PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

CARRERA DE SISTEMAS DE INFORMACIÓN

TRABAJO DE TITULACIÓN

TEMA:

APLICATIVO MÓVIL PARA PROMOCIONAR PRODUCTOS ARTESANALES CASO DE ESTUDIO “TEJIENDO MI FAJA, BORDANDO MI BLUSA” DE FUNDACIÓN FE Y ALEGRÍA

AUTORES

RICHARD ALEXANDER MONTALVO CHUQUISALA

SERGIO DAVID OLALLA MASACHE

DIRECTOR

ING. FABIÁN IGNACIO DE LA CRUZ

QUITO DM, 2022

DEDICATORIA 1

El presente trabajo de titulación se lo dedico a las siguientes personas:

A mi mamá y mi abuela

Por brindarme su apoyo incondicional y haberme aconsejado en los momentos difíciles. Son mi fuente de inspiración, siempre haciendo todo lo que estuvo a su alcance y más para que yo pudiese triunfar en mi formación profesional.

A mi papá

Por ser comprensivo, cariñoso y ser aquella persona que me relata sus anécdotas de vida para hacerme reflexionar en mis acciones.

A mi hermano

Por siempre ser la persona con la que, he compartido muchas alegrías y tristezas, te has convertido en una razón para esforzarme más cada día y ser un buen modelo a seguir.

Gracias a todos.

Sergio David Olalla Masache

AGRADECIMIENTO

Quisiera expresar mi agradecimiento a mi compañero Richard Montalvo y mi tutor Fabián de la Cruz porque el trabajo en equipo es una habilidad que se desarrolla con personas que persiguen un mismo sueño y sin ellos esta investigación no hubiera sido posible. También, agradezco a mis profesores de carrera porque sin su dedicación y experiencia no hubiera obtenido los conocimientos para transfórmame en un profesional que quiere servir a la sociedad. De igual manera, a mis compañeros de carrera, con los cuales he compartido vivencias para mirar con otras perspectivas las circunstancias que se nos presentaron días tras día.

DEDICATORIA 2

Dedico mi familia

Por ser el motor y guía emocional a través del desarrollo de este escrito. Sin su apoyo incondicional no hubiese sido posible pasar a través de todo el proceso.

A mi papa

Por eso un guía y modelo de admiración en el desarrollo de este escrito. Por enseñarme a superar cualquier obstáculo y sobre todo a pensar con cabeza fría cada movimiento.

A mi hermana y madre

Por ayudarme a pensar de manera creativa y darme ánimos en los momentos adversos cuando toda parecía desmoronarse. Por eso ellas mismas cuando no era necesario.

Richard Alexander Montalvo Chuquisala

AGRADECIMIENTO

Me gustaría agradecer primeramente a mi compañero, socio y casi colega Sergio Olalla. Sus ideas, pensamientos y propuestas sin duda proporcionaron un antes y después no solo en el desarrollo de este escrito sino en mi persona y a nivel profesional. Además, me gustaría agradecer al nuestro tutor Fabian de la Cruz el cual a pesar de las circunstancias ha sabido ser un mentor y guía pragmático y a la vez muy capaz. Agradezco igualmente a mis profesores en los cuales he obtenido no solo mucho conocimiento técnico sino valores humanos y éticos. Finalmente agradezco a mis compañeros de curso los cuales igualmente me han sabido dar su amistad sin compromiso, carisma y momentos inolvidables que siempre se quedaran impresos en mi memoria.

RESUMEN

Dentro de las metodologías de desarrollo de software Scrum es una de las más aplicadas debido a su versatilidad, resiliencia y enfoque en la mejora continua del valor agregado. De esa forma el presente proyecto tiene como meta el desarrollo de un prototipo de un aplicativo móvil con propósito de mercadotecnia para las plataformas que ejecuten Android. Para la correcta implementación de la metodología Scrum se tiene los Sprints los cuales son desarrollos incrementales que poco a poco llevan al producto final que en este caso es un aplicativo móvil. Primeramente, se tiene en cuenta la correcta toma de requerimientos y asignación de roles del equipo para luego pasar al diseño y análisis exhaustivo de cómo se puede llevar a cabo lo solicitado por el cliente o en general por el Stakeholder. Se propuso usar una metodología ágil ya que se adapta adecuadamente a los requerimientos cambiantes que solicitaba el Stakeholder. Finalmente se pudo desarrollar el aplicativo con las funcionalidades claves solicitadas. En las pruebas unitarias igualmente fueron satisfactorias mostrando la efectividad de la aplicación de la metodología. Dentro del proceso de desarrollo se descubrió que el uso del Flutter combinado con arquitecturas limpias y una correcta gestión de estados igualmente ayudaron a finalizar el proceso en el tiempo razonable.

Palabras clave: Flutter, Scrum, Desarrollo, Android, Gestión de estados

Abstract

Within the development methodologies Scrum is one of the most applied in Software Development due to its versatility, resiliency, and approach in continuous improvement cycle. Having said that, the present project has as goal the development of a prototype of a mobile app with purpose of marketing for android platforms. A correct implementation of Scrum methodology implies Sprints which are incremental developments that slowly get to the final product, which in this case is mobile app. Firstly, it must be taken in count the proper engineering of requirements and role assignation of the team. Consequently, it follows and exhaustive analysis and design about how to implement the requested by the client by the

stakeholder. It was proposed to use agile methodology since it adapts smoothly to the changing requirements requested by the stakeholder. Finally, the application could be developed with the requested key functionalities. In the unit tests they were also satisfactory, showing the effectiveness of the methodology. Within the development process, it was discovered that the use of Flutter combined with clean architectures and correct state management also helped to finish the process in a reasonable time.

Keywords: Flutter, Scrum, Development, Android, State management

ÍNDICE

Contenido

ÍNDICE DE FIGURAS, GRÁFICOS Y TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I: INTRODUCCIÓN

1.1. JUSTIFICACIÓN

Actualmente existen proyectos de vinculación con la comunidad y fortalecimiento cultural en Ecuador. En este contexto el proyecto denominado “Tejiendo Mi Faja, Bordando Mi Blusa” el cual es impulsado por la Fundación Fe y Alegría, y busca la confección de prendas de vestir ancestrales, bisutería, entre otros, para vincular a las comunidades de escasos recursos en actividades de empoderamiento cultural. Este tipo de proyectos tiene dos inconvenientes. Por un lado, no cuenta con suficiente visibilidad en los medios modernos, y, por otro lado, los integrantes de esos proyectos no tienen los medios, ni el conocimiento para empezar a publicitarse y comercializar los productos con ayuda de la tecnología. Proyectos como este existen en el país, ya sean comunitarios o simplemente negocios familiares que están perdiendo una gran oportunidad al ser excluidos del mundo digital.

1.2. Planteamiento del problema

El proyecto “Tejiendo mi faja, bordando mi blusa” no tiene un sistema de promoción efectivo que aproveche las nuevas tendencias en marketing digital para mejorar su visibilidad y aumentar sus ventas.

El trabajo de titulación consistió en el desarrollo de una plataforma de marketing para la promoción del proyecto “Tejiendo mi Faja, Bordando Mi Blusa” de la Fundación Fe y Alegría en Guamote. El proceso de desarrollo de la plataforma incluye la toma de requerimientos, aplicación de metodología Scrum, diseño de interfaces gráficas enfocadas al usuario, modelo arquitectónico de la aplicación y pruebas de usabilidad entre otros. Todo con el fin de solucionar el caso de estudio presente.

Se plantea la siguiente pregunta principal:

¿Cómo y por qué automatizar los procesos de mercadotecnia para el proyecto “Tejiendo mi Faja, Bordando Mi Blusa”?

Preguntas secundarias:

- ¿Qué tecnologías de desarrollo, marcos de trabajo se basará el actual proyecto de titulación?
- ¿Cuál es la solución técnica para los problemas de diseño y alcance del aplicativo de marketing?
- ¿El aplicativo móvil cumplió total o parcialmente los objetivos para el cual fue desarrollado o pensado?

1.3. Objetivo General

Desarrollar una aplicación móvil que permita publicitar y controlar los procesos de registro de productos, de una forma fácil e interactiva para el usuario, solucionando la falta de visibilidad a los microempresarios de las comunidades apartadas.

1.4. Objetivos Específicos

- Realizar un análisis de los requerimientos funcionales y no funcionales necesarias para el aplicativo.
- Aplicar la metodología SCRUM para la implementación de funcionalidades.
- Diseñar la interfaz a utilizar en la aplicación para mejorar la navegación del usuario.
- Desarrollar las pruebas e implementación de la aplicación móvil.

1.5. Alcance

El trabajo de titulación plantea como alcance desarrollar un prototipo funcional de plataforma de marketing, donde el pequeño artesano tenga la posibilidad de publicitar sus productos elaborados a mano, definiendo como público objetivo de estos productos a mujeres y niños.

El tiempo para desarrollar este aplicativo es de 3 meses y contará con las siguientes funcionalidades:

- **Catálogo de productos:** Se presentará los diferentes tipos de productos ofrecidos por los artesanos de una forma organizada e interactiva.
- **Cambio de idioma español/inglés:** La aplicación tendrá un botón que permita al cliente cambiar de idiomas a los componentes visuales para ampliar el público objetivo del aplicativo.
- **Registro de usuario:** La primera vez que el cliente desee registrar un producto o realizar un pedido se desplegará una pantalla de registro donde el cliente ingrese su información personal, tanto para coordinar el pedido o para controlar quien ha realizado una publicación de un producto.
- **Botón de pedidos:** En la selección de un producto, la aplicación desplegará un mensaje de solicitud de pedido al cliente, se enviará la petición de un pedido hacia el departamento de ventas de Fe y Alegría.
- **Edición de cuenta de usuario:** La aplicación contará con validación de campos en el registro de información de usuario para que la información sea clara y no entorpezca el proceso de venta. También permitiendo que en caso de ser necesario el usuario o administrador modifique su información.

CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

2. Herramientas de Desarrollo

En el siguiente capítulo se describirán las herramientas elegidas para diseñar el aplicativo teniendo en cuenta que debe ser interactivo y fácil de manejar. Por tanto, para el software se describirá las características de Flutter SDK y Firebase, los pasos y artefactos necesarios para implementar la metodología SCRUM, el rol de marketing digital para promocionar los productos y los tipos de diagramas que explicarán el funcionamiento del aplicativo.

2.1. Flutter SDK

Yuliya Datsyuk (2021) señala que Flutter simplifica el trabajo para los desarrolladores, ya que, en el pasado, se necesitaba crear diferente código base tanto para Play Store (Android) y otro para Play Market (IOS), manteniendo dos bases de código separadas. Además, existía problemas al ajustar la interfaz de usuario a todos los tamaños de pantalla posibles.

En este contexto surge Flutter conjunto de herramientas de interfaz de usuario de Google para crear aplicaciones compiladas de forma nativa para dispositivos móviles, web y de escritorio a partir de una única base de código. Flutter utiliza el lenguaje de programación Dart y una tecnología basada en widgets lo cual puede modificar o personalizar las interfaces con facilidad.

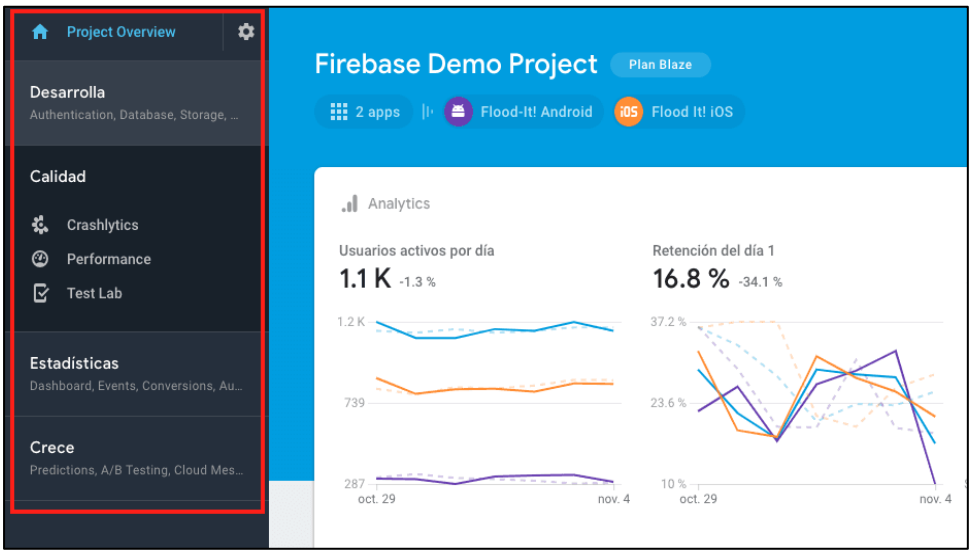
2.2. Base de datos Firebase

Es una plataforma de desarrollo conocida originalmente por su base de datos en tiempo real que sigue siendo, en esencia, una base de datos clave-valor de múltiples nodos optimizada para sincronizar datos, a menudo entre los teléfonos inteligentes y el almacenamiento centralizado en la nube.

Firebase es esencialmente una base de datos NoSQL alojada en la nube que almacena datos como JSON, por sus siglas JavaScript Object Notation se trata de un formato de intercambio de información entre aplicaciones de forma rápida y liviana, además de que su lectura expresiva resulta fácil tanto para personas como para máquinas.

Una de las principales ventajas de la base de datos en tiempo real de Firebase es que funciona sin conexión mediante el uso de la memoria caché local en el dispositivo para almacenar los cambios realizados.

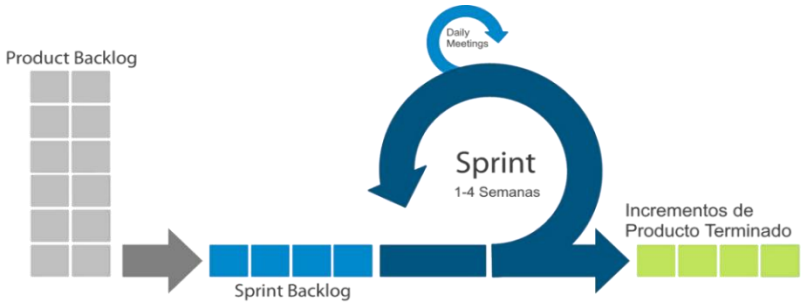
Figura 1: *Presentación del Dashboard de Firebase*



Nota. La imagen fue obtenida de (Google Analytics For Firebase: La solución de reporting para apps mobile, 2018).

2.3. Metodología Scrum

Figura 2: *Etapas de la Metodología SCRUM*



Nota. La imagen fue obtenida de (Kaizenia, 2020)

Para el desarrollo del aplicativo móvil se utilizó la metodología ágil con el marco de trabajo SCRUM, ya que permite dividir el proyecto en fases que deben completarse en pequeños intervalos de tiempo, creando bucles donde se recopila e integran los comentarios de los clientes optimizando el aplicativo.

Se puede identificar las siguientes etapas:

- Etapa de Planificación: Se especifica las tareas a realizar, similar a un cronograma de actividades.
- Etapa de desarrollo: Con la información obtenida en la fase de planificación se empieza a implementar las tareas especificadas en el calendario.
- Retrospectiva y revisión del sprint: A través de reuniones con el interesado se mostrarán los resultados del sprint. Se plantean las siguientes preguntas:
  - ¿Qué hicimos bien?, ¿Qué debemos mejorar?, ¿Qué se debe dejar de hacer?
- Entrega: Esta fase se centra en proporcionar al cliente los entregables finales.

2.4. Componentes de Metodología

Artefactos:

- **Product Backlog:** Una lista de funcionalidades que tendrá el aplicativo, debe ser presentada en forma de historias de usuario y dar a conocer los detalles del negocio.
- **Sprint Backlog:** Se trata de un conjunto de funcionalidades escogidas del Product Backlog y que deben ser realizadas en un Sprint.

Reuniones:

- **Reunión Previa:** Se trata de la definición de tareas una vez realizada la toma de requerimientos, estas tareas se ordenan por prioridades y esto se registra en el artefacto Product Backlog.
- **Planificación del Sprint:** Se determina cual es el trabajo específico por realizar y sus objetivos eligiendo elementos de Product Backlog.

Roles:

- **Scrum Master**: Se asegura que todos los eventos de Scrum se cumplan y proveen técnicas para la definición efectiva de la meta del producto.
- **Product Owner**: Profesional enfocado en entregar el mejor producto posible desarrollando una visión del proyecto a desarrollar.
- **Equipo de desarrollo:** Encargados de la programación y cumplir con las responsabilidades asignadas por el Scrum Master.
- **Stakeholder:** Interesados del proyecto que conocen los detalles del negocio y definen los requerimientos del proyecto.

2.5. Herramientas de Scrum

- **Tablero Kanban:** Es una herramienta que permite al equipo de trabajo enfocarse en las tareas que deben ser completadas, está compuesto por columnas etiquetadas con tres estados: “Pendiente”, “En progreso”, “Concluido” y una última sección marcando el espacio para las historias de usuario. Esta herramienta es muy utilizada ya que brinda una orientación visual organizada, es decir que las tareas van cambiando a un estado completado a medida que se avanza con el proyecto.

2.6. Marketing Digital

Las herramientas de marketing digital muestran mayor tendencia de crecimiento hoy en día, convirtiéndose en parte esencial de cada negocio de acuerdo con la siguiente cita “Marketing es una parte clave de todas las decisiones comerciales de una empresa, desde el desarrollo de productos y fijación de precios hasta las relaciones públicas” (Kingsnorth, 2019). El apalancamiento de las tecnologías de la información para la mercadotecnia cambió desde la introducción de las tecnologías de la información. Las herramientas digitales para posicionamiento de SEO, los chatbots, los embudos de ventas optimizados con IA son el actual presente del marketing digital. El objetivo del marketing digital es obtener la mayor parte de la atención de los clientes para que realicen acciones determinadas. Comprar un producto, consumir un servicio o compartir sus experiencias positivas del producto con su círculo social. Esto requiere un nivel de confianza tanto del producto, el medio de promoción y el origen del producto o servicio que puede ser una compañía u organización.

