PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

CARRERA DE INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Trabajo de Titulación

Tema: Desarrollo de un prototipo de aplicación móvil y sistema web para el seguimiento y análisis del ciclo logístico de bouquets de flores mediante códigos QR.

AUTOR:

DANILO ISMAEL ARIAS VILLALBA

TUTOR:

MSC. EDISON MORA

QUITO DM, ENERO DE 2026

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO I: INTRODUCCIÓN

1.1. Justificación

La experiencia operativa dentro del área de Investigación y Desarrollo (R&D) de una empresa productora y exportadora de flores ha permitido seguir de cerca la problemática de desperdicio de flor durante el empaque y traslado del producto.

Un caso de desperdicio de flor consiste en la extracción y desecho de bouquets afectados por maltrato, quemaduras en pétalos o tallo y/o presencia de patologías como botrytis, caracterizada por la aparición de una masa de esporas grises y descomposición de los tejidos, proliferando especialmente en ambientes con humedad elevada y escasa circulación de aire (Cárdenas, 2025), lo que impide que las especies florales cumplan estándares de calidad necesarios para su comercialización.

La industria florícola constituye un pilar fundamental para la economía ecuatoriana, ubicando al país como el tercer mayor exportador a nivel global con ingresos que superaron los 1.000 millones de dólares en 2024, aportando una cuota significativa tanto al PIB agrícola como al no petrolero (Betancourt Espinoza et al., 2025); por lo tanto, optimizar el ciclo logístico del producto floral desde su cadena de abastecimiento hasta su entrega, es fundamental para minimizar pérdidas y garantizar la máxima calidad del producto exportado, manteniendo así un puesto constante en el mercado global.

Sin embargo, los métodos tradicionales de control de calidad como registros manuales y reportes asincrónicos para la toma de decisiones, tiene como resultado la falta de datos en tiempo real, generando una brecha para la detección de fallos, es aquí donde se presenta la necesidad de implementar soluciones tecnológicas ágiles que digitalicen el proceso de trazabilidad de los productos.

Bajo el contexto mencionado, este trabajo de titulación propone el desarrollo de un prototipo de aplicación móvil y sistema web que permita registrar, monitorear y analizar el ciclo logístico de bouquets de flores mediante códigos QR (Quick Response) o de respuesta rápida, impresos y adheridos a la manga o etiqueta del producto, con el objetivo de identificar puntos críticos dónde se genera el desperdicio de flor.

1.2. Planteamiento del problema

El ciclo logístico de la exportación de bouquets de flores tiene su inicio mediante la negociación entre un representante de ventas y un cliente en el mercado internacional, ambas partes tratan ideas para una colección de flores que se venderá en una temporada específica, culminada la negociación el representante de ventas confirma los ítems al área de Investigación y Desarrollo para el surtido de flor de la colección.

R&D cotiza los bouquets de flores que conforman la venta para posteriormente confirmar con el área de Abastecimiento (Supply) la disponibilidad de flor, de este modo se evalúa la posibilidad de una recotización y/o cambios en los componentes del bouquet, se comunica oportunamente al representante de ventas las directrices proporcionadas por Supply para la confirmación o renegociación de la colección final que va a desarrollarse. La figura 1 representa gráficamente el flujo de desarrollo de una colección de bouquets de flores.

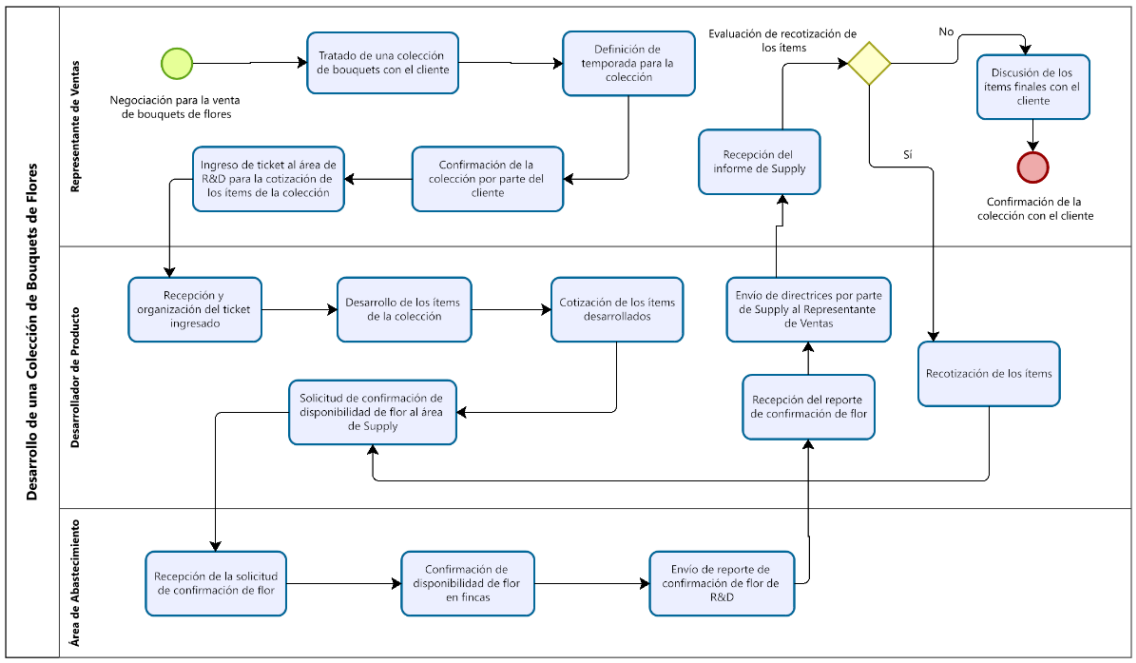


Figura 1. Diagrama de flujo de desarrollo de una colección de bouquets de flores.
Nota. Elaboración propia a través de Bizagi Modeler. Diagrama diseñado con información referencial con el fin de resguardar la privacidad de la información.

Una vez aprobada la colección, el surtido de flor es ejecutado, las flores son transportadas desde finca hasta el centro de acopio (origen), en esta etapa los bouquets ahora son mencionados cómo “ítems” de una venta, son armados en base a la receta final aprobada por el representante de ventas y son empacados en diferentes cajas según su estructura, incorporando materiales secos como cartón separador, mangas y etiquetas previo a su despacho aéreo hacia la bodega de distribución en el exterior (destino).

Al arribar a la bodega de distribución, las áreas de Control de Calidad (QA) y Diseño Floral guían a un operario para la evaluación del estado de los ítems, determinando así su calidad comercial. En este filtro crítico los ítems son clasificados como aptos o como desperdicio de flor, aquellos ítems positivos en su control de calidad pueden atravesar un proceso de rearmado con materiales secos distintos o cambio de caja según corresponda.

Durante el ciclo logístico expuesto, el control de los ítems que son aprobados y rechazados es llevado en papel para posteriormente comunicarlo mediante correo electrónico de forma asincrónica culminada la distribución diaria o semanal de los ítems, a las áreas y mandos interesados en dicha información, sin embargo, el detalle completo de los ítems que no son aprobados no es detallado por parte de QA o Diseño Floral a menos que se consulte a las áreas mencionadas los motivos respectivos del caso como se muestra en la figura 2.

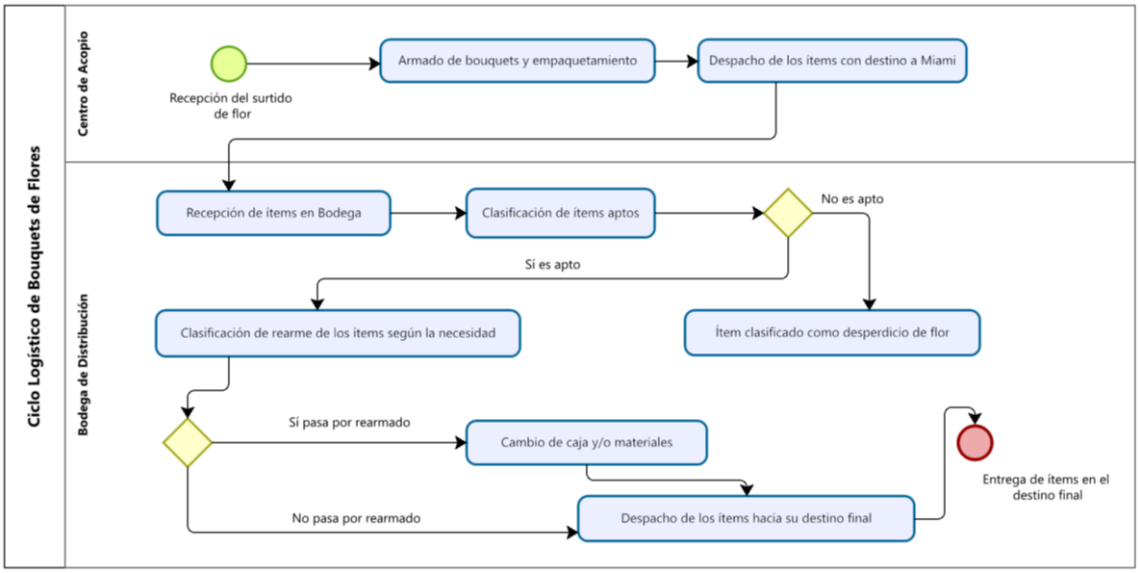


Figura 2. Diagrama de flujo del ciclo logístico de bouquets de flores.
Nota. Elaboración propia a través de Bizagi Modeler. Diagrama diseñado con información referencial con el fin de resguardar la privacidad de la información.

La ausencia de comunicación rápida y efectiva en tiempo real de los ítems descartados y sus motivos sesgan el resto de las colecciones florales que se encuentran en lista de desarrollo, decisiones cómo; cambiar el tamaño de la caja, optimizar la cantidad de ítems que se empacan en esta, probar otro tipo de armado del bouquet, retirar materiales secos, aporta para evitar el desperdicio de flor.

Ante este escenario, se plantea un prototipo de aplicación móvil que registre los ítems que se exportan desde su punto de origen mediante códigos QR codificados con información relevante como identificador, nombre del ítem y colección al que pertenece. Esta herramienta tecnológica permitiría llevar un registro en tiempo real de los bouquets hacia la bodega de destino para digitalizar su control de calidad a su llegada, facilitando así la clasificación de ítems aptos para su venta y aquellos que no cumplen el estándar de calidad en su estado, justificando el por qué son descartados.

A la par, un prototipo de sistema web que centralice los datos de despacho y verificación de los productos en tiempo real, reemplazaría la toma de información en papel para su posterior transcripción y envío de correos electrónicos, mejorando así los tiempos de trabajo y facilitando el despliegue de información de manera ágil para la toma de decisiones.

1.3. Objetivos

1.3.1. General

Desarrollar un prototipo de aplicación móvil y sistema web para el seguimiento y análisis del ciclo logístico de bouquets de flores mediante códigos QR.

1.3.2. Específicos

- Definir la arquitectura de software y el modelo de base de datos del sistema, estableciendo los flujos de información necesarios para el seguimiento del ciclo logístico.
- Desarrollar el prototipo funcional de la aplicación móvil y el sistema web utilizando el framework Flutter y la plataforma Supabase para la gestión de datos en la nube, integrando la generación y lectura de códigos QR para el registro de información.
- Identificar, registrar y analizar las causas de desperdicio de flor en los procesos de exportación y recepción de bouquets.

- Evaluar el funcionamiento y eficiencia del prototipo de la aplicación móvil y el sistema web en la detección y reducción del desperdicio de flor a través de entornos simulados.

1.4. Metodología ágil, SCRUM

SCRUM se consolida como una metodología ágil orientada al desarrollo incremental, conceptualizada como un marco de trabajo que permite abordar problemas complejos y adaptativos para entregar productos de alto valor con eficiencia y creatividad (Martins, 2025). Su objetivo es la organización de tareas dentro de ciclos cortos denominados "sprints", donde se implementan funcionalidades del producto de forma continua, mismas que se encuentran ajustadas a los requerimientos finales.

Los roles definidos para este proyecto bajo el contexto académico y organizacional son los siguientes:

- Product Owner: Msc. Erick Almeida, en calidad de Especialista de Información de R&D (Experto de la Industria). Su función es validar que las historias de usuario y funcionalidades del prototipo respondan a las necesidades reales del sector florícola, aportando el criterio experto sobre los procesos de calidad y logística.
- Scrum Master: Msc. Edison Mora, en su papel de director del trabajo de titulación. Su responsabilidad es garantizar la aplicación de la metodología de trabajo, supervisar el cumplimiento de los sprints y asegurar que el desarrollo se alinee con los estándares académicos necesarios.
- Developer: Danilo Ismael Arias, encargado de la arquitectura, desarrollo del prototipo de aplicación móvil y sistema web.

Los artefactos que componen el proyecto del trabajo de titulación:

- Product Backlog: Detalle del proyecto, lo que se conoce y es necesario para llevar a cabo la aplicación móvil y el sistema web.
- Sprint Backlog: Descripción del trabajo que se realizará en cada sprint para el cumplimiento de los requerimientos del sistema.
- Incremento del producto: Hitos que han sido cumplidos en cada sprint para cumplir el concepto de desarrollo ágil e incremental.

Finalmente, los eventos que se llevarán a cabo para el desarrollo del trabajo de titulación:

- Sprints: Tiempo definido para la ejecución de tareas concretas.
- Sprint Planning: Planificación de cada sprint, qué y cómo se realizará.
- Sprint Review: Revisión de los resultados que evidencian un incremento del proyecto.

Al aplicar SCRUM en este trabajo de titulación, la entrega de sprints está prevista para cumplirse de manera quincenal, permitiendo planificar tareas específicas como el diseño de la base de datos, el desarrollo de módulos de lectura de códigos QR en la aplicación móvil, la construcción del sistema web y la implementación de pruebas funcionales en entornos que simulen casos reales.

1.5. Alcance

El alcance del prototipo expuesto en este documento como trabajo de titulación está delimitado para cumplir su relevancia y viabilidad, permitiendo registrar, monitorear y analizar el ciclo logístico de bouquets de flores mediante códigos QR.

El prototipo por desarrollarse a través de Flutter incluirá módulos puntuales para la recopilación del despacho y recepción de los ítems, la lectura de códigos QR y la visualización de reportes básicos respecto al desperdicio de flor. La línea de comunicación entre la aplicación móvil y el sistema web será implementada mediante la base de datos en la nube Supabase.

El desarrollo del prototipo se enfocará en la funcionalidad óptima de la aplicación móvil y del sistema web, descartando el apartado de despliegue de ambas partes, así como la integración de los datos almacenados en Supabase con otros sistemas de información empresariales (ERP) y/o la conexión de otro tipo de herramientas IoT que pueden involucrase en el seguimiento de los ítems durante su ciclo logístico.

Las respectivas pruebas y validaciones se realizarán en entornos controlados de simulación de exportación y recepción de los ítems con datos de ejemplo prácticos, no datos reales, por lo tanto, este trabajo de titulación no constituye un despliegue comercial de la aplicación móvil ni del sistema web.

CAPÍTULO II: MARCO TEÓRICO Y CONCEPTUAL

2.1. Conceptos clave

2.1.1. Trazabilidad logística en la industria florícola

En el contexto ecuatoriano, el sector florícola destaca como una actividad exportadora estratégica, puesto que impulsa significativamente el desarrollo económico y la captación de divisas no petroleras, a la vez que garantiza empleo y bienestar social para las familias en las regiones rurales productoras (Betancourt Espinoza et al., 2025). Bajo esta premisa, la implementación de herramientas tecnológicas orientadas a la trazabilidad logística se vuelve indispensable no solo para optimizar procesos y reducir desperdicios, sino como una medida necesaria para resguardar una fuente vital de la economía nacional.

La trazabilidad logística se define como la competencia para rastrear el historial y la ubicación de un producto a lo largo de toda la cadena de suministro, esta visibilidad otorga a las empresas la facultad de reaccionar con agilidad ante incidencias tales como defectos de fabricación, contaminación o reclamos de clientes, al permitir la localización exacta del origen del fallo para su inmediata corrección (Marín, 2024). En la producción florícola, este monitoreo detallado resulta crucial para garantizar que los bouquets cumplan con los rigurosos estándares de calidad que exige el mercado internacional.

La implementación de herramientas tecnológicas enfocadas en el registro y análisis de datos en la trazabilidad logística permite transformar procesos manuales que toman tiempo y son propensos a errores por sistemas de información accesibles y seguros con datos en tiempo real.

En la industria florícola, dónde la vida útil del producto es corta, contar con datos de manera inmediata para la identificación rápida de puntos críticos de desperdicio de flor, permite tomar decisiones para acciones inmediatas ante fallas que se pueden presentar en la manipulación y transporte de los ítems.

2.1.2. Desperdicio de flor

El desperdicio de flor es entendido cómo la pérdida de producto que se presenta a partir de un bouquet de flores o ítem que debe ser retirado y desechado en un punto de su ciclo logístico debido al maltrato físico, quemaduras en pétalos, tallos o presencia de patologías biológicas propias de una flor como botrytis, los ítems que presentan dichos casos son

clasificados mediante un proceso de control de calidad previo a distribución del producto para su venta.

Un ítem clasificado cómo desperdicio de flor, no se limita únicamente a sufrir daños desde su salida de finca al presentar pétalos quemados o contener botrytis, el caso puede presentarse en diferentes etapas del proceso, un empaquetado brusco previa exportación o daños en el transbordo, es por ello que el identificador para el desperdicio de flor debe contarse en dos puntos clave; el despacho desde el centro de acopio (origen) y el proceso de verificación o rearme en la bodega de distribución en el exterior (destino).

