el sistema consulta la API, entonces se muestran los gráficos de barras y contadores numéricos actualizados. - Escenario 2 (Negativo): Dado que no hay datos en el mes actual, cuando cargo el dashboard, entonces las gráficas muestran un estado de "Sin datos disponibles".							
Pre-condiciones							
• Rol de Administradora activo. • Conexión a la base de datos MySQL establecida.							
Pasos y condiciones de ejecución							
• Iniciar sesión como Admin. • Cargar la carga de los componentes de analítica en el inicio.							
Resultado esperado							
• Visualización correcta de KPI (Ventas totales, Total de emprendedoras y Pedidos por vender).							
Estado de prueba		<table border="1"> <thead> <tr> <th>Exitos</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>		Exitos	Fallo	SI	No
Exitos	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE					
Sede Santo Domingo Escuela de Historia, Infraestructura y Creatividad							
CASO DE PRUEBA 10		Fecha: 25/02/2026					
Nombre caso de prueba: Administración Pedidos		Sprint: 4					
Medio de sección a evaluar: Administración Pedidos		Historia de usuario asociada: 10					
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐		Tipo: Prueba de Aceptación					
Descripción: Como Administradora Quiero revisar los comprobantes de pago y actualizar el estado de las órdenes Para gestionar el flujo de ventas de la plataforma.							
- Escenario 1 (Exitoso): Dado que el pago es verificado, cuando presiono "Aprobar", entonces el pedido pasa a estado "Aprobado" y se descuenta el stock. - Escenario 2 (Exitoso): Dado que el comprobante es falso o emfeco, cuando presiono "Rechazar", entonces el pedido vuelve a estado "Pendiente de Pago".							
Pre-condiciones							
• Pedidos en estado "En Validación" existentes.							
Pasos y condiciones de ejecución							
• Entrar a Gestión de Pedidos. • Abrir la imagen del comprobante. • Seleccionar acción (Aprobar/Rechazar).							
Resultado esperado							
• Actualización inmediata en la base de datos y feedback visual mediante un Toast verde.							
Estado de prueba		<table border="1"> <thead> <tr> <th>Exitos</th> <th>Fallo</th> </tr> </thead> <tbody> <tr> <td>SI</td> <td>No</td> </tr> </tbody> </table>		Exitos	Fallo	SI	No
Exitos	Fallo						
SI	No						
Errores asociados:							
 MSc. Ximena Toro PRODUCT OWNER							

Pontificia Universidad Católica del Ecuador		PUCE	
Sede Santa Domínguez Facultad de Historia, Infraestructura y Construcción		Sede Santa Domínguez Facultad de Historia, Infraestructura y Construcción	
CASO DE PRUEBA 11	Fecha: 25/02/2026	CASO DE PRUEBA 12	Fecha: 03/03/2026
Nombre caso de prueba: Admin/Gestión/Productos	Serie: 4	Nombre caso de prueba: Admin/Gestión/Usuarios	Serie: 4
Metodología a evaluar: Admin/Gestión/Productos	Historia de usuario asociada: 11	Metodología a evaluar: Admin/Gestión/Usuarios	Historia de usuario asociada: 12
Técnica de prueba: Caja Negra ☒ Caja Blanca ☐	Tipo: Prueba de Aceptación	Técnica de prueba: Caja Negra ☒ Caja Blanca ☐	Tipo: Prueba de Aceptación
Descripción: Como Administrador Quiero gestionar el stock y las categorías de los productos Para asegurar que la oferta en la aplicación sea correcta y actualizada.		Descripción: Como Administrador Quiero gestionar los roles y el acceso de las empresadoras Para mantener la seguridad y la correcta organización de los perfiles en el sistema.	
- Escenario 1 (Exitoso): Dado que actualizo el stock de un producto a cero, cuando guardo, entonces el sistema le asigna automáticamente la etiqueta "Agotado". - Escenario 2 (Negativo): Dado que ingreso un precio con caracteres especiales, cuando intento guardar, entonces el sistema rechaza la entrada por validación de tipo.		- Escenario 1 (Exitoso): Dado que selecciono una empresadora, cuando cambio su rol a "Administrador", entonces el sistema actualiza sus permisos de acceso. - Escenario 2 (Exitoso): Dado que detengo un perfil inactivo, cuando cambio su estado a "Suspendida", entonces el usuario no podrá iniciar sesión. - Escenario 3 (Exitoso): Dado que busco por nombre en la barra de búsqueda, cuando escribo, entonces la tabla filtra los resultados instantáneamente.	
Pre-condiciones • Acceso al panel de administración de productos.		Pre-condiciones • Existencia de múltiples usuarios en la base de datos.	
Pasos y condiciones de ejecución - Seleccionar un producto. - Modificar cantidad de inventario o precio. - Guardar cambios.		Pasos y condiciones de ejecución - Entrar a Gestión de Usuarios. - Reservar una Suscripción. - Cambiar el rol de un usuario de prueba.	
Resultado esperado • Los cambios se reflejan en la vista del cliente en tiempo real.		Resultado esperado • Los permisos de seguridad se actualizan en el servidor y el usuario afectado recibe el cambio en su próximo inicio de sesión.	
Estado de prueba	Exitos Falló ☒ ☐	Estado de prueba	Exitos Falló ☒ ☐
Errores asociados:		Errores asociados:	
 MSc. Ximena Toro PRODUCT OWNER		 MSc. Ximena Toro PRODUCT OWNER	

Anexo 6. Evidencias de entrega y reconocimiento de la aplicación web y móvil





Anexo 7. Manual de usuario móvil

10 DE MARZO DE 2026

MANUAL DE USUARIO

IEEE 1063-2001

MARCO SILVA— LEONARDO DIAZ
PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
Santo Domingo

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2

REGISTRO DE CAMBIOS

FECHA	USUARIO	VERSIÓN	ACCIONES
10-MAR-26	MARCO	1.2	AJUSTE DE ESTRUCTURA Y FIGURAS
05-FEB-26	MARCO	1.0	CREACIÓN

Manual de Usuario | IEEE 1063 – 2001

1

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2

Tabla de contenido

1	INTRODUCCIÓN	4
1.1	Alcance del manual	4
1.2	Público objetivo	4
2	CONCEPTO DE LAS OPERACIONES	4
3	PROCEDIMIENTOS	5
3.1	Acceso inicial y opciones de entrada	5
3.1.1	Objetivo	5
3.1.2	Pasos	5
3.2	Registro de cuenta	6
3.2.1	Objetivo	6
3.2.2	Requisitos previos	6
3.2.3	Registro como cliente o emprendedora	6
3.2.4	Resultado esperado	7
3.2.5	Registro para administración	10
3.3	Inicio de sesión y verificación de correo	11
3.3.1	Objetivo	11
3.3.2	Pasos para iniciar sesión	11
3.3.3	Verificación de correo	12
3.4	Recuperación de contraseña	12
3.4.1	Objetivo	12
3.5	Navegación general dentro de la aplicación	16
3.5.1	Objetivo	16
3.6	Perfil, configuración y cierre de sesión	17
3.6.1	Objetivo	17
3.6.2	Pasos para editar el perfil	17
3.6.3	Cierre de sesión	17
3.7	Explorar emprendedoras y contenido destacado	19
3.7.1	Objetivo	19
3.7.2	Contenido destacado y historias	19
3.7.3	Mapa de puntos de venta	20
3.7.4	Listado y detalle de emprendedoras	20
3.8	Catálogo, carrito y realización de pedidos	23
3.8.1	Objetivo	23

Manual de Usuario | IEEE 1063 – 2001

2

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2

3.8.2	Requisitos previos	23
3.9	Pagos, comprobantes y seguimiento del pedido	27
3.9.1	Objetivo	27
3.9.2	Pedidos activos	27
3.9.3	Historial y filtros	27
3.9.4	Detalle y comprobante	27
3.10	Solicitud al proyecto para emprendedoras	30
3.10.1	Objetivo	30
3.10.2	Requisitos previos	30
3.11	Ubicación de venta para emprendedoras	32
3.11.1	Objetivo	32
3.11.2	Requisitos previos	32
3.12	Notificaciones	34
3.12.1	Objetivo	34
3.13	Funciones administrativas	35
3.13.1	Objetivo	35
3.13.2	Requisitos previos	35
3.13.3	Panel administrativo	35
3.13.4	Gestión de pedidos	35
3.13.5	Verificación de pago	35
3.13.6	Agendamiento de entregas	36
3.13.7	Zonas de venta y acuerdos privados de ubicación	36
4	PREGUNTAS FRECUENTES	42
4.1	¿Puedo usar la aplicación como visitante?	42
4.2	¿Puedo explorar productos sin iniciar sesión?	42
4.3	¿Puedo guardar favoritas sin iniciar sesión?	42
4.4	¿Qué ocurre si no verifico el correo?	42
4.5	¿Puedo subir el comprobante después de crear el pedido?	42
4.6	¿Las funciones administrativas están disponibles para cualquier cuenta?	42
5	Guía rápida de uso	43
6	GLOSARIO	44
7	REFERENCIA	45

Manual de Usuario | IEEE 1063 – 2001

3

1 INTRODUCCIÓN

La aplicación móvil 100 Mujeres Trabajando fue desarrollada para apoyar la interacción entre clientes, emprendedoras y personal autorizado del proyecto, reuniendo en un solo entorno digital procesos de acceso, exploración, pedidos, seguimiento y gestión operativa.

Con la aplicación móvil, tanto clientes como emprendedoras y el personal administrativo tendrán un medio digital para registrarse, consultar información, realizar acciones según su rol y dar seguimiento a los procesos principales de la app.

1.1 Alcance del manual

Este documento cubre los flujos principales de acceso, registro, inicio de sesión, recuperación de contraseña, navegación general, perfil, exploración, pedidos, comprobantes, solicitud al proyecto, ubicación de venta, notificaciones y funciones administrativas.

No reemplaza la documentación técnica del sistema. Cuando una pantalla no cuenta con evidencia visual suficiente en el material disponible, el manual la identifica como validación pendiente.

1.2 Público objetivo

- Visitantes que desean conocer la aplicación antes de crear una cuenta.
- Clientes que usan la app para explorar, comprar y seguir pedidos.
- Emprendedoras que completan solicitud al proyecto y registran su ubicación de venta.
- Administradoras con acceso a panel, pedidos, comprobantes, zonas y acuerdos.

2 CONCEPTO DE LAS OPERACIONES

- Los requerimientos mínimos para que la aplicación funcione correctamente son los siguientes:
- Dispositivo móvil compatible y conexión a internet estable.
- Permisos de cámara, galería o archivos cuando la app solicite adjuntos.
- Inicio de sesión y verificación de correo cuando el flujo de seguridad o la función utilizada lo exijan.

Advertencia: La app permite explorar parte del contenido como visitante, pero funciones como crear pedidos, guardar favoritos, editar perfil de forma persistente o registrar ubicación requieren una sesión activa.

3 PROCEDIMIENTOS

3.1 Acceso inicial y opciones de entrada

3.1.1 Objetivo

Explicar cómo iniciar el uso de la app desde la pantalla de bienvenida y qué opciones están disponibles antes de autenticarse.

Desde la pantalla de bienvenida es posible registrarse, iniciar sesión, continuar como visitante o abrir el cuadro de accesibilidad para ajustar la visualización.

3.1.2 Pasos

1. Abra la aplicación.
2. En la bienvenida seleccione Registrarse, Iniciar sesión o Continuar como visitante.
3. Si desea cambiar la visualización, toque el acceso de Accesibilidad.
4. Ajuste tamaño de texto, tema claro, oscuro o automático, y cierre el cuadro para volver a la pantalla anterior.

Nota operativa: El cuadro de Accesibilidad muestra vista previa, control de tamaño de texto, selección de tema y acciones fijas: Restablecer y Cerrar.



Figura 2. Pantalla de bienvenida de la aplicación.



Figura 3. Widget de accesibilidad y preferencias de visualización.

Validación pendiente: No se identificó una captura independiente del estado de inicio como visitante dentro de HomeTab. Conviene validarlo antes del cierre final del manual.

3.2 Registro de cuenta

3.2.1 Objetivo

Describir la creación de cuentas para cliente, emprendedora y miembro del equipo administrativo.

3.2.2 Requisitos previos

- No se requiere una sesión previa.
- Para el registro administrativo se necesita un código de invitación válido.

3.2.3 Registro como cliente o emprendedora

1. Desde la bienvenida toque Registrarse.
2. En la selección de rol elija Soy cliente o Soy emprendedora.
3. Presione Continuar.
4. Complete el Paso 1 con los datos solicitados.
5. Complete el Paso 2 con la información de perfil.
6. Acepte la política correspondiente y finalice con Crear cuenta.

Advertencia: Si no se selecciona un rol, el sistema muestra Por favor selecciona un rol. En el paso 2, si faltan datos o no se acepta la política, se impide continuar.

Nota operativa: Durante el registro se reutilizan validaciones como Este campo es obligatorio, Correo no válido, Teléfono no válido y Contraseña inválida.

3.2.4 Resultado esperado

- La cuenta cliente queda registrada y puede continuar al uso general.
- La cuenta emprendedora queda registrada y ofrece completar la solicitud al proyecto ahora o después desde Perfil.



Figura 3. Pantalla principal de selección de rol.



Figura 4. Tarjetas de selección visual para los roles cliente y emprendedora.

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2



Figura 5. Registro paso 1 para el rol emprendedora.



Figura 6. Registro paso 2 para el rol cliente.

Manual de Usuario | IEEE 1063 – 2001

8

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2

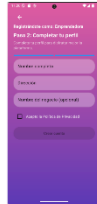


Figura 7. Registro paso 3 para el rol emprendedora.

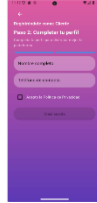


Figura 8. Registro paso 2 para el rol cliente.

Manual de Usuario | IEEE 1063 – 2001

9

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2

3.2.5 Registro para administración:

1. En la selección de rol use el acceso ¿Eres parte del equipo?.
2. Ingrese el código de invitación y pulse Verificar.
3. Complete el paso Credenciales con correo real y contraseña.
4. Complete el paso Perfil y finalice la creación de la cuenta.

Nota operativa: El flujo administrativo puede mostrar un bloqueo temporal de carga mientras verifica el código o crea la cuenta.



Figura 9. Ingreso y validación del código de invitación para el equipo administrativo.



Manual de Usuario | IEEE 1063 – 2001

10

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2

Figura 10. Paso 1 del asistente administrativo: credenciales.



Figura 11. Paso 2 del asistente administrativo: perfil.

3.3 Inicio de sesión y verificación de correo

3.3.1. Objetivo

Explicar cómo acceder a la cuenta y qué ocurre cuando el sistema exige verificar el correo electrónico.

3.3.2. Pasos para iniciar sesión.

1. En la bienvenida toque Iniciar sesión.
2. Ingrese Usuario y Contraseña.
3. Pulse el botón de ingreso.

Nota operativa: El campo Usuario acepta correo o número de documento. Durante el envío el botón cambia a ingresando....

FAQ: Si el identificador no puede resolverse, el sistema puede mostrar el mensaje No se pudo resolver credenciales para este identificador.

Manual de Usuario | IEEE 1063 – 2001

11

APP MÓVIL : 100 MUJERES TRABAJANDO

V 1.2



Figura 12. Pantalla de inicio de sesión.

3.3.3 Verificación de correo

Cuando la cuenta requiere verificación, la app redirige a una pantalla específica antes de permitir el acceso completo. La guía visible indica abrir el correo, verificar la cuenta y regresar a la aplicación para continuar.

