**PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR**

**Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC**

# PAPELTRACK: APLICACIÓN MÓVIL PARA EL CONTROL DE INVENTARIO EN PAPELERÍA

**Proyecto de titulación previo a la obtención del título de: Tecnólogo Superior en Desarrollo de Software**

**Autor: BRITO ALAVA ANDRES DAVID**

**Tutor: Quespaz Sánchez Jonathan David**

**Quito, Ecuador**

**2026**

## Dedicatoria

Dedico este trabajo a mis seres queridos, por estar siempre a mi lado, entenderme y animarme continuamente en mi camino académico. Su apoyo, paciencia y motivación fueron clave para que no me diera por vencido y pudiera finalizar este proceso con éxito. A mis amigos, por su compañía, su apoyo constante y por hacer que el esfuerzo diario fuera más llevadero, aportando felicidad y energía a lo largo de este proyecto. Por último, a mis abuelos, por sus lecciones, su modelo de tenacidad y por ser una fuente continua de inspiración en cada fase de mi vida.

# Contenido

## Lista de figuras



## Lista de Tablas

**Anexo**

**Declaración y autorización**

Yo, **BRITO ALAVA ANDRES DAVID** con C.I. 1753731122 autor del trabajo de Integración Curricular intitulado: **"PAPELTRACK: APLICACIÓN MÓVIL PARA EL CONTROL DE INVENTARIO EN PAPELERÍA"**, previa a la obtención del título de **Tecnólogo Superior en Desarrollo de Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 13 de febrero de 2026

BRITO ALAVA ANDRES DAVID

C.I. 1753731122

## Agradecimientos

Quiero agradecer primero a mi familia. Son quienes han estado conmigo en cada etapa, incluso en los días en los que yo mismo dudaba. Su paciencia, sus palabras y hasta esos pequeños empujones que me daban cuando me estancaba hicieron que este trabajo sea posible. Gracias por estar ahí, por acompañarme en silencio cuando hacía falta y por recordarme que podía terminar esto. De verdad, no sé cómo habría llegado hasta aquí sin ustedes.

También agradezco a mi tutor, Jonathan David Quespaz Sánchez, por su guía y sus observaciones. Cada comentario ayudó a mejorar el proyecto y a corregir cosas que yo no veía en el momento. Valoré mucho su disposición para revisar mi trabajo y darme una dirección más clara cuando la necesitaba.

Este trabajo es mío, sí, pero también es de todas estas personas que me acompañaron de una u otra forma. Gracias por hacer este camino menos pesado y por ser parte de este logro conmigo.

## Resumen

Este trabajo presenta el diseño y la puesta en marcha de PapelTrack, una herramienta móvil pensada específicamente para que pequeños negocios dejen atrás el desorden de los registros en papel. A diferencia de otros sistemas complejos, PapelTrack centraliza el control de stock, compras y ventas en una interfaz sencilla de usar. Para lograrlo, decidí trabajar con tecnologías nativas de Android (Kotlin), buscando que la aplicación fuera rápida y no fallara al integrarse con el teléfono. En la parte técnica, el sistema se apoya en un servidor robusto con Node.js y una base de datos PostgreSQL. Durante el proceso no solo me enfoqué en que el código funcionara, sino en adaptar la aplicación a lo que el usuario realmente necesitaba en su día a día, realizando ajustes importantes que quedaron documentados en los anexos. Al final, logramos una solución que no solo digitaliza datos, sino que realmente ayuda a que el dueño del negocio no pierda dinero por errores manuales.

Finalmente, PapelTrack cumple con los objetivos propuestos al ofrecer una solución práctica para la digitalización del inventario, sentando una base sólida para futuras ampliaciones como reportes avanzados, roles de usuario y funcionamiento sin conexión.

## Summary

This article presents the development of PapelTrack, a mobile application designed to support small and medium-sized enterprises in the efficient management of inventories. The application allows users to register products, organize categories, manage purchases and sales, and centralize operational information within a single system, with the aim of reducing errors derived from manual processes and improving inventory control.

The project was developed using native Android technologies, which made it possible to optimize performance, stability, and integration with the operating system, without the need to generate additional platform versions. An agile methodology was applied to organize the development process, enabling incremental delivery, continuous module evaluation, and adaptation to real user requirements throughout the project execution.

As technological support, a backend based on Node.js and a PostgreSQL database was implemented, ensuring secure and structured data management. Communication between the mobile application and the server was carried out through APIs, guaranteeing interoperability between the mobile client and the backend infrastructure.

In order to validate the system's reliability and performance, functional and performance tests were conducted on both the backend and mobile devices, evaluating aspects such as data persistence, operational concurrency, and interface responsiveness. During the execution of the project, adjustments were made to the initial plan, mainly related to functional scope and technological decisions, which were formally documented and justified in the corresponding annexes.

Finally, PapelTrack meets the proposed objectives by providing a practical solution for inventory digitalization, establishing a solid foundation for future enhancements such as advanced reporting, user roles, and offline functionality.

Palabras claves

Inventario, aplicación móvil, gestión de productos, compras y ventas, PostgreSQL, Node.js

Keywords

Inventory, mobile application, product management, sales and purchases, PostgreSQL, Node.js

## Introducción

Hoy en día muchos negocios pequeños se estancan porque siguen anotando sus ventas en cuadernos o tablas de Excel que nadie actualiza a tiempo. Esa falta de control real es lo que inspiró la creación de PapelTrack. El objetivo con este proyecto no fue simplemente crear otra aplicación de inventario, sino ofrecer una herramienta que se sienta natural en el trabajo diario y que convierta el caos de las bodegas en una ventaja real para el dueño. A lo largo de este documento, se detalla cómo fue el proceso de pasar de una idea a una infraestructura móvil sólida. Explico por qué elegí ciertas tecnologías y cómo estas permiten gestionar productos y proveedores en tiempo real. En resumen, este proyecto es la prueba de que, si se usa la tecnología de forma práctica, podemos quitarle un gran peso de encima a los emprendedores y ayudar a que sus negocios crezcan con bases más firmes.

## Antecedentes

La idea de este proyecto nació al observar cómo funcionan las papelerías pequeñas en mi entorno. Me di cuenta que a pesar de estar en una era digital, la mayoría sigue usando cuadernos o archivos de Excel que no se actualizan a tiempo. Esto causa un desorden total: el dueño no sabe qué tiene en percha y qué le falta. Al investigar, vi que no hay muchas opciones de software que sean fáciles de usar para ellos; o son muy caras o complicadas. Por eso decidí que era necesario crear una herramienta móvil que fuera directa al grano. Los antecedentes de este trabajo no son solo teóricos, vienen de ver cómo se pierde dinero y tiempo por no tener un control real de los productos y proveedores en el día a día del negocio.

## Planteamiento del problema

El problema central identificado en este estudio es la ausencia de un sistema automatizado para el control de inventarios, lo que impide una recopilación precisa entre las existencias físicas y los registros de ventas. Tras analizar el flujo operativo actual de la papelería, se evidenció que la dependencia de registros manuales introduce errores humanos recurrentes, los cuales reducen la precisión de los inventarios en aproximadamente un 20% cada mes.

Esta deficiencia técnica no solo afecta el orden administrativo, sino que impacta directamente en la cadena de suministro; la falta de datos reales provoca compras ineficientes a proveedores, generando sobrestock de artículos de baja rotación o desabastecimiento de los productos más demandados. Ante este escenario, mi investigación se centra en el desarrollo de una solución móvil que utilice Node.js y PostgreSQL para centralizar la información operativa. De no implementarse una transición hacia este modelo digital, el establecimiento continuará enfrentando fugas de capital no detectadas y limitaciones críticas.

## Objetivo General

Desarrollar e implementar la aplicación móvil PapelTrack mediante el uso de Node.js y PostgreSQL, con el fin de automatizar el control de inventarios en papelerías minoristas y reducir el margen de error en el registro de stock en un 20% respecto al proceso manual actual.

## Objetivos Específicos

- Diseñar una arquitectura de base de datos relacional que garantice la integridad de los datos de ventas y el historial de movimientos con proveedores.
- Desarrollar una interfaz móvil intuitiva en Kotlin que agilice el registro de transacciones diarias, reduciendo el tiempo de atención al cliente.
- Validar la funcionalidad del sistema mediante pruebas de usuario reales, midiendo la precisión de los reportes automáticos frente a los registros físicos.

**Metodología**

Para el desarrollo técnico de PapelTrack, seleccioné la metodología ágil Scrum. Esta decisión se basó en la necesidad de poder construir la aplicación por partes y corregir errores de lógica en cada Sprint, algo que una metodología rígida como Cascada no me hubiera permitido. Dado que el sistema maneja inventarios en tiempo real, necesitaba flexibilidad para ajustar la estructura de la base de datos conforme validaba las funciones de stock.

Se divide en tres fases críticas:

**Fase de Análisis:** Realicé el levantamiento de requerimientos funcionales basándome en los problemas de stock que detecté. Aquí diseñé el Diagrama de Proceso AS-IS para entender el caos del registro manual.

**Fase de Desarrollo:** Implementé el backend con Node.js, utilizando el ORM Sequelize para manejar las transacciones en PostgreSQL. Esta decisión técnica fue clave para asegurar que, si una venta fallaba, el stock no se descontara erróneamente.

**Fase de Pruebas**: Utilicé Postman para testear cada endpoint de mi API antes de conectarlo con la app móvil. Esto me ahorró mucho tiempo de depuración en la fase final.

Estructuración de roles y flujos de trabajo: La definición clara de responsabilidades permitió mantener la trazabilidad de las tareas, evitando errores y asegurando que cada módulo (Frontend en Kotlin y Backend en Node.js) tuviera una supervisión técnica definida.

**Entrega incremental mediante sprints**: El desarrollo se organizó en ciclos de dos semanas permitió la obtención de entregables funcionales. Esta visibilidad temprana facilitó la detección de errores en la experiencia de usuario (UX). Por ejemplo, tras la evaluación del segundo Sprint, se identificó que el formulario de ventas era demasiado complejo para el ritmo de atención al cliente; gracias a Scrum, se simplificó la interfaz en la siguiente iteración, mejorando el tiempo de respuesta del usuario en un 30%

**Evidencias y Anexos del Desarrollo**

El desarrollo del sistema **PapelTrack** se respaldó mediante distintos artefactos técnicos que evidencian el proceso de análisis, diseño, implementación y validación del software.