2.7. Diagramas UML

Los diagramas UML permiten realizar representaciones esquemáticas del funcionamiento de un determinado software. El uso de representaciones visuales permite comprender de mejor manera los aciertos o errores en los procesos que se están realizando.

Existen varios tipos de diagramas de UML, sin embargo, para este aplicativo se consideró necesario utilizar el diagrama de clases para explicar el diseño del aplicativo y los diagramas de casos de uso que describirán los siguientes elementos:

- **Actores:** Aquellos que interactúan con la aplicación, puede ser una persona particular, una empresa o una aplicación externa.
- **Relaciones:** Muestra como la interacción entre actores y el sistema.

CAPÍTULO III: PLANIFICACIÓN

3. Análisis y Diseño del Aplicativo Móvil

A continuación, se presentará el capítulo de la elaboración del diseño del sistema, teniendo como punto de partida los requerimientos dados por el cliente, luego se describirá la arquitectura de software elegida. Por último, se explicará la funcionalidad que se tiene planeada para satisfacer los requerimientos y el modelo de la base de datos.

3.1. Clasificación de Requerimientos

Para la toma de requerimientos se empleó la técnica de entrevistas, por ende, se produjeron dos reuniones entre el equipo de desarrollo y el Stakeholder Francisco Rodríguez. La primera donde se tomó nota de las funcionalidades que debería tener el aplicativo y se las clasificó para simplificar el desarrollo en el tiempo estipulado. Y la segunda reunión donde se presentó avances de la arquitectura del aplicativo y algunos requerimientos no funcionales deseables del aplicativo.

Por tanto, se presenta a detalle la clasificación de requerimientos que se consideró necesaria en la implementación del aplicativo.

Primeramente se recopila a los requerimientos críticos, que son aquellos que se implementan de forma obligatoria en este trabajo de titulación, luego se tiene los requerimientos deseables que se refieren a aquellos requerimientos que se implementarán una vez que hayan finalizado con requerimientos críticos y si existe tiempo sobrante de desarrollo y por último se recopila a los requerimientos opcionales, aquellos que se considerarán a implementar a futuro, pero no son abarcados en este trabajo de titulación.

Requerimientos Críticos

- Pantalla de Inicio
- Cambio de idioma español/inglés
- Gestión de Productos, desplazamiento por el aplicativo visualizando todos los productos registrados.
- Registro de Clientes, para publicación y visualización de productos.
- Edición de registro de información de vendedor, para evitar errores de tipeo o deseo de actualización de la información.
- Almacenamiento de Información de proveedores y productos en la base de datos.
- Registro de Imágenes de Productos con un tamaño estandarizado, para una visualización adecuada de los productos,
- Botón de Pedidos, informará al cliente que su pedido va a ser procesado.
- Notificaciones de productos solicitados, al departamento de ventas de Fe y Alegría.

Requerimientos Deseables

- Modificación de Información de Productos, para realizar cambios necesarios debido algún error tipográfico o un aumento de características del producto.
- Barra de Búsqueda de Productos
- Despliegue de misión y visión de Fe y Alegría
- Promociones de Productos, actualizaciones de ofertas semanales sobre determinados productos.
- Pestaña para visualizar los productos comprados por el .

Requerimientos Opcionales

- Botón de compra, una implementación de un sistema PayPal para la compra directa en el aplicativo.
- Sitio Web de distribución, centralización de las compras en un portal web que será gestionado por Fe y Alegría.

3.2. Requerimientos Generales

En la Tabla 2 se presentan los requerimientos generales del aplicativo, los cuales se trasladan a funcionalidades a desarrollar en los Sprints.

Tabla 1: *Requerimientos generales del Aplicativo*

1	Se solicita, la realización de una aplicación móvil que contenga una interfaz de usuario que refleje la identidad de las comunidades indígenas de la organización Fe y Alegría.
2	Se solicita, que el aplicativo contenga un formulario para registro de vendedores.
3	Se solicita, que el aplicativo contenga un formulario para publicar los productos.
4	Se solicita, una pantalla de inicio que cuente con el logo de Fe y Alegría, y cuente con la opción de cambio de idioma español-inglés.
5	Se solicita, que la aplicación cuente con una pantalla donde despliegue información general como misión, visión y valores de Fe y Alegría.
6	Se solicita, la información detallada para cada producto con fines promocionales: tamaños, precio, materia prima y fotografías. A su vez, se debe contar con una barra de búsqueda que permitan encontrar de forma rápida un producto en específico.
7	Se solicita, que la aplicación tenga un botón de pedido el cual envíe una advertencia al cliente sobre su pedido procesado y notifique de esta solicitud al departamento de ventas de Fe y Alegría.
8	Se solicita, una sección de catálogo donde se pueden visualizar todos los productos ofertados.

Nota. Se enumeran los requerimientos tomados de la reunión con el Stakeholder.

3.3. Descripción del Sistema

El aplicativo está desarrollado bajo una arquitectura cliente/servidor, ya que “La estructura de este modelo es capaz de facilitar mucho más la integración de nuevas tecnologías y el continuo crecimiento de la infraestructura computacional” (Arquitectura cliente-servidor: ¿De qué trata?, 2020). Como se analiza en la cita, en caso de ser necesario implementar otras soluciones para mejorar la aplicación, la arquitectura cliente/servidor lo permite.

Este aplicativo cuenta con tres capas, descritas a continuación:

- La capa de visualización se trabaja a nivel del cliente y su interacción con la aplicación es a través de la interfaz de usuario. La aplicación permitirá interactuar con sus componentes, es decir, ordenar, registrar y visitar la página de proveedores a través mayormente de una interfaz gráfica.
- La capa de lógica del negocio o middleware es donde se implementan los casos de uso y la lógica del negocio. Esta capa contiene es el puente que conecta la capa de visualización con la capa de datos. Los datos son procesados y enviados a la capa de visualización, así mismo las interacciones de los clientes y vendedores en la capa de visualización se procesan aquí, para luego enviarse a la capa de datos. Así en esta capa se almacenan procesos.
- La capa de datos es donde se almacenan los datos. Puede ser cualquier repositorio o gestor de base de datos relacional o no relacional. Los datos residen en esta capa de datos a la espera de recibir solicitudes del middleware.

3.4. Funcionalidades del Aplicativo

Las principales funcionalidades del sistema están clasificadas en dos grupos: las tareas programadas a realizar por el sistema y las tareas que debe realizar el cliente, tal como se presentan en la Tabla 2.

Tabla 2: *Funcionalidades del Aplicativo*

ID Funcionalidad	Nombre	Descripción
FS1	Catálogo de Productos	La aplicación almacenará distintos tipos de productos por tanto es necesario crear estas categorías en la base de datos para luego desplegarlas si el cliente lo requiere.
FS2	Botón de pedido	El botón mostrará una notificación al cliente informándole que su pedido será atentado por el departamento de ventas de Fe y Alegría. A su vez se genera un aviso del pedido.
FS3	Conexión con Base de Datos	El aplicativo contará con archivos de compatibilidad para conectividad a Firebase que permitirán el almacenamiento de la información de usuarios y productos.

3.5. Funcionalidades del Cliente

Tabla 3: *Funcionalidades del Cliente*

ID Funcionalidad	Nombre	Descripción
FC1	Selección de Idiomas	La aplicación debe contener archivos de traducción español/inglés de componentes.
FC2	Registro de Información	Se debe crear etiquetas de texto para mostrar texto donde solicita el tipo de información personal del proveedor de telares, como su nombre o cédula, y otro componente de tipo cuadro de texto donde se registra esa información y se la pasa a la base de datos.
FC3	Registro de Productos	De la misma forma se contará con etiquetas para registrar información del producto y también que acepte archivos de imágenes con tamaño estandarizado. Promocionando de mejor manera el producto.

3.6. Diseño de la Base de datos Firebase

Figura 3: *Base de datos- Formato JSON Propuesta 1*

```
{
    "id": 1111111,
    "password": "contraseña",
    "names": ["Nombre", "Apellido", "MiddleName"],
    "gender": "male",
    "e-mail": "example@email.com",
    "address": "Dirección completa",
    "phone_number": 9999999,
    "language_pref": "english",
    "seller_state": true,
    "age": 99,
    "published_products":[
        {
            "name":"Producto1",
            "photo_carousel_paths": ["htttp://images.com","htttp://images.com"],
            "description": "Esta es una descripción",
            "price": 1,
            "category": "categoria1",
            "sizes_avaliable": ["pequeño", "grande", "extra grande"],
            "color": "verde",
            "material": "lana"
        },
        {
            "name":"Producto2",
            "photo_carousel_paths": ["htttp://images.com","htttp://images.com"],
            "description": "There is not description",
            "price": 2,
            "category": "category2",
            "sizes_avaliable": ["pequeño" ],
            "color": "rojo",
            "material": "algodon"
        }
```

Nota. Modelado de los componentes que se deben crear en Firebase, el archivo se presenta como JSON, ya que no se utiliza un motor de base de datos relacional.

Se representa el primer diseño de modelo de base de datos NoSQL en formato JSON. Donde se distingue a los principales actores: usuarios y productos. Creándose así dos colecciones de datos. La primera se denomina “Productos” que contiene la descripción respectiva de cada producto como categoría, descripción y carrusel de fotos entre otros. La otra colección denominada” Usuarios”, almacena la información de usuarios que deseen publicar productos. De ahora en adelante se distinguen dos tipos de usuarios, primero existen los usuarios no registrados denominados “usuarios clientes” que simplemente buscan visualizar el catálogo o hacer pedidos y el segundo tipo usuarios vendedores denominados “usuarios registrados” que pueden realizar las actividades del usuario y cliente y además llevar un control más personalizado en la app. Así para los usuarios registrados se guardarán sus datos personales como nombre, correo, número

telefónico entre otros. Nótese que en el esquema actual se han incluido preferencias de idioma y números de contacto del usuario registrado.

El esquema actual no debe considerarse como definitivo ya que no está incluido el registro de clientes. El esquema de base de datos NoSQL permite flexibilidad al momento de realizar el diseño por lo que facilita su modificación posterior.

Figura 5: *Almacenamiento de Productos*

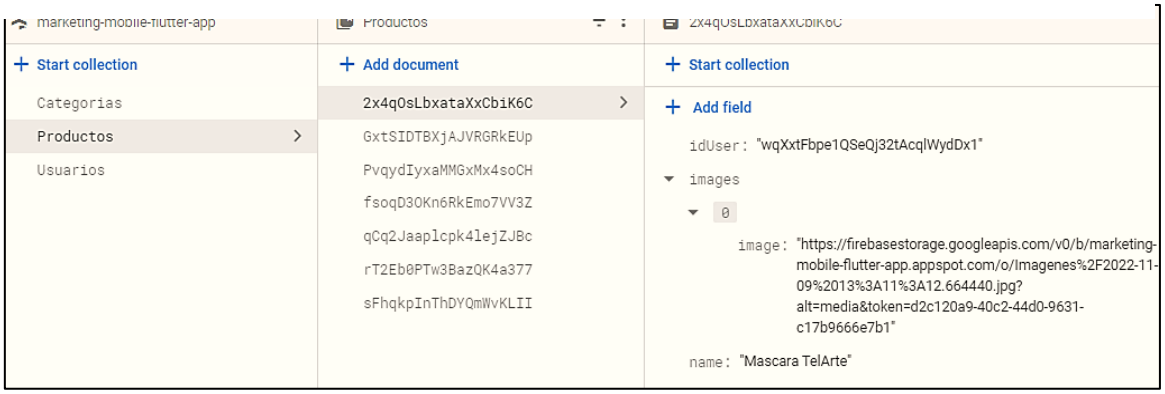
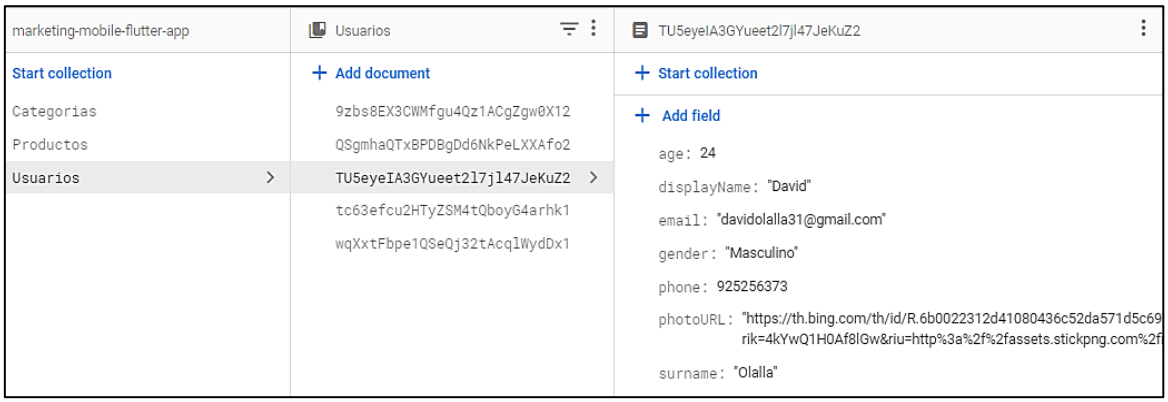


Figura 4: *Almacenamiento de Usuarios*



Para la base de datos se usó Firestore, “una base de datos NoSQL alojada en la nube a la que pueden acceder tus apps para Apple, Android y la Web directamente desde los SDK nativos”, (Firestore |, s. f.) y Cloud Storage para almacenamiento de clientes y vendedores además del carrusel de fotos de productos respectivamente.

3.7. Consideraciones de Implementación

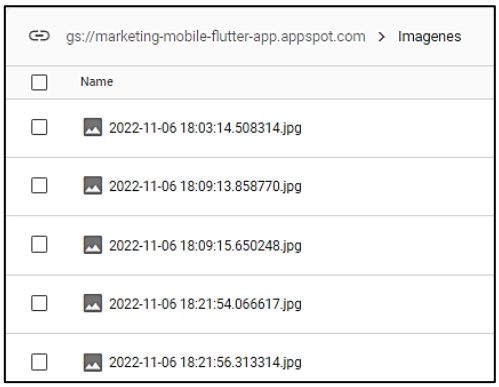
Para el correcto funcionamiento de la app debe considerarse primeramente el archivo google-services.json dentro del directorio del proyecto de la aplicación debe de ser el

adecuado y directamente proporcionado por el asistente de creación de proyecto de aplicaciones dentro de la consola de Firebase.

En segundo lugar, el id de aplicación debe coincidir con el que se encuentra dentro del archivo *../Android/app/build.gradle.com*. Adicionalmente se deben tener habilitados tanto los servicios de autenticación como de almacenamiento en Firebase.

Por último, para el guardado de imágenes del proyecto se creó una carpeta llamada Imágenes que contienen las imágenes de los productos cuyas rutas están almacenadas en Firestore.

Figura 6: *Almacenamiento de Imágenes*



Finalmente, como el aplicativo se desarrolló en modo prueba las reglas de seguridad permiten a cualquier persona que tenga acceso editar y leer la información. Sin embargo, una vez se lleve a producción la aplicación es obligatorio que se deben establecer reglas de acceso claras que protejan la integridad de los datos.

3.8. Organización de Componentes en Flutter

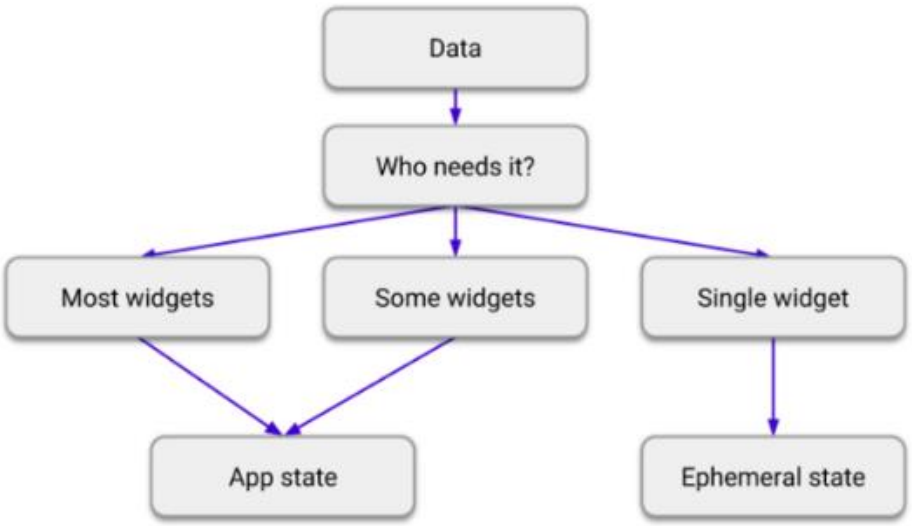
Se trata de los diferentes niveles que tendrá nuestra aplicación, separando archivos dentro del proyecto para un desarrollo ordenado y rápida localización de errores, el objetivo es evitar tener todo el código acumulado en una sola pestaña.

Las capas de esta arquitectura se presentan a continuación:

1. **UI:** Se trata la capa de representación de datos, llamado también frontend, que en Flutter es representado a través de widgets y páginas de navegación. En esta capa se realizan las importaciones de las funcionalidades que se desarrollarán en las otras capas, es decir el frontend depende del backend y no viceversa.
2. **Data:** Esta capa se responsabiliza de manejar dependencias externas para la obtención de datos de la app, en nuestro caso la conexión con la base de datos Firebase.
3. **Dispositivo:** Programación de funcionalidades nativas del dispositivo, por ejemplo, la utilización de la cámara para la toma de fotos de productos que serán publicados en el aplicativo.
4. **Dominio:** Capa orientada a cumplir las funcionalidades definidas en los casos de uso, programación de métodos utilizando el lenguaje de programación Dart.

3.9. Gestor de Estados en Flutter

Figura 7: *Estados en Flutter*



Nota. Imagen obtenida de (Diferencia entre estado efímero y estado de app, 2022). En algún punto del ciclo de vida de una aplicación en Flutter se encontrará en cualquier de los siguientes estados local/efímero o compartido.