2.1.3. Código QR

Un código "Quick Response" en sus siglas QR, se define como una matriz bidimensional capaz de almacenar información alfanumérica accesible mediante las cámaras de dispositivos móviles. Su diseño fue pensado específicamente para garantizar una lectura ágil a través de equipos electrónicos, facilitando su aplicación en procesos de etiquetado y gestión de inventarios en contextos de producción masiva (Padrón et al., 2020). Adicionalmente, su arquitectura interna ofrece robustez, permitiendo la recuperación íntegra de los datos incluso ante deterioros físicos que pueda sufrir el código al estar impreso.

La elección por los códigos QR frente a los códigos de barras tradicional tiene su justificación en la eficiencia de captura de datos. Su diseño espacial permite un acceso inmediato a la información, característica que, según Gordon Graell (2023), ha desplazado tecnológicamente a su predecesor de barras hasta volverlo obsoleto, esta decisión toma fuerza al considerar que el código de barras posee una capacidad de almacenamiento limitada y exige hardware específico como máquinas de escáner láser. En contraste, la facilidad para realizar el escaneo directamente mediante teléfonos móviles fue el factor determinante para su implementación en este trabajo de titulación.

2.1.4. Backend as a Service (BaaS)

Backend as a Service (BaaS) es un modelo de arquitectura computacional alojada en la nube que proporciona a los desarrolladores herramientas integradas para la gestión de una base de datos cloud. Este enfoque se apoya en proveedores que suministran APIs documentadas y kits de desarrollo (SDKs) con el fin de facilitar la implementación ágil

de funciones preestablecidas y asegurar una interacción fluida entre la lógica del servidor (backend) y la interfaz de usuario (frontend) (Uunonen, 2025).

La ventaja principal del uso de un BaaS es la abstracción de la complejidad del lado del servidor, permitiendo al desarrollador enfocar el proyecto en la lógica de negocio y la experiencia de usuario, siendo un componente clave para aplicaciones móviles y sistemas web gracias a la capacidad de acelerar los ciclos de implementación de funcionalidades y reducir la carga de trabajo que implica el mantenimiento, escalabilidad y seguridad de la infraestructura del backend.

Posterior al análisis del modelo BaaS, la elección de la herramienta que cumple este concepto para el presente trabajo de titulación es Supabase, plataforma de código abierto construida para ser empleada con PostgreSQL (Structured Query Language) que destaca por extender las capacidades de la base de datos tradicional con funcionalidades en tiempo real a través de su propio servidor (Supabase, 2025).

2.1.5. Desarrollo de aplicaciones móviles

Las aplicaciones móviles están diseñadas para ejecutarse en dispositivos como celulares o tablets aprovechando recursos de hardware integrados como su pantalla táctil, cámara, sensores y acceso a redes móviles de internet, permitiendo realizar tareas de forma rápida y precisa, a diferencia de las aplicaciones de escritorio tradicionales que requieren mayor capacidad de cómputo, las aplicaciones móviles ofrecen facilidad y acceso inmediato a la información desde cualquier ubicación.

Bajo este contexto aprovechando las integraciones que ofrecen los dispositivos móviles, surge la idea de brindar una herramienta tecnológica para el control de desperdicio de flor a través del registro de códigos QR adheridos a la manga o etiqueta de un bouquet de flores mediante la cámara de un teléfono celular, permitiendo así la digitalización del control de calidad de los productos en vivo, agilizando la captura de información en tiempo real y eliminando retrasos operativos que genera recopilar datos de forma manual en lápiz y papel.

Para el desarrollo del prototipo móvil, se seleccionó el framework Flutter debido a su eficiencia multiplataforma, esta herramienta permite unificar el flujo de trabajo mediante una única base de código ejecutable tanto en Android como en iOS, lo que optimiza significativamente los tiempos de desarrollo al evitar la programación simultánea en

entornos nativos separados, es precisamente, esta capacidad de emplear un solo lenguaje para generar aplicaciones funcionales en ambos sistemas operativos constituye la principal ventaja competitiva valorada por los desarrolladores.

2.1.6. Sistema web

Un sistema web es definido como una aplicación o plataforma que es accesible a través de un navegador de internet sin necesidad de requerir instalar el software de manera local, teniendo su diferencia más grande con la aplicación móvil expuesta, cuyo enfoque es la captación y escritura de datos desde el campo operativo, el sistema web cumplirá un rol en el campo administrativo al centralizar y gestionar la visualización de la información.

El prototipo para el sistema web propuesto cumple la función de ser un dashboard logístico, dónde se muestre a través de tablas y gráficos la información en tiempo real sobre los bouquets de flores que se encuentran en un proceso de despacho, recepción, aprobación y rechazo, facilitando el monitoreo de los ítems y la detección rápida de desperdicio de flor en la cadena de suministro para la toma de decisiones efectiva.

Adicionalmente el sistema web incorpora un módulo para la generación de los códigos QR a partir de archivos Excel con tablas que contengan información relevante para la construcción del código e identificación del ítem, permitiendo así tener un QR único para cada bouquet, garantizando la trazabilidad del producto desde su origen hasta su destino. De esta forma el sistema web complementa a la aplicación móvil desarrollada en conjunto al proveer el software necesario para la gestión de la información del ciclo logístico de los bouquets de flores. Con el fin de mantener el mismo el mismo framework de desarrollo para el prototipo, la elección para llevar a cabo el sistema web fue Flutter.

2.1.7. Prototipo

Un prototipo es una versión preliminar de un sistema de información que permite validar sus funcionalidades sin constituir un producto final, el propósito de un prototipo es demostrar la viabilidad técnica de la solución propuesta a un problema, permite también detectar fallos, identificar mejoras y evaluar el desempeño del sistema frente a escenarios simulados.

Para este trabajo de titulación el prototipo resultante es una aplicación móvil y sistema web desarrollados en Flutter, compuestos por la base de datos en la nube Supabase, su

función es demostrar la capacidad de registro, monitoreo y análisis del ciclo logístico de bouquets de flores mediante el uso de datos de prueba que representen los flujos de trabajo reales de exportación y recepción de bouquets de flores.

2.2. Antecedentes

Un referente clave en el seguimiento del ciclo logístico de productos es Amazon, cuya eficiencia operativa se sustenta en el despliegue de dispositivos de escaneo y herramientas tecnológicas interconectadas en la nube, lo que aporta a los operarios para mantener un registro actualizado y en tiempo real sobre el estado de la mercancía (Diego, 2025). Este nivel de control y trazabilidad es la base de su éxito en la organización del transporte de productos, abarcando todo el proceso desde el empaquetado en bodegas hasta la entrega en la dirección del cliente.

Amazon ha consolidado su prestigio y reconocimiento global gracias a un modelo de negocio donde la cadena de suministro actúa como eje central. Su estrategia se distingue por asumir el control integral del flujo de mercancías, gestionando desde la recolección en fábricas internacionales y trámites aduaneros hasta el almacenamiento y la distribución al cliente, asegurando con ello un monitoreo constante de cada etapa del proceso logístico (Diego, 2025).

Sin embargo, bajo el contexto de este trabajo de titulación orientado a proponer una solución para el caso de desperdicio de flor en el ciclo logístico de bouquets de flores, la implementación de dispositivos de tracking similares a los aplicados por Amazon implicaría posterior personalización de sus componentes y sistemas con el fin de cumplir dicho objetivo, además representa un conjunto de procesos cómo; investigación, cotización y adquisición del hardware y software.

En contraste, el desarrollo de un prototipo de aplicación móvil y sistema web diseñado a medida y adaptado a la necesidad del caso, permite replicar los beneficios fundamentales de la trazabilidad y monitoreo de la cadena de abastecimiento y entrega de los ítems. Esta propuesta representa una optimización significativa en los tiempos de implementación gracias a la reducción de pasos para la obtención de la herramienta final.

2.3. Marco metodológico

2.3.1. Selección de la metodología

La selección de la metodología ágil SCRUM para este trabajo de titulación sobre distintos modelos tradicionales como Cascada, es debido a la experimentación constante en la que se ve envuelto el desarrollo del prototipo, dado que pueden surgir ajustes durante el análisis, diseño y programación, este enfoque resulta preciso al permitir la incorporación de cambios durante la ejecución de los sprints, con la única condición de que no se alteren los compromisos de entrega pactados (Rodríguez & Vicente, 2015).

SCRUM tiene su éxito en la entrega continua de avances que generan un incremento del producto, este concepto se define como el resultado acumulativo de todos los elementos del Product Backlog sumados durante la ejecución de cada sprint (Quedena, 2019). Esta característica permite validar avances funcionales de manera constante, a diferencia de las metodologías tradicionales o en cascada que suelen evidenciar resultados en las fases finales del proyecto.

2.3.2. Roles, artefactos y eventos

Esta metodología ágil se caracteriza por la interacción entre 3 componentes clave que garantizan la transparencia y la capacidad de respuesta ante cambios durante la ejecución del desarrollo del proyecto, estos son los roles, artefactos y eventos.

Los roles definen las responsabilidades dentro del equipo para asegurar que el desarrollo se mantenga alineado a los objetivos y resultados esperados. El Product Owner es el encargado de darle valor al producto, siendo el responsable de gestionar y priorizar los requerimientos según las necesidades del producto, el Scrum Máster actúa como un tutor que guía la adaptación de la metodología, finalmente el desarrollador es el encargado de transformar los requerimientos en funcionalidades, actúa con autonomía para decidir las tareas a realizar con el fin de ser eficiente y asegurar la entrega de incrementos funcionales en cada sprint.

En la gestión de la información y entregables, se emplean artefactos diseñados para comprobar el estado del proyecto en todo momento. Como primer artefacto se define el Product Backlog funciona como un registro que enlista las características y mejoras deseadas para el producto, el Sprint Backlog es derivado de la lista mencionada y contiene las tareas a ser completadas en un periodo de tiempo.

Los eventos se definen cómo Sprints, una iteración de tiempo donde se ejecutan las tareas para llegar al producto final, cada ciclo comienza con una sesión llamada Sprint Planning, donde se definen las tareas y estrategias para cumplirlas, culminado el ciclo de trabajo se ejecuta el Sprint Review dónde se verifican los avances obtenidos y se brinda retroalimentación que permita ajustar el rumbo del proyecto.

2.4. Herramientas tecnológicas

2.4.1. Framework de desarrollo: Flutter VS React Native

Para el desarrollo de herramientas tecnológicas ejecutables en dispositivos móviles y navegadores web, la elección del framework es un factor determinante para la correcta programación de los sistemas mencionados, realizando una evaluación entre dos marcos de trabajo más utilizados para el desarrollo móvil y web, se comparó la viabilidad de Flutter y React Native en el presente trabajo de titulación.

Mientras que React Native se basa en un puente de comunicación para la interacción de los componentes del sistema, Flutter compila directamente el código sin la necesidad de aplicar enlaces entre los componentes, esta característica es vital para el prototipo de la aplicación móvil, ya que se garantiza un escaneo de códigos QR sin retrasos operativos y para el sistema web, se asegura una visualización de datos consistente, sin interrupciones.

La elección por Flutter fue tomada gracias a su motor de renderizado propio, útil para el registro de desperdicio de flor en bodega, dónde operarios utilizan aplicaciones móviles en distintos dispositivos tanto de marca, tamaño o resolución. El sistema web se beneficia de esta tecnología también al presentar una interfaz administrativa robusta y de alto rendimiento. La tabla 1 muestra la comparativa entre ambos frameworks de desarrollo en base a 5 criterios; lenguaje, arquitectura, interfaz de usuario, rendimiento y entornos aplicables.

Tabla 1
Comparativa entre Flutter y React Native

Criterio	Flutter	React Native
Lenguaje	Dart (compilado)	JavaScript (interpretado)
Arquitectura	Sin puente (directo al CPU)	Puente de comunicación
Interfaz (UI)	Componentes propios	Componentes nativos del sistema

Rendimiento	Alto	Dependiente del puente JavaScript
Entornos	Aplicación móvil y sistema web	Aplicación móvil y soporte web externo

Auto: Elaboración propia.

2.4.2. Backend as a Service: Supabase VS Firebase

Para la gestión de datos en la nube y sincronización de la información en tiempo real, se compararon dos plataformas más utilizadas en proyectos de aplicaciones móviles y sistemas web que involucran backend como servicio, siendo estas Supabase y Firebase.

El motor de base de datos integrado en Supabase es PostgreSQL, dada la necesidad de una base de datos relacional que permita establecer vínculos entre ítems, colecciones y reportes de desperdicio de flor, Supabase es la elección ideal para evitar inconsistencias en la trazabilidad. A diferencia de Firebase, propiedad de Google, que basa su estructura de base de datos en formato NoSQL, Supabase garantiza la integridad referencial de los ítems para los procesos logísticos. La tabla 2 compara ambos proveedores de BaaS en base a 5 criterios; base de datos, licenciamiento, integridad de los datos, sincronización y tipo de consultas.

Tabla 2
Comparativa entre Supabase y Firebase

Criterio	Supabase	Firebase
Base de Datos	Relacional (PostgreSQL)	No relacional
Licenciamiento	Código abierto	Google
Integridad de datos	Claves foráneas y reglas	Datos duplicados
Sincronización	Tiempo real web mobile y desktop	Tiempo real nativo de la plataforma
Consultas	Soporte complejo para SQL	Limitación a filtros simples

Autor: Elaboración propia.

Además, la selección de Supabase tiene su fundamentación en la capacidad de respuesta en tiempo real de la plataforma gracias al uso del protocolo WebSocket, este estándar permite establecer un canal de comunicación bidireccional para la recepción de actualizaciones desde el servidor de manera instantánea, eliminando la necesidad de

realizar consultas constantes por parte del cliente y reduciendo significativamente la sobrecarga de datos en el intercambio de información (Murley et al., 2021). La figura 3 a continuación presenta un diagrama simple del protocolo WebSocket aplicado en el presente trabajo de titulación.

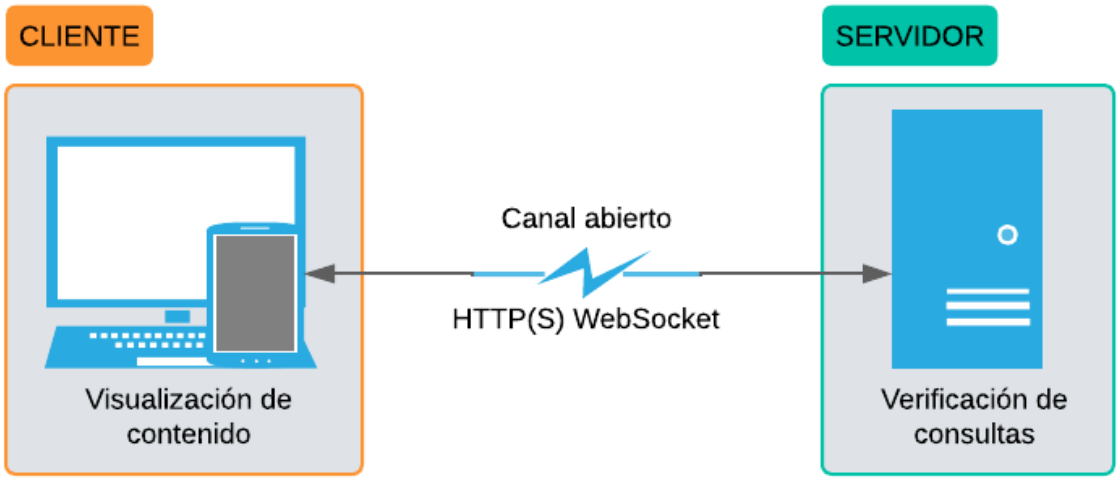


Figura 3. Protocolo WebSocket
Nota. Elaboración propia a través de Lucidchart.com.

2.4.3. Visual Studio Code

El entorno de desarrollo elegido para el desarrollo del prototipo de aplicación móvil y sistema web es Visual Studio Code, debido a su compatibilidad con el ecosistema de Flutter y Dart. La capacidad de gestionar extensiones facilita la detección de errores de manera ágil y rápida, optimizando así el flujo de trabajo de programación, un punto fundamental para mantener el ritmo de desarrollo que exige la metodología ágil elegida.

2.4.4. Control de versiones mediante Git y GitHub

La integridad del código fuente y la gestión de los avances del desarrollo del prototipo de aplicación móvil y sistema web se garantizan mediante el uso de Git como sistema de control de versiones. Esta herramienta asegura el control del proyecto al permitir que el desarrollador posea una copia local del código base. Git facilita un trabajo no lineal a través de su capacidad para generar ramas de forma ágil, lo que permite experimentar con nuevas funcionalidades de manera independiente sin comprometer la estabilidad de la línea de desarrollo principal (Tu et al., 2022).

GitHub funciona como un repositorio remoto que permite cargar el código fuente de forma segura en la nube, esta plataforma actúa como un centro estratégico donde se

gestionan los incrementos de un sistema, lo que asegura que cada modificación sea monitoreada. De esta forma, se establece un entorno controlado que facilita el seguimiento continuo del proyecto y la coordinación eficiente entre los componentes.

2.5. Arquitectura de software

2.5.1. Modelo Vista Controlador

La arquitectura de software elegida para el presente trabajo de titulación es Modelo Vista Controlador (MVC), su implementación permite dividir el sistema en tres capas independientes. Esta estructura facilita la organización del código al separar las responsabilidades de procesamiento, manejo de datos y presentación, lo que optimiza el mantenimiento y la escalabilidad del software frente a futuros requerimientos (Paolone et al., 2020). La figura 4 presenta un diagrama de la arquitectura Modelo Vista Controlador con la que contará el prototipo de aplicación móvil y sistema web.