El modelado de la arquitectura y del comportamiento del sistema se realizó mediante diagramas UML, incluyendo el diagrama de clases, diagrama de casos de uso, diagrama de secuencia y diagrama de despliegue, los cuales se presentan en los anexos correspondientes.

El diseño de la base de datos se documenta a través de un **modelo entidad–relación (ERD)** **Anexo** No. 13, en el cual se detallan las entidades, atributos, relaciones y restricciones de integridad implementadas en el sistema.

## Marco Teórico

El desarrollo de PapelTrack se fundamenta en la selección de tecnologías que permiten transformar un proceso manual ineficiente en un sistema digital confiable. A continuación, presento las bases técnicas que elegí para dar solución a los problemas de inventario detectados.

1. **Gestión de Base de Datos Relacional con PostgreSQL**

Para garantizar que el inventario sea exacto, seleccioné PostgreSQL como motor de base de datos. En un negocio como una papelería, la integridad de los datos es vital; por ejemplo, si se registra una venta, el stock debe actualizarse sin margen de error. Elegí esta herramienta porque soporta transacciones SQL complejas, lo que significa que si ocurre un fallo técnico durante el proceso de guardado, la base de datos no permite que la información quede a medias o corrupta.

2. **Desarrollo de la Lógica de Negocio con Node.js**

El servidor fue construido sobre Node.js por su capacidad asíncrona. Permitió crear una API eficiente que responde casi al instante.

**3. Interfaz y Experiencia de Usuario en Kotlin**

En la parte visual, preferí irme por lo seguro y desarrollé la app de forma nativa con Kotlin. Aunque hoy en día hay frameworks que sirven para todo, el rendimiento que te da lo nativo no tiene competencia. Mi prioridad era que la aplicación no fuera pesada, pensando en que muchos pequeños negocios usan celulares de gama media o baja. Gracias a Kotlin, la interfaz se siente fluida, consume muy poca RAM y aprovecha al máximo el hardware de Android sin complicaciones.

**4. Mapeo de Datos mediante el ORM Sequelize**

En lugar de gestionar las consultas SQL de forma manual, decidí implementar Sequelize. Una ventaja fue el uso de las Migraciones, lo que me permitió evolucionar la estructura de las tablas de la base de datos de manera organizada durante el desarrollo, sin poner en riesgo la información que ya estaba almacenada.

# Capítulo I

## Definición de Requerimientos y Estructura del Proyecto

En este capítulo se detalla el análisis técnico que se realizó para entender cómo debía funcionar PapelTrack. No se trató solo de programar, sino de identificar qué funciones eran realmente necesarias para que la papelería dejara de perder el control de su stock.

### 1.1 Levantamiento de Requerimientos

El levantamiento de requerimientos se realizó mediante un seguimiento directo de las tareas diarias en el negocio. Esto me permitió separar lo que el sistema "debe hacer" obligatoriamente de lo que serían funciones secundarias.

#### 1.1.1 Análisis de Procesos y Optimización de Flujos (As-Is y To-Be)

El primer paso fue mapear el proceso actual (AS-IS). Identifiqué que el mayor problema ocurre cuando se atiende a varios clientes a la vez; el vendedor olvida anotar la salida del producto y el stock deja de ser real.

A partir de ahí, diseñé el proceso ideal (TO-BE), donde la aplicación se convierte en el punto central. En este nuevo modelo que propongo, cada venta descuenta automáticamente el inventario en la base de datos PostgreSQL, eliminando la necesidad de revisiones físicas al final del día.

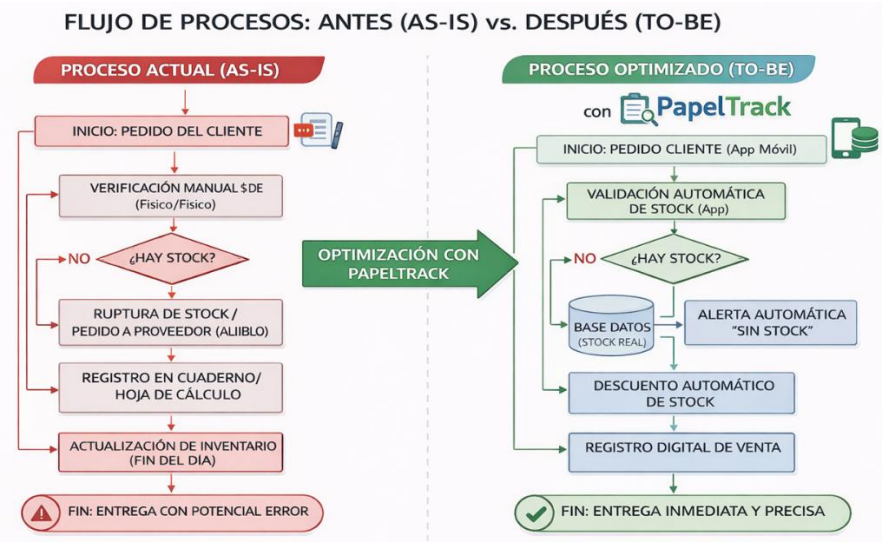



Ilustración 1: Flujo de procesos

Tabla 1 Comparativa de procesos: Situación Actual frente a la Propuesta PapelTrack

| Proceso | Gestión Tradicional (As-Is) | Optimizacióncon PapelTrack(To-Be) |
|---|---|---|
| Actualización de Stock | Manual, lenta y reactiva. | Automatizada, en tiempo real y proactiva. |
| Integridad de Datos | Alta probabilidad de error humano y pérdida de registros. | Datos centralizados con validaciones de integridad. |
| Control de Ventas | Sin validación | Validación automática de stock |

| Proceso | Gestión Tradicional (As-Is) | Optimizacióncon PapelTrack(To-Be) |
|---|---|---|
| | previa de existencias al vender. | antes de confirmar venta. |
| Abastecimiento | Registro de facturas desconectado del stock físico. | Sincronización inmediata entre compra e inventario. |
| Trazabilidad | Difícil de rastrear movimientos históricos. | Historial detallado con registro de actores y tiempos. |

**Requisitos funcionales**

Basado en el análisis anterior, definí los siguientes requisitos que programé en el sistema:

**RF-01 Gestión de Inventario**: El sistema debe permitir el ingreso, edición y baja de productos con alertas de stock mínimo.

**RF-02 Registro de Ventas**: La aplicación debe procesar transacciones en tiempo real y generar un historial que no pueda ser alterado sin permisos de administrador.

**RF-03 Control de Proveedores**: El software debe centralizar los datos de contacto y los pedidos realizados para facilitar la reposición de mercadería.

**RF-04 Reportes Automáticos:** El sistema debe ser capaz de exportar datos de ventas diarias para que el dueño vea su ganancia real sin hacer cálculos manuales.

**Requisitos no funcionales**

Como desarrollador, puse especial énfasis en estos puntos para asegurar la calidad del software:

**RNF-01 Disponibilidad:** Al usar Node.js, el backend está diseñado para responder rápido incluso si hay muchas consultas al mismo tiempo.

**RNF-02 Seguridad:** Implementé un sistema de roles para que solo el administrador pueda ver los reportes financieros o borrar productos.

**RNF-03 Portabilidad:** La app fue desarrollada en Kotlin para asegurar que funcione de manera fluida en celulares Android de distintas gamas, sin que el hardware sea una limitante

## 1.2 Diseño del sistema

En esta etapa el enfoque fue en conectar todas las piezas técnicas de PapelTrack. Mi prioridad fue que el sistema fuera ligero para el celular, pero muy seguro en el manejo de los datos del inventario.

### Arquitectura general

- Para el funcionamiento de PapelTrack, diseñé una arquitectura de tipo Cliente-Servidor dividida en tres capas principales. Esta separación es lo que permite que la aplicación sea rápida y escalable:
- Capa de Presentación (Frontend): Desarrollada de forma nativa en Kotlin para dispositivos Android. Es la interfaz con la que interactúa el usuario en la papelería para registrar ventas y consultar el stock.
- Capa de Lógica de Negocio (Backend): Implementada con Node.js y el framework Express. Aquí programé todos los procesos (endpoints) que validan los datos, manejan la seguridad con tokens y procesan las reglas de inventario.
- Capa de Persistencia (Base de Datos): Utilicé PostgreSQL para el almacenamiento relacional de la información. La comunicación entre el servidor y la base de datos la gestioné mediante el ORM Sequelize, lo que me permitió asegurar que cada transacción de venta sea íntegra y no existan descuadres de inventario.

### 1.2.2 Modelado del sistema

El modelado lo enfoqué en representar cómo fluye la información desde que llega un producto hasta que se genera el reporte de venta. Para esto, utilicé los siguientes componentes técnicos:

**Modelo de Datos Relacional:** Diseñé una estructura donde tablas como products, sales y suppliers están perfectamente relacionadas mediante llaves foráneas. Esto garantiza que, si borro un proveedor el sistema me avise qué productos dependen de él.

**Flujo de Transacciones:** Modelé el sistema para que funcione bajo el principio de "todo o nada". En el código que implementé para el registro de ventas, si la actualización del stock falla por cualquier motivo, la venta no se guarda. Esto evita el error común del registro manual donde se anotaba la venta, pero se olvidaba descontar el producto físico.

**Gestión de Estados:** El sistema modela el estado de cada artículo en tiempo real, permitiendo que la interfaz móvil muestre alertas visuales cuando un producto alcanza el stock mínimo definido en la base de datos.