1. Use Enviar correo de verificación si todavía no recibió el mensaje.
2. Abra el correo registrado y pulsa el enlace de verificación.
3. Regrese a la app y toque Ya verifiqué, continuar.

Advertencia: Si la persona intenta retroceder sin verificar, la app puede mostrar: Debes verificar tu correo para continuar. También existe un acceso temporal limitado visible según el estado de la cuenta.

Validación pendiente: En el caso de que la figura disponible no se encuentre una captura independiente de EmailVerificationPage. Conviene agregarla para que esta sección quede totalmente visual.

3.4 Recuperación de contraseña

3.4.1 Objetivo

Describir el flujo para recuperar el acceso mediante documento, correo o teléfono.

1. En la pantalla de inicio de sesión toque ¿Olvidaste tu contraseña?.
2. Elija buscar la cuenta por Documento o por Correo.
3. Toque Buscar cuenta y revise los métodos disponibles.
4. Seleccione Correo o SMS según el canal habilitado.
5. Complete la confirmación requerida por el sistema.
6. Si el flujo es por SMS, ingrese el código OTP.
7. Defina la nueva contraseña y confirme el cambio.

Manual de Usuario | IEEE 1063 – 2001

12

APP MÓVIL : 100 MUJERES TRABAJANDO

V 1.2

Nota operativa: La pantalla de recuperación usa el encabezado Recuperar acceso y, cuando la actualización es correcta, muestra Contraseña actualizada. Inicia sesión con la nueva.

Advertencia: Si no hay canales disponibles, la app informa No encontramos métodos para recuperar tu cuenta. Vuelve e intenta con otro dato.

FAQ: En la verificación por teléfono puede aparecer un contador para renovar. Si el código no tiene seis dígitos o expiró, el sistema solicita uno nuevo.



Figura 13. Búsqueda de cuenta mediante documento.



Figura 14. Búsqueda de cuenta mediante correo.

Manual de Usuario | IEEE 1063 – 2001

13

APP MÓVIL : 100 MUJERES TRABAJANDO

V 1.2



Figura 15. Selección del canal de recuperación disponible.



Figura 16. Confirmación del correo para envío del enlace de recuperación.

Manual de Usuario | IEEE 1063 – 2001

14

APP MÓVIL : 100 MUJERES TRABAJANDO

V 1.2



Figura 17. Confirmación del teléfono antes del envío de SMS.



Figura 18. Ingreso y validación del código OTP por SMS.

Manual de Usuario | IEEE 1063 – 2001

15

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2



Figura 19. Cambio de contraseña al final del flujo de recuperación.

3.5 Navegación general dentro de la aplicación

3.5.1 Objetivo

Ubicar a la persona usuaria dentro de las áreas principales de la app después del acceso o del ingreso como visitante.

La navegación principal conduce a Inicio, Catálogo, Pedidos y Perfil. Cuando la persona entra como visitante puede recorrer contenido público, pero las acciones que guardan información o ejecutan operaciones reales exigen autenticación.

Nota operativa: El Dashboard principal cambia según el rol. Los perfiles cliente y emprendedora comparten una vista operativa general, mientras que la administración accede a un panel protegido.

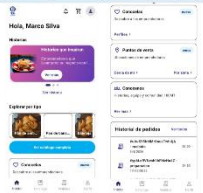


Figura 20. Vista general del inicio para usuario autenticado.

Manual de Usuario | IEEE 1063 – 2001

16

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2

3.6 Perfil, configuración y cierre de sesión

3.6.1 Objetivo

Describir cómo revisar datos personales, editar el perfil, cambiar la foto y cerrar sesión.

3.6.2 Pasos para editar el perfil

1. Entre a Perfil desde la navegación principal.
2. Use la acción Editar.
3. Modifique los datos necesarios y guarde los cambios.
4. Si desea cambiar la foto, abra el selector de imagen y elija el origen disponible.

Nota operativa: Si se intenta salir con cambios sin guardar, la app solicita confirmación. Durante la carga o guardado puede mostrarse un bloqueo temporal.

3.6.3 Cierre de sesión

1. Abra Perfil.
2. Pulse Cerrar sesión.
3. Confirme la acción en el cuadro de diálogo.

Advertencia: En modo invitado la acción puede mostrarse como Volver al inicio. La confirmación evita cierres de sesión accidentales.



Figura 21. Vista general de la pantalla Perfil.

Manual de Usuario | IEEE 1063 – 2001

17

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2



Figura 22. Pantalla Editar perfil.



Figura 23. Selector de imagen para foto de perfil.

Manual de Usuario | IEEE 1063 – 2001

18

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2



Figura 24. Estado de Perfil como invitado.

Figura 25. Acceso de cierre de sesión desde Perfil.



Figura 26. Diálogo de confirmación de cierre de sesión.

3.7 Explorar emprendedoras y contenido destacado

3.7.1 Objetivo

Presentar el recorrido de descubrimiento: historias, mapa de puntos de venta, listado de emprendedoras, detalle y favoritos.

3.7.2 Contenido destacado y historias

En inicio se muestra el bloque Para ti y una sección de Historias. Desde allí puede abrirse una historia para revisar la imagen principal, el título y la descripción ampliada.

Manual de Usuario | IEEE 1063 – 2001

19

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2

3.7.3 Mapa de puntos de venta

La pantalla Mapa permite revisar pines, contador de resultados y filtros. Si no existen coincidencias, la app muestra un estado sin resultados y permite limpiar filtros.

3.7.4 Listado y detalle de emprendedoras

1. Abra el listado de emprendedoras.

2. Revise las tarjetas disponibles.

3. Toque una tarjeta para abrir el perfil detallado.

4. Marque favorita si desea conservar la emprendedora en su lista personal.

Advertencia: El detalle puede verse sin cuenta, pero guardar favoritos de forma persistente requiere iniciar sesión.



Figura 27. Sección Historias en la pantalla inicio.

Manual de Usuario | IEEE 1063 – 2001

20

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2



Figura 28. Vista ampliada de una historia o contenido destacado.



Figura 29. Mapa de puntos de venta con búsqueda y filtros.

Manual de Usuario | IEEE 1063 – 2001

21

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2



Figura 30. Estado sin resultados en el mapa de puntos de venta.



Figura 31. Estado principal de emprendedoras.



Figura 32. Perfil detallado de una emprendedora.

Manual de Usuario | IEEE 1063 – 2001

22

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.2



Figura 33. Indicador visual de favorita aplicado en una tarjeta.

Validación pendiente: Conviene verificar capturas adicionales del bloque de contacto, filtros del mapa y tarjeta seleccionada para volver esta sección aún más visual.

3.8 Catálogo, carrito y realización de pedidos

3.8.1 Objetivo

Explicar cómo explorar productos, agregarlos al carrito y completar el checkout.

3.8.2 Requisitos previos

- Para crear el pedido se necesita una sesión iniciada.
- El catálogo y el carrito pueden abrirse sin autenticación.

1. Entre al Catálogo.

2. Abra el detalle del producto y seleccione presentación o cantidad.

3. Agregue el producto al carrito.

4. Revise el carrito y pulse Continuar al checkout.

5. Complete los datos de contacto.

6. Seleccione el método de entrega: Retiro en punto o Entrega coordinada.

7. Si elige retiro, seleccione un punto. Si elige entrega, ingrese dirección y teléfono.

8. Revise la cuenta bancaria, adjunte comprobante si ya lo tiene y envíe el pedido.

Advertencia: Si la persona no ha iniciado sesión, el checkout informa Debes iniciar sesión para crear un pedido. También bloquea la salida cuando el envío o la carga del comprobante están en proceso.

Nota operativa: El flujo puede recordar los datos de contacto para pedidos futuros y permite cambiar entre Retiro en punto y Entrega coordinada según la disponibilidad.

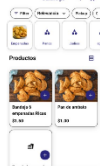


Figura 34. Entrada principal al catálogo de productos.

Manual de Usuario | IEEE 1063 – 2001

23



Figura 35. Opción del producto con selección de cantidad.



Figura 36. Carrito de compras previo al checkout.



Figura 37. Formulario principal de checkout.



Figura 38. Selector del método de entrega.



Figura 39. Despliegue de puntos de entrega disponibles.



Figura 40. Checkout con la modalidad Entrega coordinada.



Figura 41. Pantalla de confirmación luego de crear el pedido.

3.9 Pagos, comprobantes y seguimiento del pedido

3.9.1 Objetivo

Describir cómo revisar pedidos activos, historial, detalle y estado del comprobante.

3.9.2 Pedidos activos

La sección Pedidos muestra los pedidos en curso y, según el estado, acciones como Ver detalle, Subir comprobante o Cómo llegar. Si no existen pedidos activos, la pantalla invita a volver al catálogo.

3.9.3 Historial y filtros

El historial permite buscar por número o producto y aplicar filtros por periodo o estado. Los periodos visibles incluyen última semana, últimos 15 días, últimos 30 días, últimos 3 meses y últimos 6 meses.

3.9.4 Detalle y comprobante

El detalle del pedido se abre normalmente como hoja inferior. Allí pueden aparecer estados como Pago pendiente de comprobante, Pago en revisión y Pago confirmado. La sección de comprobante puede mostrar Sin comprobante adjunto, Comprobante listo para enviar o Comprobante disponible.

Nota operativa: Para adjuntar el comprobante, la app abre un selector con Cámara, Galería o Archivo. Si el proceso es exitoso, muestra Comprobante enviado.

Advertencia: Algunos cambios de comprobante dependen del estado del pedido y del estado actual del pago.



Figura 42. Pedidos activos con acceso a detalle y subida de comprobante.



Figura 43. Sección del comprobante visible dentro del detalle del pedido.



Figura 44. Historial de pedidos del cliente.



Figura 45. Hoja previa de filtros dentro el historial.

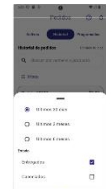


Figura 46. Hoja de filtros por periodo y estado.



Figura 47. Pedido con modalidad de retro desde la vista de pedidos.

Figura 48. Detalle del punto de retro dentro del pedido.

3.10 Solicitud al proyecto para emprendedoras

3.10.1 Objetivo

Guiar a la emprendedora en el envío de su solicitud al proyecto.

3.10.2 Requisitos previos

- Se requiere una sesión iniciada para continuar con el formulario completo.

- Entre al flujo desde el mensaje posterior al alta, desde Perfil o desde el acceso visible para emprendedoras no aprobadas.
- Pulse 'Crear solicitud'.
- Complete los datos requeridos y adjunte los soportes solicitados.
- Envíe la solicitud y revise la confirmación final.

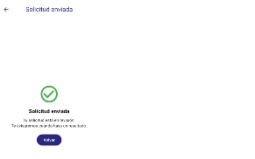
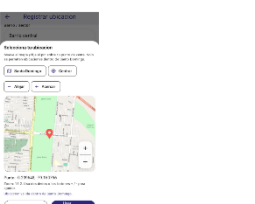
Nota operativa: Si no hay sesión, la pantalla de entrada puede ofrecer: iniciar sesión o registrarse antes de continuar.



Figura 49. Pantalla de entrada a la solicitud al proyecto.



Figura 50. Formulario principal de solicitud al proyecto.

APP MÓVIL - 100 MUJERES TRABAJANDO V 1.2  Figura 51. Confirmación de envío exitosa de la solicitud. 3.11 Ubicación de venta para emprendedoras 3.11.1 Objetivo Explicar cómo registrar o actualizar la ubicación de venta y la foto del punto. 3.11.2 Requisitos previos - Se requiere una sesión activa. - La opción está disponible para perfiles emprendedora. - Abra Mi ubicación de venta desde Perfil o desde el acceso visible del negocio. - Ingresa el barrio, sector o referencia y la descripción. - Seleccione el punto de venta en el mapa. - Ajunte la foto del punto de venta. - Guarde la información. Advertencia: El formulario valida descripción, punto de venta y foto obligatoria. Si ya existe una ubicación previa, la app pide confirmación antes de reemplazarla. Nota operativa: La pantalla introduce el flujo con una tarjeta que explica que la ubicación también puede prellenar la solicitud al proyecto. El guardado exitoso muestra Ubicación guardada. Manual de Usuario IEEE 1063 – 2001 32	APP MÓVIL - 100 MUJERES TRABAJANDO V 1.2  Figura 52. Formulario principal para registrar la ubicación de venta.  Figura 53. Estado de bloqueo visual durante guardado e procesamiento de imagen. Manual de Usuario IEEE 1063 – 2001 33
APP MÓVIL - 100 MUJERES TRABAJANDO V 1.2  Figura 54. Confirmación para reemplazar una ubicación existente. Validación pendiente: Sería útil incorporar una captura adicional del selector de mapa. Seleccione tu ubicación para completar el paso a paso visual. 3.12 Notificaciones 3.12.1 Objetivo Explicar cómo revisar la bandeja de notificaciones y abrir el contenido relacionado. - Abra la sección Notificaciones. - Revisa la lista de avisos. - Toque una notificación para abrir la pantalla relacionada. Nota operativa: Las notificaciones no leídas se distinguen visualmente. Cuando existen varias no leídas, puede aparecer la acción Marcar todas. FAQ: Si no hay registros, la pantalla informa No tienes notificaciones. Si la carga falla, ofrece Reintentar. Manual de Usuario IEEE 1063 – 2001 34	APP MÓVIL - 100 MUJERES TRABAJANDO V 1.2  Figura 55. Bandeja de notificaciones y ejemplo de estado asociado. 3.13 Funciones administrativas 3.13.1 Objetivo Presentar las funciones exclusivas para perfiles administrativos dentro del panel protegido. 3.13.2 Requisitos previos - La cuenta debe tener rol admin o super. - El acceso puede quedar condicionado por la verificación del correo. 3.13.3 Panel administrativo El panel administrativo es la vista principal para cuentas autorizadas. Desde ahí se accede a pedidos, zonas, acuerdos y otros módulos de gestión. 3.13.4 Gestión de pedidos La bandeja Pedidos permite buscar por cliente o teléfono, aplicar filtros y abrir el detalle de un pedido. El detalle se visualiza como hoja inferior y concentra el estado operativo, el pago y el agendamiento de entregas. 3.13.5 Verificación de pago La sección de pago usa estados como Sin comprobante, Comprobante subido, Comprobante en revisión, Comprobante validado y Comprobante rechazado. Si se rechaza el comprobante, la app solicita un motivo opcional. Manual de Usuario IEEE 1063 – 2001 35

3.13.6 Agendamiento de entregas

Para pedidos con modalidad de entrega se puede programar fecha y hora. También puede registrarse un costo de entrega opcional.

3.13.7 Zonas de venta y acuerdos privados de ubicación

La administración dispone de módulos para crear y revisar zonas de venta, así como para listar, abrir, crear o editar acuerdos privados de ubicación.

Advertencia: El contenido administrativo real está protegido por el componente AdminOnly. Si la cuenta no tiene el rol correcto, la interfaz administrativa no se renderiza.

FAQ: Si un deep link o una notificación intenta abrir un pedido inexistente, el sistema puede mostrar No se encontró el pedido solicitado o No se pudo abrir el detalle de este pedido.



Figura 54. Vista general del panel administrativo y bloque principal de indicadores.



Figura 57. Rendimiento operativo de pedidos en administración.



Figura 58. Detalle de pedido administrativo en hoja inferior.

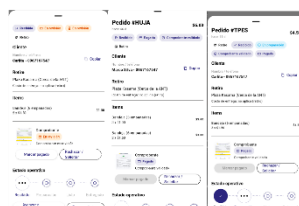


Figura 59. Estados administrativos del pago antes y después de validación.



Figura 60. Diseño para rechazo de comprobante.



Figura 61. Rendimiento administrativa con filtros por rango de fechas.



Figura 62. Flujo de agendamiento de entregas.