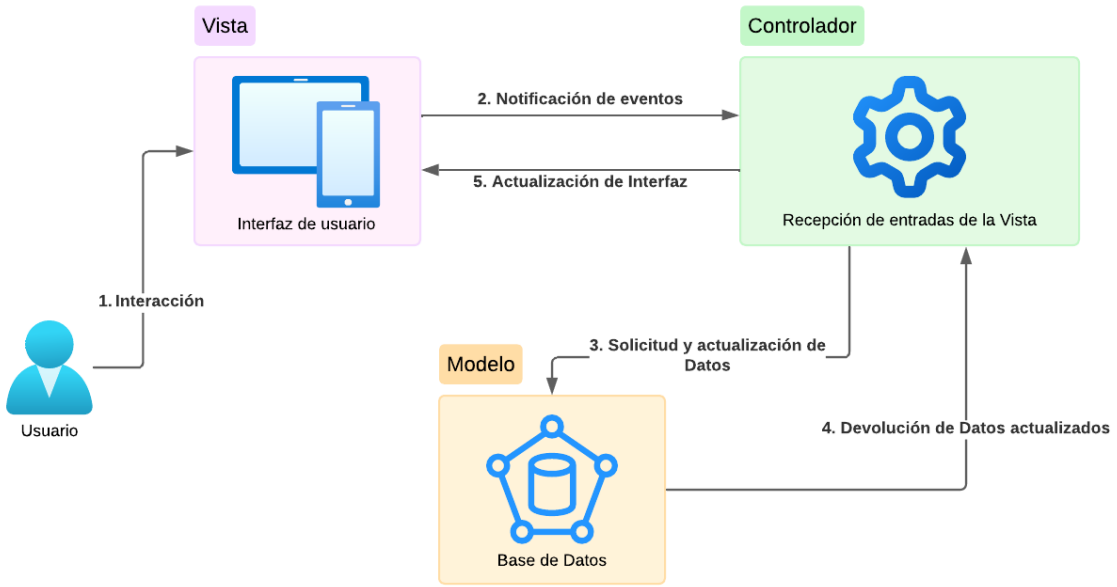


Figura 4. Diagrama de arquitectura Modelo Vista Controlador (MVC)
Nota. Elaboración propia a través de Lucidchart.com.

- Modelo: Es la capa encargada de la gestión de los datos y las reglas del negocio. En este prototipo, el modelo define la estructura de la información sobre los bouquets y registros de desperdicio de flor, esta capa se conecta directamente con la plataforma Supabase, transformando las tablas de la base de datos, en sus siglas "BDD", en objetos que el sistema procesa y valida.

- Vista: Esta capa constituye la interfaz con la que interactúan los usuarios, desarrollada a través del framework Flutter, se divide en 2 entornos específicos; aplicación móvil para el escaneo de códigos QR y registro de información, y el sistema web diseñado para la visualización administrativa de los datos.
- Controlador: Actúa como el puente de comunicación entre las capas de vista y modelo. Su función es recibir las acciones del usuario, en el presente caso, la lectura de códigos QR y ejecutar la lógica correspondiente para actualización de registros o consulta de información en el backend.

2.5.2. Sincronización mediante WebSockets

El éxito del seguimiento logístico de los bouquets de flores para la toma de decisiones tiene una gran dependencia en la velocidad con la que la información se transmite. Para lograr una actualización inmediata de la información, la arquitectura del prototipo incorpora el protocolo WebSocket a través de la plataforma Supabase.

Las conexiones tradicionales basadas en protocolos HTTPS que operan a través de un modelo unidireccional de solicitud y respuesta, requieren peticiones constantes desde el cliente para la verificación de actualizaciones. Estas arquitecturas presentan limitaciones significativas, el intercambio de información mediante texto plano en los protocolos HTTPS puede resultar ineficiente para el procesamiento de datos en sistemas de información, lo que deriva en altas latencias y retardos operativos que comprometen la fluidez de la comunicación entre las partes (Silvestre, 2021).

A diferencia del protocolo WebSocket, que mantiene un canal de comunicación bidireccional abierto entre el servidor y las interfaces, se permite que la transmisión de datos sea instantánea. Bajo este contexto, cuando se registra un caso de desperdicio de flor en la aplicación móvil, el controlador envía los datos al modelo y simultáneamente se emite la información en tiempo real hacia el sistema web. De esta manera se evitan los retrasos operativos que involucran los procesos manuales de recopilación y envío de información.

CAPITULO III: ANÁLISIS, DISEÑO Y DESARROLLO DEL PROTOTIPO

3.1. Análisis de requerimientos

3.1.1. Requerimientos funcionales

Los requerimientos funcionales definen las acciones y servicios que se deben ejecutar conforme el usuario interactúe con el sistema. Estos requerimientos representan lo que debe hacer el prototipo, detallando desde la captura de datos mediante el escaneo de códigos QR hasta la información desplegada en el sistema web. Los requerimientos funcionales de la aplicación móvil y sistema web son detallados a través de las tablas 3 y 4 respectivamente.

Tabla 3
Requerimientos funcionales Aplicación Móvil

Código	Requerimiento	Descripción
RF-01	Registro de despacho	El operario del centro de acopio debe escanear todos los ítems que conforman el envío de una colección de bouquets.
RF-02	Registro de aprobación o rechazo	El operario de la bodega de distribución debe escanear los ítems para clasificarlos como aprobados o rechazados según su calidad.
RF-03	Especificación de causas de rechazo	Al escanear un ítem no apto, la aplicación debe permitir al operario seleccionar o detallar el motivo del rechazo.
RF-04	Asociación automática de la colección	El sistema debe identificar automáticamente la colección a la que pertenece el ítem mediante la lectura del código QR.

Autor: Elaboración propia

Tabla 4
Requerimientos funcionales Sistema Web

Código	Requerimiento	Descripción
RF-05	Generación de etiquetas QR mediante archivos Excel	El sistema debe permitir la carga de archivos Excel para la generación de códigos QR únicos vinculados a cada ítem y colección.
RF-06	Dashboard de monitoreo	El sistema debe permitir visualizar mediante tablas y gráficos el estado de los ítems despachados y recibidos en tiempo real.
RF-07	Cálculo automático de ítems	El sistema debe realizar los respectivos cálculos entre los ítems aprobados y rechazados para la toma de decisiones.
RF-08	Reporte de incidencias	El sistema debe desplegar las razones específicas del desperdicio recolectadas por la aplicación móvil parag la toma de decisiones.

Autor: Elaboración propia.

3.1.2. Requerimientos no funcionales

Los requerimientos no funcionales establecen las propiedades que rigen al sistema de información. Estos requerimientos no describen tareas, describen cómo debe comportarse el prototipo en términos de rendimiento, seguridad e integridad y disponibilidad de los datos alojados en la nube. La tabla 5 presenta los requerimientos no funcionales para el prototipo.

Tabla 5
Requerimientos no funcionales

Código	Requerimiento	Descripción
RNF-01	Sincronización en tiempo real	La comunicación entre el registro de ítems en la aplicación móvil y la visualización en el sistema web debe ser inmediata.
RNF-02	Integridad de los datos	La base de datos debe asegurar la integridad de los ítems registrados y la disponibilidad de estos de forma remota en la nube.

RNF-03	Rendimiento	La aplicación móvil debe operar de manera fluida aprovechando el motor de renderizado del framework Flutter.
RNF-04	Seguridad	El sistema debe implementar la seguridad de los códigos QR, que estos no contengan información sensible en texto plano, en su lugar, identificadores únicos que solo puedan ser interpretados por la base de datos del prototipo.

Autor: Elaboración propia.

3.2. Modelado del sistema

3.2.1. Casos de uso

Se han identificado 3 actores principales involucrados en la interacción del prototipo; administrador, operario de acopio y operario de distribución, sus interacciones se centran en garantizar la trazabilidad de los ítems y la eficiencia en la captura de casos de desperdicio de flor.

- Administrador (Sistema Web): Sus acciones son: carga de ítems mediante archivos Excel para la generación de códigos QR únicos, monitoreo del dashboard para la visualización del estado de los ítems y el análisis de los casos de desperdicio de flor como se presenta en la figura 5.

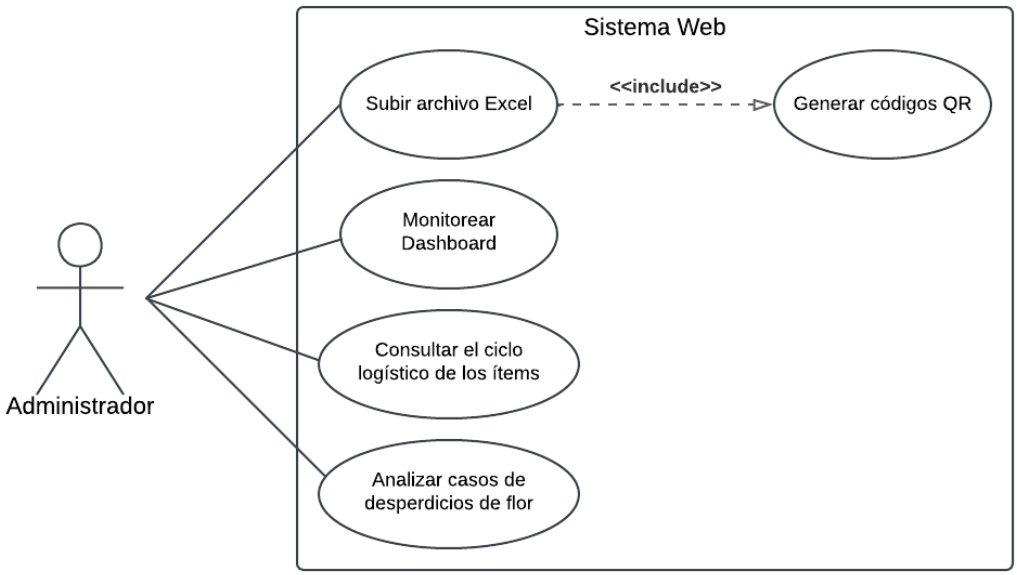


Figura 5. Caso de uso del Administrador
Nota. Elaboración propia a través de Lucidchart.com.

- Operario de Acopio (App Móvil): El operario del centro de acopio representa el origen del ciclo logístico, su interacción con la aplicación móvil consiste en el registro del despacho de ítems, escaneando los bouquets de una colección que se va a despachar como se presenta en la figura 6.

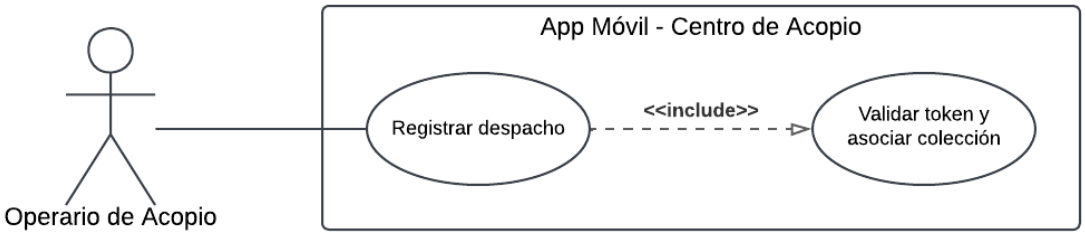


Figura 6. Caso de uso del Operario de Acopio
Nota. Elaboración propia a través de Lucidchart.com.

- Operario de Distribución (App Móvil): El operario del centro de distribución interactúa con el sistema para escanear los ítems y clasificarlos como aprobados o rechazados, en caso de presentar un rechazo, el sistema vincula automáticamente el ítem a su colección y se solicita especificar el motivo del rechazo como se presenta en la figura 7.

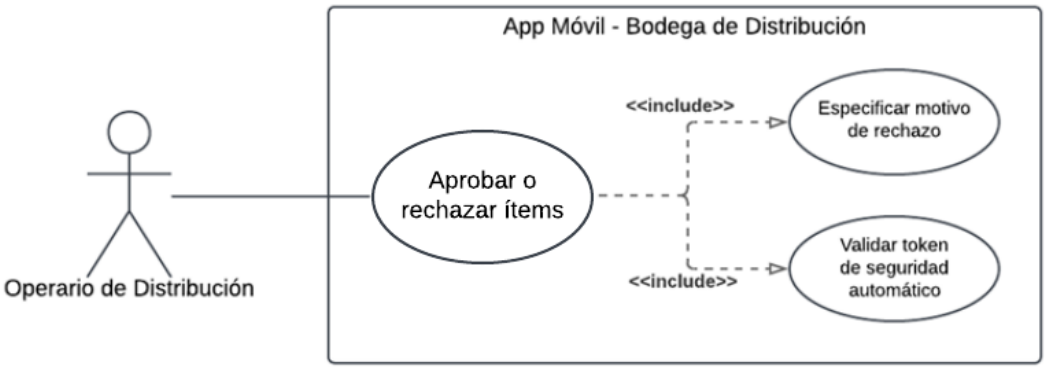


Figura 7. Caso de uso del Operario de Distribución
Nota. Elaboración propia a través de Lucidchart.com.

3.2.2. Diagrama de clases

El diagrama de clases expuesto para el presente trabajo de titulación describe la estructura del prototipo mediante la definición de clases, atributos y las relaciones bajo la arquitectura Modelo Vista Controlador (MVC), estas clases representan la capa de

Modelo, asegurando de esta forma que la lógica de negocio esté correctamente organizada.

Clases principales del sistema:

- Ítem: Es la entidad central del sistema que contiene el qr_token en formato UUID (Universally Unique Identifier) para el resguardo de información sensible y su código de referencia en formato SKU (Stock Keeping Unit) un formato internacional, con el fin de serializar los datos de manera estratégica y resguardar la privacidad de la información
- Colección: Agrupa un conjunto de ítems.
- ReporteDesperdicio: Se genera únicamente cuando el operario de la bodega de distribución marca un ítem como "rechazado".
- MotivoRechazo: Justificación del rechazo de un ítem, por ejemplo, botrytis o maltrato físico.
- Usuario: Sus permisos son definidos según el rol sea este Administrador u Operario.

La figura 8 representa el diagrama de clases, sentando la base del desarrollo.

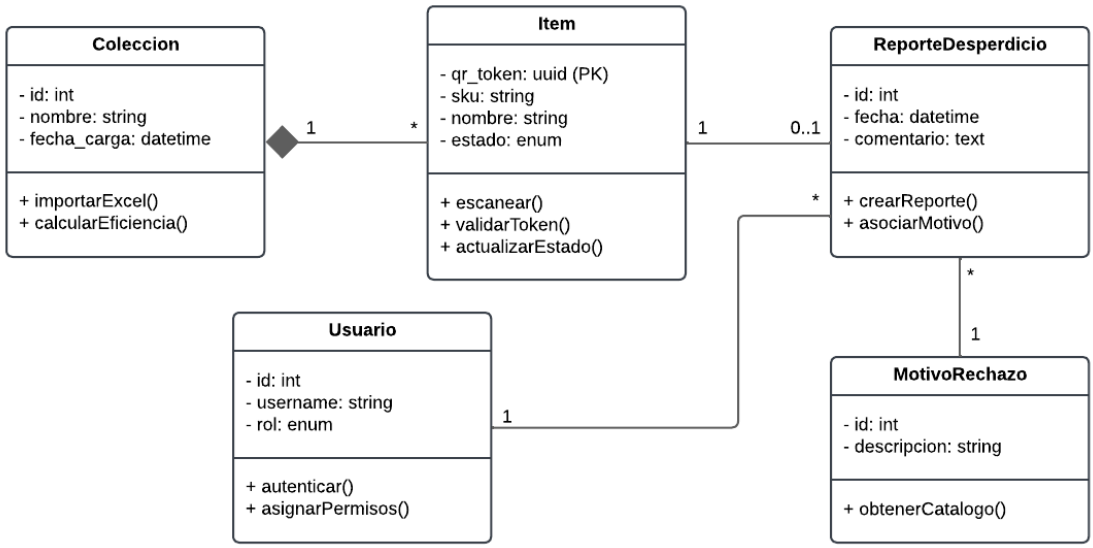


Figura 8. Diagrama de Clases
Nota. Elaboración propia a través de Lucidchart.com.

3.2.3. Diagrama de arquitectura

La arquitectura del prototipo es Modelo Vista Controlador (MVC), esta arquitectura se fundamenta en un modelo de Backend as a Service (BaaS) a través de la plataforma Supabase.

- Capa de Presentación (Vista): Desarrollada en Flutter, esta capa se despliega en dos entornos; la aplicación móvil que permite la captura de datos en el centro de acopio (origen) y la bodega de distribución (destino) y el sistema web, el cuál actúa como un centro de monitoreo para la visualización de información.
- Capa de Comunicación (Controlador): Gestiona el intercambio de información mediante dos protocolos; API REST y WebSocket.
- Capa de Datos (Modelo): Gestionado a través de la plataforma Supabase bajo el motor de base de datos PostgreSQL, en esta capa se almacena el diccionario interno que vincula los UUID de los códigos QR con los ítems, asegurando la integridad referencial y seguridad de los datos.

La figura 9 presenta el diagrama de arquitectura del prototipo.

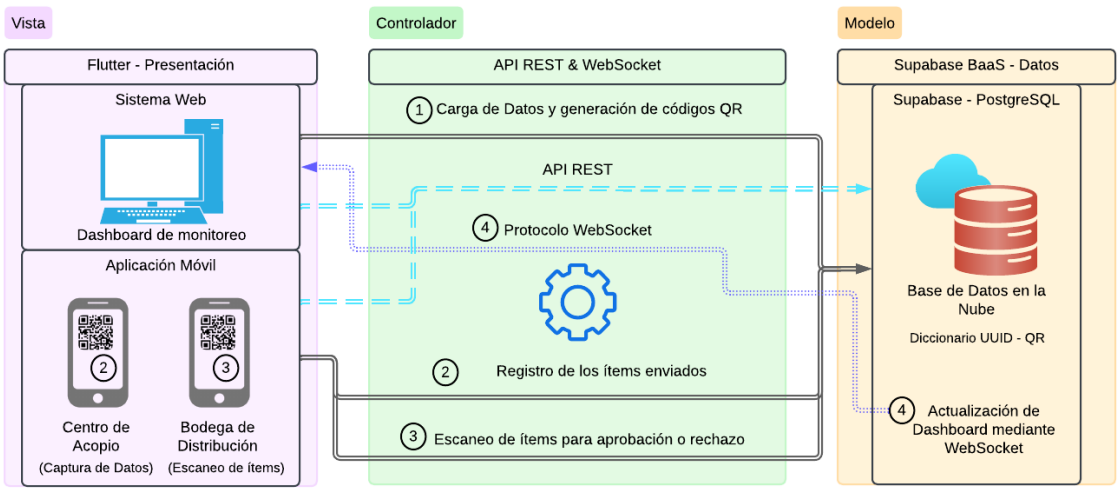


Figura 9. Diagrama de Arquitectura del prototipo
Nota. Elaboración propia a través de Lucidchart.com.