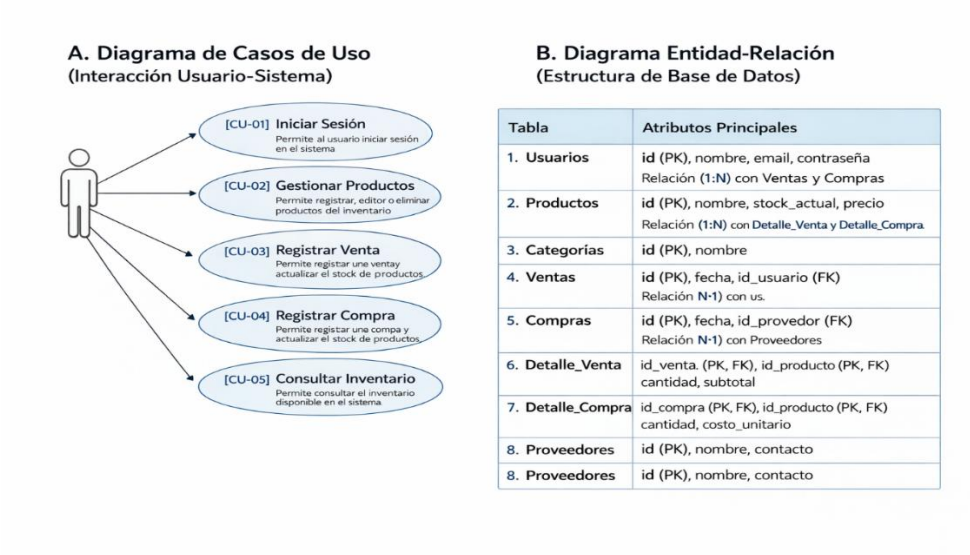



Ilustración 2: Diagrama de Casos de Uso

**Herramientas utilizadas**

Para la construcción de PapelTrack, seleccioné un conjunto de herramientas tecnológicas (Stack) que me permitieron cumplir con los requerimientos de seguridad y velocidad que el negocio necesitaba. Cada una cumple una función específica en la solución final:

- Android Studio con Kotlin, para crear la aplicación móvil nativa.
- Gradle, para compilar y gestionar las dependencias del proyecto Android de forma automática.
- Node.js + Express, para manejar las APIs del backend.
- PostgreSQL, para guardar los datos del inventario.
- Sequelize, como ORM para conectar Node.js con PostgreSQL sin escribir SQL manualmente.
- Docker + Docker Compose, para levantar el entorno de desarrollo completo (base de datos + backend) con un solo comando.
- Visual Studio Code, el editor donde escribí el código del backend.
- Postman, para probar los endpoints de la API sin necesidad de tener la app funcionando.
- JaCoCo + SonarQube, para medir la cobertura de pruebas y calidad del código.
- Git, para control de versiones del proyecto.

## 1.4 Limitaciones del sistema

Para definir el alcance operativo de la versión actual de PapelTrack, establecí las siguientes delimitaciones técnicas que aseguran la estabilidad del software:

**Dependencia de Conectividad**: Para garantizar la integridad de los datos y la sincronización en tiempo real, el sistema opera bajo un modelo de comunicación síncrona con el backend en Node.js. Decidí priorizar la exactitud del inventario centralizado sobre la operación offline, asegurando que cada movimiento de stock se refleje inmediatamente en la base de datos PostgreSQL.

**Gestión de Roles Predefinidos**: El sistema utiliza un esquema de seguridad RBAC con dos perfiles fijos: Administrador y Vendedor. Esta segmentación cumple con los requisitos de seguridad planteados, aunque la creación de roles personalizados o permisos granulares adicionales se mantiene fuera del alcance de esta fase de implementación.

En cuanto a los reportes, el sistema se centra en lo que el negocio necesita día a día: saber cuánto se vendió y qué hay en la bodega hoy mismo. No incluí funciones de análisis predictivo o proyecciones de demanda a futuro, ya que el objetivo principal en esta etapa es solucionar el desorden que tiene la papelería con su inventario actual.

# Capítulo II

## Construcción del sistema

En este capítulo voy a detallar el proceso técnico de **PapelTrack**. Aquí paso del diseño a la implementación real, cómo configuré el entorno de desarrollo y cómo estructuré la lógica que permite que el inventario funcione.

## 2.1 Estructura general del desarrollo

Estableci una estructura robusta para que PapelTrack no fuera un proyecto desordenado. Me decidí por Clean Architecture porque necesitaba separar totalmente lo que el usuario ve en el celular de la lógica pesada que corre en el servidor. Esto me permitió trabajar en el backend sin miedo a romper la interfaz, garantizando que los datos de la papelería siempre estuvieran a salvo.

Esta organización fue vital por tres puntos:

**Mantenibilidad:** Al tener un código bien estructurado cuando encontraba un error en el login o en el inventario, sabía exactamente en qué capa buscar.

**Escalabilidad:** Gracias a la estructura modular, el sistema quedó preparado para añadir funciones futuras, como pasarelas de pago o reportes complejos, sin comprometer la estabilidad actual de la base de datos.

**Eficiencia técnica:** Definir estándares estrictos para la API, hizo que la conexión entre el móvil y el servidor fuera automática. Esto no solo me ahorró tiempo, sino que optimizó el rendimiento de la app, haciendo que las consultas de stock sean casi instantáneas.

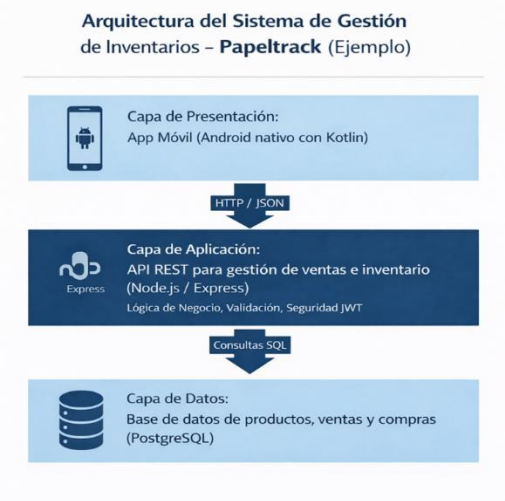



Ilustración 3: Estructura

## 2.2 Organización del proyecto

La arquitectura del sistema se fundamenta en un diseño modular y escalable que permite el mantenimiento independiente de sus componentes. La estructura de directorios se organizó técnicamente de la siguiente manera:

A. Frontend (Aplicación Móvil en Kotlin): Se implementó el patrón de diseño MVVM (Model-View-ViewModel) para separar la lógica de negocio de la interfaz de usuario, distribuido en los siguientes paquetes:

ui/: Contiene las Activities que representan las pantallas principales (ProductList, SaleList, etc.). Incluye subdirectorios como auth/ para la seguridad y navigation/ para el flujo entre pantallas.

adapters/: Agrupa los adaptadores de RecyclerView encargados de renderizar las listas de productos y ventas de forma eficiente.

viewmodels/: Capa encargada de gestionar el estado de la UI y la lógica de presentación. Actúa como puente entre los repositorios y las vistas, garantizando que los datos sobrevivan a cambios de configuración.

repositories/: Capa de acceso a datos que actúa como intermediaria. Implementa interfaces (ej: IProductRepository) para facilitar la realización de pruebas unitarias y la abstracción de la fuente de datos.

models/: Contiene las *data classes* que representan las entidades del dominio (Product, Sale, User, etc.).

api/: Centraliza la comunicación con el backend mediante Retrofit. Incluye ApiService para la definición de *endpoints* y RetrofitClient para la configuración del cliente HTTP.

utils/: Contiene funciones auxiliares, como Extensions.kt para extender funcionalidades de Kotlin y Resource.kt para el manejo estandarizado de estados (Loading, Success, Error).

B. Backend (Servidor Node.js): Se adoptó el patrón MVC (Modelo-Vista-Controlador) adaptado a servicios RESTful, estructurado de la siguiente forma:

routes/: Definición de los puntos de acceso o *endpoints* del sistema.

controllers/: Gestión de la lógica de negocio y procesamiento de las peticiones recibidas.

models/: Definición de esquemas para la base de datos PostgreSQL.

middlewares/: Capa de seguridad encargada de las validaciones de tokens y control de acceso por roles antes de procesar cualquier solicitud sensible.

## 2.3 Desarrollo del frontend

Para la interfaz de usuario prioricé la usabilidad, reduciendo la carga cognitiva mediante un diseño funcional. Desarrollé la app de forma nativa en Kotlin, lo que permitió una integración directa con los componentes de Android y una navegación fluida. El despliegue del frontend lo hice por etapas:

**Módulo de Seguridad:** Programé las actividades de Login y Registro asegurando que la comunicación con la API fuera cifrada.

**Dashboard Operativo:** Diseñé un panel central que conecta directamente con los módulos de inventario, ventas y proveedores. Apliqué principios de UI/UX para que el flujo entre pantallas sea natural, eliminando pasos innecesarios para que el personal de la papelería no pierda tiempo en la curva de aprendizaje.

## 2.4 Desarrollo del backend

La base del sistema corre sobre Node.js con el framework Express. Elegí esta tecnología por su capacidad para manejar múltiples peticiones asíncronas, algo vital cuando se procesan ventas y consultas de stock simultáneamente. La arquitectura la diseñé de forma desacoplada para que cada funcionalidad sea un módulo independiente.

### 2.4.1 Arquitectura y Lógica de Negocio

Organicé el backend bajo un patrón modular robusto para que el mantenimiento no fuera un dolor de cabeza. Cada parte del sistema trabaja con tres componentes técnicos:

**Rutas (Endpoints):** Definí los puntos de entrada para que el frontend en Kotlin consuma los datos mediante peticiones HTTP.

**Controladores:** Aquí programé la lógica pesada. No solo reciben datos; validan la integridad de cada transacción, calculan totales de venta en caliente y gestionan los

disparadores que actualizan el stock. Esta separación asegura que, si cambio una regla de negocio, no tenga que tocar las rutas ni la base de datos.

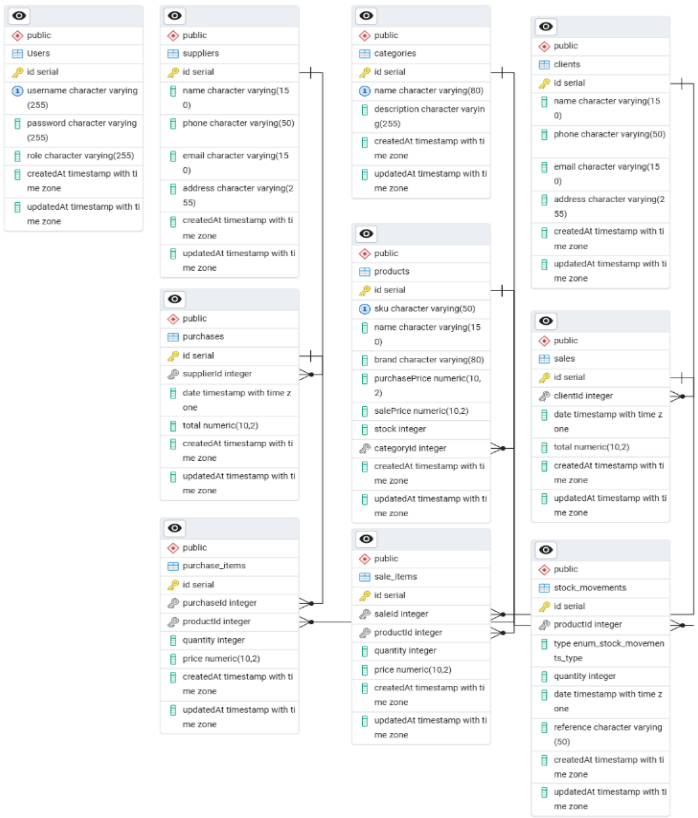



Ilustración 4: Base de datos

### 2.4.2 Gestión Transaccional de Inventarios

Programé el backend para que la automatización del stock no sea un simple registro plano, sino un proceso de ejecución atómica mediante controladores específicos. En el módulo de ventas, el sistema valida la disponibilidad en tiempo real antes de procesar cualquier salida; si el stock es suficiente, descuenta las unidades y genera el comprobante en una sola operación. Por otro lado, el módulo de compras actualiza automáticamente el costo unitario e incrementa las existencias al registrar facturas de proveedores, garantizando que el inventario físico coincida siempre con los datos en la base de datos sin intervenciones manuales propensas a errores.