Figura 63. Módulo de zonas de vector



Figura 64. Listado de acuerdos privados de ubicación



Figura 65. Detalle de acuerdo privado de ubicación.



Figura 66. Formulario de acuerdo privado de ubicación.

Validación pendiente: Los textos visibles de `AdminSalesZonesPage`, `AdminLocationAgreementsPage` y `AdminDashboardPage` conviene validarlos otra vez con capturas finales antes de cerrar la versión de entrega.

4 PREGUNTAS FRECUENTES

4.1 ¿Puedo usar la aplicación como visitante?

4.1 ¿Puedo usar la aplicación como visitante?
 Si. El acceso como visitante permite explorar parte del contenido, aunque acciones como crear pedidos, guardar favoritos o registrar ubicación requieren sesión.

4.2 ¿Puedo explorar productos sin iniciar sesión?

Sí. Es posible abrir catálogo, detalle de producto y carrito sin autenticarse. Para crear el pedido sí se necesita una cuenta activa.

4.3 ¿Puedo guardar favoritas sin iniciar sesión?

No. El estado visual puede mostrarse, pero la persistencia del favorito requiere autenticación.

4.4 ¿Qué ocurre si no verifico el correo?

La app puede redirigir a la pantalla de verificación y bloquear el avance hasta completar ese paso o hasta agotar el acceso temporal disponible.

4.5 ¿Puedo subir el comprobante después de crear el pedido?

Si, siempre que el estado del pedido lo permita. El acceso puede aparecer desde la pantalla de éxito o desde Pedidos.

4.6 ¿Las funciones administrativas están disponibles para cualquier cuenta?

No. Requieren un rol administrativo y los controles de acceso correspondientes.

5 Guía rápida de uso

Flujo	Ruta resumida
Acceso inicial	Pantalla inicial → Registrarse / Iniciar sesión / Continuar como visitante.
Registro de cuenta	Bienvenida → Registrarse → Selección de rol → Paso 1 → Paso 2 → Confirmación.
Registro administrativo	Selección de rol → ¿Es parte del equipo? → Código de invitación → Credenciales → Perfil → Finalización.
Inicio de sesión	Bienvenida → Iniciar sesión → Credenciales? → Verificación de correo si aplica.
Recuperación	Inicio de sesión → ¿Olvidaste tu contraseña? → Buscar cuenta → Elegir método → Validar → Cambiar contraseña.
Perfil	Navegación principal → Perfil → Editar / Configurar / Cerrar sesión.
Catálogo y pedido	Catálogo → Producto → Carrito → Checkout → Confirmación.
Seguimiento	Pedidos → Detalle → Acciones sobre comprobante o retiro.
Exploración	Inicio / acceso correspondiente → Mapa o listado → Detalle → Favoritos si hay sesión.
Solicitud emprendedora	Perfil o acceso visible → Crear solicitud → Completar formulario → Enviar.
Ubicación de venta	Perfil o acceso visible → Mi ubicación de venta → Completar datos → Guardar.
Administración	Acceso con cuenta admin → Panel administrativo → Módulo requerido.

APP MÓVIL - 100 MUJERES TRABAJANDO V 1.2 6 GLOSARIO • Accesibilidad: Opciones de visualización que permiten ajustar el tamaño de texto y el tema de la aplicación. • Administradora: Persona usuaria con permisos para acceder a funciones internas de gestión y control. • Carrito: Sección donde se agrupan los productos seleccionados antes de confirmar un pedido. • Catálogo: Módulo en el que se visualizan los productos disponibles dentro de la aplicación. • Checkout: Proceso final de confirmación del pedido, donde se completan datos de contacto, entrega y pago. • Cliente: Persona usuaria que explora productos, realiza pedidos y consulta su seguimiento. • Comprobante de pago: Archivo o imagen que respalda el pago realizado por un pedido. • Correo verificado: Estado de la cuenta cuando la persona usuaria confirma su correo electrónico mediante el enlace enviado por el sistema. • Emprendedora: Persona usuaria que participa en el proyecto y puede acceder a funciones específicas dentro de la aplicación. • Entrega coordinada: Modalidad de entrega en la que se registran datos para coordinar el despacho del pedido. • Historial de pedidos: Registro de pedidos anteriores que puede consultarse mediante búsqueda o filtros. • Inicio de sesión: Proceso mediante el cual una persona usuaria ingresa a la aplicación con sus credenciales. • Notificación: Aviso generado por la aplicación para informar cambios, novedades o acciones pendientes. • OTP: Código de verificación de un solo uso utilizado en ciertos procesos de recuperación de acceso. • Pedido activo: Pedido que aún se encuentra en proceso y no ha finalizado completamente. • Perfil: Sección donde la persona usuaria consulta o edita su información personal y opciones de cuenta. • Punto de retiro: Lugar seleccionado para recoger un pedido cuando se utiliza esa modalidad de entrega. • Recuperación de contraseña: Flujo que permite restablecer la contraseña mediante correo o teléfono. • Registro: Proceso de creación de una nueva cuenta dentro de la aplicación. • Visitante: Persona que accede a la aplicación sin iniciar sesión y con funciones limitadas. Manual de Usuario IEEE 1063 – 2001 44	APP MÓVIL - 100 MUJERES TRABAJANDO V 1.2 7 REFERENCIA • IEEE (2001). 1063-2001 - IEEE Standard for Software User Documentation [Internet]. Recuperado de: https://ieeexplore.ieee.org/document/974401 Manual de Usuario IEEE 1063 – 2001 45
--	---

Anexo 8. Manual de usuario Web

13 DE MARZO DE 2026

MANUAL DE USUARIO

IEEE 1063-2001

MARCO SILVA- LEONARDO DIAZ
PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
Santo Domingo

APP WEB - 100 MUJERES TRABAJANDO

V 1.2

REGISTRO DE CAMBIOS

FECHA	USUARIO	VERSION	ACCIONES
05-FEB-26	LEONARDO	1.0	CREACIÓN
10-MAR-26	LEONARDO	1.1	ACTUALIZACIÓN DEL MANUAL DE USUARIO WEB
13-MAR-26	MARCO	1.2	INTEGRACIÓN, DEPURACIÓN Y CONSOLIDACIÓN

Manual de Usuario | IEEE 1063 – 2001

1

Tabla de contenido

1	INTRODUCCIÓN	3
1.1	Alcance del manual	3
1.2	Público objetivo	3
2	CONCEPTO DE LAS OPERACIONES	4
3	PROCEDIMIENTOS	5
3.1	Menú principal - llamado a la acción.....	¡Error! Marcador no definido.
3.2	Registro de emprendedoras	¡Error! Marcador no definido.
3.3	Inicio de sesión de emprendedora	¡Error! Marcador no definido.
3.4	Inicio de sesión de administradora	¡Error! Marcador no definido.
3.5	Perfil de emprendedora	¡Error! Marcador no definido.
3.6	Realizar pedido	¡Error! Marcador no definido.
3.7	Comprobante y ruta	¡Error! Marcador no definido.
3.8	Historial de pedidos.....	¡Error! Marcador no definido.
3.9	Panel administrativo.....	¡Error! Marcador no definido.
3.10	Gestión administrativa de pedidos.....	¡Error! Marcador no definido.
3.11	Gestión administrativa de productos	¡Error! Marcador no definido.
3.12	Gestión administrativa de usuarios.....	¡Error! Marcador no definido.
4	PREGUNTAS FRECUENTES	5
5	Guía rápida de uso	21
6	GLOSARIO	21
7	REFERENCIA	21

1 INTRODUCCIÓN

La aplicación web 100 Mujeres Trabajando fue desarrollada para apoyar la gestión de pedidos y la operación administrativa del proyecto desde una plataforma centralizada. Su finalidad es reducir la dependencia de procesos manuales dispersos y facilitar el acceso a tareas cotidianas mediante una interfaz web organizada por roles.

Desde la perspectiva de uso, las emprendedoras pueden registrarse, iniciar sesión, actualizar su perfil, realizar pedidos, consultar su historial y reportar comprobantes. De forma complementaria, el personal administrativo puede controlar pedidos, productos, usuarios y seguimiento operativo desde un panel de trabajo unificado.

1.1 Alcance del manual

El presente manual describe los procedimientos principales confirmados en los dos manuales revisados y consolida la información en una sola versión. El contenido se organiza por flujos reales de uso, priorizando las tareas que una persona ejecuta dentro del sistema antes que la descripción de componentes internos.

Cuando una acción no quedó completamente evidenciada con captura actual o presenta diferencias entre versiones, se la señala como validación pendiente para no introducir funciones no confirmadas.

1.2 Público objetivo

Este documento está dirigido a dos grupos de usuarios.

- El primero corresponde a las emprendedoras que utilizan la plataforma para registrar pedidos, reportar comprobantes de pagos y consultar su información de los datos su perfil.
- El segundo correspondiente al personal administrativo encargado del control de productos, usuarios, pagos y avance operativo de los pedidos, que controla la operación interna del sistema.

También resulta útil para personas responsables de inducción o soporte funcional, ya que resume el uso esperado del aplicativo con un lenguaje orientado a tareas.

2 CONCEPTO DE LAS OPERACIONES

Los requerimientos mínimos para que la aplicación web funcione correctamente son los siguientes:

- Disponer de un navegador web actualizado, preferiblemente Google Chrome, Microsoft Edge, Mozilla Firefox o Safari.
- Contar con conexión estable a internet durante el uso de la plataforma.
- Utilizar una cuenta activa y el acceso correspondiente al rol que se desea usar.
- Permitir la carga de imágenes cuando el sistema solicite foto de perfil, comprobantes o imágenes de producto.
- Habilitar el permiso de ubicación del navegador cuando se trabaje con la vista de mapa o coordenadas.

El sistema distingue, como mínimo, dos perfiles visibles de operación. La emprendedora accede a un dashboard para gestionar pedidos, historial, comprobantes, perfil y ubicación. La administradora accede a un panel de control para revisar pedidos, productos y usuarios.

De la revisión comparada de ambos manuales se concluye además que el acceso administrativo no se obtiene mediante un registro público independiente. El cambio de rol depende de una cuenta con privilegios administrativos.

No se confirmó visualmente en esta revisión un flujo público de recuperación de contraseña ni una captura actualizada de todos los módulos administrativos. Por ello, las secciones respectivas se mantienen con enfoque funcional y se señalan sus límites cuando corresponde.

3 PROCEDIMIENTOS

En esta sección se describen los procedimientos principales del sistema, organizados según el flujo operativo de emprendedoras y personal administrativo.

3.1 Acceso inicial y opciones de entrada

Objetivo: presentar el punto de entrada del sistema y orientar a la persona usuaria hacia el registro o el inicio de sesión según su perfil.

Requisitos previos: abrir la aplicación web desde un navegador con conexión estable a internet.

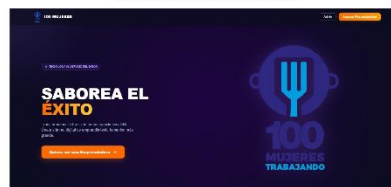
Pasos:

- Ingresar a la pantalla principal de la aplicación.
- Identificar las opciones visibles de registro e inicio de sesión.
- Seleccionar el acceso correspondiente si ya dispone de una cuenta.
- Seleccionar la opción de registro si todavía no cuenta con acceso.

Resultado esperado: la persona usuaria continúa hacia la pantalla de registro o al formulario de autenticación que corresponde a su rol.

Observaciones: esta vista cumple una función introductoria y de navegación. No se evidenciaron operaciones transaccionales dentro de esta pantalla.

Figura 1. Pantalla de bienvenida - llamado a la acción.



3.2 Registro de cuenta

Objetivo: permitir la creación de una cuenta para emprendedoras y aclarar cómo se habilita el acceso administrativo.

Requisitos previos: no se requiere una sesión activa previa.

Pasos para crear una cuenta de emprendedora:

- Ingresar desde la pantalla inicial a la opción de registro.
- Completar los datos solicitados en el formulario.

3. Elegir el medio principal de registro visible en la interfaz, ya sea correo o código, cuando la pantalla lo permita.
4. Definir y confirmar la contraseña.
5. Finalizar el proceso con la acción Crear cuenta.

Resultado esperado: si la información es válida, la cuenta queda creada y el sistema dirige a la pantalla de inicio de sesión o confirma que ya es posible ingresar.

Observaciones: la revisión de ambos manuales permite confirmar el registro de emprendedoras. En cambio, un registro público independiente para administración no quedó evidenciado.

Restricciones: el acceso administrativo debe ser habilitado desde una cuenta con privilegios de administración mediante la actualización del rol del usuario.

Figura 2. Formulario de registro de emprendedora.



3.3 Inicio de sesión de emprendedora

Objetivo: permitir el acceso de la emprendedora a su dashboard operativo.

Requisitos previos: contar con una cuenta registrada y utilizar la pantalla de acceso para emprendedoras.

Pasos:

1. Abrir la opción de inicio de sesión para emprendedora.
2. Ingresar la credencial correspondiente. Según el flujo revisado, puede utilizarse correo o código cuando esa alternativa se encuentre disponible.
3. Escribir la contraseña registrada.
4. Presionar el botón Iniciar sesión.

Resultado esperado: cuando las credenciales son correctas y el rol corresponde a emprendedora, el sistema permite el ingreso al panel principal.

Observaciones: el ingreso mediante código aparece como alternativa funcional en una de las versiones revisadas, por lo que se mantiene como flujo válido dentro de esta consolidación.

Restricciones: si la cuenta no posee el rol correcto, el acceso puede ser bloqueado por el sistema.

Figura 3. Inicio de sesión de emprendedora.



Figura 4. Ingreso alternativo mediante código.



3.4 Panel principal y realización de pedido

Objetivo: mostrar el dashboard principal de la emprendedora y explicar el flujo básico para registrar un pedido.

Requisitos previos: haber iniciado sesión correctamente con una cuenta de emprendedora.

Pasos:

1. Ingresar al dashboard y revisar el menú lateral disponible.
2. Abrir la opción Realizar pedido o el acceso equivalente al catálogo.
3. Seleccionar los productos deseados y ajustar cantidades con los controles visibles.
4. Revisar el resumen del pedido y el total acumulado.
5. Elegir la modalidad de entrega disponible y completar los datos adicionales si el sistema los solicita.
6. Confirmar la compra para registrar el pedido.

Resultado esperado: el sistema registra el pedido y muestra una confirmación con opciones para continuar comprando o revisar el historial.

Observaciones: en la comparación de manuales, el dashboard de emprendedora se confirma como el punto central para pedidos, historial, seguimiento, perfil y cierre de sesión.

Restricciones: la compra no puede completarse cuando el carrito se encuentra vacío.

Figura 5. Vista de carga del dashboard de emprendedora.

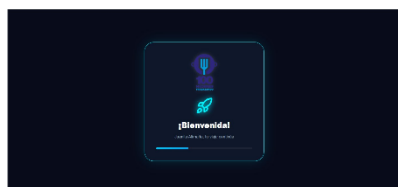


Figura 6. Vista principal del dashboard de emprendedora.



Figura 7. Panel lateral del dashboard.

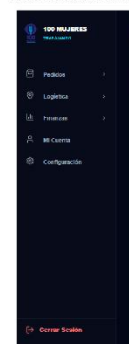
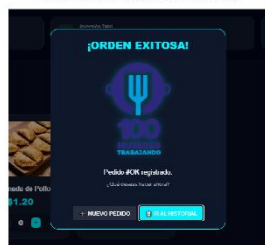


Figura 8. Selección de productos y cambio de compra.



Figura 9. Notificación de orden registrada correctamente.