En una aplicación el estado efímero se refiere al estado que simplemente afectará a un widget. Por otro lado, están el otro tipo de aplicaciones que necesitan cambiar o actualizar widgets de varias pantallas a la vez. En el primer caso simplemente con el uso de un StateFulWidget es más que suficiente. Sin embargo, la mayor parte de aplicaciones requieren de una técnica para gestión de estados como el Provider o GetX lo más utilizados.

3.10. GetX

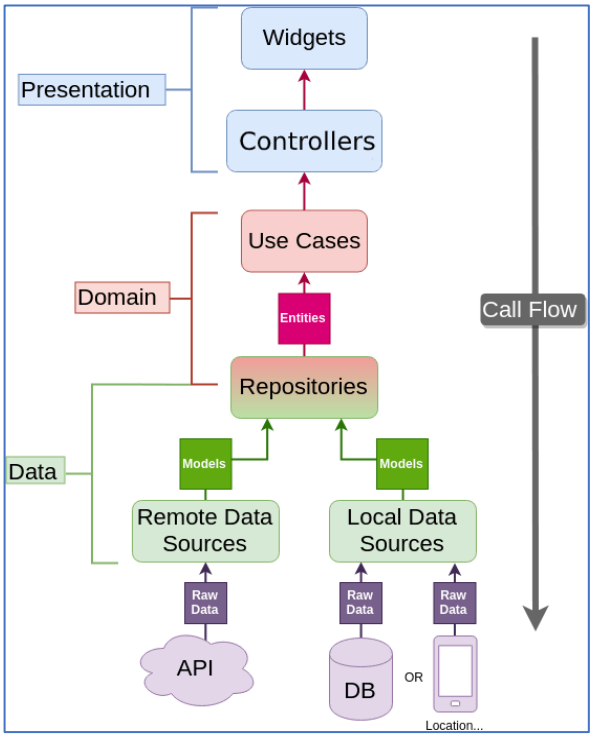
GetX es una solución extra ligera y flexible para Flutter. Combina gestión de estados de alto rendimiento, inyección de dependencias inteligente y gestión de rutas de forma rápida y práctica. GetX tiene tres principios básicos.

- **Rendimiento:** GetX esta focalizado en el rendimiento y uso ínfimo de recurso.
- **Productividad:** GetX utiliza una sintaxis fácil e intuitiva. Todo el código se simplifica bastante y por lo tanto se pueden ahorrar de desarrollo y proporcionará el máximo rendimiento que su aplicación
- **Organización:** GetX permite el desacoplamiento total de Vistas, por ejemplo, lógica de presentación, lógica de negocios, inyección de dependencia y navegación. No necesita contexto para navegar.

Para el caso de la aplicación actual se ha optado por el uso de GetX. Esto es debido a que no es solo un gestor de estados, sino que representa una facilidad al momento de gestión de rutas entre otras características.

3.11. Arquitectura de Flutter + GetX

Figura 8: *Arquitectura de Flutter + Get X*

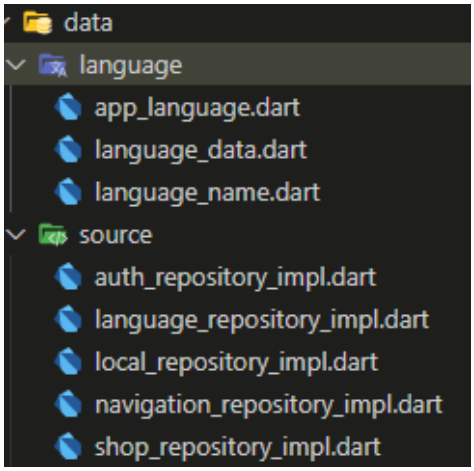


Nota. Imagen obtenida de (A sample app that implement Uncle Bob's Clean Architecture in Flutter - Best Flutter apps, 2021).

La Figura 8 indica cómo se realizó la implementación de una arquitectura limpia con GetX. Esta es la estructura de cada pantalla o vista de la aplicación. Los controladores cumplen la función de gestor de estados de cada pantalla de la UI. Igualmente se sigue los principios de una arquitectura limpia y cada capa solo podrá depender de las inferiores. Adicionalmente la capa de UI debe estar desacoplada de la lógica del negocio y la fuente de datos.

Capa de Datos

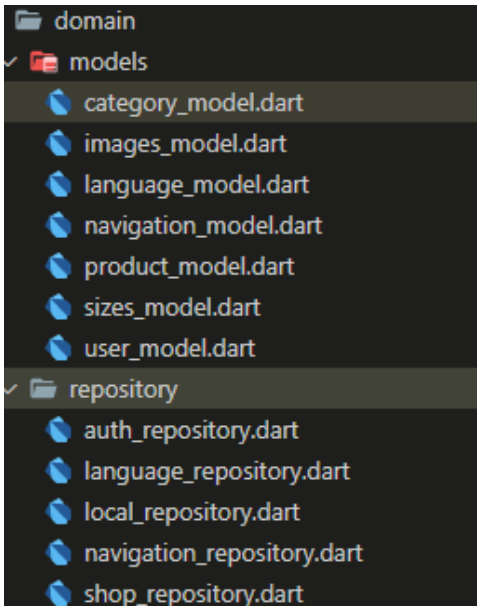
Figura 9: *Capa de Datos*



Nota. Organización de Capa de Datos dentro del Aplicativo

Capa de dominio

Figura 10: *Capa de Dominio*

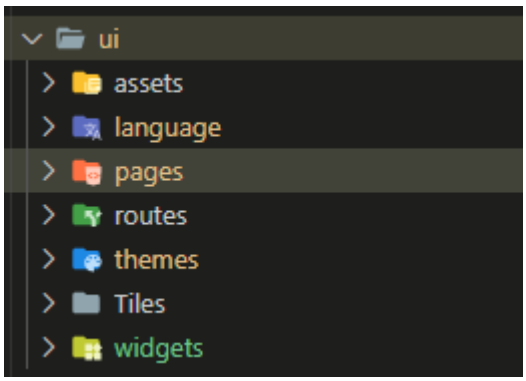


Nota. Organización de Capa de Dominio dentro del Aplicativo

En la capa de dominio se definió las interfaces que implementan la lógica de negocio que luego se puede implementar en el repositorio. Por ejemplo, se define en el repositorio la interfaz de autenticación. Se realizará una implementación mínima de los métodos necesarios luego se podrá instanciar en la capa de datos.

Capa de Presentación

Figura 11: *Capa de Presentación*



Nota. Organización de Capa de Presentación dentro del Aplicativo

3.12. GetX como Gestor de Estados

Manejo de rutas

Para el manejo de rutas se definieron dos archivos principalmente. El primer archivo route_name. dart se describen todos los nombres de las rutas en una clase RouteName. El segundo archivo describe los widgets de las vistas o pantallas de la aplicación. Nótese que gracias al uso de GetX no se utiliza “context” para el manejo de rutas.

Pantallas o Vistas de la aplicación

La aplicación en total cuenta con 7 pantallas cada una con su respectivo controlador que manejará los estados en cada pantalla, además un archivo binding, este se trata de un fichero tipo Dart que se utiliza para un manejo eficiente de inyección de dependencias

Navigation Widget: Funcionalidades

- Cambio de idioma
- Menú de navegación

Navigation Widget: Descripción del controlador

La funcionalidad de Navigation Widget es desplegar la barra de aplicación y el menú de navegación los cuales serán el método de navegación central de aplicación.

El controlador maneja la navegación entre las vistas a través de una variable que controla el índice del widget de navegación. El índice cambiará en cada interacción del icono de navegación. Hay una consideración adicional. Si el vendedor no está registrado únicamente estará disponible la pantalla home de vista de productos, caso contrario tendrá que registrarse para agregar productos, ver o editar información personal. Esta funcionalidad se verifica mediante una instancia del repositorio de autenticación.

Home Widget: Funcionalidades

- Ver catálogo de productos completo.
- Mostar cada ítem con una imagen, nombre y precio en una lista desplegable.

Home Widget: Descripción del controlador

Home Widget es el encargado de desplegar o mostrar el catálogo de productos en forma de widgets en una lista desplegable. La función del controlador es recuperar la lista de productos al cargar la página mediante la implementación de funciones del repositorio de la tienda, además envía los argumentos necesarios a la siguiente pantalla para mostrar los detalles de cada producto.

Add Product View: Funcionalidades

- Ingresar información detalla de un producto.
- Seleccionar imágenes de un producto o seleccionar de galería local.
- Subir imágenes de un producto al Storage de Firebase.
- Almacenar datos del producto en Firebase.

Add Product View: Descripción del controlador

El Add Product Widget se encarga de desplegar un formulario con los campos o widgets respectivos para añadir nuevos productos al catálogo. La función del controlador es la validación de campos mediante variables observables que no permiten agregar el producto hasta que se cumpla una seria de requerimientos. Además, se encarga de llamar a la función de añadir producto.

Esta función llamará a otra de la implementación del repositorio en la capa de datos y mediante una instancia a la base de datos cargando un documento de producto dentro de la colección “Usuarios”.

Profile Widget: Funcionalidades

- Mostrar información del usuario registrado (nombre, edad, correo electrónico, genero, foto, entre otros).
- Permitir la opción de editar información personal.

Profile Widget: Descripción de controlador

El Widget se encarga de mostrar toda la información del usuario registrado (nombre, apellido, edad, e-mail, entre otros). La función del controlador es simplemente extraer es ejecutar una consulta a Firestore y extraer los datos requeridos, mientras se mostrarán datos provisionales en lo que consigue la información.

EditProfile Widget: Funcionalidades

- Editar Información de usuario registrado.
- Validar campos vacíos.

EditProfile Widget: Descripción de controlador

El Widget es simplemente de desplegar un formulario de campos del usuario editables. El usuario registrado tiene la posibilidad de cambiar los datos. El controlador simplemente se encarga de realizar un pequeño query de actualización con los nuevos valores en los campos. Además de incluir una validación de los campos vacíos, que nuevamente se realiza mediante variables de controlador de campos de texto.

Details View: Funcionalidades

- Mostar información detallada de cada producto
- Mostrar carrusel de imágenes de cada producto
- Mostrar nombres de proveedor
- Redirigir al home page de Fe y Alegría

Details View: Controlador

En este widget encargado de mostrar toda la información detallada de cada producto (nombre, categoría, descripción, proveedor, material entre otros). El controlador simplemente se encarga de recuperar los daros del usuario que publicó el producto en cuestión mediante el campo foráneo de id del producto.

Sign in Widget: Funcionalidades

- Validación de campos de contraseña y email
- Autenticación de usuario por contraseña

Sign in Widget: Controlador

Este Widget es el encargado de Mostrar los campos de texto requerido para la autenticación del usuario. La función de controlador es la validación de campos por controladores de campos de texto y llamada a la función “Sign In” del repositorio de autenticación la cual utiliza una instancia de “FirebaseAuth” para realizar la autenticación.

Sign up Widget: Funcionalidades

- Validación de campos de usuario
- Creación de nuevo Usuario en Firestore y FirebaseAuth

Sign up Widget: Controlador

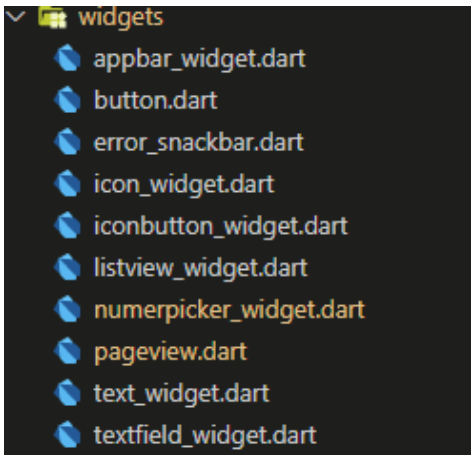
Este Widget es el encargado de construir un formulario con los campos para posteriormente ser completados y se pueda crear un nuevo usuario. El controlador se encarga de la validación en tiempo real de los campos mediante los controladores de campos de texto además de invocar a la función Sign Up que crea un nuevo usuario con usuario y contraseña como método de autenticación. Posteriormente también crea los datos en la colección de ‘Usuarios’ de Firestore.

3.13. Widgets y Vistas Adicionales

Adicionalmente de estos widgets también se tiene los siguientes widgets que están vinculados con la navegación

- **Splash Widget:** pantalla de carga inicial de la aplicación antes de ir a la vista Home
- **NoAuth Widget:** Vista auxiliar en las pantallas de Agregar producto y Profile cuando no hay un usuario autenticado. Redirige a la pantalla de Inicio de Sesión
- **Search Widget** se encarga de la función de filtrado por búsqueda en un campo de texto.

Figura 12: *Estructura de Carpetas de los Widgets*



Nota. Se presentan los widgets que son reutilizables de la aplicación en varias vistas.

3.14. Product Backlog

Se traslada las funcionalidades a implementar a historias de usuario, donde se asignó una prioridad para su desarrollo en la aplicación, siendo 1 la más alta prioridad y 3 la mínima, tal como presenta la tabla a continuación.

Tabla 4: *Product Backlog de la Aplicación*

PRODUCT BACKLOG DE LA APLICACIÓN			
TAREA ID	**HISTORIA**	**ESTIMADO (DÍAS)**	**PRIORIDAD**
2	Como usuario cliente y registrado, deseo una pantalla de inicio atractiva que promueva el uso del aplicativo.	5	1
1	Como usuario cliente y registrado, deseo poder cambiar la configuración de idioma del aplicativo	3	1
3	Como usuario cliente y registrado, deseo poder ingresar de forma fácil al catálogo de productos.	3	2
4	Como usuario cliente, deseo registrar mi información personal y crear una cuenta.	5	1
5	Como usuario registrado, deseo poder almacenar información de mis productos.	4	2
6	Como usuario cliente y registrado, deseo visualizar a detalle toda la información registrada del producto.	3	2

Nota. Se definen las actividades que se realizarán en los Sprints

4. Desarrollo de Metodología Scrum

El objetivo de este capítulo es la definición e inicio del proceso de desarrollo del aplicativo utilizando la metodología ágil SCRUM, a continuación, se especificarán las herramientas y roles de los integrantes del proyecto de titulación. Este proceso permitirá enfocarse en cada tarea para cumplir las metas propuestas.

4.1. Asignación de Roles

Tabla 5: *Tabla de Roles*

Rol	Responsable
Product Owner	Ing. Fabián de la Cruz
Scrum Master	Richard Montalvo
Development Team	Sergio Olalla
Development Team	Richard Montalvo
Stakeholder	Ing. Francisco Rodríguez

Nota. Actividades por realizar de los involucrados del proyecto.

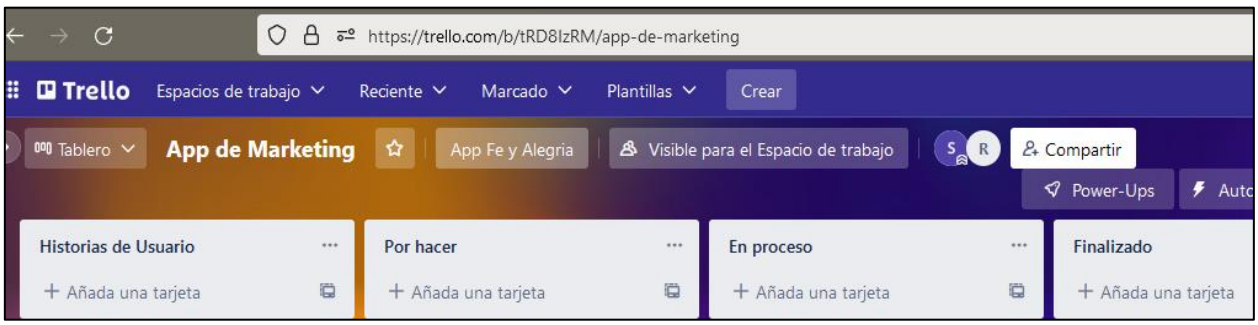
4.2. Creación de Área de Trabajo

Se utilizará una herramienta en línea, donde se colocarán notas, que permitirán organizar las tareas del equipo. La herramienta donde se organizó el tablero se denomina “Trello”, su característica principal es la administración de varios tipos de proyectos. Se define las tareas a realizar en el primer Sprint en un tiempo de 3 semanas.

El Área de trabajo consta de 4 partes:

- Historias de usuario
- Por hacer
- En progreso
- Finalizado

Figura 13: *Creación de Tablero Kanban*

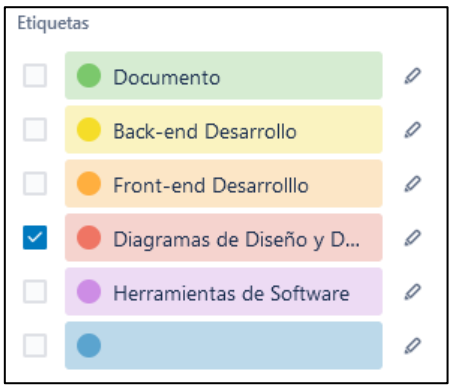


Nota. Tablero Kanban para llevar control del desarrollo

4.3. Clasificación de Actividades Kanban

Una vez que se definen las tareas a realizar se utilizará una etiqueta de colores para reconocer que tareas corresponden a cada parte del proceso de desarrollo.