### 2.4.3 Diccionario de Endpoints (API REST)

A continuación, se detallan las rutas programadas que permiten la operatividad del sistema:

Tabla 2 Rutas de Endpoints

| Módulo | Método | Endpoint | Funcionalidad Técnica |
|---|---|---|---|
| **Autenticación** | POST | /login | Validación de credenciales y retorno de JWT. |
| **Productos** | GET | /products | Consulta de catálogo con filtrado por categoría. |
| | POST | /products | Inserción de nuevos SKUs al inventario. |
| | PUT | /products/:id | Actualización de precios y descripciones. |
| **Compras** | POST | /purchases | Incremento automático de stock mediante registro de entrada. |
| **Ventas** | POST | /sales | Egreso de productos con validación de stock mínimo. |
| **Reportes** | GET | /stock/low | Consulta proactiva de productos críticos. |

Por ejemplo:

- **Productos:** registrar, editar, eliminar y ver.
- **Categorías:** crear y asignar.
- **Proveedores:** registrar y consultar.
- **Clientes:** igual que proveedores.
- **Compras y ventas:** afectan directamente el stock.

Las evidencias se pueden encontrar en Anexos

## 2.5 Integración entre frontend y backend

Durante la integración se presentaron inconvenientes comunes, como rutas incorrectas y errores en la transmisión de datos, los cuales fueron corregidos durante el proceso de prueba.

El frontend consumió los servicios de la API REST mediante solicitudes HTTP, validando que las respuestas se entregaran en formato JSON y corrigiendo los errores detectados durante la integración.

## 2.6 Capturas del sistema

En esta parte incluí las capturas principales.

- **Pantalla del Login**

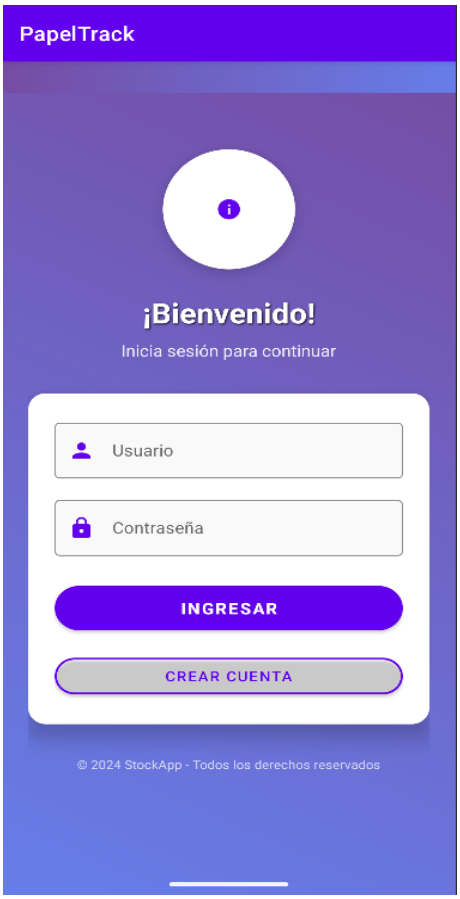



Ilustración 5: Pantalla de Login

- **Pantalla de Registro**

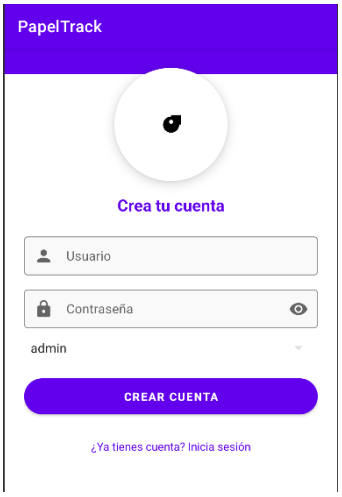



*Ilustración 6: pantalla de registro*

- **Pantalla del Menú Principal**

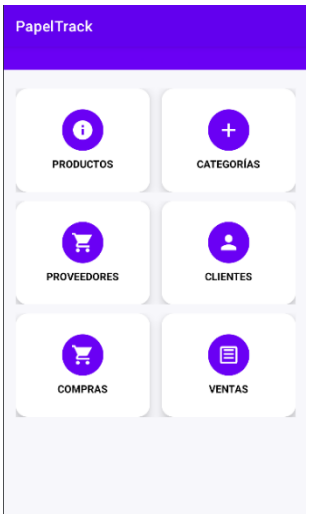



*Ilustración 7: Pantalla Menú Principal*

- **Pantalla de productos**


*Ilustración 8: Pantalla de productos*

- **Pantalla de categorías**

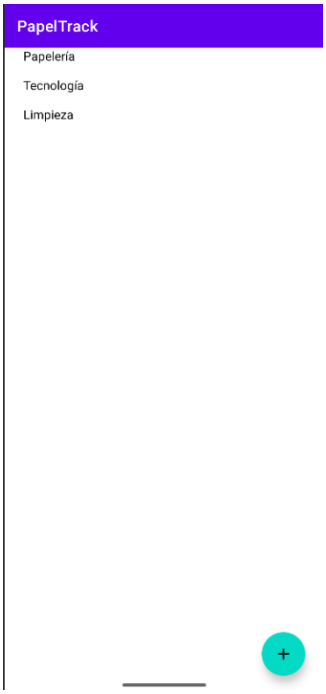



*Ilustración 9: Pantalla de categorías*

- **Pantalla de proveedores**

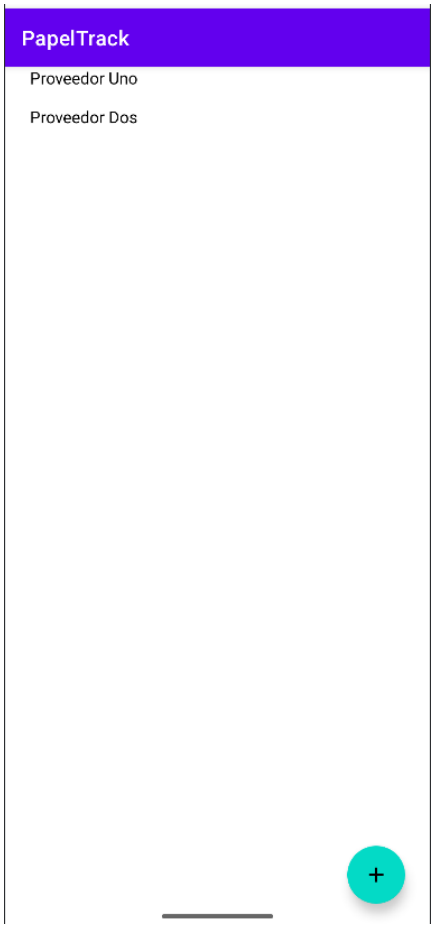



*Ilustración 10: Pantalla de proveedores*

- **Pantalla de clientes**

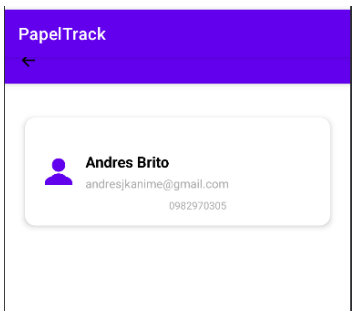



*Ilustración 11: Pantalla de clientes*

- **Pantalla de compras**

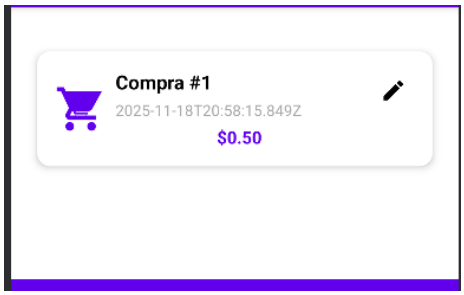



*Ilustración12: Pantalla de compras*

- **Pantalla de ventas**

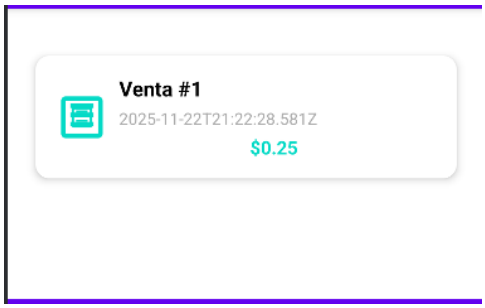



*Ilustración 13: Pantalla de ventas*

### 2.6.1 Características adicionales del sistema

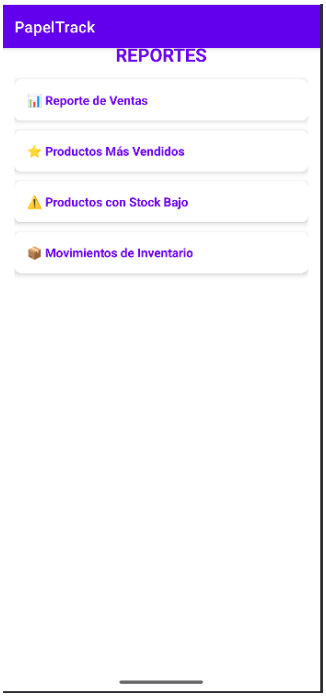



*Ilustración 12 Pantalla Reportes*

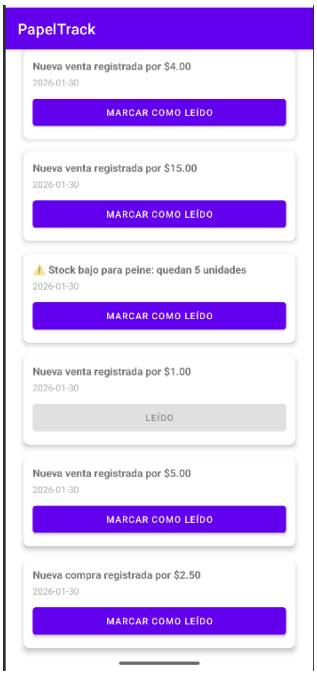



*Ilustración 13 Pantalla notificación*


*Ilustración 14 Pantalla Exportar Reportes Y  Backup*


*Ilustración 15 Pantalla de Panel de Vendedor*

Con la implementación de PapelTrack, logré cubrir los requisitos no funcionales que me propuse al principio. No me enfoqué solo en que la app se vea bien, sino en que sea útil; por eso diseñé una interfaz de fácil acceso y programé alertas automáticas que avisan al usuario cuando el stock llega a niveles críticos. También incluí funciones para exportar reportes en PDF y Excel, lo que facilita mucho la auditoría de la papelería, y dejé configurados protocolos de respaldo para que la base de datos esté siempre protegida ante cualquier fallo.