3.5 Historial de pedidos

Objetivo: permitir la consulta de pedidos anteriores y el seguimiento básico de sus estados.

Requisitos previos: ingresar al dashboard de emprendedora y abrir la opción Historial de pedidos.

Pasos:

1. Abrir el historial de pedidos desde el menú correspondiente.
2. Revisar las tarjetas o registros visibles en pantalla.
3. Aplicar filtros por fecha o por identificador cuando se requiera acotar la búsqueda.
4. Consultar el estado del pedido y las acciones disponibles para cada caso.
5. Descargar el ticket o continuar al pago cuando el sistema habilite la opción.

Resultado esperado: la usuaria puede revisar pedidos previos, distinguir su estado y acceder a acciones complementarias como descarga de ticket o registro de pago.

Observaciones: el historial funciona también como punto de entrada al flujo de comprobante, por lo que cumple una doble función de consulta y gestión posterior al pedido.

Figura 10. Vista general del historial de pedidos.

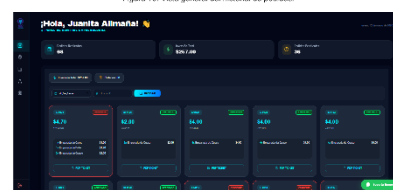
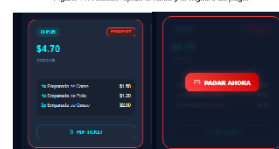


Figura 11. Acceso rápido al ticket y al registro de pago.



3.6 Comprobante de pago

Objetivo: explicar cómo notificar el pago de un pedido y adjuntar el comprobante correspondiente.

Requisitos previos: contar con un pedido que admita carga de comprobante desde el historial o desde el módulo de pago.

Pasos:

1. Abrir la opción de pago desde el historial o desde la sección financiera equivalente.
2. Seleccionar el pedido pendiente o elegible para notificar pago.
3. Verificar el método de pago disponible en la interfaz.

4. Adjuntar la imagen del comprobante.

5. Presionar la acción de envío o notificación del pago.

Resultado esperado: el sistema registra el comprobante y deja el pedido listo para revisión administrativa.

Observaciones: en los manuales revisados se evidencia como principal la carga de imágenes de comprobante. La alternativa de tarjeta aparece en una interfaz, pero no quedó completamente confirmada como flujo operativo final.

Restricciones: solo se pueden cargar comprobantes cuando el pedido se encuentra en un estado que habilita dicha acción.

Figura 12. Selección del ticket y carga del comprobante.

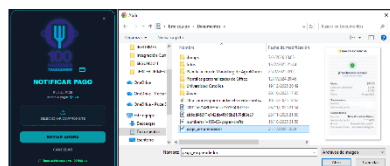
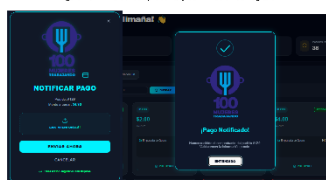


Figura 13. Envío del comprobante y confirmación del registro.



3.7 Ubicación de entrega y seguimiento

Objetivo: mostrar el uso del mapa para registrar coordenadas y describir el seguimiento básico del pedido.

Requisitos previos: haber iniciado sesión y conceder permisos de ubicación al navegador cuando el sistema lo solicite.

Pasos para registrar la ubicación:

1. Abrir la sección de perfil, dirección o ubicación según el acceso disponible.
2. Permitir que el navegador obtenga la ubicación actual.
3. Revisar el mapa y ajustar el marcador si es necesario.
4. Confirmar las coordenadas cuando la ubicación sea correcta.

Pasos para el seguimiento del pedido:

1. Abrir la vista de seguimiento desde el menú o desde el pedido activo.
2. Revisar la etapa visible del pedido dentro de la línea de progreso.
3. Consultar los cambios de estado hasta la fase de envío.

Resultado esperado: la ubicación queda asociada al perfil o al flujo logístico, y la usuaria puede visualizar el avance general del pedido.

Observaciones: la presencia del mapa y de la línea cronológica está respaldada por capturas. Sin embargo, la persistencia final de la vista Mi Ruta y algunos detalles logísticos aún requieren validación posterior.

Figura 14. Mapa para registro de coordenadas de la emprendedora.

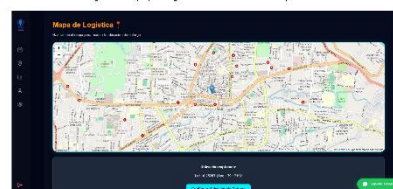


Figura 15. Ciclo cronológico inicial del pedido.

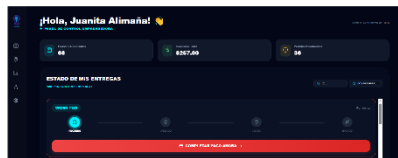
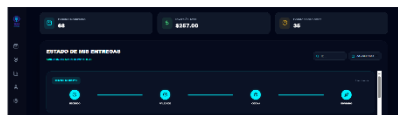


Figura 16. Ciclo cronológico final del pedido.



3.8 Configuración - Mi Perfil

Objetivo: permitir la visualización y actualización de la información personal y comercial de la emprendedora.

Requisitos previos: haber ingresado al dashboard y abrir la opción Mi Perfil o Mi Cuenta.

Pasos:

1. Revisar los datos visibles del perfil, como nombre, contacto, descripción, imágenes o redes sociales.
2. Seleccionar la opción Editar o el icono equivalente.
3. Actualizar la información necesaria y guardar los cambios.
4. Cambiar la foto de perfil o la imagen de portada si la interfaz lo permite.
5. Verificar también la sección de mapa cuando se requiera actualizar la ubicación.

Resultado esperado: el perfil queda actualizado y la plataforma muestra una confirmación cuando el guardado se realiza correctamente.

Observaciones: el perfil concierne datos personales, imagen y coordenados. Por ello, es una de las vistas más importantes para mantener operativa la atención de pedidos.

Restricciones: si se realizan cambios y no se guardan, la información puede perderse al cerrar o abandonar la edición.

Figura 17. Vista principal de Mi Perfil.



Figura 18. Modo edición de Mi Perfil.



3.9 Cierre de sesión

Objetivo: finalizar el acceso de manera segura desde la versión web.

Requisitos previos: disponer de una sesión iniciada.

Pasos:

1. Guardar los cambios pendientes antes de salir.
2. Abrir el menú del perfil o la opción visible en la parte superior de la interfaz.
3. Seleccionar Cerrar sesión.
4. Verificar que el sistema regrese a la pantalla de acceso.

Resultado esperado: la sesión finaliza correctamente y la cuenta deja de estar disponible en el navegador hasta un nuevo ingreso.

Observaciones: este paso es especialmente importante cuando se usa un equipo compartido o de acceso público.

Figura 19. Acceso visible al cierre de sesión.

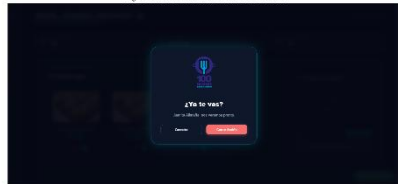
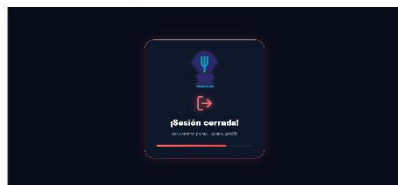


Figura 20. Confirmación de cierre de sesión.



3.10 Inicio de sesión de administradora

Objetivo: permitir el ingreso al panel de administración del sistema.

Requisitos previos: disponer de una cuenta con rol administrativo o equivalente.

Pasos:

1. Abrir la opción Admin desde la pantalla principal.
2. Ingresar la credencial autorizada y la contraseña correspondiente.
3. Presionar Iniciar sesión.

Resultado esperado: si la cuenta cuenta con permisos válidos, el sistema dirige al panel administrativo.

Observaciones: ambos manuales coinciden en que el acceso administrativo está restringido por rol. No se incorporó en esta versión una captura actualizada de esta pantalla, por lo que se conserva la descripción funcional.

Restricciones: una cuenta sin permisos administrativos no puede continuar hacia los módulos de control.

Figura 21. Inicio de sesión de administradora.



3.11 Panel administrativo

Objetivo: centralizar la supervisión operativa del sistema en una sola vista de control.

Requisitos previos: ingresar correctamente con una cuenta administrativa.

Pasos generales:

1. Revisar el resumen inicial disponible en el panel.
2. Usar el menú lateral para abrir los módulos de pedidos, productos y usuarios.
3. Cerrar la sesión al finalizar las tareas administrativas.

Resultado esperado: la administradora dispone de un centro de control desde el cual puede revisar información operativa y acceder a los demás módulos.

Observaciones: esta sección se incorporó a partir del manual elaborado por el usuario y se mantuvo por su valor funcional, aunque la captura asociada requiere actualización posterior.

Figura 22. Dashboard administrativo.



3.12 Gestión administrativa de pedidos

Objetivo: revisar pedidos, validar pagos y controlar el avance operativo de cada caso.

Requisitos previos: ingresar al panel administrativo y abrir la opción Gestión de Pedidos.

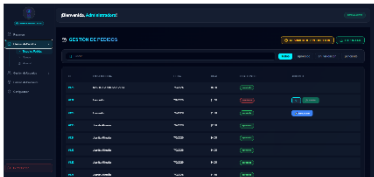
Pasos:

1. Buscar pedidos por identificador, emprendedora o fecha.
2. Aplicar filtros por estado cuando sea necesario.
3. Revisar el comprobante cargado por la emprendedora.
4. Aprobar o rechazar el pago según la validación realizada.
5. Actualizar el estado operativo o la información logística cuando corresponda.

Resultado esperado: el ciclo del pedido queda controlado por administración y cada caso avanza según su estado y validación.

Observaciones: de acuerdo con la información consolidada, este módulo es el núcleo operativo del perfil administrativo.

Figura 23. Gestión administrativa de pedidos.



3.13 Gestión administrativa de productos

Objetivo: registrar, editar y controlar la disponibilidad de los productos del catálogo.

Requisitos previos: ingresar al panel administrativo y abrir Gestión de Productos.

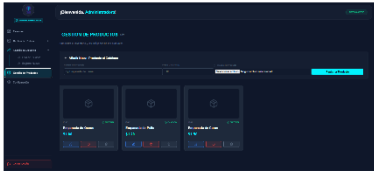
Pasos:

1. Completar los datos visibles del producto, como nombre, precio e imagen cuando aplique.
2. Guardar el nuevo registro o confirmar la edición de un producto existente.
3. Usar las acciones visibles para pausar, activar o retirar productos según la necesidad operativa.

Resultado esperado: el catálogo queda actualizado y los cambios se reflejan en la operación del sistema.

Observaciones: esta funcionalidad quedó descrita en ambos manuales como parte de la operación administrativa, aunque sin evidencia visual renovada en la versión base utilizada para las figuras.

Figura 24. Gestión administrativa de productos.



3.14 Gestión administrativa de usuarios

Objetivo: revisar las cuentas registradas y controlar roles o estados de acceso.

Requisitos previos: ingresar al panel administrativo y abrir Gestión de Usuarios.

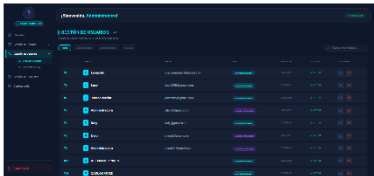
Pasos:

1. Buscar usuarios por nombre, correo o cédula.
2. Aplicar filtros por tipo de usuario o estado cuando sea necesario.
3. Abrir la edición del registro seleccionado.
4. Actualizar datos visibles, cambiar el rol o activar y desactivar el acceso según corresponda.

Resultado esperado: la base de usuarios queda organizada y alineada con los permisos requeridos por la operación.

Observaciones: esta sección también se conservó sin redundancia, tomando como base la descripción funcional más clara entre ambos manuales.

Figura 25. Gestión administrativa de usuarios.



4. PREGUNTAS FRECUENTES

• ¿Cómo me registro en la plataforma?

Debe ingresar desde la pantalla principal a la opción de registro, completar el formulario visible y confirmar la creación de la cuenta.

• ¿Puedo iniciar sesión con cédula?

Sí, el flujo revisado confirma que existe una alternativa de ingreso con cédula para emprendedoras cuando esa opción se encuentra habilitada en el formulario.

• ¿Qué hago si no puedo iniciar sesión?

Verifique que está usando el acceso correcto según su rol, que la credencial coincida con la registrada y que la contraseña haya sido escrita correctamente.

• ¿Quién puede entrar al módulo administrativo?

Solo las cuentas que ya cuentan con permisos administrativos o equivalentes.

• ¿Desde dónde puedo subir un comprobante?

Puede hacerlo desde el historial de pedidos o desde el módulo de pago equivalente cuando el pedido permite registrar comprobante.

• ¿Puedo cambiar mi ubicación de entrega?

Sí, siempre que la vista de mapa esté disponible y el sistema permita volver a confirmar coordenadas. En pedidos ya avanzados, este cambio puede requerir validación operativa.

• ¿La opción de tarjeta ya está confirmada?

En la interfaz aparece como alternativo, pero en la revisión consolidada no quedó plenamente confirmada como flujo operativo final.

5. Guía rápida de uso

La siguiente tabla resume los recorridos principales dentro del sistema.

Flujo	Ruta resumida
Acceso inicial	Pantalla principal -> registro o inicio de sesión según el rol.
Registro	Inicio -> registro -> completar formulario -> crear cuenta.
Login emprendedora	Inicio -> acceso emprendedoras -> ingresar credencial y contraseña.
Realizar pedido	Dashboard emprendedora -> realizar pedido -> seleccionar productos -> confirmar compra.
Historial	Dashboard emprendedora -> historial -> revisar estado, ticket o pago.
Comprobante	Historial o pago -> seleccionar pedido -> adjuntar imagen -> notificar pago.
Ubicación	Perfil o ubicación -> permitir acceso al mapa -> confirmar coordenadas.
Mi Perfil	Dashboard emprendedora -> Mi Perfil -> editar -> guardar cambios.
Cerrar sesión	Menú de perfil -> Cerrar sesión.
Login administradora	Inicio -> Admin -> ingresar credenciales.
Gestión de pedidos	Panel administrativo -> Gestión de Pedidos -> revisar y validar.
Gestión de productos	Panel administrativo -> Gestión de Productos -> registrar o editar.
Gestión de usuarios	Panel administrativo -> Gestión de Usuarios -> buscar, editar o cambiar rol.

6. GLOSARIO

Pedido: Solicitud registrada por una emprendedora para adquirir productos del catálogo.

Comprobante: Imagen o archivo que evidencia el pago realizado por un pedido.

Historial: Vista donde se consultan pedidos enterados, sus estados y acciones relacionadas.

Panel administrativo: Pantalla central en la cual se controla la operación interna del sistema.

Usuario: Persona con acceso registrado dentro de la plataforma.

Emprendedora: Usuario que ingresa para gestionar su perfil, pedidos y comprobantes.

Administradora: Usuario con permisos para controlar pedidos, productos y usuarios.

Rol: nivel de acceso asignado a una cuenta dentro del sistema.

Módulo: Conjunto de funciones agrupadas dentro de una misma sección de la aplicación.

Sesión: Estado de acceso activo de una cuenta dentro del sistema web.

Ruta: Vista logística relacionada con la ubicación o punto de entrega de un pedido.

Coordenadas: datos de ubicación utilizados para registrar un punto en el mapa.

7. REFERENCIA

- IEEE (2001). 1063-2001 - IEEE Standard for Software User Documentation [Internet]. Recuperado de: <https://ieeexplore.ieee.org/document/974401>

Anexo 9. Manual técnico móvil.