El funcionamiento del sistema inicia con la carga de datos desde la web, donde el modelo genera los códigos QR únicos para cada bouquet, durante el ciclo logístico, el flujo de datos opera de la siguiente manera:

- Origen: El operario del centro de acopio registra los ítems que se despachan mediante la aplicación móvil, el controlador envía los datos al Modelo en Supabase.
- Destino: El operario de la bodega de distribución escane los ítems enviados desde el origen para aprobarlos y rechazarlos, el controlador envía los datos de aprobación y rechazo al Modelo en Supabase.
- Sincronización: Mediante un evento de protocolo WebSocket se emite la información al sistema web.
- Procesamiento: El sistema ejecuta automáticamente cálculos entre ítems aprobados y rechazados, actualizando el dashboard sin intervención manual.

3.3. Diseño de la Base de Datos

3.3.1. Modelo conceptual

Para el modelo conceptual de la Base de Datos del prototipo, se identificaron las entidades principales que intervienen en el ciclo logístico de bouquets de flores y sus relaciones, estas son:

- Colección: Representa una venta negociada, agrupando un conjunto de ítems bajo una misma orden.
- Ítem: Es el bouquet de flores que forma parte de una venta, cada ítem posee una clave única y tiene 4 estados, pendiente, enviado, aprobado y rechazado.
- Reporte de Desperdicio: Entidad que se genera cuando un ítem es marcado como rechazado en la bodega de distribución.
- Motivo de Rechazo: Catálogo para las causas del rechazo de un ítem como botrytis, maltrato o quemadura, permitiendo la estandarización de los datos para el análisis de desperdicio de flor.
- Usuario: Diferenciador entre el administrados, operario de acopio y distribución.

La figura 10 presenta el modelo conceptual de la base de datos de forma gráfica en base al diagrama de clases.

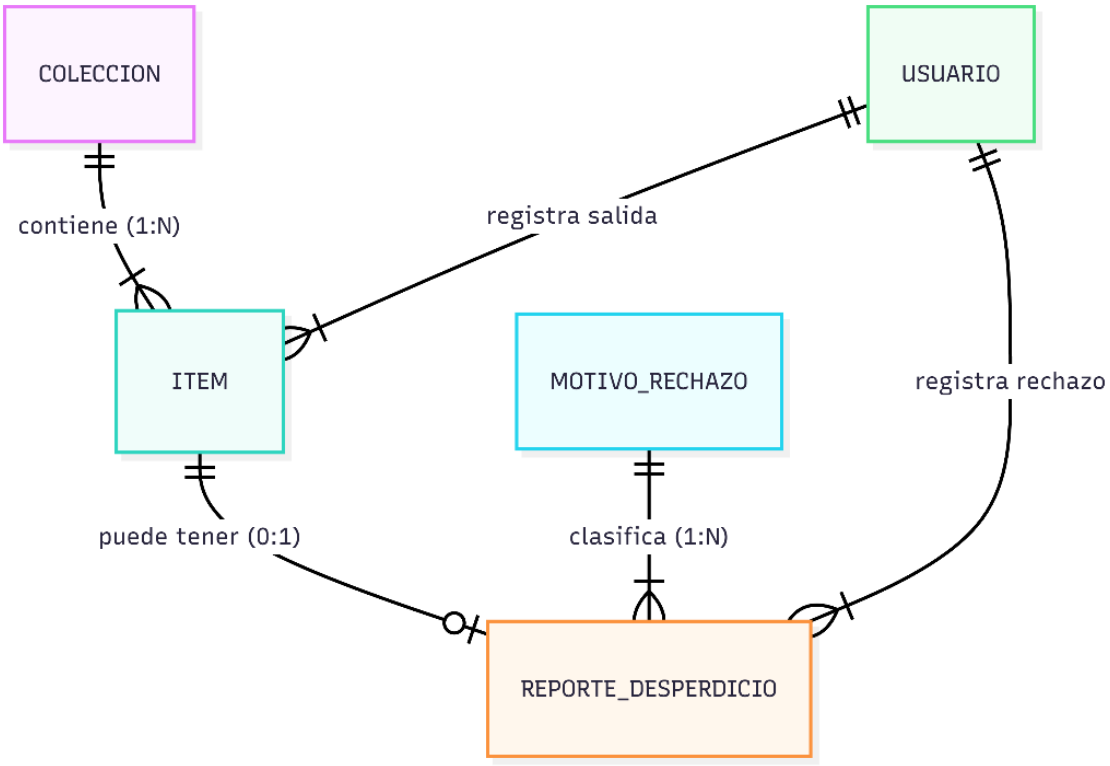


Figura 10. Modelo Conceptual de la Base de Datos
Nota. Elaboración propia a través de MermaidChart.com.

3.3.2. Modelo lógico

El modelo lógico de la Base de Datos del prototipo definió la estructura relacional para el giro de negocio del presente contexto, la arquitectura de datos asegura que no existan ítems sin colección o reportes sin justificación.

Se implementó un sistema de identificadores universales (UUID) para los ítems, garantizando que estos sean únicos, sin códigos repetidos, para mantener la integridad referencial, la tabla de ítems incluye una llave foránea hacia las colecciones, la eliminación de una colección elimina sus ítems asociados con el fin de evitar registros basura en la BDD.

Además, con el fin de optimizar el rendimiento y serializar la información, los datos relacionados a los ítems rechazados se retiran de la tabla principal y se almacenan en una tabla específica para los casos de desperdicio de flor, a continuación, en la figura 11 se presenta el modelo lógico de la base de datos de forma gráfica.

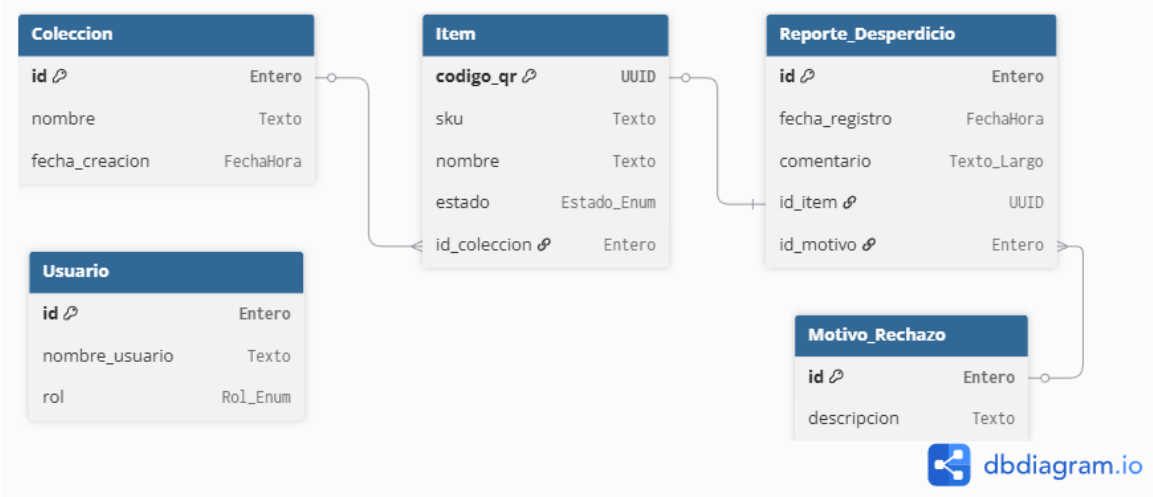


Figura 11. Modelo Lógico de la Base de Datos
Nota. Elaboración propia a través de DBDiagram.io.

3.3.3. Modelo físico

El modelo físico del prototipo corresponde a la implementación final en la Base de Datos en la nube de la plataforma Supabase, se acoplan datos específicos del motor PostgreSQL con el fin de optimizar el rendimiento y la precisión de la información, como se visualiza en la figura 12.

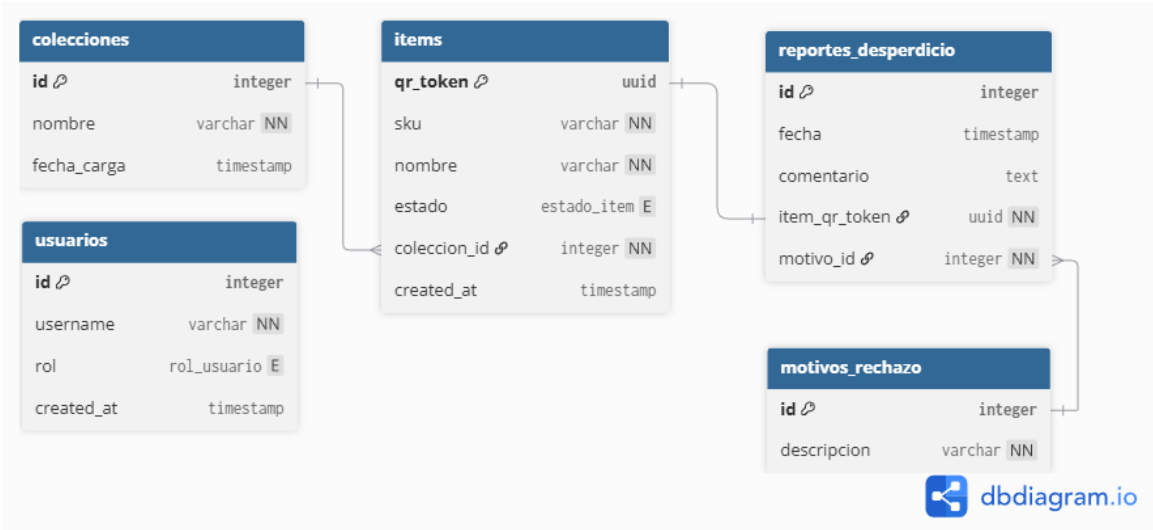


Figura 12. Modelo Físico de la Base de Datos
Nota. Elaboración propia a través de DBDiagram.io.

La tabla ítems es el eje central del sistema, dónde el campo qr_token de tipo UUID cumple la función de llave primaria, garantizando que cada código QR sea único en todo el ciclo logístico.

3.4. Desarrollo bajo metodología ágil SCRUM

3.4.1. Product Backlog

El artefacto de Product Backlog se centra en los requerimientos funcionales para transformarlos en historias de usuario, en la planificación, cada historia ha sido clasificada según su prioridad e importancia para el giro de negocio y el cumplimiento de los objetivos planteados, complejidad se define bajo la dificultad de implementación a nivel de código y lógica. La tabla 6 presenta el product backlog con los respectivos módulos del prototipo y la tabla 7 el sprint backlog con fechas establecidas para el cumplimiento de tareas del proyecto.

Tabla 6
Product Backlog

ID	Módulo	Historia de Usuario	Prioridad	Complejidad	Sprint
HU-01	Infraestructura	Como desarrollador, quiero configurar el proyecto en GitHub y Supabase.	Alta	Baja	0
HU-02	Seguridad	Como administrador, quiero autenticarme en el sistema web mediante correo y contraseña para acceder a las funciones pertinentes.	Alta	Baja	1
HU-03	Gestión web	Como administrador, quiero subir un archivo Excel con una colección para generar los códigos QR de los ítems.	Alta	Media	1
HU-04	Gestión web	Como administrador, quiero generar y descargar un PDF con los códigos QR de una colección para que estos sean impresos y	Alta	Media	1

		adheridos a la manga de los bouquets.			
HU-05	App móvil (acopio)	Como operario de acopio, quiero escanear el QR de un ítem para registrar su despacho y cambiar su estado a enviado.	Alta	Media	2
HU-06	App móvil (distribución)	Como operario de distribución, quiero escanear los ítems para marcarlos como aprobados o rechazados en el sistema.	Alta	Media	3
HU-07	App móvil (distribución)	Como operario de distribución, quiero seleccionar el motivo de rechazo de un ítem, desde un catálogo.	Alta	Media	3
HU-08	Dashboard web	Como administrador, quiero ver indicadores numéricos de ítems, pendientes, enviados y aprobados vs rechazados para monitorear los ítems.	Media	Media	4
HU-09	Dashboard web	Como administrador, quiero visualizar un gráfico y tabla con los motivos de rechazo más frecuentes para tomar decisiones.	Alta	Media	4

Autor: Elaboración propia.

3.4.2. Sprint Backlog

Tabla 7
Calendario de Sprints

Sprint	Evento	Fecha Inicio	Fecha Fin	Objetivo Principal
Sprint 0	Configuración	17/10/2025	27/10/2025	Arquitectura base, repositorio de GitHub y base de datos en Supabase.
Sprint 1	Gestión web y QR	30/10/2025	13/11/2025	Login, carga de archivos Excel y generación de QRs.
Sprint 2	App móvil (acopio)	17/11/2025	27/11/2025	Lectura de códigos QR y registro de despacho.
Sprint 3	App móvil (distribución)	27/11/2025	11/12/2025	Registro de ítems aprobados y rechazados.
Sprint 4	Dashboard y cierre	11/12/2025	22/12/2025	Dashboard web con gráficos en tiempo real y pruebas finales.
Cierre	Entrega final	02/01/2026	05/01/2026	Entrega del proyecto.

Autor: Elaboración propia.

3.4.3. Sprint 0: Arquitectura e Infraestructura

Sprint 0: Arquitectura e Infraestructura

Duración: 17/10/2025 – 27/10/2025

Objetivos: Configurar la infraestructura de Backend as a Service (BaaS) en Supabase, implementar el modelo físico de la BDD relacional y habilitar los canales de comunicación WebSockets para el futuro Dashboard.

Sprint Planning: HU-01

- Creación de organización y proyecto en Supabase.
- Ejecución del script SQL para la creación de tablas y relaciones.
- Inserción de datos para el catálogo de motivos de rechazo.
- Configuración de políticas de seguridad para el acceso y manipulación de la capa Modelo durante el desarrollo del prototipo.

Ejecución de tareas:

Culminada la creación de la organización y proyecto en Supabase, se procede a la ejecución del script SQL para la creación de las 5 tablas y relaciones como se presenta en la figura 13.

```sql
-- Tabla Colecciones
CREATE TABLE colecciones (
    id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    nombre TEXT NOT NULL,
    fecha_carga TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Tabla Items
CREATE TABLE items (
    qr_token UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    sku TEXT NOT NULL,
    nombre TEXT NOT NULL,
    estado estado_item NOT NULL DEFAULT 'pendiente',
    coleccion_id BIGINT REFERENCES colecciones(id) ON DELETE CASCADE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Figura 13. Fragmento de script SQL para la creación de tablas en Supabase.
Nota. Elaboración propia a través de Carbon.now.sh.

La plataforma cuenta con un visualizador del esquema de la base de datos de forma gráfica para un mejor entendimiento del proyecto, similar a un diagrama de modelo físico, además, el visualizador permite añadir columnas y editar propiedades de las tablas de forma intuitiva sin la necesidad de ejecutar otro script, como se puede observar en la figura 14.

Figura 14. Visualizador del esquema de la BDD en Supabase.
Nota. Captura de pantalla.

Para la implementación del protocolo WebSocket en el desarrollo del prototipo, se accede al editor de tablas para activar la propiedad “Realtime”, las tablas afectadas son “reportes_desperdicio” e “item”, de esta forma el futuro dashboard contará con acceso inmediato a la información necesaria, la figura 15 demuestra la alerta de activación de la propiedad mencionada.

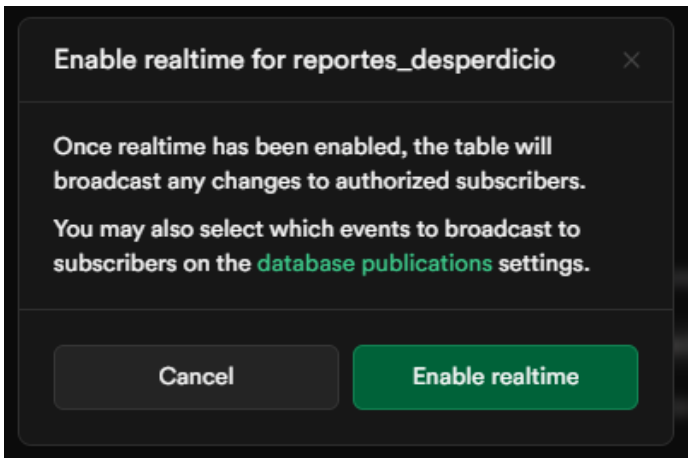


Figura 15. Advertencia de activación de propiedad Realtime en Supabase.
Nota. Captura de pantalla.

Con el fin de no estancarse en el desarrollo del prototipo debido a políticas de permisos, se permite el acceso total a las tablas a través del editor SQL de la plataforma, de esta manera, se optimiza el flujo de trabajo cumpliendo con el objetivo de aplicar una metodología ágil, la figura 16 presenta un fragmento del script.

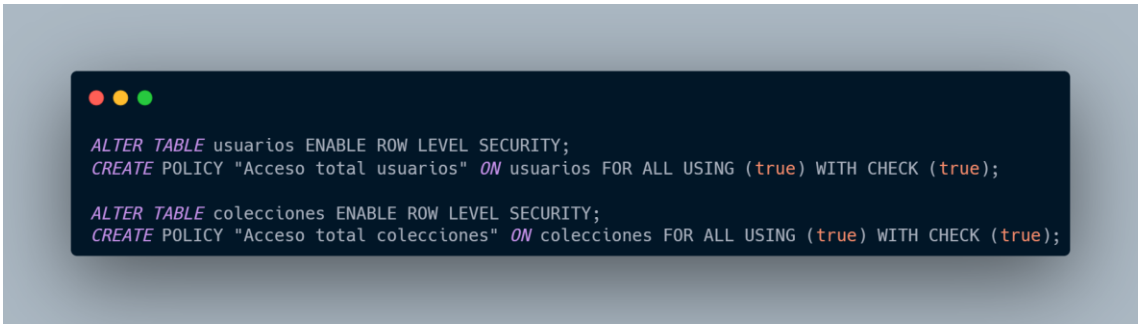


Figura 16. Fragmento de script SQL para la alteración de permisos en Supabase.
Nota. Elaboración propia a través de Carbon.now.sh.

Incremento: El inicio del prototipo con un backend operativo, contando con las tablas de ítems, colecciones, motivos de rechazo, reportes de desperdicios y usuarios listas para recibir datos, además de contar con una API REST expuesta para su uso.

Sprint Review: 27/10/2025.

Se verifica desde el visualizador del esquema de Supabase que las 5 tablas existen y que las relaciones entre estas se encuentran activas, además se comprueba que la propiedad "Realtime" se encuentra activa, como tarea pendiente se espera la creación del proyecto en Flutter para generar un repositorio en GitHub.

Actualización del Product Backlog detallado en la tabla 8:

Tabla 8
Actualización del Product Backlog post Sprint 0

ID	Historia de Usuario	Estado
HU-01	Configuración de Github, Supabase y BDD	En progreso

Autor: Elaboración propia.

3.4.4. Sprint 1: Gestión Web

Sprint 1: Gestión Web

Duración: 30/10/2025 – 13/11/2025

Objetivos: Desarrollar el módulo web que permita la autenticación de usuarios, la carga de datos de colecciones mediante archivos Excel y la generación de documentos PDF con códigos QR para los bouquets.

Sprint Planning:

Configuración y Seguridad (HU-02):

- Creación del proyecto Flutter Web y configuración de dependencias.
- Implementación de la conexión con Supabase.
- Desarrollo de la interfaz de Login y lógica de autenticación.

Gestión de Datos (HU-03):

- Implementación del selector de archivos (file_picker).
- Desarrollo del servicio de lectura de Excel y transformación a objetos.
- Escritura de la función de inserción de datos en la BDD.

Etiquetado (HU-04):

- Diseño de la plantilla de códigos QR en PDF.
- Generación de códigos QR a partir de los UUID (qr_flutter).
- Implementación de la descarga del archivo generado para impresión de los códigos.

Tarea pendiente del Sprint 0:

- Creación del repositorio en GitHub.

Ejecución de tareas:

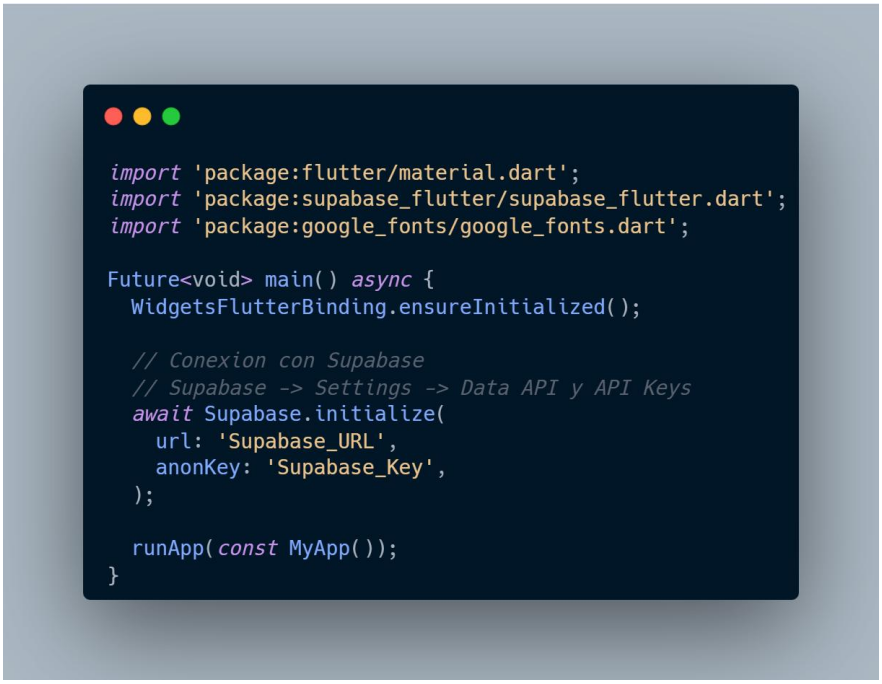
Para la creación del proyecto Flutter Web y la configuración de sus dependencias, se ejecutan los comandos de creación, navegación y adición a través de una terminal de Visual Studio Code, ubicada en la carpeta dónde se alojará el prototipo. La figura 17 demuestra la creación del proyecto en flutter a través de este método.

Figura 17. Comandos de creación, navegación y adición en terminal.
Nota. Elaboración propia a través de Carbon.now.sh.

Culminada la creación del proyecto, es necesario importar las librerías necesarias para el funcionamiento de la aplicación:

- material.dart para los componentes visuales.
- supabase_flutter para la conexión con la base de datos y servicios backend.
- google_fonts para la personalización de la interfaz.

Se inicializa Supabase en el proyecto mediante el método “Supabase.initialize()” donde se configuran la URL del proyecto y la clave pública, esta configuración permite al prototipo acceder de forma segura al proyecto alojado en Supabase. En la figura 18 se demuestra mediante un fragmento de código la conexión a la plataforma BaaS.



```dart
import 'package:flutter/material.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'package:google_fonts/google_fonts.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  // Conexion con Supabase
  // Supabase -> Settings -> Data API y API Keys
  await Supabase.initialize(
    url: 'Supabase_URL',
    anonKey: 'Supabase_Key',
  );

  runApp(const MyApp());
}
```

Figura 18. Importación de librerías y conexión del proyecto con Supabase.
Nota. Elaboración propia a través de Carbon.now.sh.

Para confirmar la conexión del proyecto con Supabase, el despliegue de un aviso de conexión exitosa es suficiente, como se demuestra en la figura 19.

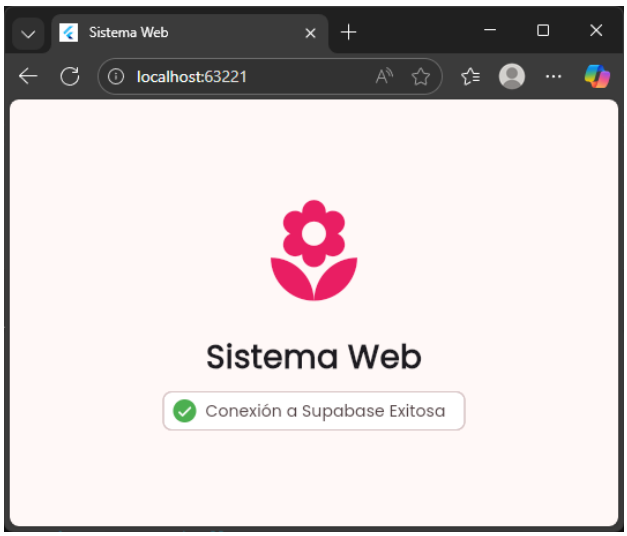


Figura 19. Comprobación de conexión con Supabase.
Nota. Interfaz de elaboración propia.

Para la creación del Log In, en la plataforma de Supabase el apartado de "Authentication" permite añadir usuarios al proyecto, se restringe la creación de usuarios a través del sistema web debido al enfoque del prototipo, dónde el acceso a la información es limitado

al administrador, culminada la creación del usuario, con su email y contraseña, se programa el ingreso, modificando el despliegue del aviso de conexión por una autenticación de usuario real, como se confirma en la figura 20.

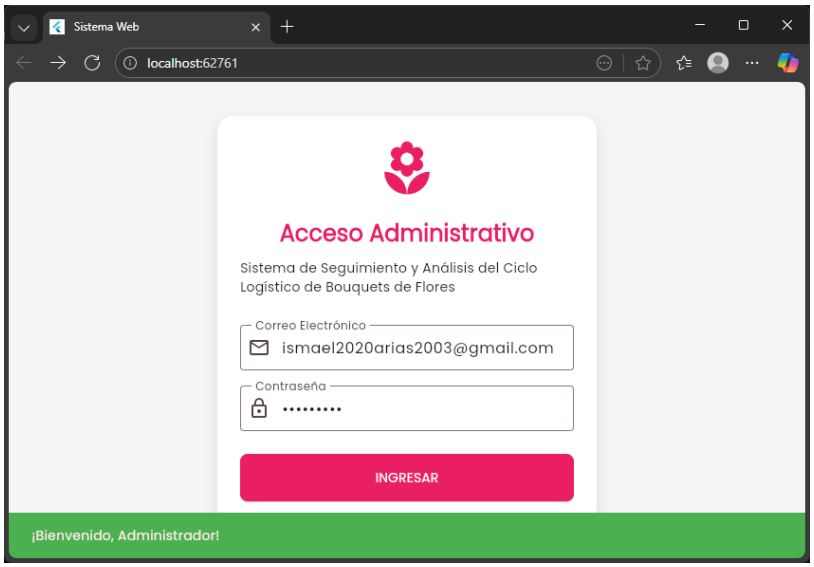


Figura 20. Log In del sistema web.
Nota. Interfaz de elaboración propia.

La carga de archivos Excel es el siguiente paso, por lo que añadir una redirección desde el Log In hacia el apartado administrativo permite organizar los siguientes pasos; realizar un archivo Excel con los ítems, dos columnas es necesario, SKU o identificador y NOMBRE, el apartado administrativo, permite la carga del archivo Excel con los ítems para la creación de los respectivos códigos QR, este módulo es observable en la figura 21.

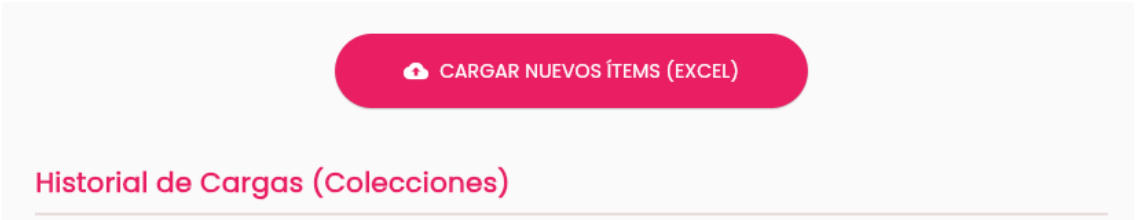


Figura 21. Módulo de carga de archivos Excel.
Nota. Interfaz de elaboración propia.

Este módulo utiliza la librería “FilePicker” para permitir al administrador seleccionar un archivo Excel desde el explorador de archivos, este se carga en memoria para su posterior procesamiento, si el administrador cancela la selección, el proceso se detiene. Posteriormente, el sistema actualiza el estado de la interfaz indicando que el archivo se

encuentra en proceso de carga, el fragmento de código empleado se lo puede observar en la figura 22, mientras que los datos de ejemplo empleados en el archivo Excel de prueba se encuentran en la tabla 9.

```
// Cargar Excel
  Future<void> _subirExcel() async {
    try {
      // 1. Seleccionar archivo
      FilePickerResult? result = await FilePicker.platform.pickFiles(
        type: FileType.custom,
        allowedExtensions: ['xlsx'],
        withData: true,
      );

      if (result == null) return;

      setState(() => _isUploading = true);
      final bytes = result.files.first.bytes;
      var excel = Excel.decodeBytes(bytes!);

      // 2. Crear la coleccion en Supabase
      final String nombreColeccion = "Carga ${DateTime.now().toString().substring(0, 16)}";

      // Insercion
      final dataColeccion = await Supabase.instance.client
          .from('colecciones')
          .insert({'nombre': nombreColeccion})
          .select()
          .single();

      final int coleccionId = dataColeccion['id'];

      // 3. Leer filas del excel e insertar items
      List<Map<String, dynamic>> itemsParaInsertar = [];
```

Figura 22. Fragmento de código para la carga de archivos Excel.
Nota. Elaboración propia a través de Carbon.now.sh.

Tabla 9
Ítems de ejemplo para Sprint 1.

SKU	NOMBRE
R-RED-01	BQT RED ROSES 50 cm
R-WHT-02	BQT WHITE ROSES 50 CM
R-PNK-03	BQT PINK ROSES 50 CM

Autor: Elaboración propia.

Para la lectura de los datos del archivo Excel, el sistema recorre cada una de las hojas del documento y procesa sus filas, omitiendo la primera ya que es la cabecera de las columnas necesarias, a partir de la segunda fila, se extraen los datos relevantes de cada registro.

Los datos se almacenan temporalmente en una estructura de tipo lista que permite agrupar todos los registros antes de enviarlos a la base de datos en la nube, cada registro se asocia a la colección creada anteriormente. El identificador del QR de cada ítem se genera automáticamente en la base de datos mediante una función UUID, la figura 23 presenta el fragmento de código utilizado para la de registros en Supabase.

```
if (itemsParaInsertar.isNotEmpty) {
        await Supabase.instance.client.from('items').insert(itemsParaInsertar);

        if (mounted) {
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
              content: Text('Se cargaron ${itemsParaInsertar.length} ítems.'),
              backgroundColor: Colors.green,
            ),
          );
        }
      }
```

Figura 23. Fragmento de código para la inserción de registros en Supabase.
Nota. Elaboración propia a través de Carbon.now.sh.

La colección es creada en Supabase en la tabla colecciones y a la par, los ítems pertenecientes a la colección son ingresados en la tabla ítems con el estado “pendiente”, cómo se puede observar a continuación en las figuras 24 y 25.

id int8	nombre text	fecha_carga timestamptz
1	Carga 2025-11-02 10:47	2025-11-02 10:47:00+00

Figura 24. Tabla “colecciones” post Sprint 1.
Nota. Captura de pantalla.

qr_token uuid	sku text	nombre text
1836bee2-d5cb-4ee6-98b3-123ef0417e59	R-RED-01	BQT RED ROSES 50 cm
3a9d1560-d7a9-4aa4-9805-28da922fc6ce	R-WHT-02	BQT WHITE ROSES 50 CM
6059d1a5-b469-4234-a3c1-cb032a500f00	R-PNK-03	BQT PINK ROSES 50 CM

Figura 25. Tabla “ítems” post Sprint 1.
Nota. Captura de pantalla.

Finalizando el Sprint 1, se implementa la creación de códigos QR con la información asociada a cada ítem, la función _crearEtiqueta() se encarga de construir el diseño de cada código. El QR se genera a partir del identificador único qrData, este es el UUID generado en Supabase, la figura 26 demuestra la implementación detallada.

```dart
// Etiqueta QR + Texto
  pw.Widget _crearEtiqueta(String sku, String nombre, String qrData) {
    return pw.Container(
      width: 150,
      height: 180,
      padding: const pw.EdgeInsets.all(10),
      decoration: pw.BoxDecoration(
        border: pw.Border.all(color: PdfColors.grey),
        borderRadius: const pw.BorderRadius.all(pw.Radius.circular(5)),
      ),
      child: pw.Column(
        mainAxisAlignment: pw.MainAxisAlignment.center,
        children: [
          // Codigo QR
          pw.BarcodeWidget(
            barcode: pw.Barcode.qrCode(),
            data: qrData, // UUID generado en Supabase
            width: 80,
            height: 80,
          ),
          pw.SizedBox(height: 10),
          pw.Text(sku, style: pw.TextStyle(fontWeight: pw.FontWeight.bold, fontSize: 10)),
          pw.Text(nombre, style: const pw.TextStyle(fontSize: 8),
          textAlign: pw.TextAlign.center, maxLines: 2),
        ],
      ),
    );
  }
```

Figura 26. Fragmento de código para la creación de códigos QR.
Nota. Elaboración propia a través de Carbon.now.sh.

Los QR generados son guardados en formato PDF en el dispositivo local, como se puede observar en la figura 27.

Figura 27. PDF con códigos QR.
Nota. Captura de pantalla.

Utilizando Google Lens, se realiza una prueba para comprobar el contenido del código QR generado, este solo debe contener el texto UUID que Supabase le asignó, de esta

forma si los códigos son escaneados por cámaras de dispositivos móviles o aplicaciones de lectura de códigos QR ajenos a la aplicación móvil, no se desplegará información sensible del ítem, simplemente se presentará un texto plano inentendible como lo expone la figura 28, comprobando así el resguardo de la información.

Figura 28. Uso de Google Lens sobre el QR generado.
Nota. Captura de pantalla.

Tarea pendiente del Sprint 0:

Al finalizar la programación del sistema web para el Sprint 1, la tarea pendiente del Sprint 0 es la creación del repositorio en GitHub para el manejo de versiones del prototipo. En github.com se crea un nuevo repositorio privado para conectar el proyecto a la nube, de esta forma se mantiene un versionamiento del prototipo en caso de necesitar regresar a versiones anteriores del desarrollo. En una terminal de Visual Studio Code, se ingresan los siguientes comandos detallados en la figura 29.

Concretada la conexión y el push, se verifica en el repositorio de GitHub que el proyecto ha sido cargado, como lo demuestra la figura 30.

Este proceso se repetirá a lo largo de la culminación de los Sprints, con el fin de mantener consistencia en los avances del prototipo.