Figura 14: *Clasificación de Actividades*



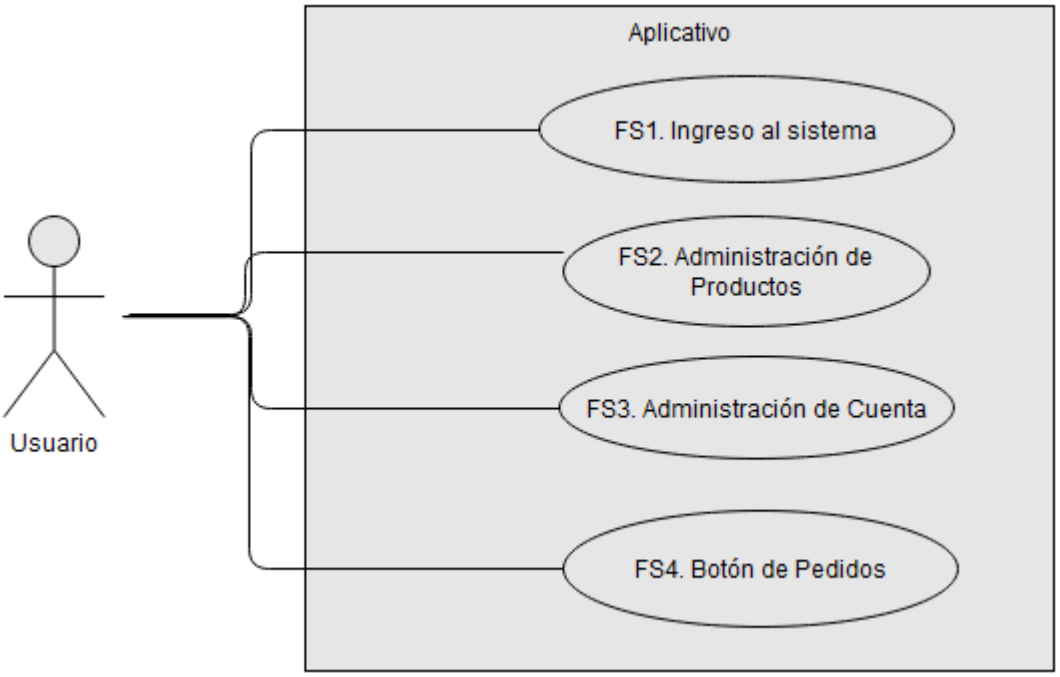
Nota. Se ha definido etiquetas para un mayor control de las actividades

4.4. Diagrama de Caso de Uso General

Se presenta una visión general del funcionamiento del sistema, que luego será descompuesto para analizar mayor profundidad las tareas a realizar.

- El término “Administrar” en el caso de uso se refiere a los procesos de registro, edición, borrado.

Figura 15: *Caso de uso General del Aplicativo*



4.5. Diagrama de Casos de usos específicos

Figura 16: *Caso de Uso de Registro de Productos*

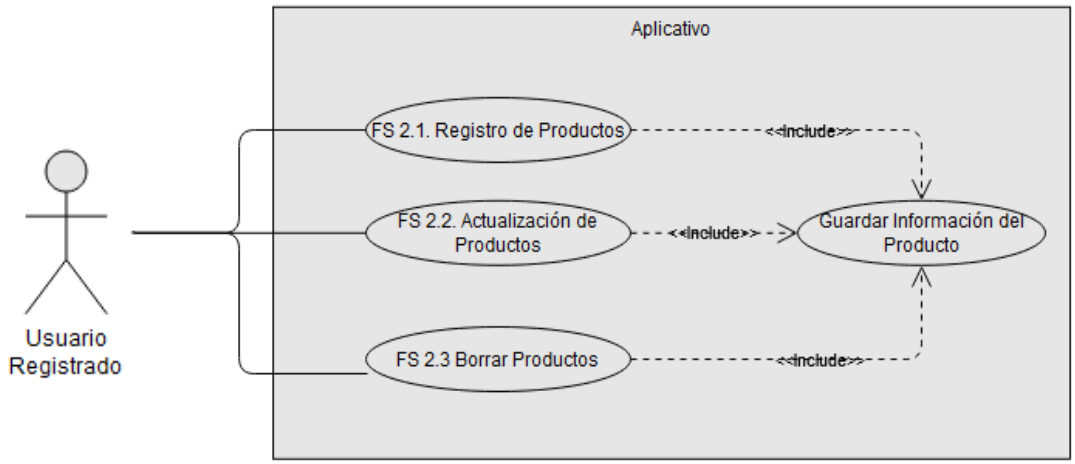


Tabla 6: *Descripción de Registro de Producto*

Sistema:	Aplicación Promoción de Productos Fe y Alegría
Caso de uso:	Registrar Productos
Actor:	Usuario Registrado
Dependencias	Insertar los datos correctamente
	Conexión con la base de datos
Precondición	Completar el formulario de registro de producto
Secuencia	1. Seleccionar Botón de añadir Producto
	2. Rellenar la información solicitada
	3. Seleccionar el botón de Guardado
	4. Editar la información de ser necesario
	5.Borrar producto
Excepciones:	Error de conexión con la base de datos
Soluciones:	Solicitar al administrador del sistema

Figura 17: *Caso de Uso de Registro de Clientes*

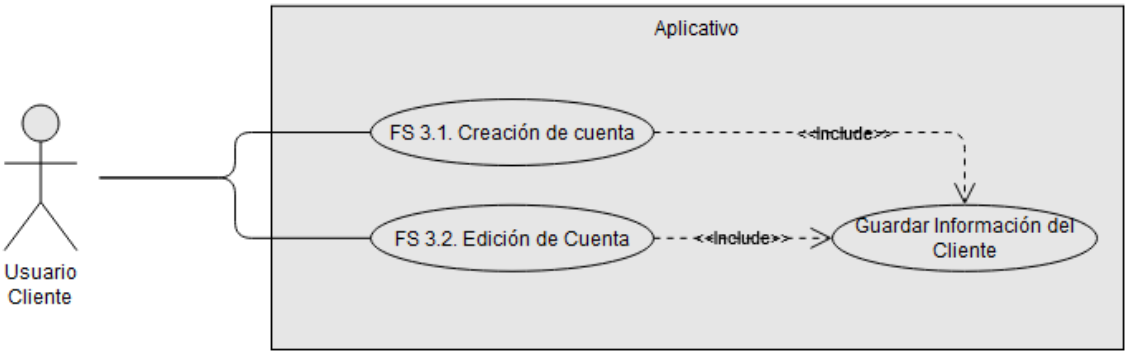


Tabla 7: *Descripción de Registro de Clientes*

Sistema:	Aplicación Promoción de Productos Fe y Alegría
Caso de uso:	Registro de Información Personal
Actor:	Usuario Cliente
Dependencias	Insertar los datos correctamente
	Conexión con la base de datos
Precondición	Completar el formulario de registro de proveedor
Secuencia	1. Seleccionar Botón de añadir Proveedor
	2. Rellenar la información solicitada
	3. Seleccionar el botón de Guardado
	4. Editar la información de ser necesario
Excepciones:	Error de conexión con la base de datos
Soluciones:	Solicitar al administrador del sistema

Figura 18: *Caso de Uso Realizar Pedido*

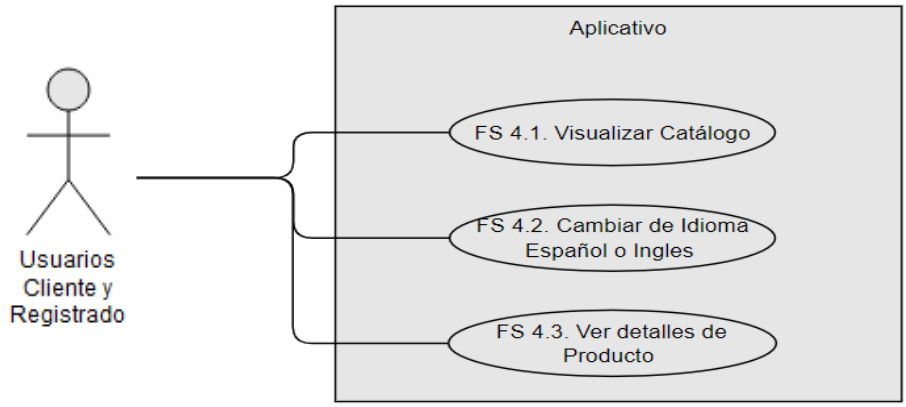


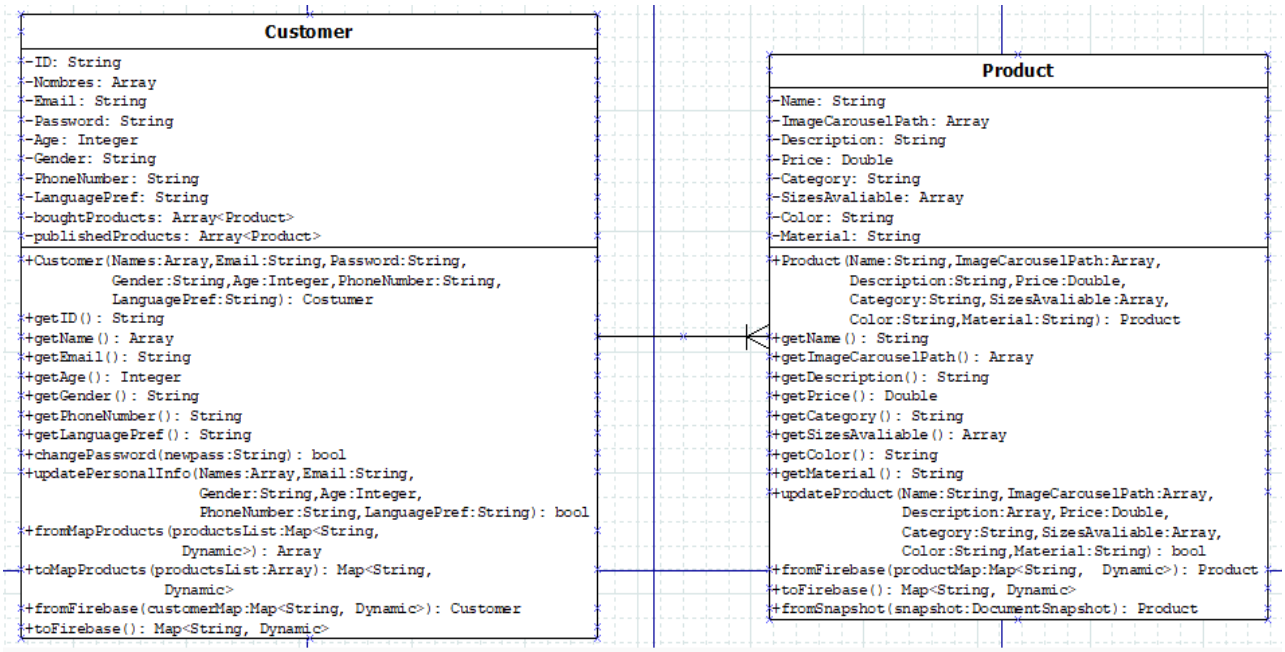
Tabla 8: *Descripción Realizar Pedido*

Sistema:	Aplicación Promoción de Productos Fe y Alegría
Caso de uso:	Realizar pedido
Actor:	Usuario Cliente y Registrado
Dependencias	Conexión con la base de datos
Precondición	Ninguna
Secuencia	1. Seleccionar botón de Iniciar
	2. Observar y escoger el producto deseado
	3. Seleccionar el producto
	4. Visualizar información detallada del producto
	5. Seleccionar el botón de hacer pedido y ser redireccionado a la Pagina de fe y Alegría
Excepciones:	Error de conexión con la base de datos
Soluciones:	Solicitar al administrador del sistema

4.6. Diagrama de Clases

Se presenta en la Figura 10 el diagrama de clases donde se describen los objetos del sistema, sus características y funcionalidades que tendrán para el futuro uso del aplicativo.

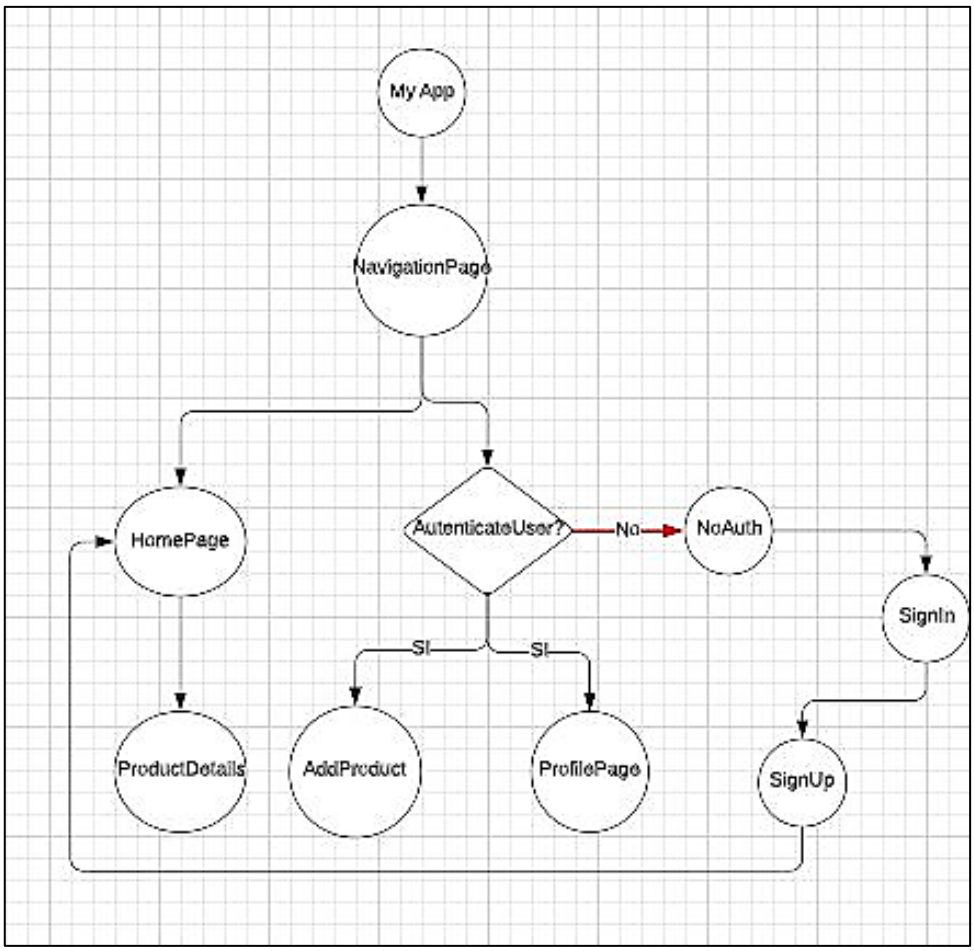
Figura 19: *Diagrama de Clase del Sistema*



4.7. Diagrama de Estados

Al principio se analizó la posibilidad de utilizar una gestión de estados más inteligente y organizada como lo es Bloc. Sin embargo, no se posible debido a la complejidad de manejo de las librerías. Al final se decidió implementar con una gestión de estados basado en controladores de página. La cual simplifico y dio como resultado el árbol de Widgets que esta abajo.

Figura 20: *Árbol de Widgets Simplificado*



Nota. Árbol de widgets simplificado solo incluyendo pantallas

CAPÍTULO V: SPRINT 1

5. Sprint 1

5.1. Sprint Backlog 1

En la Tabla 8 se presentan los elementos a trabajar en el Sprint 1 de desarrollo del aplicativo, estos elementos son seleccionados de los elementos generales del Product Backlog.

Tabla 9: *Sprint Backlog 1*

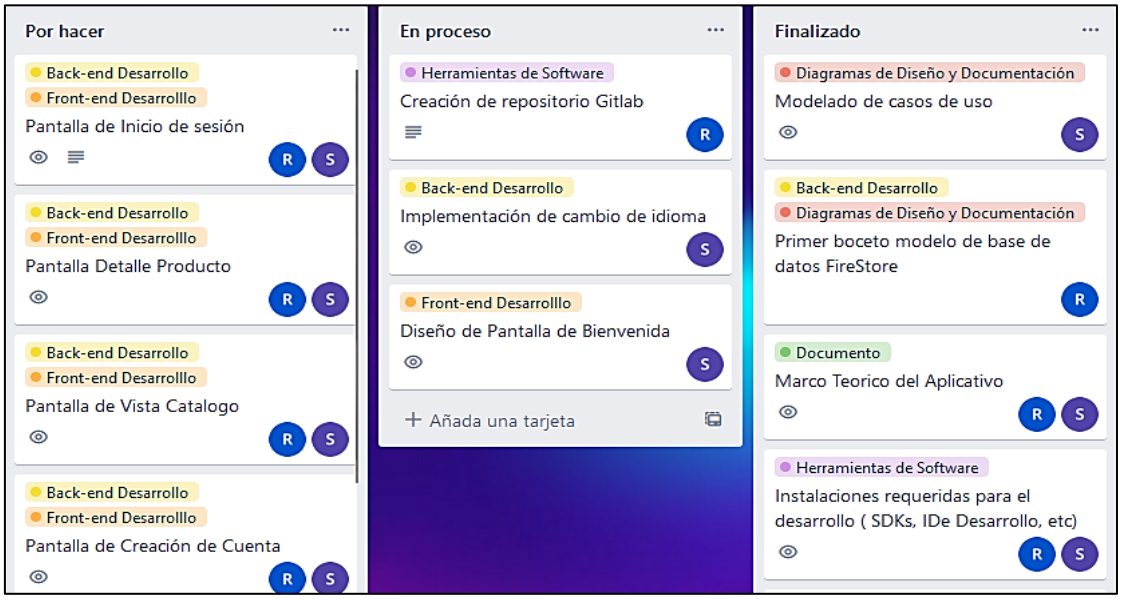
SPRINT BACKLOG 1			
Tarea ID	**Historia**	**Estimado (Días)**	**Prioridad**
2	Como usuario cliente y registrado, deseo una pantalla de inicio atractiva que promueva el uso del aplicativo.	5	**2**
1	Como usuario cliente y registrado, deseo poder cambiar la configuración de idioma del aplicativo	3	1
4	Como usuario cliente, deseo registrar mi información personal para visualizar los productos ofertados.	5	**1**

Nota. Se asigna prioridad a las actividades que llevarán más tiempo de desarrollo.

5.2. Kanban Inicial Sprint 1

Se presenta a continuación el tablero, con las actividades que el grupo de desarrollo debe empezar, estas tareas se han colocado en orden para respetar la organización de la aplicación, además que primero se debe programar las funcionalidades, probarlas con plantillas establecidas y luego integrarse con las pantallas finales de UI.