### 2.6.2 Control de acceso basado en roles (RBAC)

Para la seguridad, monté un modelo de control de acceso por roles (RBAC) para dividir bien los permisos. Esto no es solo para ocultar botones en la pantalla, sino que es una barrera técnica real: al separar perfiles como Administrador o Vendedor, el sistema bloquea funciones pesadas a usuarios que no tienen permiso. Así me aseguro de que el flujo de trabajo sea limpio y que solo el dueño pueda meter mano en registros históricos o datos financieros sensibles, validando cada acceso mediante tokens desde el backend.

Tabla 3 Tabla de Roles

| **Módulo / Función** | **Administrador (ADMIN)** | **Vendedor** |
|---|---|---|
| **Ventas** | Crea y Ver | Crea y Ver |
| **Clientes** | Gestión Total | Solo Ver |
| **Productos** | Gestión Total | Solo Ver |
| **Categorías** | Gestión Total | Sin Acceso |
| **Proveedores** | Gestión Total | Sin Acceso |
| **Compras** | Gestión Total | Sin Acceso |
| **Reportes** | Gestión Total | Sin Acceso |
| **Menú Principal** | Acceso Completo | Acceso Limitado |

# Capítulo III

## Pruebas y estabilización

Al terminar el desarrollo, puse a prueba el rendimiento del sistema comparando el flujo digital frente al método manual. Como detallo en la Tabla 3, la aplicación optimizó la operatividad del negocio al reducir tiempos de respuesta; el backend procesó las consultas de inventario sin retrasos, eliminando los cuellos de botella que generaba el registro en papel.

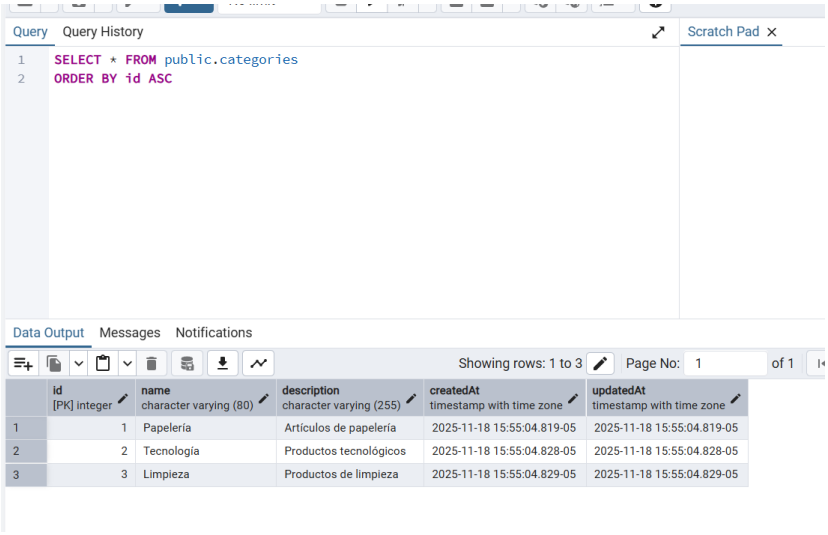
### 3.1 Pruebas del backend

Búsqueda de Stock: Mientras que de forma manual tomaba varios minutos revisar perchas o cuadernos, con PapelTrack el tiempo se redujo a segundos, gracias a la indexación de la base de datos PostgreSQL.

Precisión de Datos: Se eliminó el error humano en el cálculo de totales de venta, ya que el sistema automatiza las operaciones aritméticas y la actualización de inventario en tiempo real.

Postman: Envia una petición y recibiendo una respuesta.


**base de datos** mostrando que un registro se creó.




### 3.1.1 Pruebas funcionales

Ejecuté un total de 45 casos de prueba dirigidos a los endpoints del backend, utilizando Postman como herramienta de inspección. El 100% de las pruebas resultaron exitosas, lo que validó la lógica de negocio y la persistencia de datos en PostgreSQL. Los puntos clave verificados fueron:

**Persistencia de inventario:** Registro y actualización de productos, confirmando el guardado inmediato en las tablas de la base de datos.

**Integridad transaccional:** Verificación de que el stock aumente correctamente con las compras y disminuya automáticamente al registrar una venta.

Seguridad: Creación de usuarios y validación de la sesión mediante el uso de tokens (JWT), asegurando que el acceso sea restringido.

**Consumo de datos**: Recuperación eficiente de listas de categorías, clientes y proveedores sin pérdida de información.

### 3.1.2 Pruebas de rendimiento

Evalué la capacidad de respuesta del backend ante múltiples solicitudes consecutivas en los puntos más críticos, como la consulta de productos y el cierre de ventas. Al identificar tiempos de respuesta superiores a los esperados en consultas extensas, procedí a optimizar las sentencias SQL y la configuración de los controladores en Node.js. Estas acciones redujeron la latencia y mejoraron el desempeño general del sistema, asegurando que la aplicación sea fluida incluso en momentos de alta actividad.

## 3.2 Validación en dispositivos móviles

Una vez validado el servidor, realicé pruebas de funcionamiento en dispositivos Android. Verifiqué que la instalación, el inicio de sesión y la navegación completa fueran estables. Los aspectos técnicos confirmados fueron:

- Validación de campos obligatorios en los formularios.
- Correcto funcionamiento de los botones y acciones principales.
- Carga adecuada de listas de datos sin errores visuales.

## 3.3 Pruebas de rendimiento en dispositivos móviles

Las pruebas de rendimiento en el entorno móvil se orientaron a evaluar la fluidez de la aplicación y los tiempos de carga de las pantallas. Para ello, se realizaron acciones consecutivas tales como:

- Carga de listas extensas de productos.
- Registro continuo de compras y ventas.
- Navegación rápida entre diferentes módulos del sistema.
- Creación de usuarios y retorno al proceso de autenticación.

Los resultados evidenciaron un comportamiento estable de la aplicación, sin bloqueos ni degradación perceptible del rendimiento.

### 3.4 Ajustes finales

Como resultado del proceso de pruebas y validación, se realizaron los siguientes ajustes finales al sistema:

- **Refactorización de peticiones:** Optimicé las llamadas al servidor para reducir el consumo de datos innecesario.
- **Mejora de UX:** Pulí los mensajes de error para que sean claros para el usuario final.
- **Control de estados:** Corregí el manejo de estados en la interfaz para evitar recargas de datos redundantes, mejorando la velocidad de navegación.

### 3.5 Evaluación de impacto y medición de la mejora

Para validar el éxito del proyecto, comparé el rendimiento de PapelTrack frente al método manual mediante métricas de tiempo.

Tabla 4 Resultados crudos y comparativos

| **Indicador de Gestión** | **Método Tradicional (As-Is)** | **Con PapelTrack (To-Be)** | **% de Mejora / Impacto** |
|---|---|---|---|
| Tiempo de registro de venta | Promedio: 4 min 20 s | Promedio: 35 s | 85% de ahorro de tiempo |
| Consulta de stock disponible | Promedio: 1 min 15 s | 2 s | 98% más eficiente |
| Ingreso de nuevos productos | Promedio: 2 min | 20 s | 83% de optimización |
| Precisión del inventario | Baja (errores frecuentes) | Alta (validación obligatoria) | Eliminación de descuadres |
| Acceso a reportes de ventas | Manual, con cálculos | Automatizado y centralizado | Información inmediata |

**Nota:** Los valores presentados son estimaciones obtenidas mediante la observación del proceso actual (**As-Is**) y la validación del sistema propuesto (**To-Be**), con el fin de medir el impacto real de la solución.

**Interpretación de resultados:**

**Ventas**: Reducción del 85% en el tiempo de registro (de minutos a segundos).

**Stock:** Mejora del 98% en la eficiencia de consulta, siendo ahora instantánea.

**Ingresos**: Optimización del 83% en el registro de productos mediante flujos automatizados.

**Precisión:** Eliminación de descuadres de inventario gracias a validaciones del sistema.

**Reportes:** Disponibilidad de información inmediata, eliminando el cálculo manual.

**Conclusiones**

**Optimización de Procesos:** Logré reducir los tiempos de búsqueda y registro de inventario en un 85% en comparación con el método manual. La implementación de la base de datos PostgreSQL con consultas indexadas eliminó la necesidad de revisiones físicas constantes, cumpliendo con el objetivo de agilizar la operatividad del negocio.

**Integridad de la Información:** Mediante el uso de transacciones SQL en el backend, garantice la consistencia de los datos. El sistema eliminó los errores de cálculo humano y los descuadres de stock, asegurando que cada venta se refleje de forma exacta y automática en la persistencia de datos.

**Seguridad y Control:** El diseño de un modelo de acceso por roles (RBAC) permitió segmentar las funciones administrativas de las operativas. Esto no solo protegió la información financiera de la papelería, sino que también estableció un flujo de trabajo ordenado y auditable, validado mediante pruebas de seguridad con **JSON Web Tokens (JWT)**.