4 DE FEBRERO DE 2026

APP MÓVIL



100 MUJERES TRABAJANDO

MANUAL TÉCNICO
IEEE 1063-2001

MARCO SILVA— LEONARDO DÍAZ
PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
Santo Domingo

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.0

REGISTRO DE CAMBIOS

FECHA	USUARIO	VERSIÓN	ACCIONES
05-FEB-26	MARCO	1.0	CREACIÓN

Manual Técnico | IEEE 1063 – 2001

1

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.0

Tabla de Contenidos

1 INTRODUCCIÓN3

2 CONCEPTO DE LAS OPERACIONES.....3

2.1 Requerimientos del dispositivo3

2.2 Requerimientos del sistema3

2.3 Flujo operativo por rol4

2.4 Permisos y consideraciones de operación4

3 Procedimientos.....5

3.1 Clonar repositorio GitHub5

3.2 Estructura general del proyecto (Flutter).....5

3.2.1 Punto de entrada y composición de la app.....6

3.2.2 Navegación centralizada y control de acceso por rol.....7

3.2.3 Tema, accesibilidad y preferencias de visualización8

3.2.4 Organización por módulos (Clean Architecture por feature).....9

3.2.5 Flujos representativos: pedidos, entrega y comprobantes.....10

3.2.6 Módulo administrativo12

3.2.7 Backend serverless: Cloud Functions13

3.3 Configuración mínima14

3.4 Ejecución.....14

3.5 Compilación de entrega (APK/AAB)14

4 ACCESO A LOS DATOS.....14

4.1 Consola de Firebase (Firestore/Functions).....15

4.2 Consola de Cloudinary (almacenamiento multimedia).....16

5 GLOSARIO17

6 REFERENCIAS18

Manual Técnico | IEEE 1063 – 2001

2

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.0

1 INTRODUCCIÓN

Este manual técnico describe la estructura y organización de la aplicación móvil "100 Mujeres Trabajando", desarrollada en Flutter y soportada por servicios de Firebase y Cloud Functions. Se presenta la arquitectura adoptada (Clean Architecture por módulos), la navegación por rutas, el manejo de estado y los componentes principales que sustentan autenticación, pedidos, administración y contenido comunitario.

Además, se detallan los requisitos mínimos del entorno y los procedimientos esenciales para clonar, configurar y ejecutar el proyecto, incluyendo la compilación de entrega. Finalmente, se resumen las integraciones (Firestore, Storage y Functions) y consideraciones de acceso a datos, a fin de facilitar el mantenimiento y la revisión técnica del sistema.

2 CONCEPTO DE LAS OPERACIONES

A continuación, se detallan los requerimientos mínimos para el correcto uso y despliegue de la aplicación móvil.

2.1 Requerimientos del dispositivo

- Teléfono inteligente o tablet compatible con Android (recomendado).
- Conectividad a Internet (para operaciones con Firestore/Functions/Storage).

2.2 Requerimientos del sistema

- Flutter SDK compatible con Dart ^3.7.0 (según pubspec.yaml).
- Android toolchain configurado y dispositivo/emulador disponible.
- Firebase configurado para Android (android/app/google-services.json) y opciones en lib/firebase_options.dart.
- Cloud Functions (si aplica): Node.js 22 + TypeScript (functions/package.json).
- Cloudinary (si aplica): parámetros en lib/modules/core/config/storage_config.dart y secretos en functions/src/config/cloudinary.ts.

Manual Técnico | IEEE 1063 – 2001

3

APP MÓVIL: 100 MUJERES TRABAJANDO

V 1.0

3.2.3 Tema, accesibilidad y preferencias de visualización

La configuración visual global se define en lib/modules/core/config/app_theme.dart. La accesibilidad (tamaño de texto y tema claro/oscuro/automático) se gestiona mediante el diálogo lib/modules/core/widgets/settings_dialog.dart y su estado se mantiene con lib/modules/core/blocs/settings_cubit.dart.

Figura 6. Configuración de tema global (app_theme.dart).

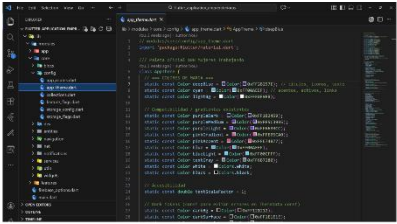
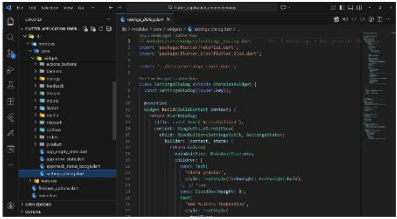


Figura 7. Diálogo de accesibilidad (settings_dialog.dart).



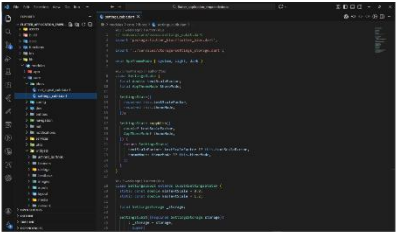
Manual Técnico | IEEE 1063 – 2001

8

APP MÓVIL: 100 MUJERES TRABAJANDO

V 1.0

Figura 8. Gestión de estado de accesibilidad (settings_cubit.dart).

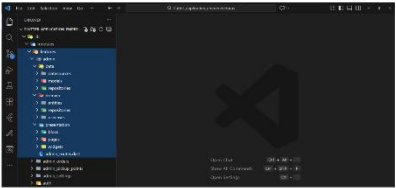


3.2.4 Organización por módulos (Clean Architecture por feature)

Cada módulo en lib/modules/features/<feature>/ sigue Clean Architecture, separándose:

- Presentation (UI): pantallas, widgets, navegación y estado (Bloc/Cubit).
- Domain: entidades, casos de uso y contratos (repositorios/interfaces).
- Data: modelos, repositorios concretos y datosources (Firestore/Auth/Storage/Functions).

Figura 9. Estructura de un feature (ejemplo: admin).



Manual Técnico | IEEE 1063 – 2001

9

APP MÓVIL: 100 MUJERES TRABAJANDO

V 1.0

3.2.5 Flujos representativos: pedidos, entrega y comprobantes

El módulo de pedidos implementa el flujo de checkout, selección de método de entrega/retrio y registro de comprobantes. Los puntos clave se ubican en archivos del módulo orders y en widgets especializados.

Figura 10. Pantalla de checkout (checkout_from_cart_page.dart).

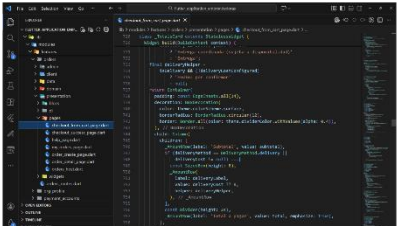
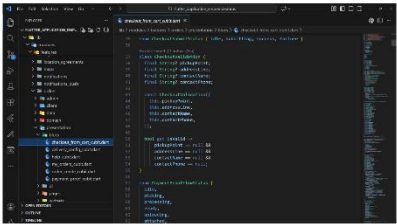


Figura 11. Lógica de checkout (checkout_from_cart_cubit.dart).



Manual Técnico | IEEE 1063 – 2001

10

APP MÓVIL: 100 MUJERES TRABAJANDO

V 1.0

Figura 12. Selector de método de entrega (delivery_method_selector.dart).

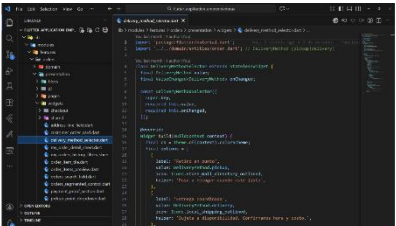
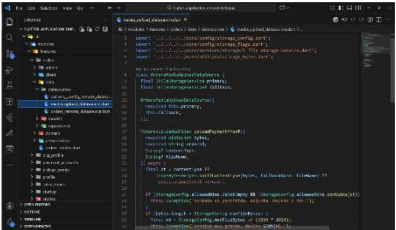


Figura 13. Carga de comprobantes (media_upload_datasource.dart).



Manual Técnico | IEEE 1063 – 2001

11

3.2.6 Módulo administrativo

El acceso administrativo se concentra en el módulo admin y sus submódulos (p. ej., admin_orders). La navegación interna y la estructura de vistas se soportan en archivos como admin_routes.dart y admin_shell.dart.

Figura 14. Rutas del módulo Admin (admin_routes.dart).

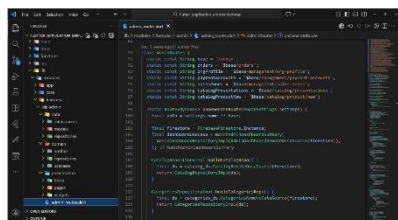
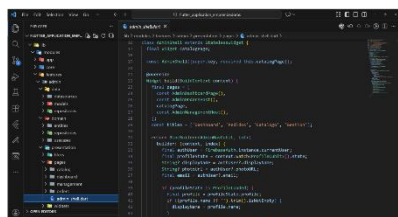


Figura 15. Contenedor de navegación del Admin (admin_shell.dart).



3.2.7 Backend serverless: Cloud Functions

Las Cloud Functions se ubican en functions/src. El archivo functions/src/index.ts registra las funciones expuestas y cada callable/trigger se implementa en functions/src/functions/.

Figura 16. Registro de funciones (functions/src/index.ts).

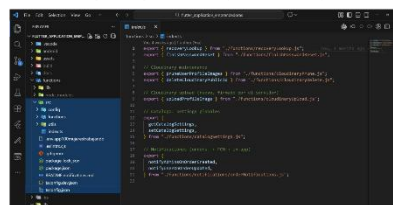
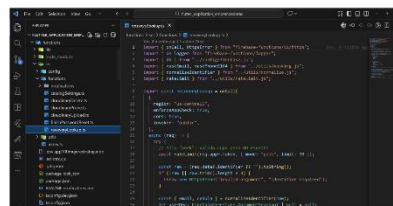


Figura 17. Callable representativa (recoveryLookup.ts).



3.3 Configuración mínima

1. Instalar dependencias Flutter: 'flutter pub get'.
2. Verificar Firebase: 'lib/firebase_options.dart' y 'android/app/google-services.json' (además de 'firebase.json' y '.firebaseignore').
3. Si se usan Functions: en 'functions/' ejecutar 'npm install' y 'npm run build'.
4. Verificar parámetros de almacenamiento (Cloudinary/Firebase Storage) según 'lib/modules/core/config/storage_config.dart'.
5. Preparar datos iniciales en el proyecto Firebase (por ejemplo, catálogo y parámetros operativos) para evitar estados vacíos en la app. Si se utilizan puntos de retiro, revisar la guía 'functions/src/config/pickup_points.seed.md' como referencia de carga/estructura.

3.4 Ejecución

- Ejecutar la app en dispositivo/emulador: 'flutter run'.
- Si se requiere validar funciones localmente: 'firebase emulators:start --only functions'.

3.5 Compilación de entrega (APK/AAB)

- Generar APK: 'flutter build apk'.
- Generar AAB: 'flutter build appbundle'.
- Configurar firma y ajustes de release en el módulo Android, según el entorno de entrega.

4 ACCESO A LOS DATOS

La persistencia se gestiona principalmente con Firestore, cuyas reglas están definidas en 'firestore.rules' y sus índices en 'firestore.indexes.json'. La carga de archivos se gestiona mediante servicios de almacenamiento (Cloudinary como principal si está habilitado; Firebase Storage como alternativa según configuración). La comunicación con Cloud Functions se realiza mediante un cliente de funciones en el módulo core.

A nivel conceptual, el sistema gestiona los siguientes tipos de datos:

- Usuarios y roles (acceso y rutas habilitadas tras autenticación).
- Pedidos (estado operativo, método de entrega/retiro, estado de pago y evidencia asociada).
- Catálogo (productos, categorías y presentaciones consumidas en módulos cliente y administración).
- Solicitudes y acuerdos (registro de nuevas emprendedoras y acuerdos de ubicación).
- Comunidad (historias, perfiles y puntos de venta visibles en mapa y sección social).

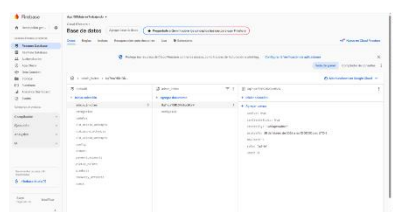
Para verificación operativa, se recomienda revisar la Consola de Firebase (Firestore/Functions/Storage), así como los logs de ejecución cuando se reproduzcan incidencias.

4.1 Consola de Firebase (Firestore/Functions)

La verificación de datos y eventos del sistema se realiza desde la Consola de Firebase, principalmente en Firestore (documentos) y Functions (logs/ejecución).

- Seleccionar el proyecto correspondiente en Firebase Console.
- Abrir Firestore Database para revisar registros creados por la app (usuarios, pedidos, catálogo y solicitudes) y validar estados/fechas.
- Revisar Cloud Functions para confirmar ejecución de funciones callable y analizar logs cuando exista un error.
- Validar reglas e índices cuando se presenten fallos de permisos o consultas (firebase.rules / firestore.indexes.json).

Figura 18. Consola de Firebase (Sección de la Base de Datos Firestore)



APP MÓVIL - 100 MUJERES TRABAJANDO

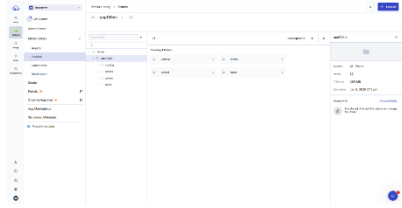
V 1.0

4.2 Consola de Cloudinary (almacenamiento multimedia)

Cuando el almacenamiento multimedia está habilitado, los archivos (por ejemplo, comprobantes) se gestionan en Cloudinary y se registra su URL de referencia en los datos del sistema.

- Abrir Cloudinary Console → Media Library
- Ubicar recursos cargados (por carpetas o por ficha) y validar que correspondan al pedido/registro esperado.
- Si un archivo no aparece, revisar límites de tamaño/formato y la configuración de subida definida en el cliente y/o Functions.

Figura 19. Servidor de imágenes en la Nube Cloudinary



Manual Técnico | IEEE 1063 – 2001

16

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.0

5 GLOSARIO

- **Flutter:** Framework para desarrollo multiplataforma basado en widgets.
- **Dart:** Lenguaje de programación utilizado por Flutter.
- **Clean Architecture:** separación por capas (presentation/domain/data) por feature para mantenibilidad y pruebas.
- **Presentation:** capa de UI: páginas, widgets, navegación y manejo de estado (BloC/Cubit).
- **Domain:** capa de negocio: entidades, casos de uso y contratos (interfaces de repositorio).
- **Data:** capa de acceso a datos: modelos/DTO, datasources e implementaciones concretas de repositorios.
- **BLoC/Cubit:** Patrones para gestión de estado reactiva y separación de lógica de UI.
- **Repository Pattern:** patrón para desacoplar dominio de proveedores de datos (Firestore/Cloudinary).
- **Dependency Injection:** inyección de dependencias para componer repositorios, casos de uso y cubits/blocs.
- **Firebase Auth:** Servicio de autenticación y gestión de sesiones.
- **Firestore:** base de datos NoSQL orientada a documentos (Firestore).
- **Cloud Functions:** funciones serverless para lógica backend (callable/trigger).
- **FCM:** Firebase Cloud Messaging para notificaciones push (si aplica).
- **App Check:** mecanismo para proteger recursos Firebase validando integridad de la app (si está habilitado).
- **Cloudinary:** servicio externo de almacenamiento/transformación de archivos multimedia (si está habilitado).
- **ThemeMode:** preferencia de tema (claro/oscuro/auto) aplicada a la UI.
- **DTO/Modelo:** estructura de datos usada para transportar o mapear información entre capas.
- **Ruta (Route):** identificador de navegación para abrir una pantalla dentro de la app.