Figura 21: *Kanban Inicial Sprint 1*



5.3. Prototipo de Interfaces

Prototipo de Pantalla de Bienvenida:

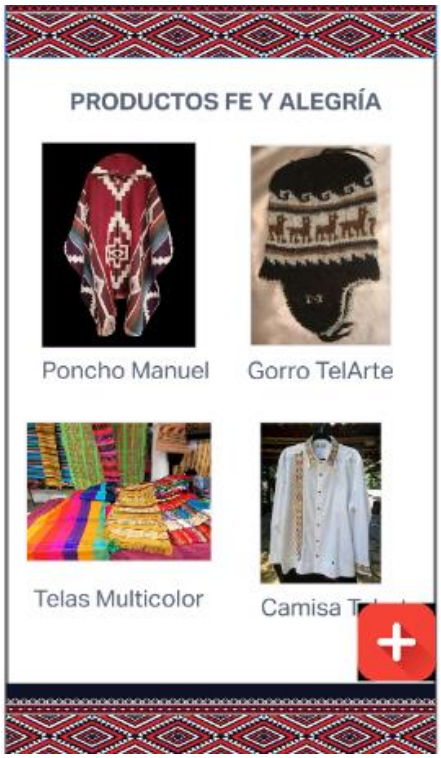
Se trata de la primera pantalla que observa tanto usuario cliente como registrado por tanto debe ser llamativa y provocar curiosidad sobre el aplicativo.

Figura 22: *Prototipo pantalla de Bienvenida*

Prototipo de Catálogo de Productos:

Se debe mostrar todos los productos que se han registrado en la aplicación, para que el usuario seleccione el más llamativo. Además, en la parte inferior derecha cuenta con el botón de añadir productos.

Figura 23: *Prototipo de Catálogo*



Prototipo de Pantalla de Pedidos:

Se despliega un carrusel de imágenes del producto, se muestra el título del producto, tamaños y precio.

Figura 24- *Prototipo Detalle de Producto*

5.4. Interfaces Sprint 1

Se presentan las interfaces que se observarán al iniciar la aplicación, para la realización de estas se tomó en cuentas los requerimientos funcionales y la gama de colores con los que trabaja Fe y Alegría.

a) Pantalla de Bienvenida de la Aplicación

- Primera Pantalla que se visualizará el usuario cliente y registrado al abrir la aplicación, se utilizó mucha decoración ya que se busca despertar el interés por el mismo.

Figura 25: *Pantalla de Bienvenida*

Figura 26: *Fragmento de código Bienvenida*

```
class _LoginState extends State<Inicio> {
  //CREACIÓN DE WIDGET
  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: BoxDecoration(
        image: DecorationImage(
          // INSERCIÓN DE GIF
          image: const
AssetImage("assets/diablo.gif"),
          fit: BoxFit.cover,
          colorFilter: ColorFilter.mode(
            const
Color(0xff4B1F4D).withOpacity(0.1),
            BlendMode.color,
          ),
        ),
      ),
```

b) Registro de Cliente

Una vez que el usuario avanza de la pantalla de bienvenida, visualizará el catálogo general de productos. En la parte inferior de la aplicación se ubica una barra de desplazamiento que conduce a una pestaña de añadir producto, la cual cuenta con indicaciones para registro de usuario y que le permita publicar sus productos.

Figura 27: *Pantalla de Registro de Cliente*

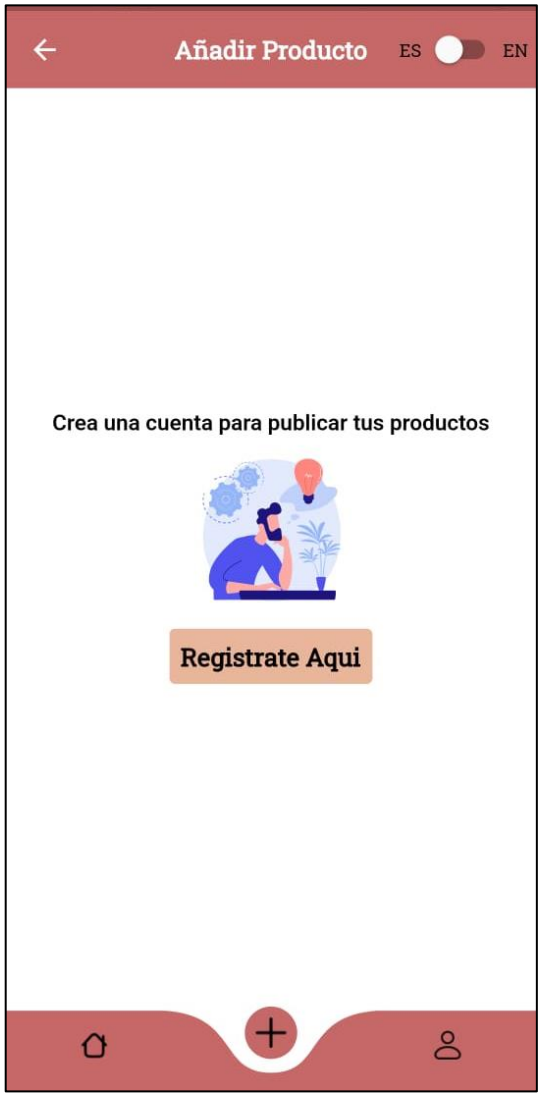


Figura 28: *Fragmento de código Registro de usuario*

```
Container(
  child: Text(  // Widget que le avisa al usuario que tiene que autenticarse para añadir productos
    'create_acount'.tr,
    style: const TextStyle(
        color: Colors.black,
        fontWeight: FontWeight.w600,
        fontSize: 16), // TextStyle
  ), // Text
), // Container
const SizedBox( //separador
  width: 15,
), // SizedBox
Container( //Widget Container que muestra imagen
  height: 136,
  width: 132,
  decoration: const BoxDecoration(
    image: DecorationImage(
        image: AssetImage("assets/Login1.gif"), fit: BoxFit.fill), // DecorationImage
  ), // BoxDecoration
), // Container
const SizedBox(
  width: 15,
), // SizedBox
TextButton( // boton que llama al metodo iniciar sesión
  onPressed: signIn,
  style: TextButton.styleFrom(
    backgroundColor: Color.fromARGB(236, 231, 175, 147),
  ),
```

Figura 29: *Funcionalidad de cambio de pantalla a Inicio de Sesión*

```
class _NotAuthPageState extends State<NotAuthPage> {
  // Esta función lleva a la pagina de inicio de sesión mediante GetToNamed de GetX
  void signIn() {
    Get.toNamed(RouteName.sign_in); //
  }
```

c) Inicio de Sesión

Una vez que el usuario registrado pulsa el botón de Registrar Productos, se presentará la pantalla de inicio, donde se indica la opción de creación de cuenta o si está ya registrado, ingresar sus credenciales y acceder a la pantalla de publicación de productos.

Figura 30: *Pantalla de Inicio de Sesión*

Figura 31: *Fragmento de Código Inicio de Sesión*

```
Container(   // Container que contruye el Widget del Logo de Fe y Alegria
  height: 146,
  width: 142,
  decoration: const BoxDecoration(
    image: DecorationImage(
        image: AssetImage("assets/FAlogo.png"), fit: BoxFit.fill), // DecorationImage
  ), // BoxDecoration
), // Container
const SizedBox(height: 20),
TextFormFieldWidget( // Campo Personalizado que contiene  el campo de email
  controller: controller.emailController,
  hint: 'email'.tr,
), // TextFormFieldWidget
const SizedBox(height: 20),
TextFormFieldWidget( // Campo Personalizado que contiene  el campo de contraseña
    controller: controller.passwordController,
    hint: 'password'.tr), // TextFormFieldWidget
const SizedBox(height: 35),
SizedBox(  //Boton que ejecuta la función del controlador de verificación de datos
  width: size.width * 0.85,
  height: 45,
  child: Buttonwidget(
    context: context,
    text: 'singin'.tr,
    onTap: controller.verify,
  ),
```

Figura 32: *Fragmento de Código de Controlador de Inicio de Sesión*

```
class SignController extends GetxController {
 //Se definen dos variables de campos de texto a ser validados
 TextEditingController emailController = TextEditingController();
 TextEditingController passwordController = TextEditingController();
 List<String> listMessages = []; // esta lista contiene la lista de los mensajes de error

 /* El metodo Onclose se sobreescribe
  El metodo on close es invocado al momento que se termine de ejecutar la pantalla y por tanto
  se disponen de algunos elementos que no puedan ocacionar fugas de memoria como streams de datos
  animaciones y otros */

 @override
 void onClose() {
   super.onClose();
   // Se elminan los controladores de los campos de email y contraseña
   emailController.dispose();
   passwordController.dispose();
 }

 //Función que se encarga del proceso de validación de campos antes de intentar autenticar al usuario
 void verify() {
   //Limpiar la lista de mensajes de error
   listMessages.clear();

   if (emailController.text.isEmpty) {
     listMessages.add('validate_email'.tr);
   }
   if (passwordController.text.isEmpty) {
     listMessages.add('validate_password'.tr);
   } else {
     signIn();
```

d) Registro de Usuario

Si el usuario cliente elige la opción “Crear una cuenta” se desplegará otra pantalla donde se le solicitará llenar los campos para generar una nueva cuenta y que así se pueda publicar productos. Mientras que en la base de datos se enlazan los productos con su respectivo usuario.

Figura 33: *Pantalla de Registro de Usuario*

Figura 34: *Fragmento de código de controlador de creación de cuenta*

```
void signUp() { //FUNCION DE INICIO DE SESIÓN
  Get.showOverlay(
    asyncFunction: () async {
      //Creación de usuario
      try {
        await const AuthFirebase()
            .createUserWithEmailAndPassword(
          emailController.text.trim(),
          passwordController.text.trim(),
        )
            .then((value) {
          /*Depues de crear el usuario con contraseña para autenticación se crea
            actualiza la base de datos  con los datos de nuevo usuario */
          FirebaseFirestore.instance
              .collection('Usuarios')
              .doc(value.user!.uid)
              .set(
                UserModel.toMap( // antes de enviar los datos primer hay que convertirlos al formato
                  nameMap: nameController.text,
                  imageMap:
                      'https://th.bing.com/th/id/R.6b0022312d41080436c52da571d5c697?rik=4kYwQ1H0Af8lGw&ri
                  emailMap: emailController.text,
                  ageMap: int.parse(ageController.text),
                  genderMap: gender,
                  phoneMap: int.parse(phoneController.text),
                  surnameMap: surnameController.text,
                ),
              );
        });
        Get.offAllNamed(RouteName.splash); //se eliman todas las pantallas y se vuelve al inicio
      } on FirebaseAuthException catch (e) {
        Get.showSnackbar(ErrorSnackbar(e.message ?? e.code)); //muestra el mensaje de error
```

Figura 35: *Fragmento de código de pantalla de creación de cuenta*

```
TextFormFieldWidget( // campo de email
  controller: controller.emailController,
  hint: 'email'.tr,
), // TextFormFieldWidget

const SizedBox(height: 20), // separador

//name
TextFormFieldWidget( // campo de nombre
  controller: controller.nameController,
  hint: 'name'.tr,
), // TextFormFieldWidget

const SizedBox(height: 20),

//surname
TextFormFieldWidget( // campo de apellido
  controller: controller.surnameController,
  hint: 'surname'.tr,
), // TextFormFieldWidget

const SizedBox(height: 20),

//age
TextFormFieldWidget( // campo de edad
  controller: controller.ageController,
  hint: 'age'.tr,
), // TextFormFieldWidget
```

e) Agregar Producto

Una vez que el usuario cliente ha creado su cuenta e iniciado sesión se presentará esta pantalla donde tendrá la posibilidad de subir distintas imágenes de los productos que desea publicitar, una descripción. También se debe recalcar que si el producto tiene descuento este aparecerá en un carrusel de imágenes en la parte superior catálogo de productos.

Figura 36: *Pantalla de Agregar Producto*

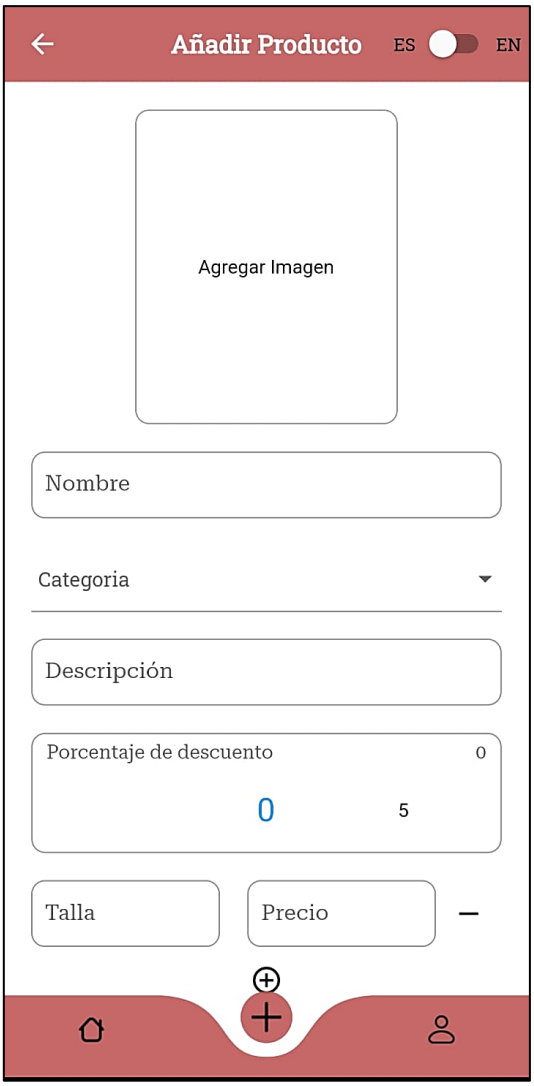


Figura 37: *Fragmento de código de Pantalla Añadir Producto*

```
Future<void> addProuct({
  required String name,
  required String description,
  required String category,
  required List<Map<String, dynamic>> listSizes,
  required List<bool> selectedMaterialsIndex,
  required int porcentage,
  required List<AddProductImages> listImages,
}) async {
  //Subir imagenes al almacenamiento de Firebase
  List<Map<String, dynamic>> newListImages = []; // Este Arreglo almacena los URL de las imagenes subidas
  isLoadingPage.value = false;
  for (var element in listImages) {
    Reference ref = FirebaseStorage.instance
        .ref()
        .child('Imagenes')
        .child("${DateTime.now()}.jpg"); // cada imagen se subira con el nombre de la fecha de subida
    UploadTask uploadTask = ref.putFile(element.fileImage!);

    await uploadTask.whenComplete(() async {
      print(ref.getDownloadURL());
      newListImages.add({'image': await ref.getDownloadURL()});
    });
  }
  //Obtiene las lista de los materiales  del producto
  List<String> madeOfMaterials = [];
  getSelectedMaterials(selectedMaterialsIndex, madeOfMaterials);
```

Nota. Método de añadir Producto del controlador de la vista Añadir Producto

Figura 38: *Fragmento de código acceso a multimedia*

```
children: [
  //La primera opción es para seleccionar una imagen del la galeria
  //Se debe recuperar la ruta de la imagen para luego conseguir
  //el objeto tipo File y luego guardala en la lista de imagenes
  IconButton(
    onPressed: () async {
      var image;
      image = await ImagePicker()
          .pickImage(source: ImageSource.gallery, imageQuality: 25); // selecciona calidad
      setState(() {
        fileImage = File(image.path);
        baseImage = base64Encode(fileImage!.readAsBytesSync());
        listImages.add(AddProductImages(fileImage: fileImage));
      });
    },
    icon: const Icon(Icons.photo_library_outlined),
  ), // IconButton
  //La segunda opción es tomar una imagen pero del dispositivo y
  //repetir el mismo procedimiento
  IconButton(
    onPressed: () async {
      var image;
      image = await ImagePicker()
          .pickImage(source: ImageSource.camera, imageQuality: 25) //selecciona la calidad
          .whenComplete(() => Navigator.pop(context));
      setState(() {
        fileImage = File(image.path);
        baseImage = base64Encode(fileImage!.readAsBytesSync());
        listImages.add(AddProductImages(fileImage: fileImage));
      });
    },
```

Nota. Extracto de código de Modal para seleccionar imágenes de galería o tomar.

f) Traducción de Componentes

Para la traducción se utilizó una colección de pares clave/valor donde se debe tener una clave asociada a la traducción del idioma que se desea cambiar, esto se puede apreciar en la siguiente figura:

Figura 39: *Función de Traducción*

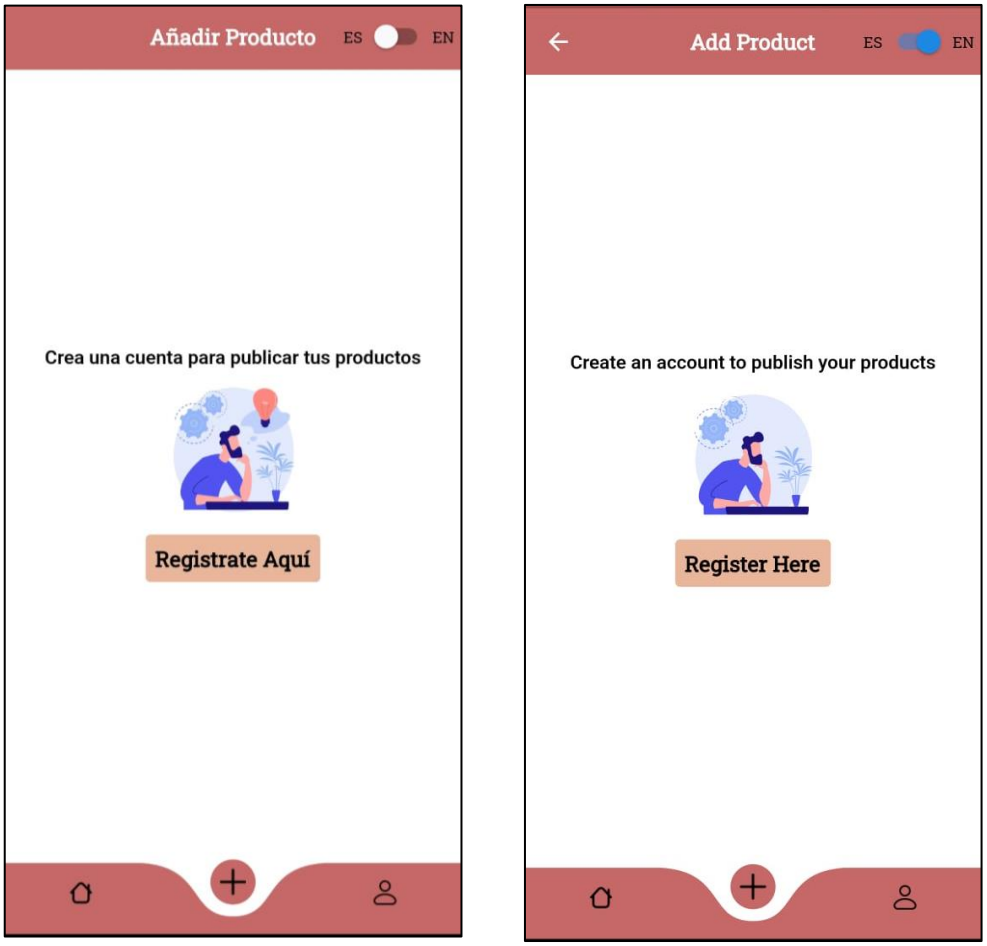


Figura 40: *Fragmento de Código Traducción*

```dart
//Clase que define los componentes de traducción de la Aplicación
class AppLanguage extends Translations {
  @override
  Map<String, Map<String, String>> get keys => {
    //ELMENTOS DE TRADUCCIÓN EL INGLES Definidos por un objeto tipo Map<String,String>
    //Cada traducción se compone de valor clave-valor. El clave de elemento debe repetirse
    //en cada idioma que se quiera traducir
        'en_US': {
          //Navigation
          'search': 'Search',
          'singin': 'Sing In',
          'start': 'Register Here',
          'sign_up': 'Create an account',
          'add_product': 'Add Product',
          'place_order': 'Place Order',
          'create_user': 'Create new user',
          'data_user': 'Data User',
          'add_image': 'Add Image',
          'size': 'Size',
          'price': 'Price',
          'porcentage': 'Discount Percentage',
          'description': 'Description',
          'category': 'Category',
          'signOut': 'Sign Out',
          'all': 'See all products',
          'my_products': 'My products',
          'options': 'Options',
          'wool': 'Wool',
          'fabric': 'Fabric',
          'manufacturer': 'Manufacturer',
```

```dart
    //ELEMENTOS DE TRADUCCIÓN EL INGLES Definidos por un objeto tipo Map<String,String>
    'es_EC': {
      //Sign
      'email': 'Correo electrónico',
      'name': 'Nombre',
      'surname': 'Apellidos',
      'age': 'edad',
      'gender': 'Género',
      'phone': 'Celular',
      'password': 'Contraseña',
      'confirm_password': 'Confirme su contraseña',
      'male': 'Masculino',
      'female': 'Femenino',

      //update profile
      'update_but': 'Guardar Cambios',

      //Validate Sign
      'validate_email': 'Debe colocar su correo',
      'validate_password': 'Debe colocar su contraseña',
      'validate_name': 'Debe colocar su nombre',
      'validate_surname': 'Debe de colocar sus apellidos',
      'validate_age': 'Debe colocar su edad',
      'validate_phone': 'Debe de colocar su celular',
      'validate_confirm_password': 'Debe de confirmar su contraseña',
      'password_confirmPassword': 'Las cotraseñas no son iguales',
      'validate_gender': 'Debe colocar su genero',
```

5.5. Pruebas Unitarias Sprint 1

Se registrarán las pruebas de funcionalidad realizadas a lo largo del desarrollo del proyecto, estas pruebas permiten encontrar errores de manera rápida para solucionarlos y poder realizar la integración adecuada y progresiva de todo el aplicativo.

Tabla 10: *Pruebas Unitarias Sprint 1*

Objetivo	Resultados esperados	Conclusión
Visualización de Catálogo	Pendiente	Se logró que los productos se puedan visualizar en el orden que son ingresados por el usuario registrado.
Registro de Usuario	Exitoso	Se logró crear una pantalla para el registro de usuario donde puede ingresar su información personal.
Creación de Cuenta	Exitoso	Se logró que todos los campos de registro almacenen la información del usuario en la base de datos.
Inicio de Sesión	Exitoso	Se logró que una vez registrado el usuario pueda iniciar sesión para luego presentar la pantalla de añadir productos.
Añadir Producto	Exitoso	La aplicación permite añadir varias imágenes, descripción, descuentos y precio. Sin embargo, la visualización de productos en el catálogo todavía está muy desorganizada.
Traducción ES/ EN de componentes	Exitoso	Se logró cambiar de idioma los componentes que conforman la interfaz sin embargo para hacer una traducción de los productos sería necesario solicitar al usuario que escriba la descripción del producto en ingles

5.6. Sprint Review 1

¿Que salió bien?

- Las pruebas de funcionamiento fueron en su mayoría satisfactorias permitiendo avanzar a tiempo con el proyecto.
- El desarrollo del frontend del aplicativo se realiza de manera rápida ya que, al tratarse de widgets, la lógica de un elemento albergando a otro es muy fácil de programar.
- El almacenamiento de usuarios y productos en la base de datos fue exitoso gracias a la compatibilidad de Flutter y Firebase.

¿Que salió mal?

- Al inicio del desarrollo se tenía problemas de compatibilidad con el lenguaje Dart ya que algunas dependencias estaban desactualizadas y no se podía ejecutar el aplicativo correctamente.
- La pantalla de catálogo estaba muy desorganizada y no recuperaba adecuadamente la información de los productos almacenados en Firebase.
- Se necesito mucha investigación para la implementación del acceso de la app a los archivos multimedia para seleccionar las imágenes de los productos

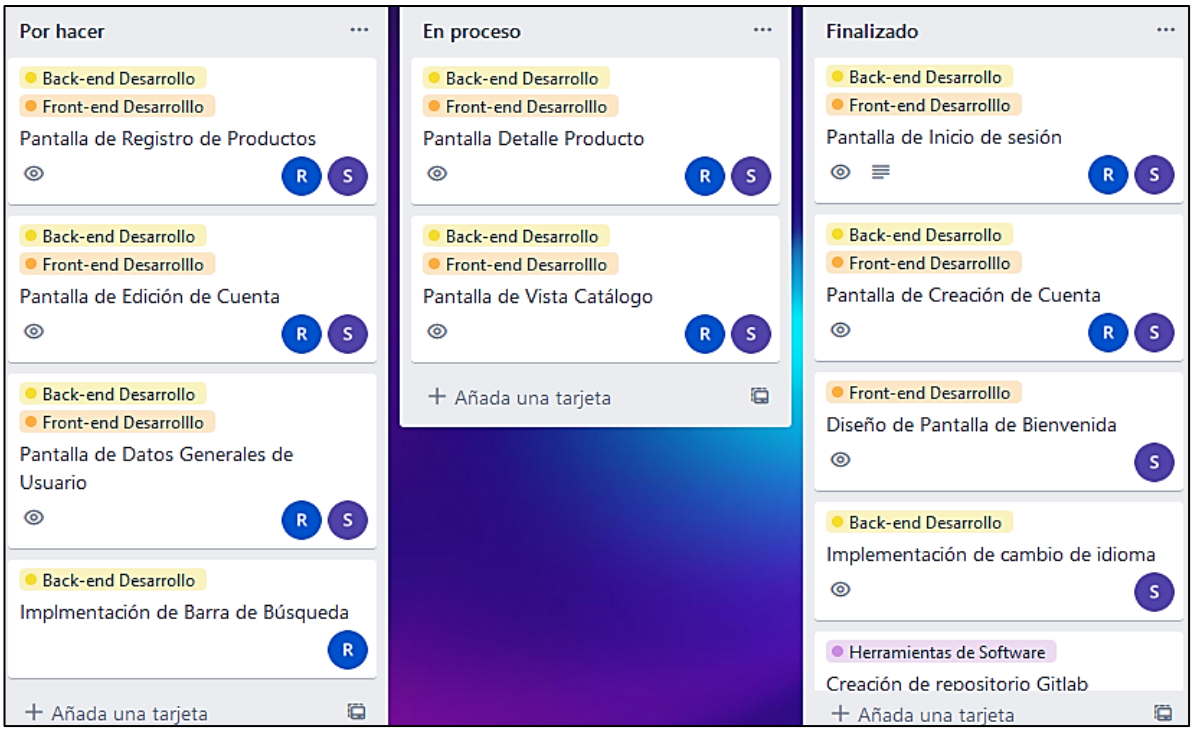
Propósito de mejora.

- Al trabajar con una traducción manual es importante recordar que todos los elementos de la interfaz no debe ser tipo string si no un objeto clave/valor.
- Implementar notificaciones en la pantalla que avise al usuario su creación de cuenta exitoso y su registro de producto en la app.

5.7. Kanban Finalizado Sprint 1

Se ha finalizado de manera correcta la mayoría de las actividades definidas en el Sprint 1, existen algunos problemas que se pudieron apreciar al momento de realizar las pruebas de funcionamiento que serán arreglados y mejorados en el Sprint 2.

Figura 41: *Kanban Finalizado Sprint 1*



CAPÍTULO VI: SPRINT 2

6. Sprint 2

Una vez finalizado el Sprint 1 como se analizó en el Sprint review el trabajo es corregir problemas de funcionalidad y trabajar sobre las actividades pendientes que se enfocan en la publicación de productos por parte del usuario.

6.1. Sprint Backlog 2

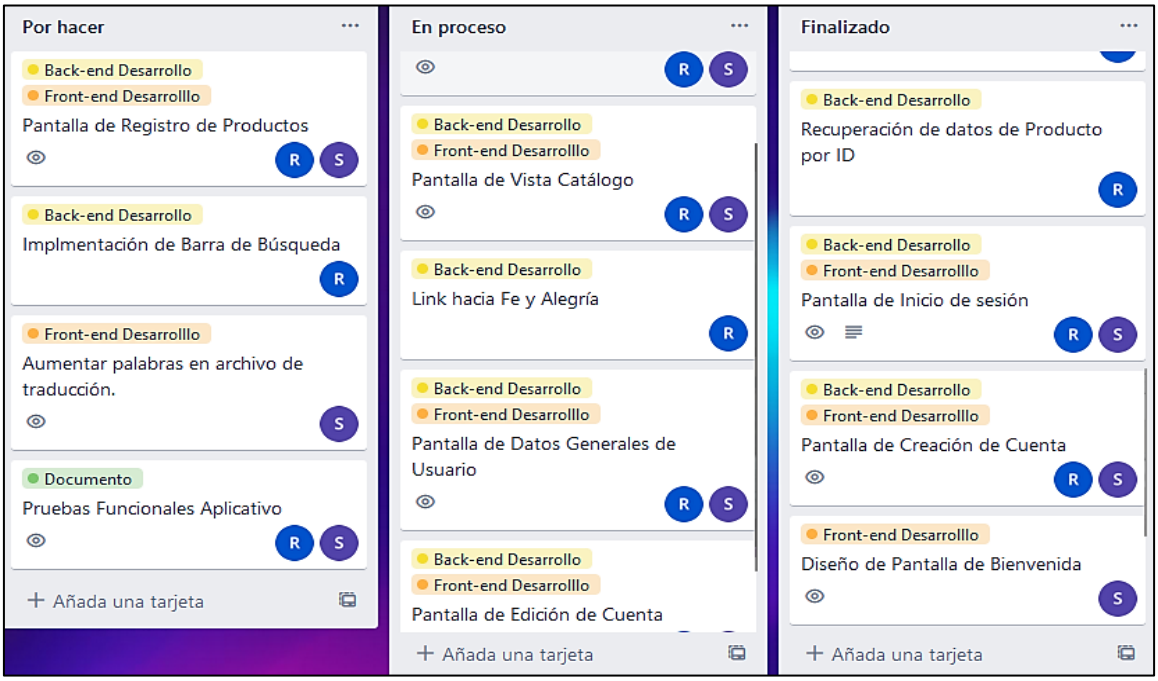
Tabla 11: *Sprint Backlog 2*

SPRINT BACKLOG 2			
Tarea ID	**Historia**	**Estimado (Días)**	**Prioridad**
3	Como usuario cliente y registrado, deseo poder ingresar de forma fácil al catálogo de productos.	**3**	**1**
5	Como usuario registrado, deseo poder almacenar información de mis productos.	**5**	**2**
6	Como usuario cliente y registrado, deseo visualizar a detalle toda la información registrada del producto.	5	**1**

6.2. Kanban Finalizado Sprint 2

Se plantean las actividades que realizarán en el siguiente y último Sprint, se considera a este como el último Sprint debido al poco tiempo restante para la entrega del proyecto, por tanto, es necesario tener claro las tareas de desarrollo que faltan por cumplir para realizar pruebas y corregir errores.

Figura 42: *Kanban Inicial Sprint 2*



6.3. Interfaces Sprint 2

a.) Pantalla de Catálogo

Se ha logrado corregir la organización de los widgets, se ha implementado un carrusel de imágenes que presentan los productos con descuentos, se cuenta con una barra de búsqueda, un icono de lista para filtrar productos registrados por usuario registrado. Y luego se encuentran todos los demás productos promocionándose, indicando una imagen inicial del producto, la categoría que pertenecen como calzado, camisetas, pantalones, entre otros y el precio normal o aplicado el descuento.

Figura 43: *Pantalla de Catálogo*

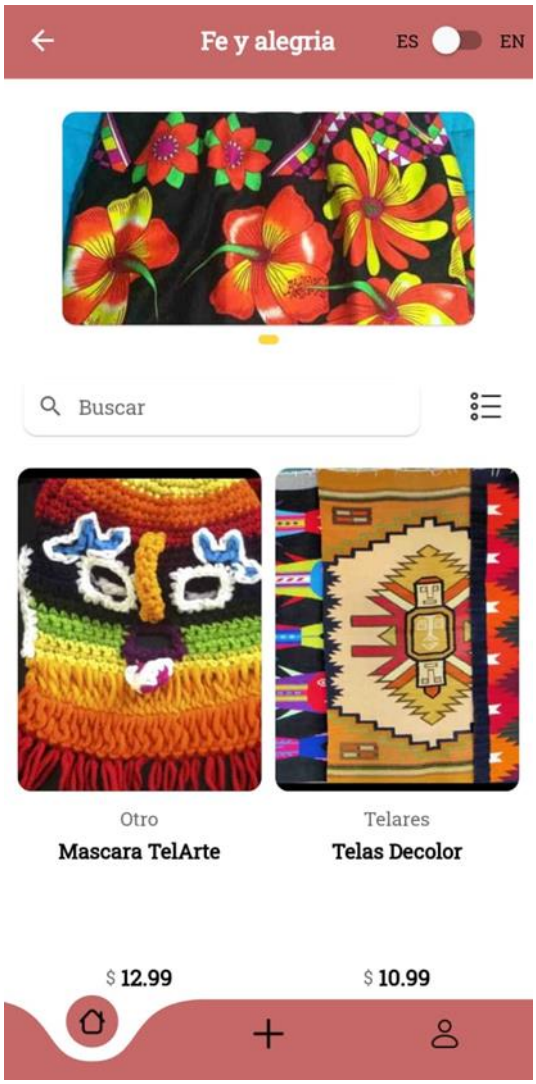


Figura 44: *Fragmento código controlador de vista de Catalogo*

```dart
@override
void onInit() { // Se ejecutación al inicio antes de contruirse el Widget
  super.onInit();
  resetSearch(1);
  getListProduct();
}

//Este método consigue la lista de productos completa  y retorna un Stream de datos de Lista
Stream<List<ProductModel>> streamProducts() {
  return store.collection('Productos').snapshots().map((QuerySnapshot query) {
    List<ProductModel> l = [];
    for (var element in query.docs) {
      l.add(ProductModel.fromJson(element.data()));
    }
    return l;
  });
}
//Establece el filtro para mostrar todos los productos 1 = Todos los productos, 2= Mis productos
void resetSearch(index) {
  search = index;
  update();
}

void getListProduct() {
  isLoadingProduct.value = false;
  _listProduct.bindStream(streamProducts());  //Contruye um Stream de Lista de productos
  isLoadingProduct.value = true;
  update();
}
```

Figura 45: *Fragmento código pantalla*

```dart
class GridViewWidget extends StatelessWidget {

  //El Widget Personlizado GridViewWidget es el responsable de mostrar la lista de productos
  //Recibe como argumento la lista de Productos y otro parametros relacionados a la UI del Widget
  const GridViewWidget({
    super.key,
    required this.spacing,
    required this.crossAxisCount,
    required this.childAspectRatio,
    required this.list,
  });

  final double spacing;
  final int crossAxisCount;
  final double childAspectRatio;
  final List<ProductModel> list;

  @override
  Widget build(BuildContext context) {
    return GridView.builder(
      itemCount: list.length,
      padding: EdgeInsets.all(spacing),
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
          crossAxisCount: crossAxisCount,
          childAspectRatio: childAspectRatio,
          mainAxisSpacing: spacing,
          crossAxisSpacing: spacing), // SliverGridDelegateWithFixedCrossAxisCount
      shrinkWrap: true,
      physics: const NeverScrollableScrollPhysics(),
      itemBuilder: (context, index) {
        ProductModel model = list[index];
        return ProductTile(model: model, index: index);
```

b.) Barra de Búsqueda de Productos

Se presenta la funcionalidad de búsqueda ubicada después del carrusel de imágenes con descuento y los productos de usuario, esta barra facilita a un usuario cliente o registrado encontrar un producto en específico o volver a visualizar un producto que atraiga su atención.