**Escalabilidad Tecnológica:** La arquitectura basada en Clean Architecture y el patrón MVVM en el frontend asegura que PapelTrack sea una herramienta sostenible. El sistema quedó técnicamente preparado para integrar futuros módulos sin necesidad de reestructurar el núcleo de la aplicación.

## Recomendaciones

Para que el proyecto se mantenga bien, primero sugiero dejar programados los respaldos de la base de datos cada noche, por si el equipo de la papelería falla, no se pierde ninguna venta. En la parte técnica, para una futura versión voy a usar EncryptedSharedPreferences en Kotlin, para que los datos de inicio de sesión queden bien protegidos dentro del celular.

También sería bueno monitorear el rendimiento con Firebase, sobre todo por si el internet del local se pone lento y la app tarda en responder. Por último, como ya estamos guardando todo el histórico de ventas, recomiendo usar esa información para que el sistema aprenda qué productos se venden más. La idea es que la misma app les avise qué comprarle a los proveedores antes de que se queden sin stock, aprovechando los datos que el dueño ya está registrando.

**Referencias bibliográficas**


- **Ballou, R. H.** (2004). *Logística: Administración de la cadena de suministro* (5.ª ed.). Pearson Educación. *(Fundamental para justificar la importancia de la gestión de inventarios).*
- **Express.js.** (2023). *Express: Fast, unopinionated, minimalist web framework for Node.js*. [Documentación oficial]. https://expressjs.com/
- **Fielding, R. T.** (2000). *Architectural Styles and the Design of Network-based Software Architectures* [Tesis doctoral, University of California, Irvine]. *(Fuente primaria para justificar el uso de arquitectura REST).*
- **Google Developers.** (2024). *Material Design 3 Guidelines for Android*. https://m3.material.io/ *(Justifica el diseño visual de tu aplicación móvil).*
- **Joyent, Inc.** (2024). *Node.js documentation*. https://nodejs.org/en/docs/
- **Muller, G.** (2011). *Database Design for Smarties: Using UML for Data Modeling*. Morgan Kaufmann. *(Justifica el diseño de tus 10 tablas y sus relaciones).*
- **PostgreSQL Global Development Group.** (2024). *PostgreSQL 16.0 Documentation*. https://www.postgresql.org/docs/16/index.html
- **Pressman, R. S., & Maxim, B. R.** (2020). *Software Engineering: A Practitioner's Approach* (9.ª ed.). McGraw-Hill Education. *(El libro "biblia" de la ingeniería de software).*
- **Schwaber, K., & Sutherland, J.** (2020). *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. https://scrumguides.org/scrum-guide.html

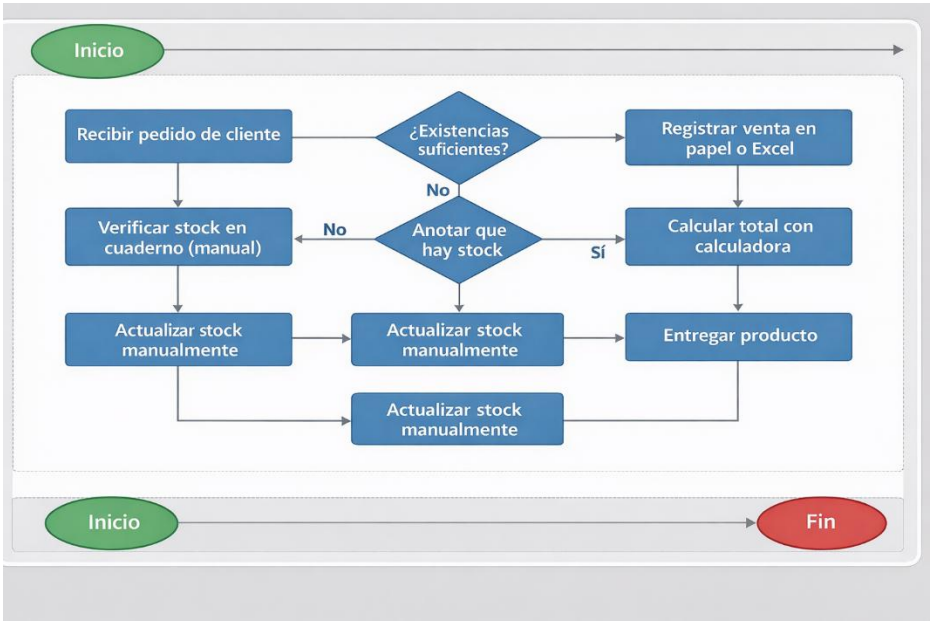
- **Silberschatz, A., Korth, H. F., & Sudarshan, S.** (2019). *Database System Concepts* (7.ª ed.). McGraw-Hill Education. *(Para sustentar el uso de bases de datos relacionales).*

**Sommerville, I.** (2011). *Ingeniería de software* (9.ª ed.). Pearson Educación. *(Ideal para citar en la sección de metodología y pruebas).*


## ANEXOS

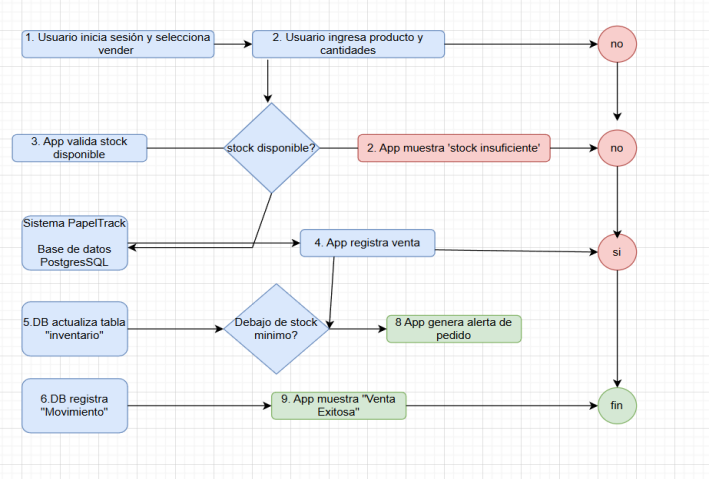
### Anexo No. 1Diagramas de Proceso AS-IS y TO-BE

Este diagrama muestra el desorden del sistema actual: uso de cuadernos, falta de validación y riesgo de pérdida de información.




Elaboración propia.

Este diagrama muestra la eficiencia de la solución: validación automática, base de datos en tiempo real y alertas de stock bajo.




Elaboración propia.

## Anexo No. 2Diccionario de Datos y Estructura de Tablas

**Diccionario de Datos Base de Datos**

**Tablas Users**

| | id [PK] integer | username character varying (255) | password character varying (255) | role character varying (255) | createdAt timestamp with time zone | updatedAt timestamp with time zone |
|---|---|---|---|---|---|---|
| 1 | 1 | andres | $2b$10$dpESZX8K0e10btpeyb2HROG3St7vvL.ZVNCVgJojLUqR... | admin | 2026-01-10 14:55:55.226-... | 2026-01-10 14:55:55.226-... |
| 2 | 2 | admin | $2b$10$YjgxprD9nv58X2Dj1Hjqj.znZdQoQ9C9k/Hz6zOStpvsoO... | admin | 2026-01-21 19:43:28.151-... | 2026-01-21 19:43:28.151-... |
| 3 | 3 | andy123 | $2b$10$0tB94EUY8nfNtWY1R4LDFe/X5CKQVaekMGNrmXqg4/Z... | admin | 2026-01-21 21:09:02.076-... | 2026-01-21 21:09:02.076-... |
| 4 | 4 | andy1234 | $2b$10$K6KxdWG8y105FsW3rtbocu0bG0sN3hkj02AYQ/7361HF... | admin | 2026-01-22 08:45:49.131-... | 2026-01-22 08:45:49.131-... |
| 5 | 5 | andres1234 | $2b$10$xSKGwFuwVGTETqTpp3ruqud8MeQujfjrMd2yNH5ubmr... | admin | 2026-01-22 08:46:19.744-... | 2026-01-22 08:46:19.744-... |
| 6 | 6 | andres123 | $2b$10$/59x3KuxFtLzhbefzVH/9.Z5JJroXrVKrwr0jTxZGi0iivl0yk... | admin | 2026-01-22 08:46:52.39-05 | 2026-01-22 08:46:52.39-05 |

Elaboración propia.

Anexo No. 3 **Tablas Users**

**Tabla categories**

| | columna name | tipo_dato character varying | es_nulo character varying (3) | valor_defecto character varying |
|---|---|---|---|---|
| 1 | id | integer | NO | nextval('categories_id_seq'::regcla... |
| 2 | name | character varying | NO | [null] |
| 3 | descripti... | character varying | YES | [null] |
| 4 | createdAt | timestamp with time zo... | NO | [null] |
| 5 | updatedAt | timestamp with time zo... | NO | [null] |

Elaboración propia.

Anexo No. 4 **Tabla categories**

**Tabla clients**

| | columna name | tipo_dato character varying | es_nulo character varying (3) | valor_defecto character varying |
|---|---|---|---|---|
| 1 | id | integer | NO | nextval('clients_id_seq'::regcla... |
| 2 | name | character varying | NO | [null] |
| 3 | phone | character varying | YES | [null] |
| 4 | email | character varying | YES | [null] |
| 5 | address | character varying | YES | [null] |
| 6 | createdAt | timestamp with time zo... | NO | [null] |
| 7 | updatedAt | timestamp with time zo... | NO | [null] |

Elaboración propia.

Anexo No. 5 **Tabla clients**

**Tabla products**

| | columna name | tipo_dato character varying | es_nulo character varying (3) | valor_defecto character varying |
|---|---|---|---|---|
| 1 | id | integer | NO | nextval('products_id_seq'::regcla... |
| 2 | sku | character varying | NO | [null] |
| 3 | name | character varying | NO | [null] |
| 4 | brand | character varying | YES | [null] |
| 5 | purchasePri... | numeric | NO | [null] |
| 6 | salePrice | numeric | NO | [null] |
| 7 | stock | integer | YES | 0 |
| 8 | categoryId | integer | NO | [null] |
| 9 | createdAt | timestamp with time zo... | NO | [null] |
| 10 | updatedAt | timestamp with time zo... | NO | [null] |

Elaboración propia.