```
# 1. Inicializar Git
git init

# 2. Agregar todos los archivos del Sprint 1
git add .

# 3. Guardar el avance
git commit -m "Finalización Sprint 1: Web, Login, Excel y QR"

# 4. Cambiar a la rama principal
git branch -M main

# 5. Conectar con Git Hub
git remote add origin LINK_DE_GITHUB

# 6. Cargar
git push -u origin main
```

Figura 29. Comandos en terminal para carga del proyecto en GitHub.
Nota. Elaboración propia a través de Carbon.now.sh.

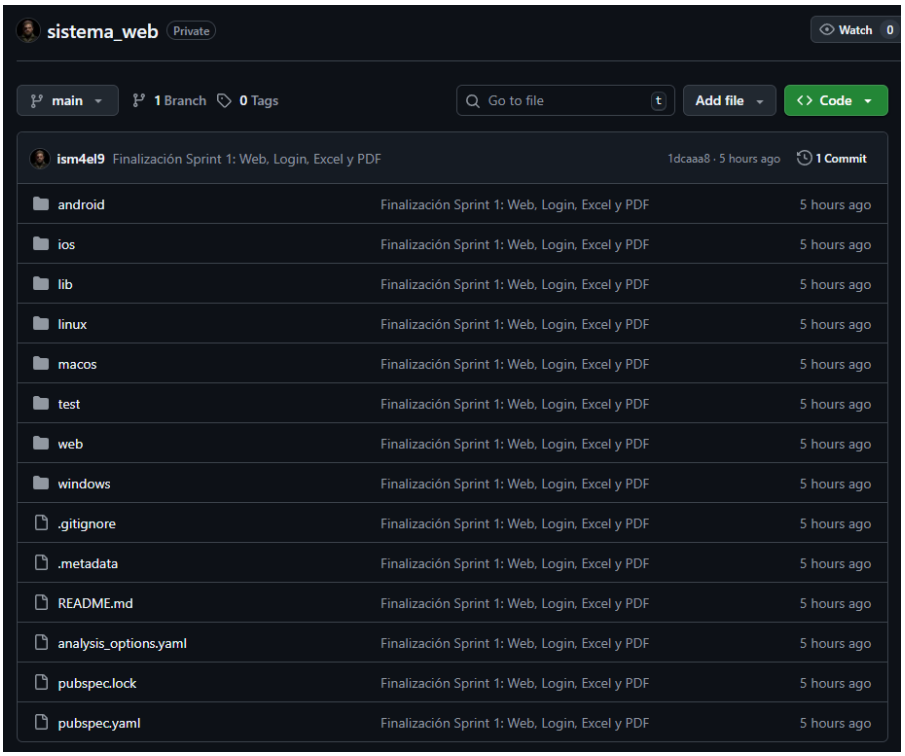


Figura 30. Comprobación del proyecto en GitHub.
Nota. Captura de pantalla.

Incremento: Sistema web desplegado que permite al administrador iniciar sesión, ingresar datos a la BDD mediante un archivo Excel y obtener los códigos QR generados a partir del archivo.

Sprint Review: 13/11/2025.

Se presenta el flujo funcional y se valida la correcta lectura del archivo Excel junto con la legibilidad de los códigos QR generados en el PDF.

Actualización del Product Backlog detallado en la tabla 10:

Tabla 10
Actualización del Product Backlog post Sprint 1

ID	Historia de Usuario	Estado
HU-01	Configuración de Github, Supabase y BDD	Terminado
HU-02	Autenticación de Administrador en Web	Terminado
HU-03	Carga de ítems vía Excel	Terminado
HU-04	Generación y descarga de PDF con códigos QR	Terminado

Autor: Elaboración propia.

3.4.5. Sprint 2: Aplicación Móvil – Módulo de Acopio

Sprint 2: Aplicación Móvil - Módulo de Acopio

Duración: 17/11/2025 – 27/11/2025

Objetivos: Desarrollar la aplicación móvil, enfocada en el perfil del operario del centro de acopio. Habilitar la lectura de códigos QR mediante la cámara del dispositivo para registrar el despacho de los bouquets y actualizar su estado en la base de datos en tiempo real.

Sprint Planning: HU-05

Infraestructura Móvil:

- Configuración inicial del proyecto y gestión de permisos de cámara en AndroidManifest.xml.
- Conexión de la aplicación móvil con Supabase.

Interfaz de Usuario:

- Diseño de la pantalla de inicio de sesión.
- Diseño de la pantalla de escaneo con cámara.

Escáner:

- Implementación de la librería mobile_scanner para la decodificación de los códigos QR generados.
- Desarrollo de la función de validación para verificar si el QR existe y actualizar el campo estado a “enviado”.

Ejecución de tareas:

Para la creación del proyecto de la aplicación móvil y la configuración de sus dependencias, se ejecutan los comandos de creación, navegación y adición a través de una terminal de Visual Studio Code, ubicada en la carpeta dónde se alojará el prototipo como se observa en la figura 31.

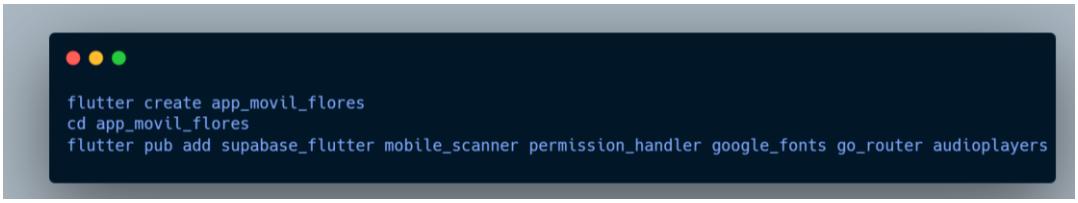


Figura 31. Comandos para la creación de la aplicación móvil.
Nota. Elaboración propia a través de Carbon.now.sh.

A diferencia del sistema web, la aplicación móvil configura el AndroidManifest.xml para brindar los accesos necesarios a internet, cámara y vibración, el fragmento de código implementado para el uso de las integraciones mencionadas se puede observar en la figura 32.

Figura 32. Edición del componente AndroidManifest.xml.
Nota. Elaboración propia a través de Carbon.now.sh.

De la misma forma que se implementó el sistema web, se conectó la aplicación a Supabase y se programó un Log In diferenciador para los operarios, implementando un selector de rol que permite al usuario seleccionar entre “Acopio” y “Distribución”. El Log In de la aplicación móvil se construyó utilizando un contenedor con ambas opciones, al seleccionar una opción, el estado de la aplicación se actualiza mediante la variable

_esAcopio, lo que permite ingresar al escaneo de códigos QR desde el centro de acopio, las figura 33 y 34 demuestran la implementación mencionada en código e interfaz.

```
Container(
  decoration: BoxDecoration(
    color: Colors.grey[200],
    borderRadius: BorderRadius.circular(12),
  ),
  child: Row(
    children: [
      // Boton ACOPIO
      Expanded(
        child: GestureDetector(
          onTap: () => setState(() => _esAcopio = true),
          child: Container(
            padding: const EdgeInsets.symmetric(vertical: 12),
            decoration: BoxDecoration(
              color: _esAcopio ? Colors.white : Colors.transparent,
              borderRadius: BorderRadius.circular(12),
              boxShadow: _esAcopio ? [const BoxShadow(color: Colors.black12, blurRadius: 4)] : [],
            ),
            child: const Center(
              child: Text("Acopio", style: TextStyle(fontWeight: FontWeight.bold)),
            ),
          ),
        ),
      ),
      // Boton DISTRIBUCION
      Expanded(
        child: GestureDetector(
          onTap: () => setState(() => _esAcopio = false),
          child: Container(
            padding: const EdgeInsets.symmetric(vertical: 12),
            decoration: BoxDecoration(
              color: !_esAcopio ? Colors.white : Colors.transparent,
              borderRadius: BorderRadius.circular(12),
              boxShadow: !_esAcopio ? [const BoxShadow(color: Colors.black12, blurRadius: 4)] : [],
            ),
            child: const Center(
              child: Text("Distribución", style: TextStyle(fontWeight: FontWeight.bold)),
            ),
          ),
        ),
      ),
    ],
  ),
),
```

Figura 33. Fragmento de código del Log In de la aplicación móvil.
Nota. Elaboración propia a través de Carbon.now.sh.

Figura 34. Log In de la aplicación Móvil.
Nota. Interfaz de elaboración propia.

Para este módulo se implementa el proceso de despacho de ítems mediante la lectura de códigos QR, la función _procesarDespacho() recibe como parámetro el identificador único del ítem (qrToken) a partir del escaneo del código QR. El sistema consulta la base de datos de Supabase para verificar si existe el ítem, si no se encuentra ningún registro, se notifica al usuario que el código no es válido o no existe en el sistema, a continuación, se demuestra la comprobación mediante un fragmento de código en la figura 35 y una prueba de escaneo de un ítem en la figura 36.

```
Future<void> _procesarDespacho(String qrToken) async {
  try {
    // 1. Busqueda del ítem por su UUID
    final List<dynamic> data = await Supabase.instance.client
        .from('items')
        .select()
        .eq('qr_token', qrToken);

    if (data.isEmpty) {
      _mostrarAlerta("Error", "Código QR no válido o no existe en sistema.", Colors.red);
      return;
    }

    final item = data.first;

    // 2. Verificacion del estado
    if (item['estado'] == 'enviado') {
      _mostrarAlerta("Atención", "Este ítem YA fue despachado anteriormente.", Colors.orange);
      return;
    }

    if (item['estado'] == 'rechazado') {
      _mostrarAlerta("Error", "Este ítem fue RECHAZADO en destino.", Colors.red);
      return;
    }

    // 3. Actualizacion (Pendiente -> Enviado)
    await Supabase.instance.client
        .from('items')
        .update({'estado': 'enviado'})
        .eq('qr_token', qrToken);

    // 4. Actualizacion completada
    try {
      await AudioPlayer().play(AssetSource('beep.mp3'));
    } catch (_) {}

    _mostrarAlerta("¡DESPACHO EXITOSO!",
    "Ítem: ${item['nombre']}\nEstado actualizado a ENVIADO.", Colors.green);

  } catch (e) {
    _mostrarAlerta("Error de Conexión", "Verifique su internet: $e", Colors.red);
  }
}
```

Figura 35. Fragmento de código del escaneo de códigos QR.
Nota. Elaboración propia a través de Carbon.now.sh.

Figura 36. Escaneo de código QR.
Nota. Interfaz de elaboración propia.

Si el ítem cumple con la validación, se actualiza su estado en la base de datos, cambiándolo de “pendiente” a “enviado”, confirmando la ejecución del despacho, para finalizar se proporciona retroalimentación al usuario, confirmando que el proceso de despacho se realizó con éxito, en caso de fallos de conexión o errores inesperados el sistema muestra un mensaje de error, la figura 37 presenta la interfaz de Supabase con registros cargados provenientes de los códigos QR.

qr_token uuid	sku text	nombre text	estado estado_item	coleccio... i...	created_at timestamptz
1836bee2-d5cb-4ee6-98b3-123ef0417e59	R-RED-01	BQT RED ROSES 50 cm	enviado	1	2025-11-02 15:47:05+00
3a9d1560-d7a9-4aa4-9805-28da922fc6ce	R-WHT-02	BQT WHITE ROSES 50 CM	pendiente	1	2025-11-02 15:47:05+00
6059d1a5-b469-4234-a3c1-cb032a500f00	R-PNK-03	BQT PINK ROSES 50 CM	pendiente	1	2025-11-02 15:47:05+00

Figura 37. Verificación de cambios de estado del ítem en Supabase.
Nota. Captura de pantalla.

Para finalizar el Sprint 2, se creó un nuevo repositorio en GitHub para la aplicación móvil, a pesar de que Flutter permite una base de código unificada, se optó por separar el repositorio administrativo del sistema web del operativo de la aplicación móvil con el fin de solo contener el código necesario para escanear los códigos QR, sin la necesidad de guardar librerías que la aplicación no necesita, la figura 38 presenta el repositorio de github de la aplicación móvil, distinto al del sistema web que conforman el prototipo.

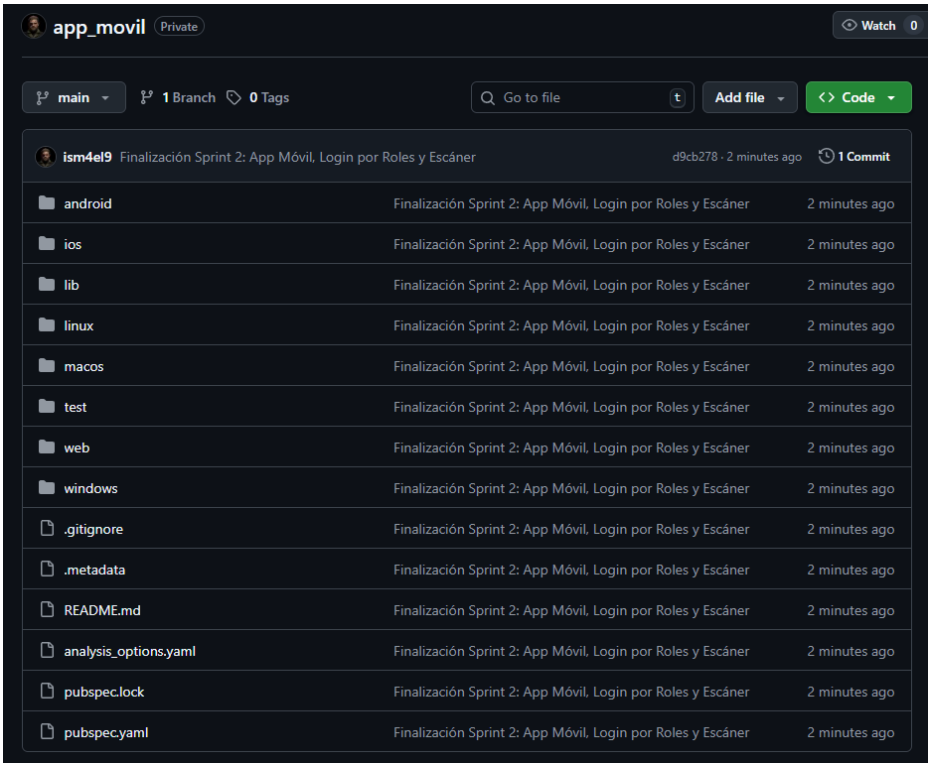


Figura 38. Repositorio en GitHub de la aplicación móvil.
Nota. Captura de pantalla.

Al igual que el Sprint 1, este proceso se repetirá a lo largo de la culminación de los Sprints.

Incremento: Aplicación móvil instalada en un dispositivo de prueba, en este caso un celular Samsung A56, que sea capaz de leer los códigos QR generados en el Sprint 1 además de tener la capacidad de cambiar el estado del ítem en la nube de “pendiente” a “enviado”.

Sprint Review: 27/11/2025.

Se realizó una demostración utilizando el teléfono móvil y los códigos QR impresos, al escanear el ítem, este aparecía inmediatamente como enviado en la base de datos.

Actualización del Product Backlog detallado en la tabla 11:

Tabla 11
Actualización del Product Backlog post Sprint 2

ID	Historia de Usuario	Estado
HU-05	Escaneo y registro de ítems.	Terminado

Autor: Elaboración propia.

3.4.6. Sprint 3: Aplicación Móvil – Módulo de Distribución

Sprint 3: Aplicación Móvil – Módulo de Distribución

Duración: 27/11/2025 – 11/12/2025

Objetivos: Implementar la lógica del módulo de distribución en la aplicación móvil, permitir al operario en destino escanear los ítems recibidos para determinar su estado "aprobado" o "rechazado", cerrando el ciclo logístico de los bouquets de flores.

Sprint Planning: HU-06

Distribución:

- Adaptación del escáner de códigos QR para el módulo de distribución.
- Implementación de validación de estados previos, recibiendo los ítems que fueron enviados desde el módulo de acopio.
- Opción A: APROBAR (cambia el estado del ítem a "aprobado").
- Opción B: RECHAZAR (cambia el estado del ítem a "rechazado" y permite ingresar el motivo).

Ejecución de tareas:

Gracias al desarrollo del Log In de la aplicación móvil en el Sprint 2, para el este módulo se implementa una ventana que permite realizar el control de calidad de los ítems recibidos. La función _mostrarDialogoCalidad() despliega una interfaz que solicita al usuario aprobar o rechazar un ítem posterior evaluación de su estándar de calidad.

El módulo ofrece dos opciones:

- Aprobar: confirma que el ítem cumple con los estándares de calidad, actualizando su estado en el sistema a "aprobado".

- Rechazar: indica que el bouquet no cumple con los estándares de calidad, desplegando opciones provenientes del catálogo de motivos de rechazo, posterior selección de la causa, el estado del ítem cambia a “rechazado”.

La figura 39 presenta el fragmento de código empleado para la funcionalidad mencionada, mientras que las figuras 40, 41 y 42 presentan la interfaz de usuario aplicando el código.