Figura 46: *Barra de Búsqueda de Productos*

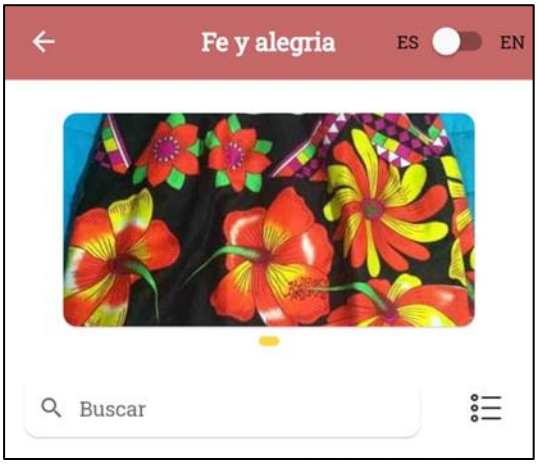


Figura 47: *Fragmento de código Barra de búsqueda*

```
// Controlador del campo de busqueda
TextEditingController searchController = TextEditingController();

@override
Widget build(BuildContext context) {
  int crossAxisCount = 2;
  double childAspectRatio = 0.48;
  double spacing = 10;
  Map value = Get.arguments; // recibe la lista de producto del home page en forma de Map
  List<ProductModel> listProducts = value['products']; // obtiene la lista de productos
  return Scaffold(
    appBar: AppBarWidget(context: context, title: 'Buscar'),
    body: Column(
      children: [
        TextFormFieldWidget(
          controller: searchController,
          hint: 'Buscar',
          onChange: (value) { // Función que es invocada cada vez que se cambia un caracter
            List<ProductModel> l;
            // l va a almacenar los elementos que coincidan con el input y el nombre del producto
            // se utiliza el metodo where para comparar los valores
            l = listProducts.where((element) {
              final name = element.name.toLowerCase();
              final input = value.toLowerCase();
              return name.contains(input);
            }).toList();
            setState(() {
              // operador ternario si no hay input lista vacia, caso contraria retorna l
              newList = value.isEmpty ? [] : l;
            });
          },
        ), // TextFormFieldWidget
```

c.) Visualización de Productos por Usuario

Se presenta la funcionalidad de visualización de los productos publicados por usuario registrado, se encuentra al lado derecho de la barra de búsqueda, con el icono de lista. Esta funcionalidad permite filtrar los productos específicos de un usuario registrado que ha iniciado sesión en el aplicativo.

Figura 48: *Visualización de Productos de Usuario*

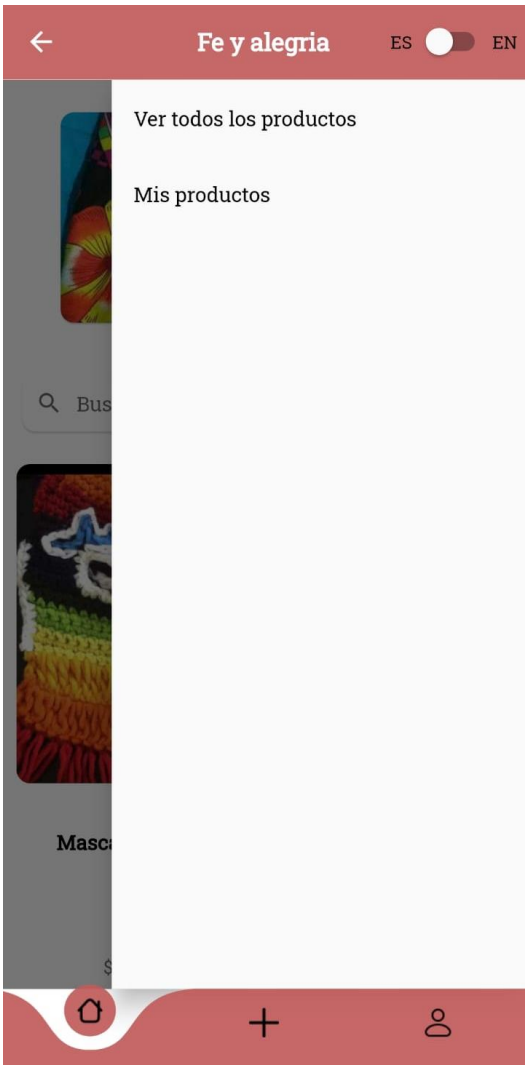


Figura 49: *Fragmento de código de Visualización de Productos de Usuario*

```
endDrawer: Drawer( // Widget que crea un scafold adicional donde poner elementos
    child: ListView(
  children: [
    //selecciona todos los productos, Modo predeterminado
    ListTile(
      title: TextWidget(context: context, text: 'all'.tr),
      onTap: () {
        controller.resetSearch(1); // funcion para filtrar productos por token de usuario
        Navigator.pop(context);
      },
    ), // ListTile
    //Verifica que el usuario este logeado para mostrar la opcion 'Mis productos'
    FirebaseAuth.instance.currentUser != null
        ? ListTile(
            title: TextWidget(context: context, text: 'my_products'.tr),
            onTap: () {
              controller.resetSearch(2);
              Navigator.pop(context);
            },
          ) // ListTile
        : const SizedBox.shrink(),
  ],
)), // ListView // Drawer
```

d.) Pantalla de Detalle de Producto

El usuario registrado después de iniciado sesión puede visualizar el catálogo, donde encontrará gran cantidad de productos, si uno de ellos le llame la atención lo seleccionará y desplegará la pantalla que se presenta a continuación.

En esta pantalla se puede observar imágenes del producto, el título o nombre del producto, tamaños, una descripción del producto o historia de este, precios, nombre del fabricante.

Por último, se dispone del botón de pedido el cual redirige al usuario cliente y registrado hacia la página de Fe y Alegría donde puede contactar con el departamento de ventas y concretar la compra.

Figura 50: *Pantalla de Detalle de Producto*

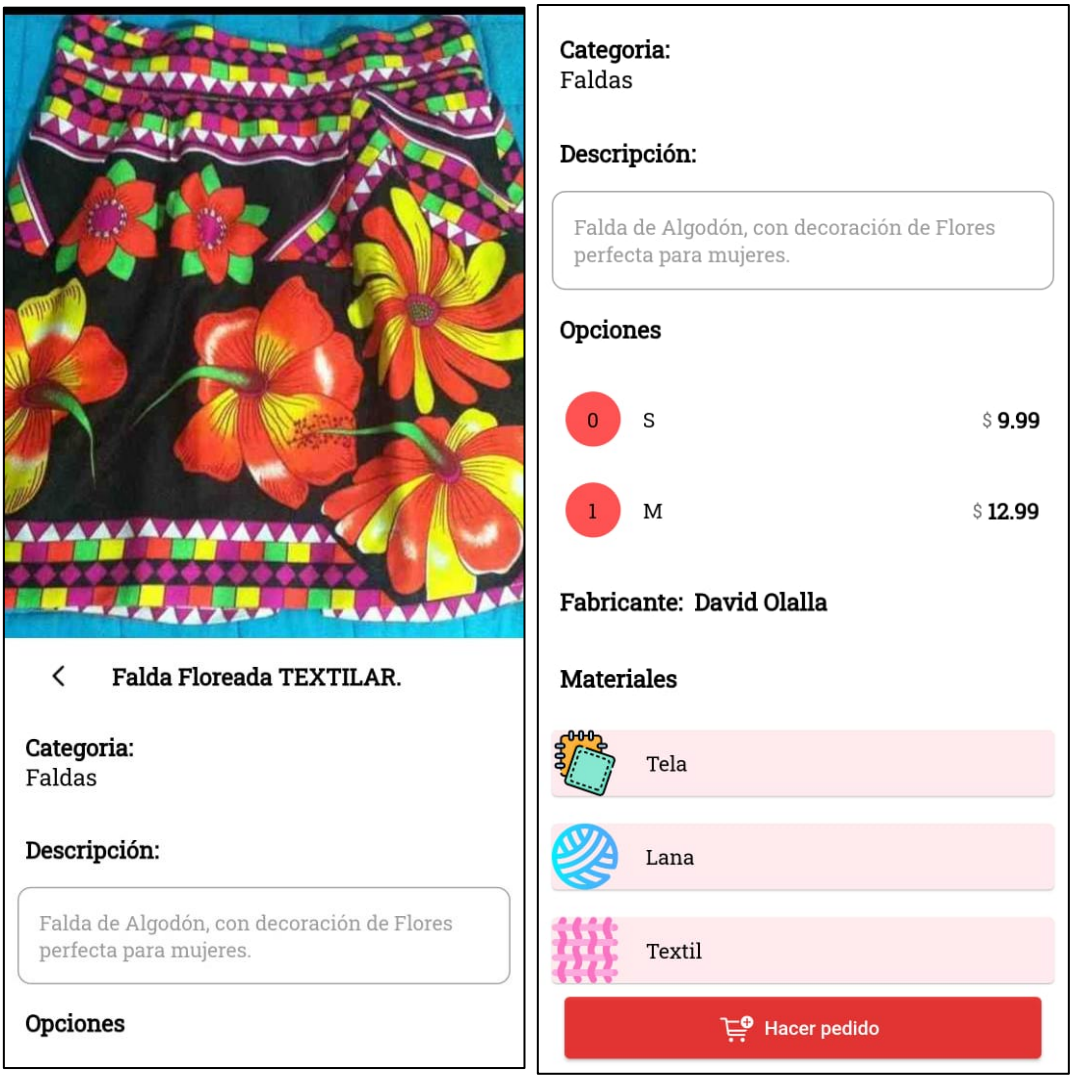


Figura 51: *Pagina web de Fe y Alegría*

Figura 52: *Fragmento de código Controlador Detalle de Producto*

```
class DetailsPageController extends GetxController {
  late String userid; // token necesario para recuperar datos del usuario
  //variables que se van a mostrar junto a la información del detalle del producto
  var productOwnerName = ''.obs;
  var productOwnerSurname = ''.obs;
  var isLoadingPage = true.obs; // indica si se esta cargando la pagina

  void loadPage(String userid) {
    this.userid = userid;
    getCurrentUser();
  }

  // esta función recuperar los datos del usuario que publicó el producto en cuestión
  Future<void> getCurrentUser() async {
    UserModel model;
    model = await FirebaseFirestore.instance
        .collection('Usuarios')
        .doc(userid)
        .get()
        .then((value) {
      return UserModel.fromJson(value.data()!);
    });
    productOwnerName.value = model.name!;
    productOwnerSurname.value = model.surname!;
    isLoadingPage.value = false;
  }
}
```

Figura 53: *Fragmento de código de Detalle de Producto*

```
// Si la pagina se encuentra cargando no mostrara el Scaffoold de la pantalla
return Obx(() => !detailsController.isLoadingPage.value
    ? SafeArea(
        child: Scaffold(
          body: SingleChildScrollView(
            padding: const EdgeInsets.only(bottom: 15),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                SizedBox(
                  width: size.width,
                  height: size.width * 1.2,
                  child: PageView.builder( // CAROUSEL DE IMAGENES DE PRODCUTO
                    itemCount: model.listImages.length,
                    itemBuilder: (context, index) {
                      ImagesModel image = model.listImages[index];
                      return Image.network(
                        image.image,
                        fit: BoxFit.cover,
                      ); // Image.network
                    },
                  ), // PageView.builder
                ), // SizedBox
                ListTile(  // NOMBRE DEL PRUDUCTO
                  leading: IconButtonWidget(context: context),
                  title: TextWidget(
                    context: context,
                    text: model.name,
                    fontSize: 17,
                    fontWeight: FontWeight.bold,
                  ),
```

e.) Pantalla de Detalles de Usuario

Como tercer elemento en la barra inferior de desplazamiento, se encuentra el icono de usuario, donde después de haber realizado el registro se podrá visualizar, los datos que han sido ingresados por el usuario registrado.

Figura 54: *Pantalla de Detalles de Usuario*

Figura 55: *Fragmento de código Detalles de Usuario*

```
class ProfileController extends GetxController {
  //Intancia un objeto tipo Usuario con datos provisionales antes de actualizarlo con los rea
  UserModel user = UserModel(
    age: 0,
    email: 'cargando...',
    gender: 'cargando...',
    name: 'cargando...',
    phone: 0,
    surname: 'cargando...',
    image:
        'https://openclipart.org/image/400px/326561',
  );

  //Esta función es la primera que se ejecuta antes de contruir el Widget
  @override
  void onInit() {
    super.onInit();
    getUserData(); // Obtiene los datos del usuario
  }

  // Obtiene los datos del Usuario mediante una instancia a Firebase
  Future<UserModel> getUserData() async {
    UserModel model;
    model = await FirebaseFirestore.instance
        .collection('Usuarios')
        .doc(FirebaseAuth.instance.currentUser!.uid) // usa como id el usuario actual
        .get()
        .then((value) {
      return UserModel.fromJson(value.data()!);
    });
```

f.) Pantalla de Edición de Cuenta de Usuario

Una vez que el usuario registrado verifique su información personal en la pantalla de detalles, en caso de que la información necesite ser actualizada se encuentra el icono de edición, debajo de su nombre lo cual despliega una pantalla adicional donde la información puede ser modificada y actualizada en la base de datos.

Figura 56: *Pantalla de Edición de Cuenta*

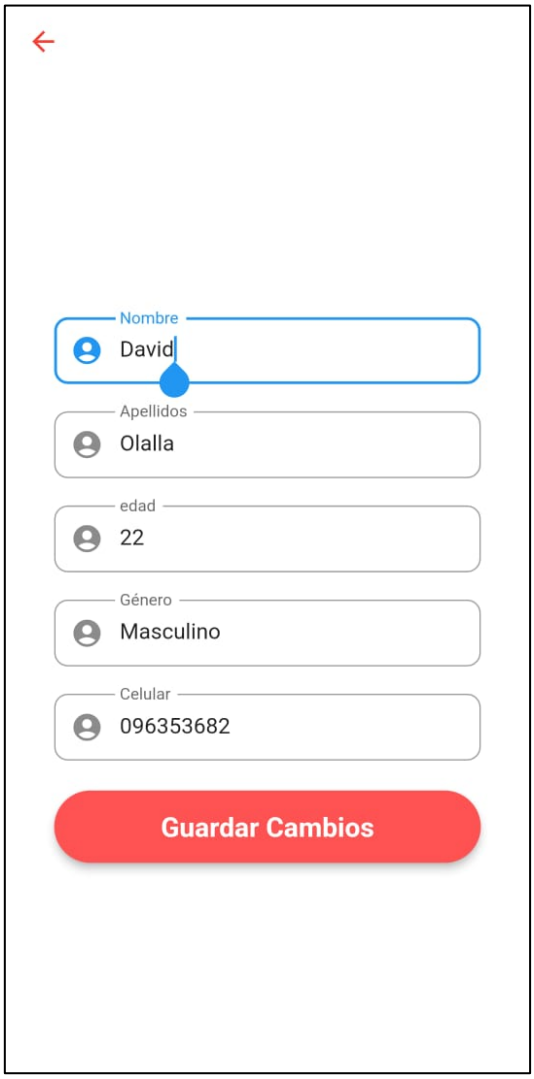
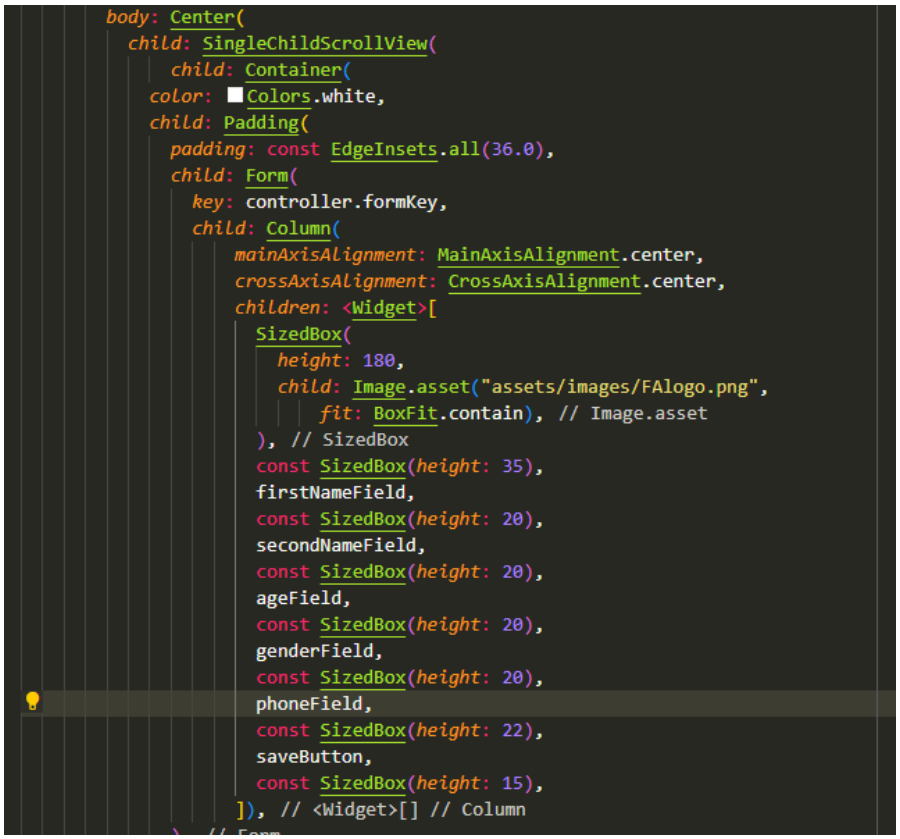


Figura 57: *Fragmento de código Edición de Cuenta*



```
body: Center(
  child: SingleChildScrollView(
    child: Container(
  color: Colors.white,
  child: Padding(
    padding: const EdgeInsets.all(36.0),
    child: Form(
      key: controller.formKey,
      child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: <Widget>[
            SizedBox(
              height: 180,
              child: Image.asset("assets/images/FAlogo.png",
                  fit: BoxFit.contain), // Image.asset
            ), // SizedBox
            const SizedBox(height: 35),
            firstNameField,
            const SizedBox(height: 20),
            secondNameField,
            const SizedBox(height: 20),
            ageField,
            const SizedBox(height: 20),
            genderField,
            const SizedBox(height: 20),
            phoneField,
            const SizedBox(height: 22),
            saveButton,
            const SizedBox(height: 15),
          ]), // <Widget>[] // Column
```

6.4. Pruebas Unitarias Sprint 2

Se registrarán las pruebas de funcionalidad realizadas a lo largo del desarrollo del proyecto, estas pruebas permiten encontrar errores de manera rápida para solucionarlos además de poder realizar la integración adecuada y progresiva de todo el aplicativo.