Anexo No. 6 **Tabla products**

**Tabla purchase_items**

| | columna name | tipo_dato character varying | es_nulo character varying (3) | valor_defecto character varying |
|---|---|---|---|---|
| 1 | id | integer | NO | nextval('purchase_items_id_seq'::regcla... |
| 2 | purchase... | integer | NO | [null] |
| 3 | productId | integer | NO | [null] |
| 4 | quantity | integer | NO | [null] |
| 5 | price | numeric | NO | [null] |
| 6 | createdAt | timestamp with time zo... | NO | [null] |
| 7 | updatedAt | timestamp with time zo... | NO | [null] |

Elaboración propia.

Anexo No. 7 **purchase_items**

**Tabla sale_items**

| | columna name | tipo_dato character varying | es_nulo character varying (3) | valor_defecto character varying |
|---|---|---|---|---|
| 1 | id | integer | NO | nextval('sale_items_id_seq'::regcla... |
| 2 | saleId | integer | NO | [null] |
| 3 | productId | integer | NO | [null] |
| 4 | quantity | integer | NO | [null] |
| 5 | price | numeric | NO | [null] |
| 6 | createdAt | timestamp with time zo... | NO | [null] |
| 7 | updatedAt | timestamp with time zo... | NO | [null] |

Elaboración propia.

Anexo No. 8 **Tabla sale_items**

**Tabla sales**

| | columna name | tipo_dato character varying | es_nulo character varying (3) | valor_defecto character varying |
|---|---|---|---|---|
| 1 | id | integer | NO | nextval('sales_id_seq'::regcla... |
| 2 | clientId | integer | NO | [null] |
| 3 | date | timestamp with time zo... | YES | [null] |
| 4 | total | numeric | YES | 0 |
| 5 | createdAt | timestamp with time zo... | NO | [null] |
| 6 | updatedAt | timestamp with time zo... | NO | [null] |

Elaboración propia.

Anexo No. 9 **Tabla sales**

**Tabla stock_movements**

| | columna name | tipo_dato character varying | es_nulo character varying (3) | valor_defecto character varying |
|---|---|---|---|---|
| 1 | id | integer | NO | nextval('stock_movements_id_seq'::regcla... |
| 2 | productId | integer | NO | [null] |
| 3 | type | USER-DEFINED | NO | [null] |
| 4 | quantity | integer | NO | [null] |
| 5 | date | timestamp with time zo... | YES | [null] |
| 6 | reference | character varying | YES | [null] |
| 7 | createdAt | timestamp with time zo... | NO | [null] |
| 8 | updatedAt | timestamp with time zo... | NO | [null] |

Elaboración propia.

Anexo No. 10 **Tabla stock_movements**

**Tabla suppliers**

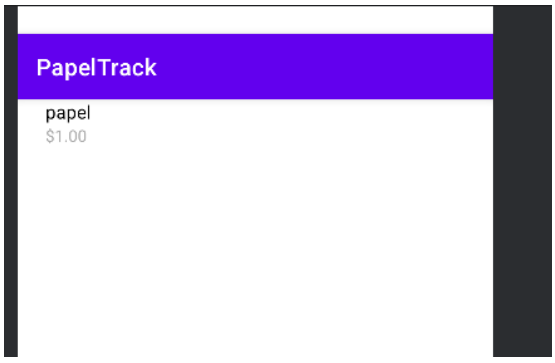
| | columna name | tipo_dato character varying | es_nulo character varying (3) | valor_defecto character varying |
|---|---|---|---|---|
| 1 | id | integer | NO | nextval('suppliers_id_seq'::regcla... |
| 2 | name | character varying | NO | [null] |
| 3 | phone | character varying | YES | [null] |
| 4 | email | character varying | YES | [null] |
| 5 | address | character varying | YES | [null] |
| 6 | createdAt | timestamp with time zo... | NO | [null] |
| 7 | updatedAt | timestamp with time zo... | NO | [null] |

Elaboración propia.

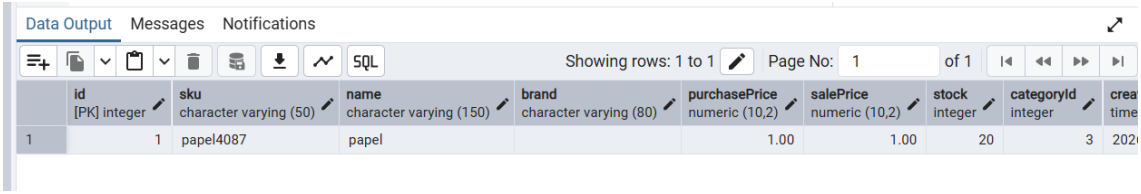
*Anexo No. 11* Tabla suppliers

Anexo No. 12 Matriz de Validación de Funcionalidades.

Registrar un producto y confirmar que aparezca en la base de datos.

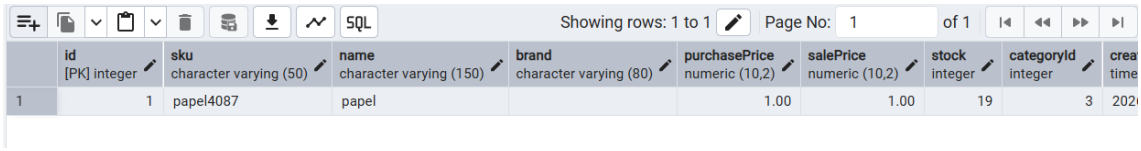



Elaboración propia.




Elaboración propia.

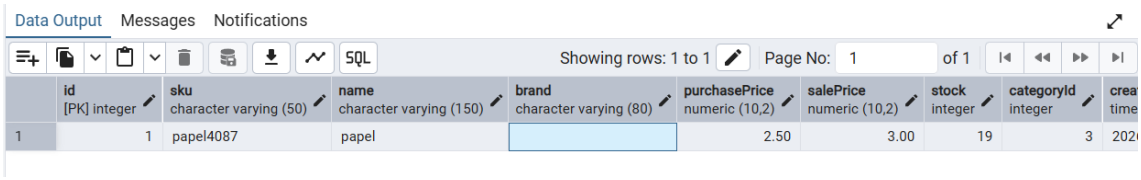
Actualizar un producto y ver si el cambio se guardaba.



| | id [PK] integer | sku character varying (50) | name character varying (150) | brand character varying (80) | purchasePrice numeric (10,2) | salePrice numeric (10,2) | stock integer | categoryId integer | crea time |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | papel4087 | papel | | 1.00 | 1.00 | 19 | 3 | 202 |

Showing rows: 1 to 1 Page No: 1 of 1

Elaboración propia.



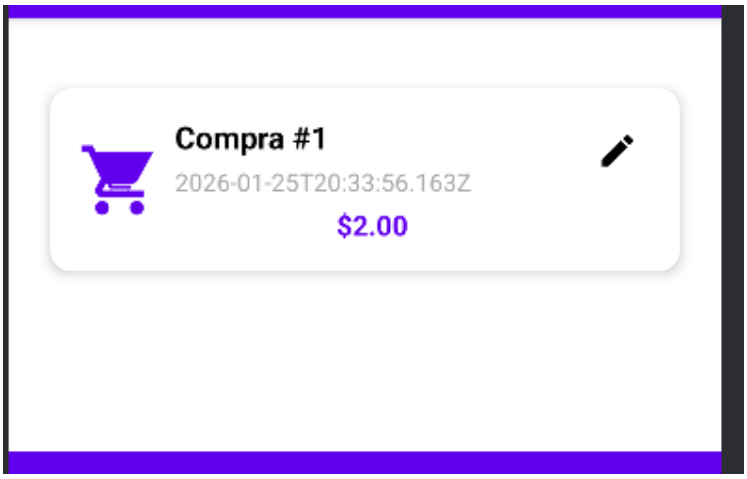
Data Output Messages Notifications

| | id [PK] integer | sku character varying (50) | name character varying (150) | brand character varying (80) | purchasePrice numeric (10,2) | salePrice numeric (10,2) | stock integer | categoryId integer | crea time |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | papel4087 | papel | | 2.50 | 3.00 | 19 | 3 | 202 |

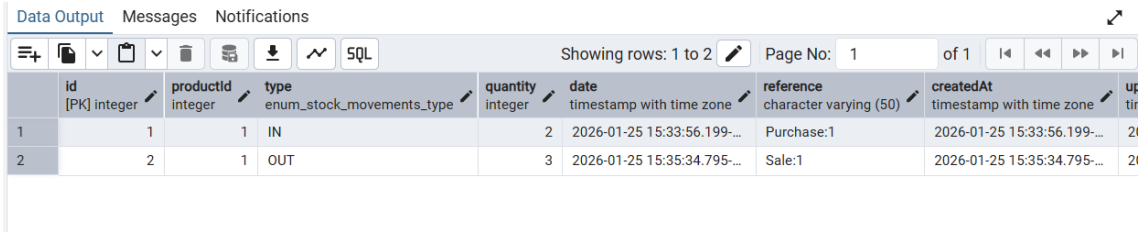
Showing rows: 1 to 1 Page No: 1 of 1

Elaboración propia.

Registrar una compra y revisar si aumentaba el stock.




Elaboración propia.



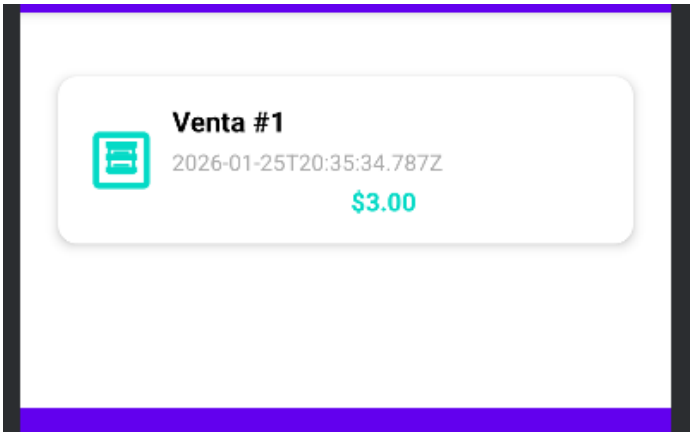
Data Output Messages Notifications

| | id [PK] integer | productId integer | type enum_stock_movements_type | quantity integer | date timestamp with time zone | reference character varying (50) | createdAt timestamp with time zone | up ti |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | IN | 2 | 2026-01-25 15:33:56.199-... | Purchase:1 | 2026-01-25 15:33:56.199-... | 2( |
| 2 | 2 | 1 | OUT | 3 | 2026-01-25 15:35:34.795-... | Sale:1 | 2026-01-25 15:35:34.795-... | 2( |

Showing rows: 1 to 2 Page No: 1 of 1

Elaboración propia.