```
// Aprobar o rechazar item
void _mostrarDialogoCalidad(String nombreItem, String qrToken) {
    showDialog(
      context: context,
      barrierDismissible: false,
      builder: (ctx) => AlertDialog(
        title: const Text("Control de Calidad", textAlign: TextAlign.center),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            const Icon(Icons.local_florist, size: 50, color: Colors.teal),
            const SizedBox(height: 10),
            Text("Item: $nombreItem", style: const TextStyle(fontWeight: FontWeight.bold)),
            const SizedBox(height: 20),
            const Text("¿Condición a la llegada?"),
          ],
        ),
        actions: [
          // Rechazar - Seleccion del motivo en el catalogo
          TextButton.icon(
            icon: const Icon(Icons.thumb_down),
            label: const Text("RECHAZAR"),
            style: TextButton.styleFrom(foregroundColor: Colors.red),
            onPressed: () {
              Navigator.pop(ctx);
              _mostrarSelectorMotivos(qrToken);
            },
          ),
          // Aprobar
          ElevatedButton.icon(
            icon: const Icon(Icons.check_circle),
            label: const Text("APROBAR"),
            style: ElevatedButton.styleFrom(backgroundColor: Colors.green, foregroundColor:
Colors.white),
            onPressed: () {
              Navigator.pop(ctx);
              _actualizarEstado(qrToken, 'aprobado', null, null);
            },
          ),
        ],
        actionsAlignment: MainAxisAlignment.spaceEvenly,
      ),
    );
  }
```

Figura 39. Fragmento de código de aprobación o rechazo de ítems.
Nota. Elaboración propia a través de Carbon.now.sh.

Figura 40. Escaneo de ítems en el Módulo de Distribución.
Nota. Interfaz de elaboración propia.

Figura 41. Selección de aprobación o rechazo del ítem.
Nota. Interfaz de elaboración propia.

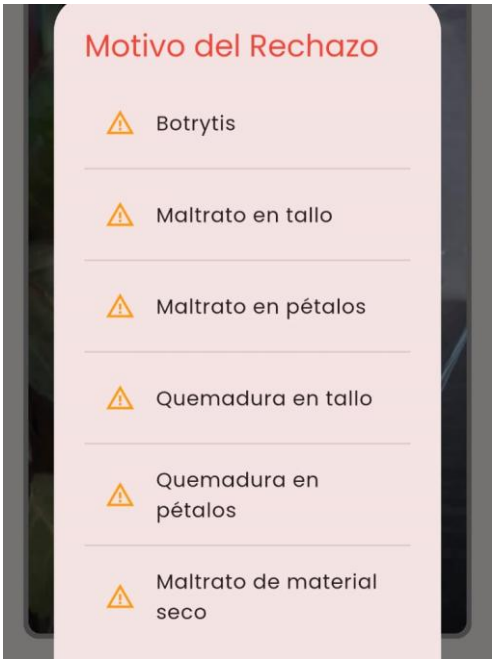
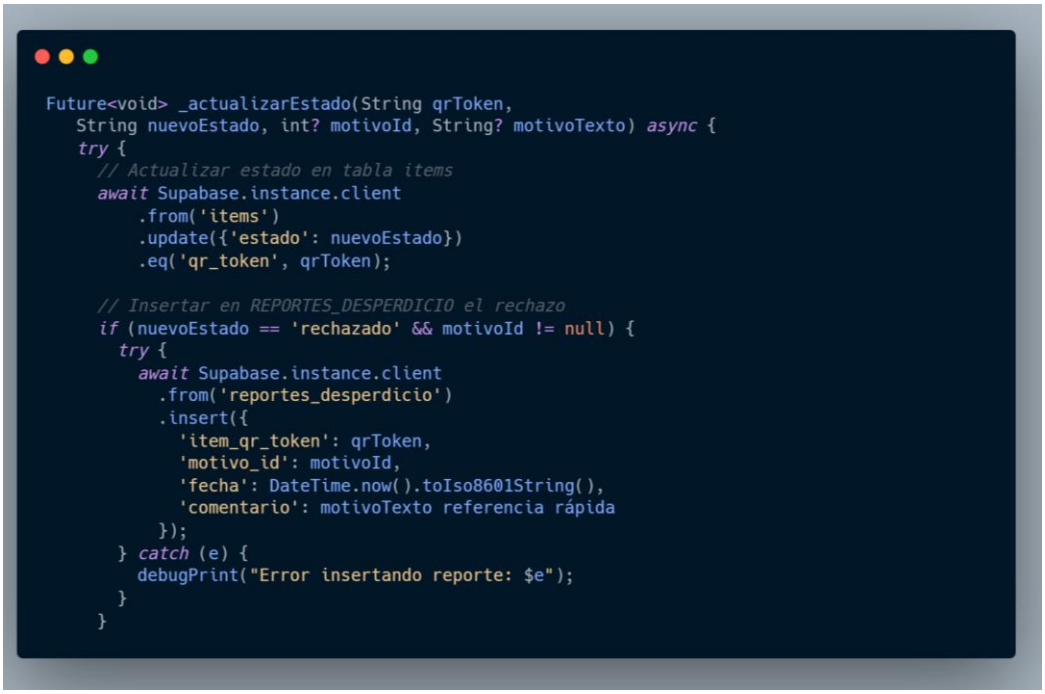


Figura 42. Selección de aprobación o rechazo del ítem.
Nota. Interfaz de elaboración propia.

Para la culminación del Sprint 3, se implementa la función _actualizarEstado() para la gestión de cambios de estado de los ítems en la base de datos, utilizando como identificador principal el token QR asociado a cada bouquet.

El sistema actualiza el estado del ítem con la selección del operario de Distribución, puede ser aprobado o rechazado, cuando el estado sea actualizado y si es un rechazo, automáticamente se registra un reporte en la tabla reportes_desperdicio, garantizando así el seguimiento del ciclo logístico de los bouquets y la recopilación de datos necesario para el dashboard del sistema web, a continuación, se presenta la actualización del estado de los ítems de ejemplo, en la figura 43 se encuentra un fragmento del código perteneciente a este módulo y las figuras 44 y 45 la comprobación del evento mencionado a través de la interfaz de Supabase.

Incremento: Actualización de la App Móvil con la integración del módulo de distribución, permitiendo cerrar el ciclo logístico de los bouquets de flores reflejando en la base de datos la aprobación o rechazo de los ítems.



```
Future<void> _actualizarEstado(String qrToken,
   String nuevoEstado, int? motivoId, String? motivoTexto) async {
   try {
     // Actualizar estado en tabla items
     await Supabase.instance.client
         .from('items')
         .update({'estado': nuevoEstado})
         .eq('qr_token', qrToken);

     // Insertar en REPORTES_DESPERDICIO el rechazo
     if (nuevoEstado == 'rechazado' && motivoId != null) {
       try {
         await Supabase.instance.client
           .from('reportes_desperdicio')
           .insert({
             'item_qr_token': qrToken,
             'motivo_id': motivoId,
             'fecha': DateTime.now().toIso8601String(),
             'comentario': motivoTexto referencia rápida
           });
       } catch (e) {
         debugPrint("Error insertando reporte: $e");
       }
     }
```

Figura 43. Fragmento de código del registro de desperdicio.
Nota. Elaboración propia a través de Carbon.now.sh.

qr_token uuid	sku text	nombre text	estado estado_item	coleccio... i...	created_at timestamptz
1836bee2-d5cb-4ee6-98b3-123ef0417e59	R-RED-01	BQT RED ROSES 50 cm	aprobado	1	2025-11-02 15:47:05+00
3a9d1560-d7a9-4aa4-9805-28da922fc6ce	R-WHT-02	BQT WHITE ROSES 50 CM	rechazado	1	2025-11-02 15:47:05+00
6059d1a5-b469-4234-a3c1-cb032a500f00	R-PNK-03	BQT PINK ROSES 50 CM	rechazado	1	2025-11-02 15:47:05+00

Figura 44. Actualización del estado de los ítems en Supabase.
Nota. Captura de pantalla.

public.motivos_rechazo | public.items | public.reportes_desperdicio | public.colecciones | public.usuarios

Filter | Sort | Insert | 1 RLS policy | Index Advisor | Role postgres

id int8	fecha timestamptz	comentario text	item_qr_token uuid	motivo_id int8
1	2025-12-08 11:49:15.101762+00	Botrytis	3a9d1560-d7a9-4aa4-9805-28da922fc6ce	1
2	2025-12-08 11:49:29.901238+00	Quemadura en pétalos	6059d1a5-b469-4234-a3c1-cb032a500f00	5

Figura 45. Creación del registro en la tabla reportes_desperdicio en Supabase.
Nota. Captura de pantalla.

Sprint Review: 11/12/2025.

Al rechazar un ítem, el sistema actualiza correctamente el estado del ítem a “rechazado” e inserta un registro en la tabla “reportes_desperdicio” para vincular el motivo específico y la fecha en la que el bouquet de flores fue rechazado.

Actualización del Product Backlog detallado en la tabla 12:

Tabla 12
Actualización del Product Backlog post Sprint 3

ID	Historia de Usuario	Estado
HU-06	Escaneo de recepción y validación de ítems, aprobado o rechazado.	Terminado
HU-07	Selección dinámica de motivos de rechazo desde catálogo	Terminado

Autor: Elaboración propia.

3.4.7. Sprint 4: Sistema Web - Tableros de Control y Reportes Gerenciales

Sprint 4: Sistema Web - Tableros de Control y Reportes Gerenciales.

Duración: 11/12/2025 – 22/12/2025

Objetivos: Implementar el módulo de inteligencia de negocios (BI) básico para el perfil administrador. El objetivo es transformar los datos operativos recolectados por los dispositivos móviles en el centro de acopio (origen) y la bodega de distribución (destino) en información visual mediante indicadores KPIs (Key Performance Indicators) llamados también como indicadores clave de desempeño y gráficos que permitan la toma de decisiones sobre la calidad y eficiencia del proceso de exportación de los ítems.

Sprint Planning: HU-08

Lógica de Negocio:

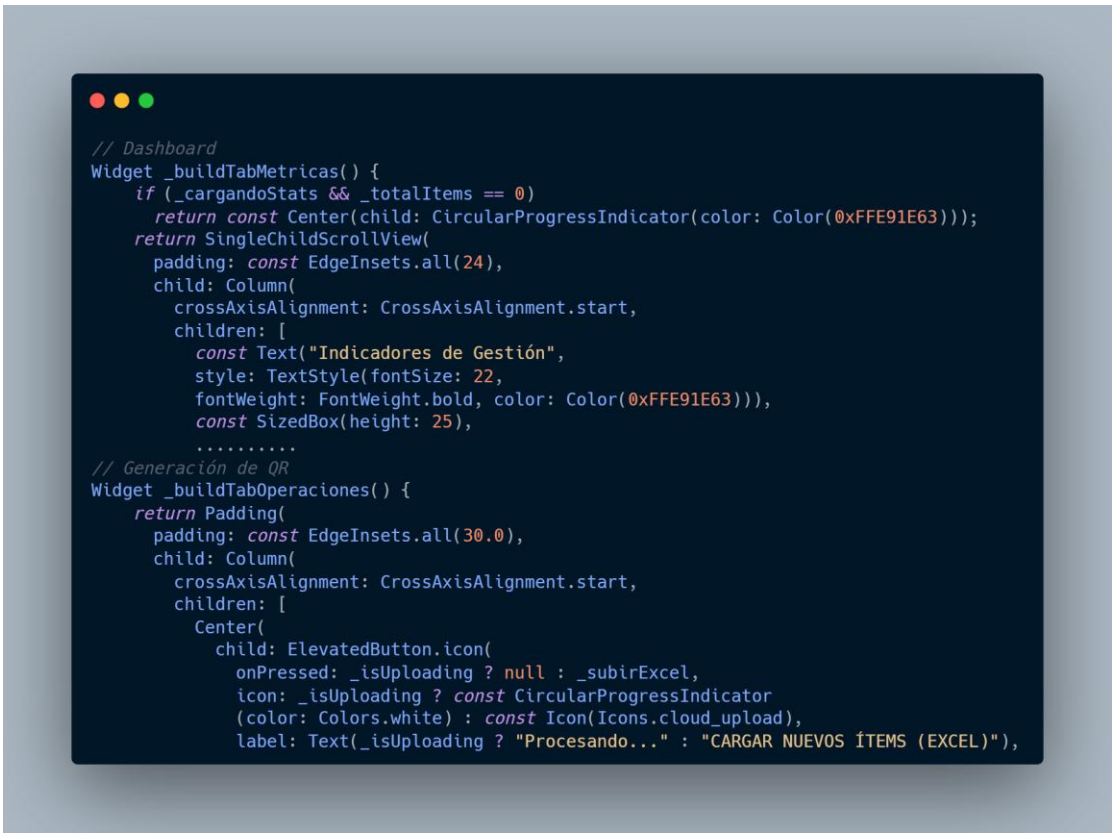
- Conteo de ítems por estados: pendiente, enviado, aprobado, rechazado.
- Despliegue de causas de desperdicio para análisis y toma de decisiones

Interfaz de Usuario: HU-09

- Integración de la librería fl_chart para la renderización de gráficos estadísticos en Flutter Web.
- Indicadores KPI.
- Implementación de gráficos.

Ejecución de tareas:

Gracias al desarrollo previo del Sistema Web en el Sprint 1 para la generación de los códigos QR a partir de un archivo Excel, este Sprint separó la generación de los códigos del dashboard a través de pestañas como se muestra en el fragmento de código a continuación en la figura 46.



```
// Dashboard
Widget _buildTabMetricas() {
    if (_cargandoStats && _totalItems == 0)
      return const Center(child: CircularProgressIndicator(color: Color(0xFFE91E63)));
    return SingleChildScrollView(
      padding: const EdgeInsets.all(24),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          const Text("Indicadores de Gestión",
          style: TextStyle(fontSize: 22,
          fontWeight: FontWeight.bold, color: Color(0xFFE91E63))),
          const SizedBox(height: 25),
          ..........
// Generación de QR
Widget _buildTabOperaciones() {
    return Padding(
      padding: const EdgeInsets.all(30.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Center(
            child: ElevatedButton.icon(
              onPressed: _isUploading ? null : _subirExcel,
              icon: _isUploading ? const CircularProgressIndicator
              (color: Colors.white) : const Icon(Icons.cloud_upload),
              label: Text(_isUploading ? "Procesando..." : "CARGAR NUEVOS ÍTEMS (EXCEL)"),
```

Figura 46. Fragmentos de código para la separación del Dashboard
Nota. Elaboración propia a través de Carbon.now.sh.

Una vez separado el dashboard del módulo de los códigos QR, la integración de indicadores KPIs y gráficos estadísticos es posible a través de la librería fl_chart y cálculos generados en este apartado.

Por ejemplo, el porcentaje de aprobación presentado en la figura 47, este mide la proporción de ítems que cumplieron con los estándares de calidad, en relación con el total de bouquets enviados.