Tabla 12: *Pruebas Unitarias Sprint 2*

Objetivo	Resultados esperados	Conclusión
Visualización de Catálogo	Exitoso	Se logró que los productos se puedan visualizar en el orden que son ingresados por el usuario registrado.
Barra de Búsqueda de Productos	Exitoso	Se logró recuperar los productos con su nombre, presentándose adecuadamente
Detalle de Producto	Exitoso	Se logró mostrar una pantalla para cada producto registrado se pueda desplegar todas las características añadidas por el usuario registrado.
Enlace a Página web de Fe y Alegría	Exitoso	Se logró agregar correctamente el vínculo en el botón de realizar pedido, ya que al ser pulsado redirige a esta página de inicio donde el usuario cliente y registrado podrá concretar la venta del producto del cual está interesado.
Visualización de detalles de usuario	Exitoso	Se logró recuperar de forma adecuada la información del usuario que se almacena al crear una cuenta, presentándola en la pantalla de detalles de usuario.
Edición de Cuenta usuario	Exitoso	Se logró de manera satisfactoria crear un botón en la pantalla de detalles de usuario, donde se colocó un botón de edición que presenta campos de información personal que deben ser actualizados, una vez que sean cambiados se notifica al usuario registrado y los valores de la base de datos se cambian.

6.5. Sprint Review 2

¿Que salió bien?

- Se pudo implementar la validación contraseña de al menos 6 caracteres en la pantalla de creación de cuenta.
- Se eliminaron pantallas que se utilizaban de plantillas para probar funcionalidades y demoraban el proceso de compilación de aplicativo.
- Se cambio de forma adecuada el icono del aplicativo el cual en un principio era aquel que viene por defecto de Flutter, una vez cambiado el icono se puede identificar de mejor manera dentro del dispositivo móvil.
- Se investigó y analizó plantillas de Flutter para implementar la funcionalidad de búsqueda, la cual funcionó correctamente mejorando la presentación del catálogo de productos.

¿Que salió mal?

- Por falta de tiempo no se pudo implementar una API de pagos en el botón de pedido.
- Existía un error de que si el usuario registrado se encontraba en la vista “Mis productos” y salía de su cuenta. Al momento de volver al catalogó general se mostraba una pantalla vacía ya que no encontraba la cuenta a la que se estaba vinculada.
- Se debe llenar todos los campos al momento de añadir un producto del contrario ocurre que son obligatorios y el producto no se almacena, mientras que en el aplicativo no permitirá avanzar a otras pantallas mientras el campo esté vacío.

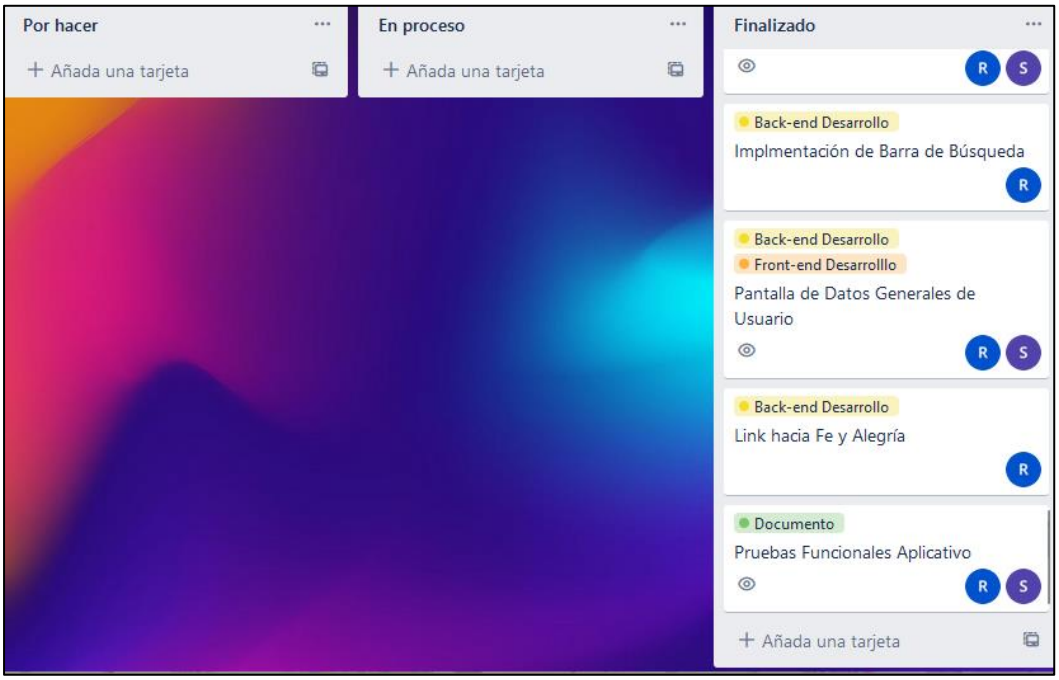
Propósito de mejora.

- Se propone para un trabajo a futuro implementar una API de pago en línea para concretar mejor la interacción del usuario cliente y registrado con el aplicativo, ya que por el momento solo esta implementado la redirección hacia página web de inicio de Fe y Alegría.
- Se puede ampliar el almacenamiento de Firestore en caso de que la cantidad de usuarios registrados aumente.

6.6. Kanban Finalizado Sprint 2

A continuación, se presenta el Tablero con todas las tareas tanto del primer como del segundo Sprint completadas, finalizando el proceso de desarrollo.

Figura 58: *Tablero Kanban Finalizado*



CAPÍTULO VII: INTEGRACIÓN

7. Pruebas de Integración

Integración por Hilos

F1.1. Visualización de Catálogo General.

Hilo: El usuario cliente y registrado desea visualizar todos los productos registrados en el aplicativo.

Pantallas que apoyan el proceso:

- Home_binding.dart
- Home_controller.dart
- Home_list_product.dart
- Home_page.dart

Resultados esperados: El usuario cliente o registrado abre la aplicación, visualiza la pantalla de bienvenida y de manera inmediata puede navegar a través del catálogo general de productos, que están almacenados en la base de datos.

F1.2. Visualización de Catálogo de usuario registrado.

Hilo: El usuario registrado presiona icono de tres líneas y selecciona para ver sus productos

Pantallas que apoyan el proceso:

- Home_binding.dart
- Home_controller.dart
- Home_list_product.dart
- Home_page.dart

Resultados esperados: El usuario registrado accede a la opción de visualización de catálogo de sus productos desde el icono de tres líneas a un lado de la barra de búsqueda y se muestra una pantalla con las opciones de 'Todos los productos y 'Mis Productos'. El usuario registrado selecciona la segunda opción y se desplaza la pantalla y se muestran los productos de catalogo del usuario registrado.

F2. Creación de cuenta

Hilo: El usuario cliente accede a esta pantalla llena los campos y se crea una cuenta

Pantallas que apoyan el proceso:

- Sign_up_controller.dart
- Sign_up_page.dart
- No_auth_page.dart

Resultados esperados: El usuario cliente accede a la pantalla, se le muestra un formulario de varios campos como nombre, email, teléfono, edad, género y otros. Los campos están válidos para no poner datos vacíos y un campo de confirmación de contraseña. Finalmente, luego de que los campos estén verificados el usuario cliente presiona el botón de crear cuenta y automáticamente está autenticado con la cuenta recién creada. El usuario cliente se convierte en usuario registrado

F3. Inicio de Sesión.

Hilo: El usuario registrado accede a la pantalla de inicio de sesión y se autentica con email y contraseña

Pantallas que apoyan el proceso:

- Sign_in_controller.dart
- Sign_in_page.dart
- No_auth_page.dart
- Sign_binding.dart

Resultados esperados: El usuario registrado accede a la pantalla de inició de sesión, se le muestra un pequeño formulario con dos campos para llenar email y contraseña. Si los campos están vacíos o no las credenciales no son válidas se

notifica al usuario registrado con un mensaje. Finalmente se autentica con el botón de iniciar sesión y se encuentra en la vista de catálogo.

F4. Agregar Productos.

Hilo: El usuario registrado usa el menú de navegación y se dirige a añadir producto para añadir un producto. Llena campos, sube imágenes y añade el producto mediante el botón de crear producto.

Pantallas que apoyan el proceso:

- Add_product_binding.dart
- Add_product_controller.dart
- Add_product.image.dart
- Add_product_page.dart
- Add_text_form_field.dart

Resultados esperados: El usuario registrado se dirige mediante el menú de navegación a la pantalla de datos de añadir Producto. Se le muestra los campos necesarios y obligatorios para añadir producto (nombre, fotos categoría, descripción etc.) Existe una validación para campos sin llenar. Para subir fotos el usuario registrado tiene dos opciones o bien selecciona de la galería o toma fotos directamente con la app. Al final presiona el botón de agregar producto.

F5. Detalles de Productos

Hilo: El usuario cliente o registrado presiona en un producto de su interés y visualiza la información detallada del producto

Pantallas que apoyan el proceso:

- Details_page_controller.dart
- Details_page.dart

Resultados esperados: El usuario cliente o registrado presiona sobre la imagen del Producto y visualiza los datos de detalle. Nombre, Categoría, Descripción, Proveedor, Talla y Precio. Posteriormente puede volver al menú de catalogo presionando la flecha de regreso.

F6. Datos de Perfil de Usuario

Hilo: El usuario registrado usa el menú de navegación y se dirige a datos personales para verificar sus datos

Pantallas que apoyan el proceso:

- Profile_binding.dart
- Profile_user_page.dart
- Profile_controller.dart

Resultados esperados: El usuario registrado se dirige mediante el menú de navegación a la pantalla de datos de Perfil de Usuario. Una vez aquí puede cerrar sesión presionando el botón de cerrar o sesión o pueda ir a la pantalla de edición de cuenta presionando el icono de Edición debajo de la inicial de su nombre.

F7. Edición de Cuenta

Hilo: El usuario registrado abre la pantalla verifica sus datos, cambia los datos necesarios y guarda cambios.

Pantallas que apoyan el proceso:

- editProfile_controller.dart
- editProfile_page.dart

Resultados esperados: El usuario registrado abre la pantalla a través de la vista de Perfil luego se le muestran todos los campos editables de sus datos personales. Luego de llenar sus datos si se han dejado campos vacíos no se permite actualizar, después automáticamente lleva a la pantalla de datos personales mostrando los datos actualizados.

7.1. Instalación

Actualmente la instalación del aplicativo es posible mediante un apk generada por el IDE Visual Studio Code, por ahora no estará disponible en ninguna tienda de

aplicaciones móviles debido a que esta aplicación forma parte de un proyecto de titulación, por tanto, los derechos los posee la Pontifica Universidad Católica del Ecuador.

La aplicación soportará dispositivos que tengan un sistema operativo con un api mínimo 33. Por último, en caso de ser necesario se debe activar la opción de *permitir aplicaciones de orígenes desconocidos.*

CONCLUSIONES Y RECOMENDACIONES

- **Conclusiones**

1. La gestión de estados permite alinear e integrar la lógica comercial o de negocio dentro de la aplicación ya se mediante el uso de librerías o código puro de alto nivel, como es Dart en el caso presente. El rendimiento de la aplicación es considerablemente afectado sin una correcta gestión de estados ya que está ligado a la arquitectura de tres capas del modelo cliente-servidor. Un código desorganizado siempre es más difícil de mantener, actualizar o corregir.
2. La programación en el lenguaje de programación Dart ha demostrado ser un lenguaje muy flexible, lleno de herramientas y librerías que facilitan el desarrollo por lo que fue el adecuado al momento de trabajar con la metodología Scrum
3. La metodología Scrum proporciona una buena base para el desarrollo de del aplicativo, sin embargo, algunas prácticas o artefactos de Scrum han sido adaptados y optimizados al tamaño del equipo de trabajo, complejidad del aplicativo y tiempo desarrollo.
4. La documentación es un parte importante del proyecto y metodología que debió empezar a desarrollarse desde el inicio. En este todos los procesos no fueron

documentados en el tiempo debido y eso conllevó a algunos retrasos en el desarrollo y la entrega de los sprint.

- **Recomendaciones**

1. En un proyecto de desarrollo el tiempo es un elemento muy valioso, si no se planifica de forma adecuada una actividad puede causar problemas para la entrega del aplicativo, por eso la importancia del tablero Kanban ya que permite llevar un control sobre las tareas que han sido completadas o están en proceso.
2. Es importante recalcar que, para el comienzo del proyecto, la programación debe ser desde el backend hacia el frontend y no viceversa, ya que depender de los elementos de interfaz para programar la funcionalidad complica el proceso de desarrollo. Una vez que se tenga la funcionalidad solo se debe personalizar el widget para poder hacer la integración de la aplicación.
3. Flutter es un SDK que se actualiza constantemente, ya que siempre busca simplificar el proceso de desarrollo, por tanto, es importante que antes de comenzar un proyecto se realice una comprobación en la consola de comandos para verificar que se esté trabajando con la versión más nueva y óptima.
4. Se recomienda trabajar con una arquitectura limpia en Flutter, de otra manera puede provocar confusiones dentro del equipo de desarrollo como múltiples archivos de widgets y código redundante, los cuales implican demoras en tiempo de ejecución y aumento de errores de código.
5. Se recomienda que en caso de continuar con el desarrollo de este proyecto se debe incluir reglas de seguridad para la base de datos, además funciones de nube para limpiar el almacenamiento cuando se elimine los productos.

BIBLIOGRAFÍA

- *6 Main Benefits of Flutter Mobile App Development in 2022 | Kindgeek*. (2022, 7 abril). KindGeek. Recuperado 7 de septiembre de 2022, de https://kindgeek.com/blog/post/6-flutter-benefits-that-made-it-the-top-choice-for-mobile-app-development-in-2021
- Kingsnorth, S. (2019, 30 abril). *Digital Marketing Strategy: An Integrated Approach to Online Marketing* (2.ª ed.). Kogan Page.
- Google Analytics For Firebase: La solución de reporting para apps mobile. (2018). https://dbibyhavas.io/es/blog/google-analytics-for-firebase/. https://dbibyhavas.io/es/blog/google-analytics-for-firebase/
- Páez, J. A., Cortes, J. A., Simanca, F. A., & Blanco, F. (2021). Aplicación de UML y SCRUM al desarrollo del software sobre control de acceso. Información tecnológica, 32(5), 57-66.
- Kaizenia, K. (2020, 30 octubre). ¿Cómo se realiza un Sprint y cuáles son los beneficios de trabajar en ellos? KZI Kaizenia cursos agile, scrum, six sigma. https://kzi.mx/como-se-realiza-un-sprint-y-cuales-son-los-beneficios-de-trabajar-en-ellos/

- ARQUITECTURA CLIENTE-SERVIDOR: ¿DE QUÉ TRATA? (2020, 16 diciembre). Nota Tecnològica. Recuperado 23 de septiembre de 2022, de https://notatecnologica.com/programas/arquitectura-cliente-servidor/
- Amit, W. & Ganapolsky, I. (2020, 26 febrero). All You Need to Know About UML Diagrams: Types and 5+ Examples. Tallyfy. Recuperado 15 de septiembre de 2022, de https://tallyfy.com/uml-diagram/
- *Firestore* |. (s. f.). Firebase. Recuperado 24 de septiembre de 2022, de https://firebase.google.com/docs/firestore
- Asodariya, T. (2022, 6 enero). *7 things you need to know about Flutter State Management*. Medium. https://medium.com/dhiwise/7-things-you-need-to-know-about-flutter-state-management-42f840ef022
- *get | Flutter Package*. (s. f.). Dart packages. https://pub.dev/packages/get
- Diferencia entre estado efímero y estado de app. (2022). Flutter. https://esflutter.dev/docs/development/data-and-backend/state-mgmt/ephemeral-vs-app
- A sample app that implement Uncle Bob's Clean Architecture in Flutter - Best Flutter apps. (2021, 22 diciembre). Best Flutter apps - Flutter app templates code. http://bestflutterapps.com/a-sample-app-that-implement-uncle-bobs-clean-architecture-in-flutter

GLOSARIO DE TÉRMINOS

SDK: Sus siglas en español significan “Kit de desarrollo de software”, como lo dice su nombre proporciona un conjunto de herramientas, documentación, guías de procesos para que los desarrolladores puedan crear aplicaciones para plataformas específicas.

Desarrollo Backend: Elementos del aplicativo que no son vistos por el usuario, este ámbito abarca el almacenamiento de datos, comunicación de componentes. Como ejemplo: servidores, base de datos, sistemas de pago, API, etc.

Desarrollo Frontend: Elementos visuales que constituyen la experiencia del usuario, Como ejemplos existen: listas desplegables, menús, botones, colores de pantalla, imágenes, etc.

Widget: Se trata de cada elemento con funcionalidad que conforma una pantalla del aplicativo. La visualización de la pantalla depende de la secuencia de widget seleccionados y su funcionamiento del árbol de widgets.

UI: Se trata de la interacción del humano con los componentes de un aplicativo. Como ejemplo existen las pantallas de visualización, botones, campos de texto.

Firestore: Se trata de una base de datos en la nube NoSQL utilizada para el desarrollo de aplicativos móviles, web y servidor. Su almacenamiento de información es en tiempo real y puede funcionar sin necesidad de conexión a internet.

Interfaz: Es un concepto que permite abstraer casos de uso de la lógica de negocio de su implementación en el sistema. Es decir, objetos hacen de “molde” o “guía” para que las clases que sí concreten estos métodos sigan las definiciones de las reglas de negocio.

Query: Se refiere a una solicitud de información, que requiere un conjunto de código para que la base de datos de un aplicativo entienda una instrucción.

StatefulWidget: Se trata de widgets que pueden cambiar sus propiedades gracias a eventos desencadenados por su interacción con el usuario. Ejemplo: botones, checkbox y campos de texto.

Scaffold: Se trata de una clase de Flutter que proporciona muchos widgets, que ocuparán el espacio disponible. Este funciona como un marco de referencia para el diseño de aplicativos.

Excepción: Se trata de un evento que se produce durante la compilación de un programa, pero interrumpe el flujo definido de instrucciones.

ANEXOS

Anexo A: Link al repositorio del Proyecto

https://gitlab.com/ELRichi/mi-qatu_comercio-electronico/-/tree/main