Manual Técnico | IEEE 1063 – 2001

17

APP MÓVIL - 100 MUJERES TRABAJANDO

V 1.0

6 REFERENCIAS

- IEEE (2001). 1063-2001 - IEEE Standard for Software User Documentation [Internet]. Recuperado de: <https://ieeexplore.ieee.org/document/974401>

Manual Técnico | IEEE 1063 – 2001

18

Anexo 10. Manual técnico Web

12 DE MARZO DE 2026



MANUAL TÉCNICO
IEEE 1063-2001

MARCO SILVA- LEONARDO DIAZ
PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
Santo Domingo

APP WEB - 100 MUJERES TRABAJANDOv 2.0

REGISTRO DE CAMBIOS

FECHA	USUARIO	VERSIÓN	ACCIONES
05-FEB-26	LEONARDO	1.0	CREACIÓN
13-MAR-26	MARCO	2.0	ACTUALIZACIÓN DEL MANUAL TÉCNICO WEB PARA EL SISTEMA WEB REACT + FIREBASE.

Manual Técnico | IEEE 1063 – 20011

APP WEB - 100 MUJERES TRABAJANDOv 2.0

Tabla de Contenidos

1INTRODUCCIÓN.....3

2CONCEPTO DE LAS OPERACIONES.....3

2.1Requerimientos de acceso web.....3

2.2Requerimientos del entorno técnico.....3

2.3Flujo operativo por rol.....4

2.4Consideraciones y restricciones operativas.....4

3Procedimientos.....5

3.1Clonar repositorio GitHub.....5

3.2Estructura general del proyecto web.....5

3.2.1Punto de entrada y composición del frontend web.....6

3.2.2Navegación y control de acceso por rol.....6

3.2.3Arquitectura modular del frontend web.....7

3.2.4Servicios principales e integraciones.....7

3.2.5Flujos representativos del sistema web.....8

3.2.6Panel administrativo.....10

3.2.7Scripts de migración y soporte técnico.....12

3.3Configuración mínima.....12

3.4Ejecución.....12

3.5Build y despliegue en Firebase Hosting.....13

4ACCESO A LOS DATOS.....13

4.1Firebase Authentication y Cloud Firestore.....14

4.2Cloudinary y verificación operativa.....14

5GLOSARIO.....16

6REFERENCIAS.....16

Manual Técnico | IEEE 1063 – 20012

APP WEB - 100 MUJERES TRABAJANDOv 2.0

1 INTRODUCCIÓN

Este manual técnico documenta el sistema web "100 Mujeres Trabajando" implementado en React 19 y Vite 6 sobre la rama "firebase-migration". El sistema opera como una SPA accesible desde navegador, autenticada con Firebase Authentication, persistida en Cloud Firestore, con gestión multimedia en Cloudinary y despliegue en Firebase Hosting.

El alcance de este documento se restringe al sistema web. Aunque la solución comparte el proyecto Firebase "app100mujerestrabajando" con la aplicación móvil, no describe la arquitectura ni los procedimientos de Flutter. Asimismo, "server.js" permanece en el repositorio como activo legado y no participa en la ejecución normal del frontend web migrado.

Las capturas operativas sugeridas para la versión diagramada final se indican dentro de las secciones correspondientes mediante referencias de figura, con el fin de orientar la inserción editorial final sin mantener referencias visuales móviles que no puedan sustentarse en el estado actual del repositorio.

2 CONCEPTO DE LAS OPERACIONES

El sistema se utiliza desde un navegador con conectividad activa hacia Firebase y Cloudinary. La operación diaria se concentra en dos flujos funcionales, emprendedora y administradora, mientras que el acceso público queda limitado a la pantalla inicial y a los formularios de autenticación y registro.

Figura 1. Pantalla de inicio web y acceso por rol (/).

2.1 Requerimientos de acceso web

- Navegador moderno compatible con módulos ES, almacenamiento local y carga de archivos (Chrome, Edge, Firefox o Safari actualizados).
- Conexión a Internet para autenticación, lectura y escritura en Firestore, consumo de rutas SPA y carga de archivos a Cloudinary.
- JavaScript habilitado y permiso del navegador para seleccionar archivos locales cuando se registren comprobantes, imágenes de producto o fotografías de perfil.
- Resolución recomendada de escritorio o portátil; la interfaz responde en Tablet y móvil, aunque la operación administrativa resulta más cómoda en pantallas medianas o grandes.

2.2 Requerimientos del entorno técnico

- Node.js 22 o superior, preferentemente 22 LTS, alineado con ".nvmrc", ".node-version" y "package.json > engines.node".

Manual Técnico | IEEE 1063 – 20013

APP WEB - 100 MUJERES TRABAJANDO

v 2.0

- npm instalado y dependencias del proyecto resueltas mediante 'npm install' en la raíz activa del repositorio.
- Archivo '.env.local' con variables 'VITE_FIREBASE_*', 'VITE_AUTH_TECH_EMAIL_DOMAIN', 'VITE_CLOUDINARY_*' y 'VITE_ENABLE_PAYMENT_EVENTS' según 'docs/firebase-migration/setup/firebase_env_setup_guide.md'.
- Firebase CLI autenticado y con acceso al proyecto 'app100mujerstrabajando' para tareas de despliegue web.
- Acceso al espacio Cloudinary correspondiente, incluyendo 'cloud_name', unsigned preset y carpeta basic 'app100mt' o la que defina el entorno.
- Archivo '.env' adicional solo cuando se requieran scripts de Admin SDK o migración, con 'FIREBASE_SERVICE_ACCOUNT_*', 'ADMIN_EMAILS' y 'LEGACY_MYSQL_*'.
- Trabajo sobre la rama 'firebase-migration', ya que esa rama documenta el modo Firebase directo y el despliegue hosting-only.

2.3 Flujo operativo por rol

El control de acceso se resuelve por rutas web y por rol de negocio normalizado en Firestore. La resolución del rol se apoya en 'users/{uid}' y en los helpers de normalización, admitiendo 'admin', 'super' y la compatibilidad legado 'administradora'. Cuando el acceso se realiza con cédula, el sistema intenta recuperar el correo real desde la colección 'cedulas'; si no existe un mapeo previo, utiliza un correo técnico con dominio configurable.

Actor o ámbito	Módulos y responsabilidades principales
Público o visitante	Pantalla de inicio, acceso a login de administradora, login de emprendedora y registro web.
Emprendedora	Autenticación, catálogo, creación de pedidos, historial, carga de comprobantes, seguimiento, perfil y logística referencial.
Administradora	Dashboard administrativo, resumen ejecutivo, gestión de pedidos, validación de pagos, gestión de usuarios y catálogo de productos.
Operación técnica	Configuración local, build, despliegue hosting-only y ejecución controlada de scripts de migración y bootstrap con Admin SDK.

2.4 Consideraciones y restricciones operativas

La web comparte el mismo proyecto Firebase con la aplicación móvil, por lo que la operación debe respetar un contrato común de colecciones, reglas e índices. El frontend migrado trabaja en modo directo contra Firebase y no depende de un backend Express para su operación normal.

- La sesión se mantiene con 'browserLocalPersistence' de Firebase Auth y con respaldo en 'localStorage' para nombre, rol y último acceso.

Manual Técnico | IEEE 1063 – 2001

4

APP WEB - 100 MUJERES TRABAJANDO

v 2.0

- El repositorio web no debe desplegar automáticamente 'firebase.rules' ni 'firebase.indexes.json'; cualquier ajuste debe pasar por merge manual con la base compartida de Flutter.
- La escritura extendida sobre 'entrepreneur_profiles' puede degradarse de forma segura hacia 'users/{uid}' si las reglas compartidas restringen la actualización owner-write.
- La subcolección 'orders/{orderId}/payment_events' permanece desactivada por defecto ['VITE_ENABLE_PAYMENT_EVENTS=false'] mientras no se apruebe una regla compartida.
- El módulo 'Logística' utiliza Leaflet y OpenStreetMap para selección visual de ubicación, pero la persistencia operativa sobre 'entrepreneur_locations' sigue catalogada como pendiente de cierre.

3 Procedimientos

3.1 Clonar repositorio GitHub

La fuente principal de verdad para este manual es el repositorio web React activo, no la carpeta 'empanada-frontend-simple' ni la aplicación móvil. El flujo de obtención recomendado es el siguiente:

1. Clonar el repositorio web: 'git clone https://github.com/marcogamer55/React-Web100MujeresTrabajando.git'
2. Ingresar a la carpeta raíz activa: 'cd React-Web100MujeresTrabajando'
3. Cambiar a la rama de trabajo documentada: 'git checkout firebase-migration'
4. Instalar dependencias del frontend: 'npm install'
5. Confirmar que los cambios y validaciones se realizan sobre la raíz del repositorio, no sobre 'empanada-frontend-simple'.

3.2 Estructura general del proyecto web

La estructura activa del sistema web se concentra en la raíz del repositorio. La organización real observada en la rama 'firebase-migration' es la siguiente:

- 'src': aplicación React principal y única fuente activa del frontend.
- 'src/pages/': pantallas completas ('AdminLogin', 'AdminDashboard', 'EmprendedoraLogin', 'EmprendedoraDashboard', 'Register').
- 'src/components/': módulos funcionales reutilizados por los dashboards, como pedidos, historial, perfil, pagos, seguimiento, sidebars y administración.
- 'src/services/': integración con Firebase, Firestore, Cloudinary y normalización de estados compartidos.
- 'src/context/': 'AdminContext' y hooks relacionados para agregación y refresco del panel administrativo.
- 'src/config/', 'src/constants/', 'src/helpers/' y 'src/mappers/': configuración de auth y media, constantes de colecciones y roles, utilitarios y mapeo de documentos Firestore.

Manual Técnico | IEEE 1063 – 2001

5

APP WEB - 100 MUJERES TRABAJANDO

v 2.0

- 'public/' y 'src/assets/': logos, imágenes de apoyo y favicon consumidos por la interfaz web.
- 'docs/firebase-migration/': documentación de setup, despliegue, auditorías, gobernanza Firestore y cierre de migración.
- 'scripts/': utilitarios de migración MySQL -> Firestore y bootstrap de administradoras mediante Admin SDK.
- 'firebase.json', 'firebase.rules', '.env.example' y 'vite.config.js': archivos clave para configuración del cliente y despliegue hosting-only.
- 'server.js', 'uploads/' y 'empanada-frontend-simple/': activos legados que no forman parte del flujo runtime del sistema web documentado.

3.2.1 Punto de entrada y composición del frontend web

El documento 'index.html' declara el contenedor 'root', el favicon '/logo.png' y la carga del módulo '/src/main.jsx'. A partir de ese punto, 'src/main.jsx' monta 'App.jsx' dentro de 'React.StrictMode', lo que define el arranque del cliente Vite.

La composición principal reside en 'src/App.jsx'. Ese archivo configura 'BrowserRouter', define la pantalla de inicio pública y carga de forma diferida las páginas 'AdminLogin', 'AdminDashboard', 'EmprendedoraLogin', 'EmprendedoraDashboard' y 'Register' mediante 'React.lazy' y 'Suspense'. El feedback global se canaliza con 'ToastContainer', mientras que 'vite.config.js' segmenta el bundle en chunks de proveedor como 'firebase-vendor' y 'ui-vendor' para reducir el costo inicial de carga.

3.2.2 Navegación y control de acceso por rol

Las rutas públicas definidas por el frontend son '/', '/admin-login', '/emprendedora-login', '/emprendedora-register', '/admin-dashboard' y '/emprendedora-dashboard'. El acceso a los dashboards se valida en los componentes de página a partir de la información persistida en 'localStorage' ('userId', 'userName', 'userRole') y del rol normalizado por 'isAdminRole'.

Los formularios de autenticación de administradora y emprendedora permiten ingresar por correo o por cédula. 'authService.loginUser()' resuelve el correo final mediante la colección 'cedulas' o mediante un correo técnico con dominio 'takeemail.com', autentica con Firebase Auth y carga el documento 'users/{uid}' para determinar nombre, rol y permisos. El registro web de emprendedora utiliza 'registerUser()', crea la cuenta en Auth, genera 'users/{uid}', registra el mapeo en 'cedulas/{cedula}' cuando aplica y prepara 'entrepreneur_profiles/{uid}' si las reglas compartidas lo permiten.

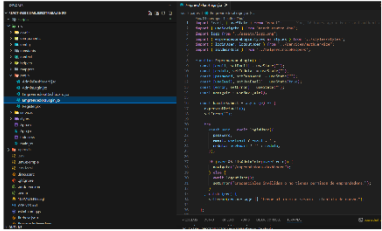
Manual Técnico | IEEE 1063 – 2001

6

APP WEB - 100 MUJERES TRABAJANDO

v 2.0

Figura 2. Login emprendedora con alternancia correo o cédula (/emprendedora-login).



3.2.3 Arquitectura modular del frontend web

La arquitectura real del frontend es modular y orientada a servicios, no una Clean Architecture estricta por feature. 'pages/' concentra las pantallas de alto nivel; 'components/' encapsula los módulos operativos reutilizados dentro de los dashboards; 'context/' centraliza el estado administrativo; y 'helpers/' junto con 'mappers/' mantienen las conversiones entre el contrato Firestore compartido y la interfaz web.

Este enfoque permite mantener compatibilidad con el código actual, donde varios componentes combinan UI, reglas de interacción y llamadas a servicios. Desde la perspectiva de mantenimiento, la capa de servicios ('src/services/') actúa como la frontera principal con Firebase, mientras que 'styles/' y los estilos inline resuelven la presentación sin un sistema CSS externo dominante.

3.2.4 Servicios principales e integraciones

- 'firebaseClient.js': valida la configuración 'VITE_FIREBASE_*', inicializa 'app', 'auth' y 'db', y aplica 'browserLocalPersistence'.
- 'authService.js': encapsula 'registerUser', 'loginUser' y 'logoutUser', persiste sesión en 'localStorage' y resuelve el fallback técnico por cédula.
- 'firebaseService.js': concentra lectura y escritura de catálogo, pedidos, historial, perfil emprendedor, usuarios y dashboard administrativo.

Manual Técnico | IEEE 1063 – 2001

7

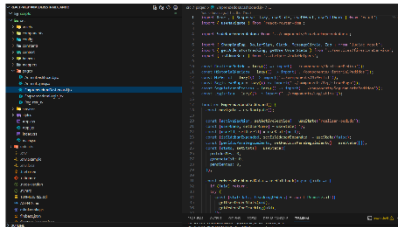
- 'cloudinaryService.js': carga imágenes directamente a Cloudinary usando un unsigned preset y devuelve URL segura, 'publicId' y tamaño del archivo.
- 'orderStatus.js': normaliza estados de pago, estados operativos y métodos de entrega para mantener compatibilidad con el contrato compartido entre web y móvil.
- 'AdminConten.js': agrega KPIs, actividad reciente y acciones administrativas, con refresco periódico y notificaciones visuales.
- 'authConfig.js' y 'mediaConfig.js': fijan el dominio técnico de autenticación y la estructura de carpetas Cloudinary ('products', 'profiles', 'payments').

3.2.5 Flujos representativos del sistema web

Los flujos funcionales observados en la rama 'firebase-migration' son los siguientes:

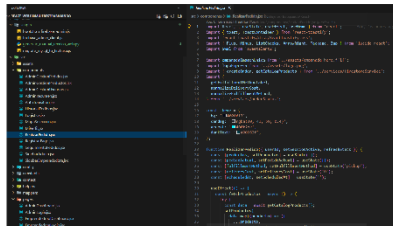
1. Login y registro web: acceso por correo o cédula, validación de credenciales en Firebase Auth y resolución de rol desde Firestore.