```
int totalProcesados = a + r;
String pctA = totalProcesados > 0 ?
  "${((a / totalProcesados) * 100).toStringAsFixed(1)}%" : "0%";
String pctR = totalProcesados > 0 ?
  "${((r / totalProcesados) * 100).toStringAsFixed(1)}%" : "0%";
      double efectividad = t > 0 ? (a / t) * 100 : 0.0;
      if (mounted) {
        setState(() {
          _listaItemsDetalle = items;
          _mapaRazonesRechazo = mapaRazones;
          _totalItems = t;
          _pendientes = p;
          _enviados = e;
          _aprobados = a;
          _rechazados = r;
          _pctAprobados = pctA;
          _pctRechazados = pctR;
          _balanceNeto = a - r;
          _porcentajeExito = efectividad;
          _conteoMotivos = conteoMotivos;
          _produccionPorFecha = conteoFechas;
          _cargandoStats = false;
        });
      }
    } catch (e) {
      debugPrint("Error Stats: $e");
      if (mounted) setState(() => _cargandoStats = false);
    }
```

Figura 47. Fragmento de código del cálculo de indicadores KPIs.
Nota. Elaboración propia a través de Carbon.now.sh.

El resultado es un dashboard web que despliega en tiempo real los datos proporcionados por la aplicación móvil gracias al protocolo WebSocket, presentando indicadores numéricos, porcentajes, gráficos estadísticos y una tabla para el monitoreo del ciclo logístico de bouquets de flores como se demuestra en las figuras 48 y 49.

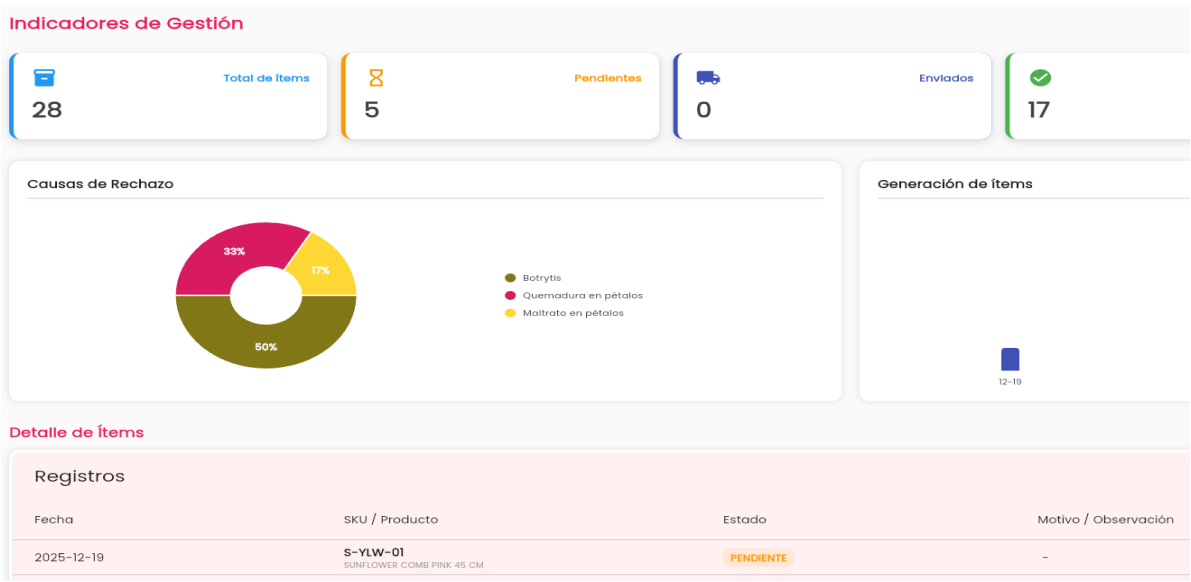


Figura 48. Despliegue de indicadores y gráficos en el Dashboard Web.
Nota. Interfaz de elaboración propia.

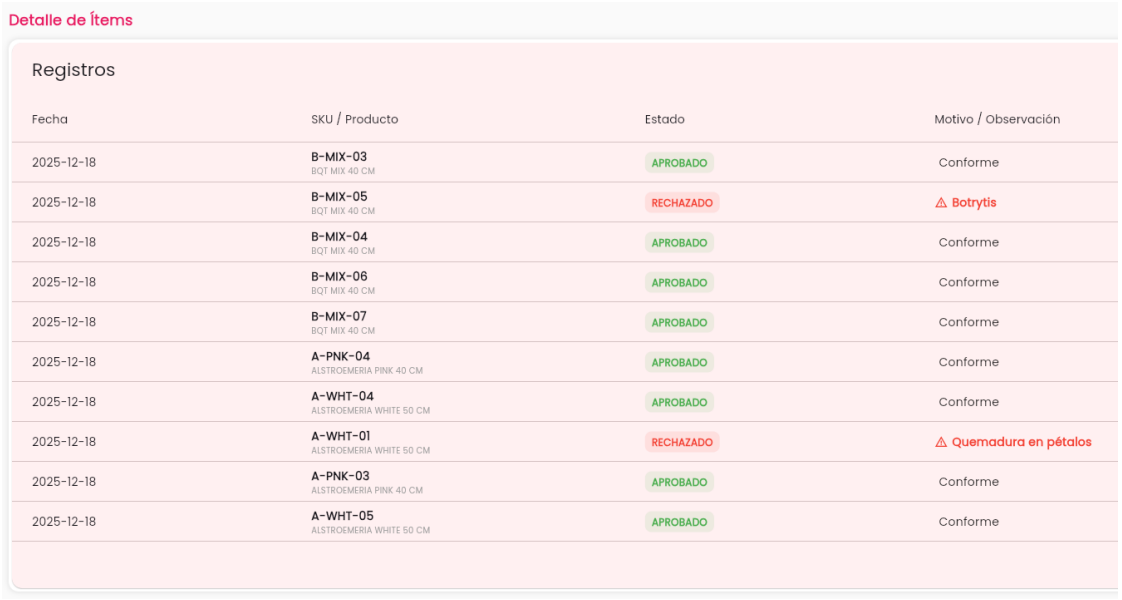


Figura 49. Despliegue de tabla y detalles en el Dashboard Web.
Nota. Interfaz de elaboración propia.

Incremento: Dashboard Web que despliega información en tiempo real de la base de datos en Supabase, el prototipo ahora cubre el ciclo logístico de bouquets de flores, la carga de ítems mediante archivos Excel para su monitoreo en el despacho de producto y control de calidad a través de la aplicación móvil.

Sprint Review: 22/12/2025

Se verificó el despliegue de indicadores para verificación de ítems pendientes, enviados, aprobados y rechazados, el cálculo automático de porcentajes de efectividad y rechazo facilita la lectura rápida del estado del negocio.

El gráfico de pastel permite identificar inmediatamente los motivos de rechazo más frecuentes, cumpliendo con el objetivo de facilitar la toma de decisiones para evitar el desperdicio de flor, finalmente, la tabla inferior ofrece trazabilidad completa, permitiendo al administrador ver el ID de la colección y el motivo específico de rechazo de cada ítem.

Actualización del Product Backlog detallado en la tabla 13:

Tabla 13
Actualización del Product Backlog post Sprint 4

ID	Historia de Usuario	Estado

HU-08	Visualización de indicadores KPIs, ítems pendientes, enviados, aprobados, rechazados.	Terminado
HU-09	Gráfico estadístico de motivos de rechazo.	Terminado

Autor: Elaboración propia.

CAPITULO IV: EVALUACIÓN DE RESULTADOS

4.1. Escenario de pruebas

Las validaciones no se limitaron a una fase final, estas fueron el resultado de pruebas constantes durante cada Sprint, este enfoque ágil permitió experimentar con el prototipo de forma continua, facilitando la medición de los tiempos del escenario de los operarios de acopio y distribución.

La validación final se realizó en un entorno controlado que simuló el ciclo logístico de exportación de bouquets de flores desde un centro de acopio hasta una bodega de distribución en el exterior, se utilizaron datos de prueba representativos y 5 ítems con los códigos QR generados por el sistema.

4.2. Pruebas funcionales

Verificación del cumplimiento de requerimientos funcionales definidos en el Capítulo III:

- Generación de códigos QR: El sistema web procesó exitosamente archivos Excel, generando etiquetas únicas en formato PDF con identificadores UUID.
- Registro de despacho: El operario de acopio logró escanear los ítems, cambiando su estado a “enviado” en la base de datos de Supabase de manera instantánea.
- Control de calidad: En el módulo de distribución se validó la capacidad de clasificar ítems como “aprobados” y "rechazados", en caso de marcar un ítem cómo rechazado, se permite seleccionar el motivo específico desde un catálogo dinámico.
- Sincronización de información en tiempo real: El dashboard web reflejó los cambios de estado y las métricas de desperdicio de flor inmediatamente después de cada escaneo gracias al uso de WebSockets

4.3. Evaluación de eficiencia: método manual vs. Prototipo

La tabla 14, presenta 3 actividades para la comparación del registro de ítems rechazados a través del método tradicional manual en contraste con el uso del prototipo, estas son; el registro de datos, procesamiento y comunicación, detallando los tiempos totales empleados en cada método.

Tabla 14
Evaluación de eficiencia

Actividad	**Método Tradicional (Manual/Correo)**	**Prototipo Digital (App/Web)**
Registro de datos	Escritura manual en papel de los ítems y sus motivos del rechazo.	Escaneo de código QR y selección de motivo.
Procesamiento	Transcripción de datos a Excel o reporte en Word.	Automatizado en la base de datos.
Comunicación	Redacción y envío de correo electrónico a interesados.	Actualización instantánea vía WebSockets en el dashboard.
Tiempo Total Estimado	7 - 12 minutos	30 segundos

Autor: Elaboración propia.

El prototipo reduce el tiempo de reporte de desperdicio de flor en un 92%, mientras que el método manual es propenso a errores de transcripción y retrasos en la toma de decisiones, considerando que la información es enviada una vez culmine una distribución de ítems diaria o semanal, el prototipo garantiza disponibilidad inmediata de la información, ahorrando días de ausencia de comunicación.

4.4. Impacto en la toma de decisiones

La implementación del prototipo permite transformar datos operativos que son capturados en campo a información estratégica visualizada en un dashboard en tiempo real. A partir de la simulación realizada con los ítems de prueba se han identificado escenarios críticos donde el sistema facilita decisiones inmediatas para reducir el desperdicio de flor como se puede observar en la figura 50.

SKU / Producto	Estado	Motivo / Observación
S-YLW-01 SUNFLOWER COMB PINK 45 CM	APROBADO	Conforme
S-YLW-02 SUNFLOWER COMB PINK 45 CM	APROBADO	Conforme
S-YLW-03 SUNFLOWER COMB PINK 45 CM	RECHAZADO	⚠ Botrytis
S-YLW-04 SUNFLOWER COMB PINK 45 CM	RECHAZADO	⚠ Botrytis
S-YLW-05 SUNFLOWER COMB PINK 45 CM	RECHAZADO	⚠ Maltrato en pétalos

Figura 50. Ítems del escenario de prueba.
Nota. Interfaz de elaboración propia.

Al registrar un ítem como rechazado debido a botrytis, el sistema envía la información al tablero de control, esta visualización inmediata permite tomar la decisión de solicitar inspecciones urgentes en la finca de la que proviene la especie floral para frenar la entrega de producto infectado, evitando que el hongo se propague a otras flores durante el transporte.

Clasificar un ítem como rechazado por quemadura en sus pétalos, puede determinar si el fallo se origina desde finca por las condiciones en las que se encuentra la flor durante su crecimiento o por parte del centro de acopio debido al uso de materiales secos inadecuados como cartón separador o mangas con material dañino, permitiendo ajustar la receta de empaque de las colecciones en desarrollo de manera inmediata.

El gráfico de pastel presentado en la figura 51, centraliza los motivos de rechazo de todos los ítems escaneados, sin limitarse a una sola colección, al visualizar, por ejemplo, que un alto porcentaje de rechazos se debe a botrytis o maltrato en pétalos se puede identificar fallos en la cadena de frío o en la manipulación de los bouquets en el centro de acopio.

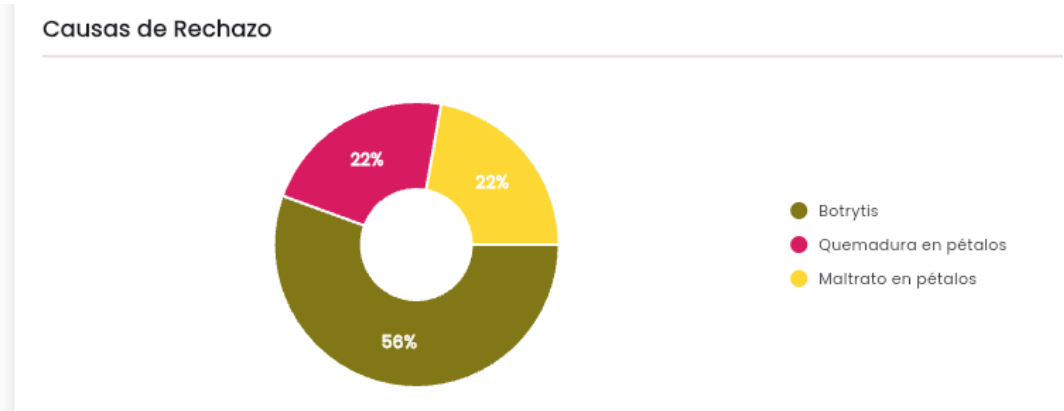


Figura 51. Gráfico de pastel.
Nota. Interfaz de elaboración propia.

Los indicadores KPIs del sistema, plasmados en la figura 52, presentan una visión general del estado de los ítems, en la simulación, el administrador observa el porcentaje de efectividad basado en el total de ítems escaneados. Esta perspectiva permite evaluar la economía y operatividad del negocio, permitiendo identificar si los estándares de calidad actuales son sostenibles a largo plazo o si se requieren ajustes.

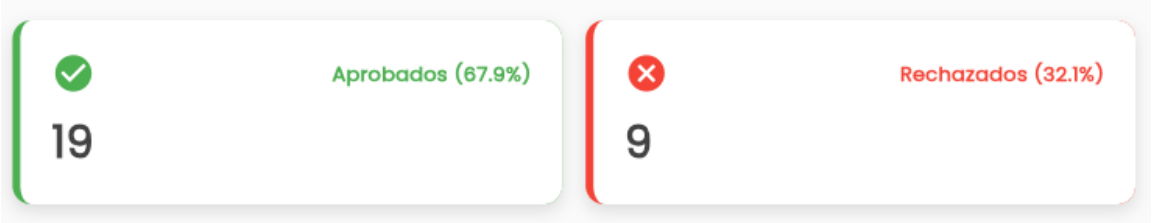


Figura 52. Gráfico de indicadores KPIs.
Nota. Interfaz de elaboración propia.

CAPITULO V: CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones del trabajo de titulación

El presente trabajo de titulación permitió evidenciar la problemática del desperdicio de flor en el ciclo logístico de los bouquets de flores, originada principalmente por una gestión tradicional basada en registros manuales, la cual es propensa a errores y limita la toma de decisiones oportunas.

La implementación de la arquitectura Modelo Vista Controlador (MVC) y el modelo Backend as a Service (BaaS) de Supabase en el prototipo, garantizaron la integridad de la información. El uso de identificadores universales (UUID) en los códigos QR permitió obtener una trazabilidad única y consistente en PostgreSQL, eliminando duplicidades en el seguimiento logístico de los bouquets de flores.

La implementación de Flutter y Dart permitió unificar la lógica de negocio en una solución multiplataforma de alto rendimiento. La elección de estas herramientas optimizó los tiempos de desarrollo y digitalizó el flujo de información mediante la integración eficiente de códigos QR en archivos PDF.

La digitalización del control de calidad permitió capturar causas puntuales de desperdicio de flor que antes eran llevadas de forma manual. El prototipo proporcionó visibilidad inmediata sobre estos motivos través de un dashboard administrativo, permitiendo contar con datos precisos para identificar puntos críticos de control, esto transforma la gestión de los datos basada en correos electrónicos a reportes gráficos e indicadores KPIs en tiempo real.

La aplicación de la metodología SCRUM facilitó el desarrollo incremental del prototipo gracias a las pruebas constantes del mismo en cada ciclo de trabajo, demostrando la reducción del tiempo de reporte de desperdicio de flor de un promedio de 10 minutos a 30 segundos. El uso del protocolo WebSocket eliminó las brechas de información mediante el monitoreo en tiempo real de los datos, agilizando la toma de decisiones para mitigar pérdidas económicas.

5.2. Recomendaciones del trabajo de titulación

Se recomienda ampliar el alcance del prototipo mediante la implementación de APIs que permitan la sincronización automática de los datos de trazabilidad logística con los sistemas de planificación de recursos empresariales (ERPs) ya existentes en la industria florícola. Esto permitiría que el registro de un ítem como desperdicio de flor afecte directamente al inventario de contabilidad y a las proyecciones de futuras ventas, optimizando la cadena de suministro desde una plataforma con información centralizada.

Para profundizar en el análisis de las causas biológicas del desperdicio de flor, como la proliferación de hongos en ambientes de alta humedad, se sugiere integrar sensores de Internet de las Cosas (IoT) que monitoreen la temperatura y humedad de los ítems durante su tránsito, los datos recabados podrían vincularse al código QR de cada bouquet, permitiendo determinar con exactitud si el daño se originó por una ruptura en la cadena de frío o por la manipulación del producto.

Finalmente se recomienda realizar un despliegue en plan piloto en un entorno comercial real, involucrando a una mayor cantidad de operarios tanto en centros de acopio como en bodegas internacionales, es fundamental diseñar programas de capacitación para el personal de operativo de campo, asegurando que el uso de dispositivos móviles y la lectura de códigos QR se convierta en una práctica común que sustituya definitivamente el uso de papel y lápiz, minimizando el error humano y garantizando la precisión y disponibilidad de la información recolectada.

REFERENCIAS

Betancourt Espinoza, A., Tuz Gia, S. N., & Salcedo Muñoz, V. E. (2025). Sector florícola: Contribución al desarrollo socioeconómico del Ecuador, 2015 – 2024. *GADE: Revista Científica, 5*(1), 755-778. https://doi.org/https://doi.org/10.63549/rg.v5i1.653

Cárdenas, N. (2025). *Identificación de Botrytis en Rosas de exportación (Rosa Sp.) de la variedad Light House en la parroquia Mulaló del cantón Latacunga.* Universidad Técnica de Cotopaxi. Latacunga: Universidad Técnica de Cotopaxi. https://repositorio.utc.edu.ec/server/api/core/bitstreams/e852b34f-7492-4874-a4e3-19228a02449c/content

Diego. (2025, octubre 12). *Amazon, el mayor caso de éxito en Logística y Supply Chain*. Sdworks: https://sdworks.cl/2024/03/26/amazon-el-mayor-caso-de-exito-en-logistica-y-supply-chain/

Gordon Graell, R. D. (2023). Ingeniería de Software: Aplicaciones e Innovaciones de los Códigos QR en los Sistemas de Información y Bases De Datos. *Synergía, 2*(1), 20-34. https://revistas.up.ac.pa/index.php/synergia/article/view/3778

Marín, A. (2024). *La Trazabilidad como Herramienta Clave en la Gestión de Cadenas de Suministro Eficientes y Seguras.* Universidad Miguel Hernández. Elche: Universidad Miguel Hernández. https://dspace.umh.es/handle/11000/33206

Martins, J. (2025, febrero 15). *Scrum: conceptos clave y cómo se aplica en la gestión de proyectos*. Asana: https://asana.com/es/resources/what-is-scrum

Murley, P., Ma, Z., Mason, J., Bailey, M., & Kharraz, A. (2021). WebSocket Adoption and the Landscape of the Real-Time Web. *WWW '21: Proceedings of the Web Conference 2021*, (pp. 1192-1203). Ljubljana, Slovenia. https://doi.org/https://doi.org/10.1145/3442381.3450063

Padrón, A., Meléndez, R., & Treviño, C. (2020). Confidencialidad de datos mediante el grabado de códigos QR cifrados: ID-óptico. *Revista de I+D Tecnológico, 16*(2). http://portal.amelica.org/ameli/journal/339/3391369007/3391369007.pdf

Paolone, G., Marinelli, M., Paesani, R., & Di Felice, P. (2020). Automatic Code Generation of MVC Web Applications. *Computers, 9*(3), 56. https://doi.org/Https://doi.org/10.3390/computers9030056

Quedena, J. (2019). *Impacto de usar Scrum dentro de la transformación digital: una revisión de la literatura científica.* Universidad Privada del Norte. Trujillo: Universidad Privada del Norte. https://doi.org/https://repositorio.upn.edu.pe/backend/api/core/bitstreams/abf82253-5b4b-4c0a-835a-da0b4892e3ec/content

Rodríguez, C., & Vicente, R. D. (2015). ¿Por qué implementar Scrum? *Revista de la Facultad de Ingeniería*(34), 15-22. https://dialnet.unirioja.es/servlet/articulo?codigo=8705520

Silvestre, T. (2021). *Estudio comparativo de las versiones más recientes de HTTP.* Universitat Politècnica de València. Valencia: Universitat Politècnica de València. https://riunet.upv.es/server/api/core/bitstreams/6a75f417-0932-49a9-9eda-f4acc6638820/content

Supabase. (2025, diciembre 15). *Database | Supabase Docs*. Supabase Docs: https://supabase.com/docs/guides/database/overview

Tu, Y.-C., Terragni, V., Tempero, E., Shakil, A., Meads, A., Giacaman, N., Fowler, A., & Blincoe, K. (2022). GitHub in the classroom: Lessons learnt. *Proceedings of the 24th Australasian Computing Education Conference* (pp. 163–172). Association for Computing Machinery. https://doi.org/https://doi.org/10.1145/3511861.3511879

Uunonen, S. (2025). *Backend as a Service in web development.* Häme University of Applied Sciences. Hämeenlinna: Häme University of Applied Sciences. https://www.theseus.fi/bitstream/handle/10024/801937/Uunonen_Samu.pdf?sequence=2