Figura 3. Dashboard de emprendedora (/emprendadora-dashboard).



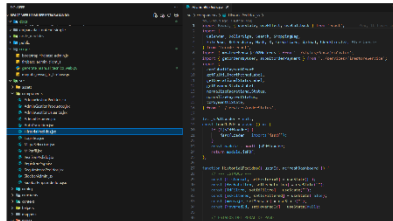
2. Catálogo y pedido: 'RealizarPedido' consulta 'products', arma el carrito, permite retiro o entrega, calcula 'deliveryCost', admite 'scheduledAt' opcional y crea el documento 'orders/{id}'.

Figura 4. Módulo de realización de pedido ('RealizarPedido').



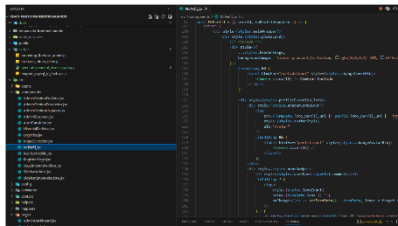
3. Historial y ticket: 'HistoriaPedidos' consulta pedidos por 'userId', filtra por fecha o identificador y genera ticket PDF con 'jspdf'.

Figura 5. Historial de pedidos con estados y acción de pago ('HistoriaPedidos').



4. Pago y comprobante: 'RegistrarPago' y el modal de historial oxigena imagen de comprobante, cargan el archivo a Cloudinary y actualizan el pedido a 'en_revision'.
5. Perfil emprendedora: 'MiPerfil' consulta 'users/{uid}' y 'entrepreneur_profiles/{uid}', permite editar descripción, lema, redes, portada y avatar, y aplica fallback seguro cuando las reglas compartidas restringen la escritura directa.

Figura 6. Perfil de emprendedora en modo edición ('MiPerfil').



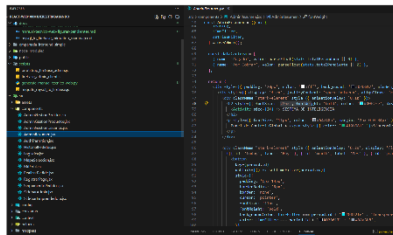
6. Seguimiento y logística: 'SeguimientoPedidos' refleja la secuencia 'recibido -> preparación -> listo -> entregado', mientras que 'Logística' ofrece selección visual de ubicación con Leaflet y OpenStreetMap.

3.2.6 Panel administrativo

El panel administrativo se publica en '/admin-dashboard' y encapsula su estado dentro de 'AdminProvider'. La shell de administración carga de forma diferida cuatro módulos principales: 'AdminResumen', 'AdminGestionPedidos', 'AdminGestionUsuarios' y 'AdminGestionProductos'.

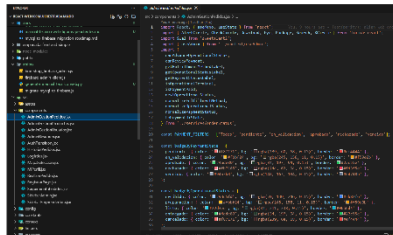
- Resumen: calcula 'totalRecaudado', 'totalPendiente', 'emprendedorasActivas', ventas por producto, actividad reciente e historico mensual con Recharts.

Figura 7. Dashboard administrativa de resumen ('AdminResumen').



- Gestión de pedidos: filtra por texto y estado de pago, visualiza comprobantes, aprueba o rechaza pagos, avanza el estado operativo, cancela pedidos, edita logística y exporta CSV operativos.

Figura 8. Gestión administrativa de pedidos ('AdminGestionPedidos').



APP WEB - 100 MUJERES TRABAJANDO

v 2.0

- Gestión de usuarios: consulta 'users', 'entrepreneur_profiles' y 'entrepreneur_locations', permite cambiar rol o estado y muestra señales de perfil, ubicación y visibilidad pública.
- Gestión de productos: realiza CRUD sobre 'products', activa o desactiva catalogo e integra carga de imágenes a Cloudinary.

3.2.7 Scripts de migración y soporte técnico

La aplicación web documentada no utiliza Cloud Functions como backend principal. En su lugar, el repositorio incorpora scripts Node de soporte para tareas administrativas fuera del navegador. 'scripts/migrate_mysql_to_firestore.js' migra datos legados desde MySQL hacia las colecciones compartidas de Firestore; 'scripts/bootstrap_firestore_admin.js' promueve cuentas de administradora mediante custom claims y sincronización del documento 'users/uid'; y 'scripts/firebase_admin_client.js' inicializa Admin SDK desde una service account o desde credenciales de entorno.

Estos scripts no participan en el runtime del frontend ni en el despliegue de Hosting. 'server.js' continua presente como backend legado monolítico y debe interpretarse como historial del proyecto, no como dependencia obligatoria del modo Firebase directo.

3.3 Configuración mínima

1. Ejecutar 'npm install' en la raíz del repositorio.
2. Crear '.env.local' a partir de '.env.example' e incorporar las claves 'VITE_FIREBASE_API_KEY', 'VITE_FIREBASE_AUTH_DOMAIN', 'VITE_FIREBASE_PROJECT_ID', 'VITE_FIREBASE_STORAGE_BUCKET', 'VITE_FIREBASE_MESSAGING_SENDER_ID', 'VITE_FIREBASE_APP_ID' y, si aplica, 'VITE_FIREBASE_MEASUREMENT_ID'.
3. Configurar 'VITE_AUTH_TECH_EMAIL_DOMAIN=fakeemail.com', 'VITE_CLOUDINARY_CLOUD_NAME', 'VITE_CLOUDINARY_UNSIGNED_PRESET', 'VITE_CLOUDINARY_FOLDER=app100mt' y mantener 'VITE_ENABLE_PAYMENT_EVENTS=false' mientras no exista aprobación manual de reglas compartidas.
4. Si se van a ejecutar scripts operativos, completar '.env' con 'FIREBASE_SERVICE_ACCOUNT_JSON' o 'FIREBASE_SERVICE_ACCOUNT_BASE64', 'ADMIN_EMAILS', 'ADMIN_ROLE' y, para migración, 'LEGACY_MYSQL_*'.
5. Autenticar Firebase CLI con 'firebase login' y seleccionar el proyecto con 'firebase use app100mujerestrabajando'.
6. Verificar que el trabajo se realice sobre 'firebase:migration' y que no se intente desplegar reglas o índices Firestore desde este repositorio web.

3.4 Ejecución

- Levantar el cliente Vite con 'npm run dev'.
- Acceder al servidor de desarrollo indicado por Vite, normalmente 'http://localhost:5173'.

Manual Técnico | IEEE 1063 – 2001

12

APP WEB - 100 MUJERES TRABAJANDO

v 2.0

- Validar ingreso de emprendedora y administradora con cuentas reales o de prueba alineadas al proyecto Firebase compartido.
- Usar 'npm run preview' solo después de generar build de producción cuando se requiera validar el artefacto 'dist/' localmente.
- En el modo Firebase directo no es necesario iniciar 'server.js' para operar la aplicación web normal.

3.5 Build y despliegue en Firebase Hosting

- Construir la versión de producción con 'npm run build'; el resultado se publica en 'dist/'.
- Desplegar la SPA con 'firebase deploy --only hosting' o con el atajo 'npm run deploy:hosting'.
- 'firebase.json' 'fj.a' 'public': 'dist' y un rewrite global hacia '/index.html', permitiendo recarga directa de rutas internas.
- 'firebase.json' anular el despliegue con el proyecto 'app100mujerestrabajando'.
- No ejecutar 'firebase deploy', 'firebase deploy --only firestore:rules' ni 'firebase deploy --only firestore:indexes' desde esta rama; el repositorio incluye un script bloqueante ('deploy:firestore:manual-review') para reforzar esa restricción.

4 ACCESO A LOS DATOS

El acceso a datos del sistema web se resuelve desde el navegador mediante Firebase Auth, Cloud Firestore y Cloudinary. La capa 'src/services/' actúa como frontera técnica del frontend y evita depender de un backend Express o de Cloud Functions en la operación cotidiana del sitio.

Las colecciones y estructuras compartidas usadas por la web son las siguientes:

- 'users': identidad operativa, rol, datos base y campos de perfil utilizados como fallback.
- 'cedulas': mapeo de documento hacia 'uid' y correo para soportar login por cedula.
- 'entrepreneur_profiles': perfil ampliado de la emprendedora, incluyendo descripción, toma, contacto y redes.
- 'entrepreneur_locations': ubicación operativa o pública asociada a la emprendedora; hoy se consulta para señales de visibilidad, pero su escritura web sigue pendiente de cierre.
- 'products': catálogo administrable de productos visibles en el flujo de pedidos.
- 'orders': pedidos con ítems, total, estado operativo, estado de pago, método de entrega, costo de entrega, programación y metadatos de comprobante.
- 'orders/{orderId}/payment_events': subcolección opcional prevista para auditoría de pagos, deshabilitada por defecto en la web actual.

Manual Técnico | IEEE 1063 – 2001

13

APP WEB - 100 MUJERES TRABAJANDO

v 2.0

4.1 Firebase Authentication y Cloud Firestore

La autenticación utiliza 'signInWithEmailAndPassword' y 'createUserWithEmailAndPassword', con persistencia local de sesión en navegador. La resolución por cédula consulta 'cedulas/{documentid}' y, cuando no existe un correo registrado, forma un correo técnico con el dominio configurado en 'VITE_AUTH_TECH_EMAIL_DOMAIN'. Una vez autenticado, el sistema consulta 'users/{uid}' para determinar rol, nombre y permisos de navegación.

- Las consultas activas de React Web no requieren índices compuestos adicionales a los ya cubiertos por el proyecto compartido.
- Las reglas e índices Firestore de este repositorio no deben desplegarse automáticamente; la decisión operativa es mantener hosting-only y aplicar cualquier cambio mediante merge manual con la base de Hutter.
- Si se desea habilitar escritura owner-write extendida en 'entrepreneur_profiles' o auditoría en 'payment_events', la aprobación debe hacerse sobre las reglas compartidas y no desde la rama web por sí sola.
- El contrato compartido de estados se normaliza en 'orderStatus.js', evitando divergencias entre 'paymentStatus', 'operationalStatus' y etiquetas de UI.

4.2 Cloudinary y verificación operativa

La gestión multimedia del sistema web se realiza desde 'cloudinaryService.js'. Las imágenes de productos, fotografías de perfil, portadas y comprobantes se suben a Cloudinary y su URL segura se persiste en Firestore. La carpeta base se define con 'VITE_CLOUDINARY_FOLDER', de donde derivan las rutas 'products', 'profiles' y 'payments'.

- La validación operativa de datos estructurados se realiza en Firebase Console, revisando el proyecto 'app100mujerestrabajando' y las colecciones compartidas consumidas por la web.

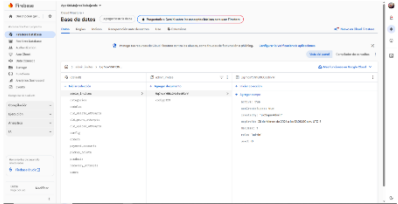
Manual Técnico | IEEE 1063 – 2001

14

APP WEB - 100 MUJERES TRABAJANDO

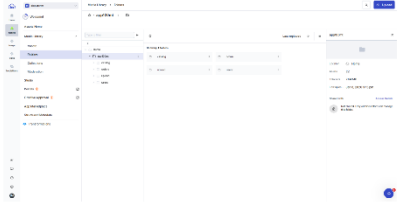
v 2.0

Figura 9. Colecciones compartidas en Firestore (Firebase Console): 'users', 'cedulas', 'products' y 'orders'.



- La validación de imágenes y comprobantes se realiza en Cloudinary Console -> Media Library, confirmando carpeta, fecha, URL y correspondencia con el pedido o perfil asociado.

Figura 10. Biblioteca multimedia en Cloudinary (Media Library): 'app100mt/products', 'app100mt/profiles' o 'app100mt/payments'.



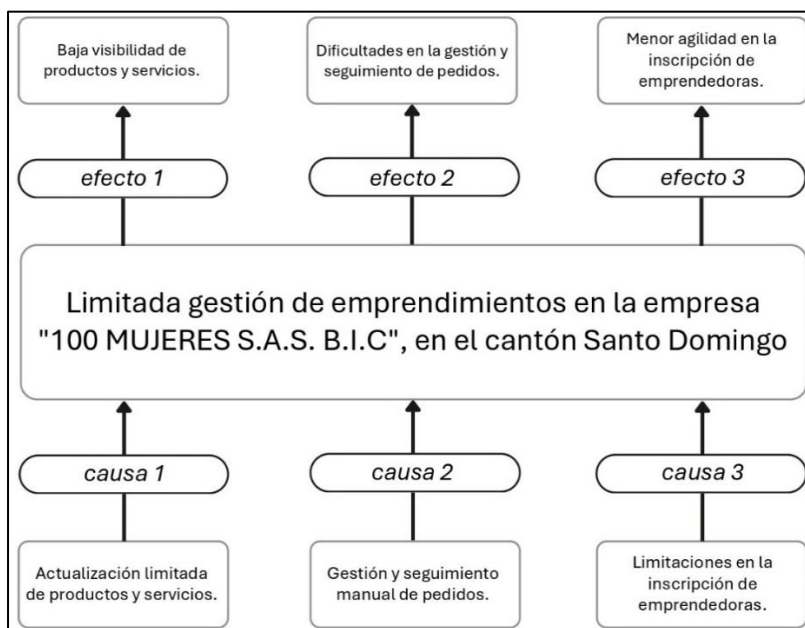
- El uso de unsigned preset simplifica la carga directa desde el navegador, pero debe endurecerse en producción mediante políticas de abuso, gobernanza de secretos y control de cuotas.

Manual Técnico | IEEE 1063 – 2001

15

APP WEB - 100 MUJERES TRABAJANDO	v 2.0
5 GLOSARIO	
• React: Biblioteca de interfaz utilizada para construir la SPA web. • Vite: Herramienta de desarrollo y build que sirve el frontend en modo local y genera el artefacto de producción. • SPA: Aplicación de página única que resuelve navegación interna sin recarga completa del documento. • React Router: Biblioteca de enrutamiento usada para publicar rutas como '/admin-dashboard' o '/emprendedora-login'. • Firestore Auth: Servicio de autenticación que valida registro, inicio y cierre de sesión. • browserLocalPersistence: Modo de persistencia de Firestore Auth que conserva la sesión en el navegador. • Cloud Firestore: Base de datos documental utilizada como persistencia principal del sistema web. • Firestore Hosting: Servicio de despliegue estático y hosting SPA utilizado para publicar 'dist'. • Cloudinary: Servicio de almacenamiento multimedia usado para imágenes de producto, perfil y comprobantes. • Lazy loading: Carga diferida de páginas o módulos con 'React.lazy' y 'Suspense'. • Vendor bundle: Chunk de dependencias de terceros generado por la configuración de build de Vite. • Service layer: Conjunto de servicios en 'src/services' que centraliza acceso a Firestore y Cloudinary. • AdminContext: Contexto React que agrega datos y acciones del dashboard administrativo. • LocalStorage: Almacenamiento del navegador usado para conservar identificadores básicos de sesión y rol. • Firestore CLI: Herramienta de línea de comandos usada para autenticación y despliegue hosting-only.	
6 REFERENCIAS	
IEEE: 1063-2001 - IEEE Standard for Software User Documentation [Internet]. Disponible en: https://ieeexplore.ieee.org/document/974401 Repositorio oficial: React-Web100MujeresTrabajando [Internet]. Disponible en: https://github.com/marcogamer555/React-Web100MujeresTrabajando.git	
Manual Técnico IEEE 1063 – 2001	16

Anexo 11. Árbol del problema



Anexo 12. Recodificación del instrumento de recolección de datos

Recodificación	Respuestas
Edad	
1	18-25
2	26-30
3	30-40
4	40-50
5	50 o más
Genero	
1	Femenino
2	Masculino

Discapacidad	
1	Auditiva
2	Visual
3	Física
4	Otro
5	Ninguna

Recodificación	Escala de Likert
Necesidad	
4	Muy necesario
3	Necesario
2	Medianamente necesario
1	Poco necesario
0	Innecesario
Frecuencia	
4	Siempre
3	Casi siempre
2	A veces
1	Casi nunca
0	Nunca
Complicación	
4	Muy complicado
3	Complicado
2	Neutral
1	Poco complicado
0	Nada complicado
De acuerdo	
4	Totalmente de acuerdo
3	De acuerdo
2	Neutral
1	Poco de acuerdo
0	Nada de acuerdo
Seguridad	
4	Totalmente segura
3	Segura
2	Moderadamente segura
1	Poco segura
0	Nada segura
Satisfacción	
4	Totalmente satisfecha
3	Satisfecha
2	Medianamente satisfecha
1	Poco satisfecha
0	Nada satisfecha
Facilidad / Dificultad	
4	Muy fácil
3	Fácil
2	Normal
1	Difícil
0	Muy difícil

Anexo 13. Evidencia de la reunión de planificación del sprint

Entrevista dirigida a la gerente de "100 MUJERES S.A.S. B.I.C" Pregunta 1: ¿Qué funciones considera más importantes para garantizar los pedidos y seguimiento de estos? Respuesta: Bueno, muchas gracias. Mi nombre es Ximena Toro. Soy gerente del proyecto 100 Mujeres trabajando. "100 Mujeres Trabajando" es una compañía SASBIC creada hace más de un año en el cantón Santo Domingo, que cuenta eh con todos y cada uno de los parámetros legales aprobados por la Superintendencia de Compañías y totalmente registrado. Los parámetros más importantes que necesita el proyecto en el tema de los pedidos, es justamente de que las compañeras que forman parte del proyecto sin mujeres trabajando tienen que hacer los pedidos siempre con antelación, para que al siguiente día sus pedidos puedan ser entregados de manera organizada, de manera ordenada y también contando con todos los registros de seguridad identidad también del proyecto y para que puedan tener ellas el producto en buen estado a la hora que lo puedan retirar. Pregunta 2: ¿Qué medidas considera necesarias para validar los comprobantes de pago subidos por las emprendedoras? y para así, garantizar la seguridad de esta información. Respuesta: Bueno, nos gustaría que en información se pueda tener el acceso para que puedan subir la captura o la fotografía de la transferencia o del depósito y que también se pueda colocar el número del documento. No sé si es que de pronto para las mujeres que no tengan el portal web de hacer la transferencia de depósito, que también exista ahí alguna información donde digan pago en efectivo o pago en transferencia. Pregunta 3: Considerando emprendedoras con diferentes niveles de educación, ¿Qué criterios de diseño, en cuanto a la aplicación móvil, considera se han tomar en cuenta? Respuesta: Bueno, tienen que tomarse en cuenta primero los colores, que la aplicación maneje los colores del proyecto, que la aplicación sea bastante familiar para ellas, sea muy accesible, que tenga mucho color. ¿En qué sentido? en lo que es las imágenes, fotografías, videos, que sea muy adaptable para que ellas la puedan manejar de la mejor manera, que sea una aplicación no complicada, que sea super accesible, amigable.	Pregunta 4: ¿Cuál tan familiarizado se encuentra con el uso de aplicaciones web? Respuesta: Bueno, yo como gerente del proyecto cien mujeres estoy familiarizada poder trabajar o conocer las apps que son herramientas importantes, pero nos va a ayudar muchísimo esta aplicación para el proyecto y para que las mujeres también sea sumamente fácil el manejo y el acceso. También nos gustaría que en el momento que esté lista la aplicación poder tener una socialización de la App con las mujeres emprendedoras. Pregunta 5: ¿considera que el desarrollo de una aplicación web y móvil de acuerdo, a sus requerimientos puede fortalecer la gestión de operaciones y logística en su empresa? Respuesta: Claro que sí. Pregunta 6: ¿considera que la aplicación de nuevas tecnologías puede facilitar la gestión de talento humano? Respuesta: Sí. Pregunta 7: ¿está de acuerdo en almacenar toda la información de sus emprendedoras y productos en una aplicación tecnológica? Respuesta: Sí. Pregunta 8: ¿Cómo guarda la información de las emprendedoras? Respuesta: De manera manual y a través de una tabla de Excel. Pregunta 9: ¿Cuál es el proceso para registrar la información de las emprendedoras? Respuesta: Hasta ahora lo estamos haciendo de manera manual cuando se acercan a la información y aquí a la gerencia del proyecto 100 mujeres a hacer parte y nosotros le hacemos llenar una ficha social de manera manual y a través de la computadora. Pregunta 10: ¿Qué medidas de control de calidad considera para asegurar la entrega de pedidos? Respuesta: Los pedidos tienen un control en el sentido de que primero van con: se ha elaborado con la materia prima del día de manera fresca y se entrega con las medidas de sanidad, de calidad y con la identificación del proyecto 100 mujeres.
Pregunta 11: ¿Qué cambios cree usted que se deben realizar para mejorar la gestión de pedidos? Respuesta: ¿Qué cambios? Yo creo que de pronto la aplicación que podría ser un cambio fundamental para nosotras para que las mujeres puedan tener la mayor facilidad de hacer los pedidos. Otra cosa que también nos va a ayudar porque la aplicación va a tener imágenes y fotografías y eso también ayudará para ver el catálogo de los productos que tenga el proyecto para que las mujeres puedan pedir y puedan solicitar. Pregunta 12: ¿La empresa tiene planes de incrementar la oferta de productos? Respuesta: Sí, vamos, queremos implementar el tema de la yuca rayada. Queremos implementar el tema de los chifles también y de las empanadas cocteleras. Pregunta 13: ¿pueden las emprendedoras vender otros productos además de la oferta de la empresa? Si, las emprendedoras pueden acoplar más productos al que ellas ya venden. Respuesta: La idea es que las personas o las mujeres que ingresen al proyecto 100 mujeres no se queden solamente con el producto del proyecto, sino que también puedan implementar más productos para que puedan generar mayor cantidad de ventas. Pregunta 14: ¿Cómo está constituida legalmente la empresa? Respuesta: La empresa está constituida a través de una compañía SASBIC legalmente constituida en el cantón Santo Domingo. Pregunta 15: ¿Qué licencias o permisos son necesarios para garantizar el trabajo de las emprendedoras? Respuesta: Eh, bueno, licencias permisos en sí municipal no necesitaría porque la mayoría de las mujeres que están trabajando con los coches necesitan solamente previa autorización de la parte privada y las demás lo hacen directamente desde sus domicilios, entonces no necesitan permiso municipal. Pregunta 16: ¿Cómo evalúa la empresa la coordinación y apoyo entre emprendedoras? Respuesta: Bueno, todavía no hemos hecho una evaluación en sí, lo que estamos nosotros queriendo retomar el tema de la capacitación y la motivación que las hacemos de manera permanente y hoy con la aplicación que queremos después es hacer una socialización de la app para que ellas puedan familiarizarse y poder vender mucho más.	Pregunta 17: ¿se da algún tipo de apoyo inicial a las emprendedoras? Respuesta: Sí, más que apoyo es una motivación que nosotros es hacemos en el momento de la inscripción. Cuando ellas se inscriben al proyecto sin mujeres trabajando, nosotros le hacemos llenar una ficha social para poder tener una base de datos de ellas, de dónde son, para saber su situación económica, su situación personal, si es soltera, si tiene hijos, etcétera, si depende económicamente de alguien. Y nosotros le entregamos es una identificación totalmente gratuita para ella. le entregamos es una camiseta, una gorra y un canguro del proyecto. Pregunta 18: ¿Cómo se realiza la distribución de las emprendedoras para asignarles las zonas de venta? Respuesta: Bueno, nosotros no focalizamos lugares específicos para ponerlas a ellas en la distribución. Nosotros hacemos un recorrido de la ciudad o del cantón e identificamos ya a mujeres que están en zonas establecidas, que pueden ser en sus lugares de trabajo, que pueden ser desde sus casas o pueden ser mujeres que ya tienen su puesto de trabajo durante muchos años y que necesitan un incentivo o que necesitan del valor del proyecto para que ellas puedan salir adelante. Por ejemplo, hay mujeres que desde sus tiendas del barrio comercializan las bandejas de empanadas, entonces ellas desde ahí ya hacen su nicho de ventas. O hay mujeres que son las otras que tienen sus coches, los cochecitos, pero ya son mujeres que están establecidas en lugares específicos, en puntos fijos de la ciudad o en locales comerciales donde ellas ya venden ahí durante muchos años, pero ahí es donde nosotros les damos la herramienta de trabajo, que es el coche, como una medida de identificación, pero también como para que ellas puedan exponer sus productos de la mejor manera y para que puedan ser más visibles pues los productos del proyecto más los adicionales que ellas venden.

[Evidencia de reunión de planificación de sprint](#)

Anexo 14. Evidencia de encuesta

[Evidencia encuesta](#)

Anexo 15. Informe del Turnitin

TTG-V7

por Marco Javier Silva Avila

Fecha de entrega: 24-may-2026 12:16p. m. (UTC-0500)
Identificador de la entrega: 2968355759
Nombre del archivo: TTG-TI-D_AZ-SILVA-202502-VF.docx (70.88M)
Total de palabras: 40300
Total de caracteres: 233458

TTG-V7

INFORME DE ORIGINALIDAD

5%	5%	2%	1%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

Marco Javier Silva Avila

TTG-F1

TTG - V2.

Trabajo Titulación Grado 202501-1

PUCE SANTO DOMINGO MOODLE

Detalles del documento

Identificador de la entrega
trnoid::1:3578070508

171 páginas

Fecha de entrega
24 may 2026, 6:09 p.m. GMT-5

29.519 palabras

Fecha de descarga
24 may 2026, 6:34 p.m. GMT-5

168.996 caracteres

Nombre del archivo
TTG-TI-D_AZ-SILVA-202502-VF_-_1.docx

Tamaño del archivo
37.0 MB

turnitin

Página 1 de 173 - Portada

Identificador de la entrega trnoid::1:3578070508

turnitin

Página 2 de 173 - Descripción general de la escritura con IA

Identificador de la entrega trnoid::1:3578070508

*% detectado como IA

La detección de IA incluye la posibilidad de que haya falsos positivos. Aunque cierto texto en esta entrega se generó probablemente con IA, los puntajes inferiores al umbral del 20 % no aparecen porque tienen una mayor probabilidad de falsos positivos.

Precaución: Se necesita revisión.

Es esencial comprender los límites de la detección de IA antes de tomar decisiones acerca del trabajo del estudiante. Te alentamos a obtener más información acerca de las funciones de detección de IA de Turnitin antes de usar la herramienta.

Aviso legal

Nuestra evaluación de escritura con IA está diseñada para ayudar a los académicos a identificar texto que podrían haberse preparado mediante una herramienta de IA generativa. Es posible que nuestra evaluación de escritura con IA no siempre sea precisa (existe la posibilidad de que identifique erróneamente redacciones probablemente generadas por humanos como generadas por IA, y redacciones probablemente generadas por IA como generadas por humanos), por lo que no debe usarse como único fundamento para aplicar sanciones a un estudiante. Para determinar si es un caso de deshonestidad académica, se necesita de un escrutinio mayor y el juicio humano, junto con la aplicación de las políticas académicas específicas de la organización.

Marco Javier Silva Avila

TTG-F2

TTG - V2

Trabajo Titulación Grado 202501-1

PUCE SANTO DOMINGO MOODLE

Detalles del documento

Identificador de la entrega
trnoid::1:3578071910

113 páginas

Fecha de entrega
24 may 2026, 6:18 p.m. GMT-5

12.135 palabras

Fecha de descarga
24 may 2026, 6:46 p.m. GMT-5

72.464 caracteres

Nombre del archivo
TTG-TI-D_AZ-SILVA-202502-VF_-_2.docx

Tamaño del archivo
34.2 MB

turnitin

Página 1 de 115 - Portada

Identificador de la entrega trnoid::1:3578071910

turnitin

Página 2 de 115 - Descripción general de la escritura con IA

Identificador de la entrega trnoid::1:3578071910

*% detectado como IA

La detección de IA incluye la posibilidad de que haya falsos positivos. Aunque cierto texto en esta entrega se generó probablemente con IA, los puntajes inferiores al umbral del 20 % no aparecen porque tienen una mayor probabilidad de falsos positivos.

Precaución: Se necesita revisión.

Es esencial comprender los límites de la detección de IA antes de tomar decisiones acerca del trabajo del estudiante. Te alentamos a obtener más información acerca de las funciones de detección de IA de Turnitin antes de usar la herramienta.

Aviso legal

Nuestra evaluación de escritura con IA está diseñada para ayudar a los académicos a identificar texto que podrían haberse preparado mediante una herramienta de IA generativa. Es posible que nuestra evaluación de escritura con IA no siempre sea precisa (existe la posibilidad de que identifique erróneamente redacciones probablemente generadas por humanos como generadas por IA, y redacciones probablemente generadas por IA como generadas por humanos), por lo que no debe usarse como único fundamento para aplicar sanciones a un estudiante. Para determinar si es un caso de deshonestidad académica, se necesita de un escrutinio mayor y el juicio humano, junto con la aplicación de las políticas académicas específicas de la organización.

Anexo 16: Certificado de participación.

 Pontificia Universidad Católica del Ecuador	SANTO DOMINGO
Otorga el siguiente CERTIFICADO	
A: SILVA AVILA MARCO JAVIER	
En reconocimiento a su destacada contribución en el ámbito de la Investigación e Innovación, mediante su participación activa con el tema: APLICACIÓN WEB Y MÓVIL PARA LA GESTIÓN DE EMPRENDIMIENTOS EN LA EMPRESA 100 MUJERES S.A.S. B.I.C, DEL CANTÓN SANTO DOMINGO Su compromiso y excelencia representan un ejemplo inspirador para nuestra institución y futuras generaciones de profesionales.	
 Mg. Franklin Carrasco Ramírez COORDINADOR DE LA CARRERA DE SISTEMAS DE INFORMACIÓN	
QUITO - AMAZONAS - AMBATO - ESMERALDAS - IBARRA - MANABÍ - SANTO DOMINGO	

 Pontificia Universidad Católica del Ecuador	SANTO DOMINGO
Otorga el siguiente CERTIFICADO	
A: DÍAZ BARREIRO LEONARDO ANDRÉS	
En reconocimiento a su destacada contribución en el ámbito de la Investigación e Innovación, mediante su participación activa con el tema: APLICACIÓN WEB Y MÓVIL PARA LA GESTIÓN DE EMPRENDIMIENTOS EN LA EMPRESA 100 MUJERES S.A.S. B.I.C, DEL CANTÓN SANTO DOMINGO Su compromiso y excelencia representan un ejemplo inspirador para nuestra institución y futuras generaciones de profesionales.	
 Mg. Franklin Carrasco Ramírez COORDINADOR DE LA CARRERA DE SISTEMAS DE INFORMACIÓN	
QUITO - AMAZONAS - AMBATO - ESMERALDAS - IBARRA - MANABÍ - SANTO DOMINGO	