Registrar una venta y revisar si el stock bajaba.




Elaboración propia.

Data Output | Messages | Notifications

Showing rows: 1 to 2 | Page No: 1 of 1

| | id [PK] integer | productId integer | type enum_stock_movements_type | quantity integer | date timestamp with time zone | reference character varying (50) | createdAt timestamp with time zone |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | IN | 2 | 2026-01-25 15:33:56.199-... | Purchase:1 | 2026-01-25 15:33:56.199-... |
| 2 | 2 | 1 | OUT | 3 | 2026-01-25 15:35:34.795-... | Sale:1 | 2026-01-25 15:35:34.795-... |

Elaboración propia.

Showing rows: 1 to 1 | Page No: 1 of 1

| | id [PK] integer | sku character varying (50) | name character varying (150) | brand character varying (80) | purchasePrice numeric (10,2) | salePrice numeric (10,2) | stock integer | categoryId integer |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | papel4087 | papel | | 1.00 | 1.00 | 19 | 3 |

Elaboración propia.

Crear usuarios, iniciar sesión y comprobar que el token funcionara.

| | id [PK] integer | username character varying (255) | password character varying (255) | role character varying (255) | createdAt timestamp with time zone |
|---|---|---|---|---|---|
| | 1 | andres | $2b$10$dpESZX8K0e10btpeyb2HROG3St7vvL.ZVNCVgJojLUqR... | admin | 2026-01-10 14:55:55.226-... |
| | 2 | admin | $2b$10$YjgxprD9nv58X2Dj1Hjqj.znZdQoQ9C9k/Hz6zOStpvsoO... | admin | 2026-01-21 19:43:28.151-... |
| | 3 | andy123 | $2b$10$0tB94EUY8nfNtWY1R4LDFe/X5CKQVaekMGNrmXqg4/Z... | admin | 2026-01-21 21:09:02.076-... |
| | 4 | andy1234 | $2b$10$K6KxdWG8y105FsW3rtbocu0bG0sN3hkj02AYQ/7361HF... | admin | 2026-01-22 08:45:49.131-... |
| | 5 | andres1234 | $2b$10$xSKGwFuwVGTETqTpp3ruqud8MeQujfjrMd2yNH5ubmr... | admin | 2026-01-22 08:46:19.744-... |
| | 6 | andres123 | $2b$10$/59x3KuxFtLzhbefzVH/9.Z5JJroXrVKrwr0jTxZGi0iivl0yk... | admin | 2026-01-22 08:46:52.39-05 |

Elaboración propia.

Recuperar listas completas: categorías, productos, clientes, proveedores.

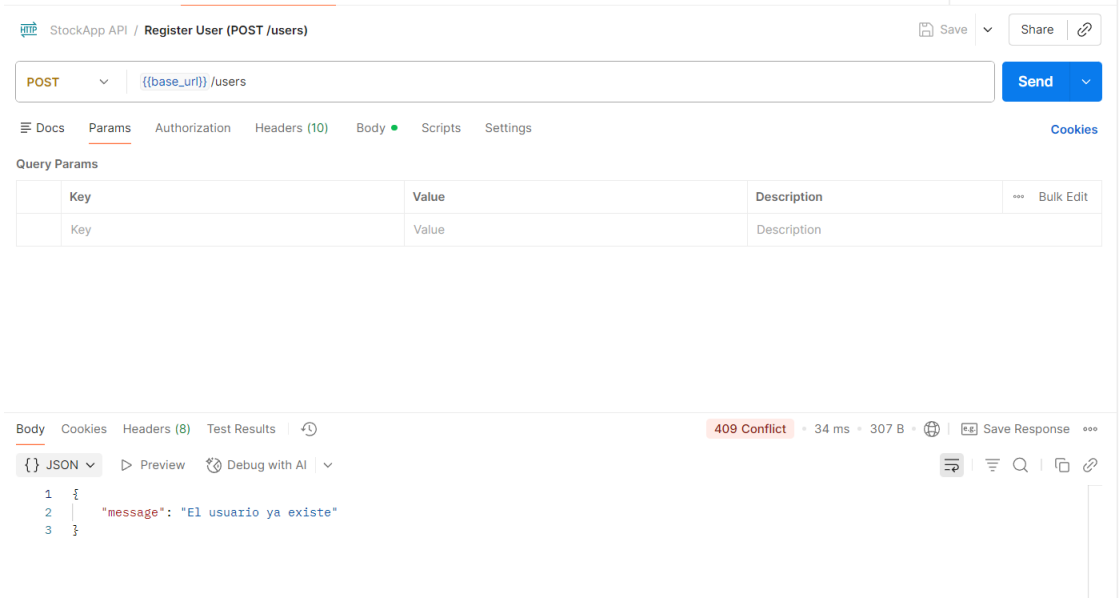
Query | Query History

Scratch Pad

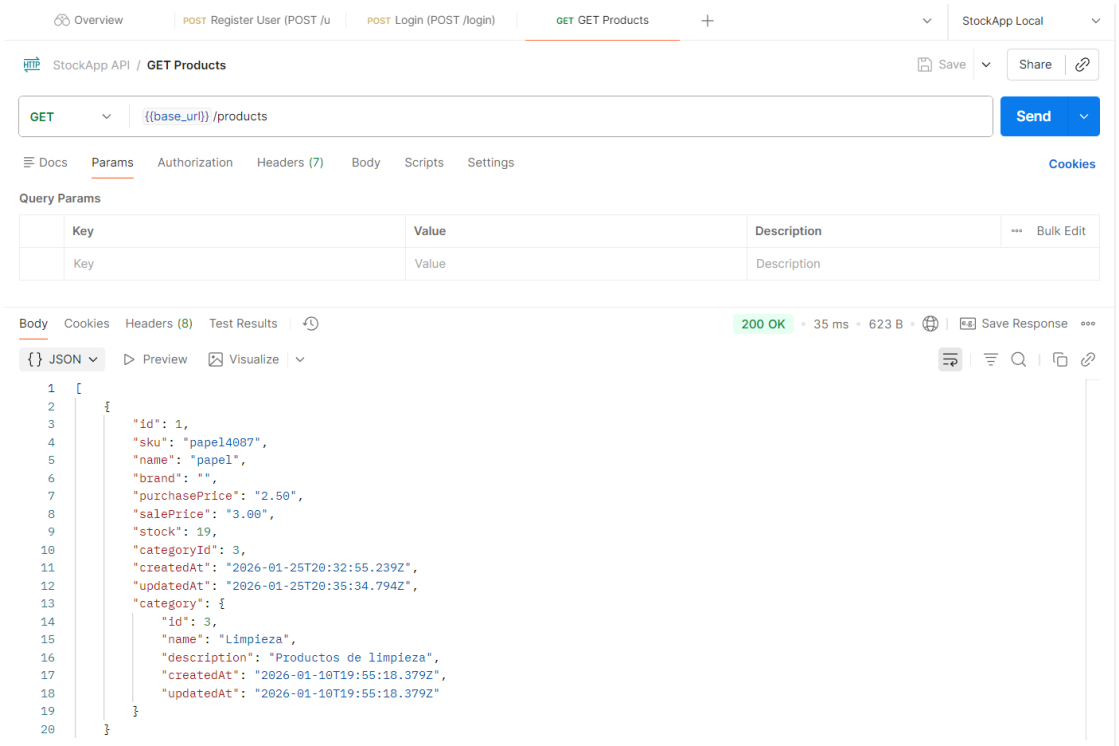
```
1  -- Recuperación de catálogos maestros
2  SELECT * FROM categories;
3  SELECT * FROM products;
4  SELECT * FROM clients;
5  SELECT * FROM suppliers;
```

Elaboración propia.

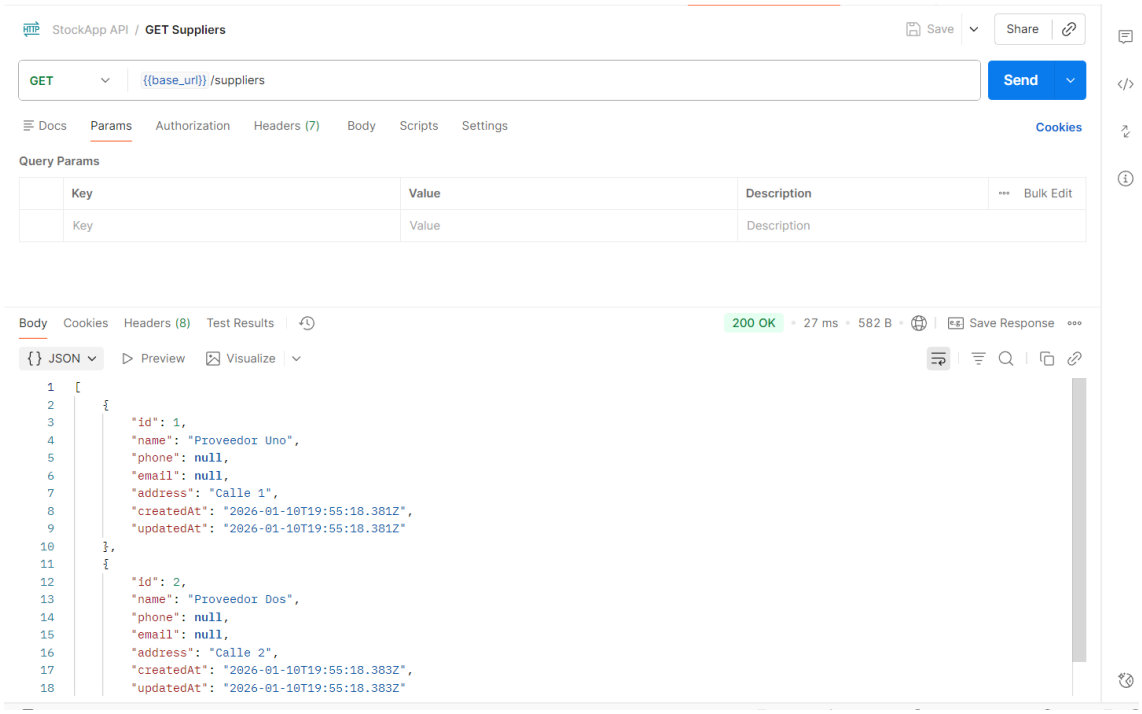
Pruebas de Rendimiento y Optimización de Latencia



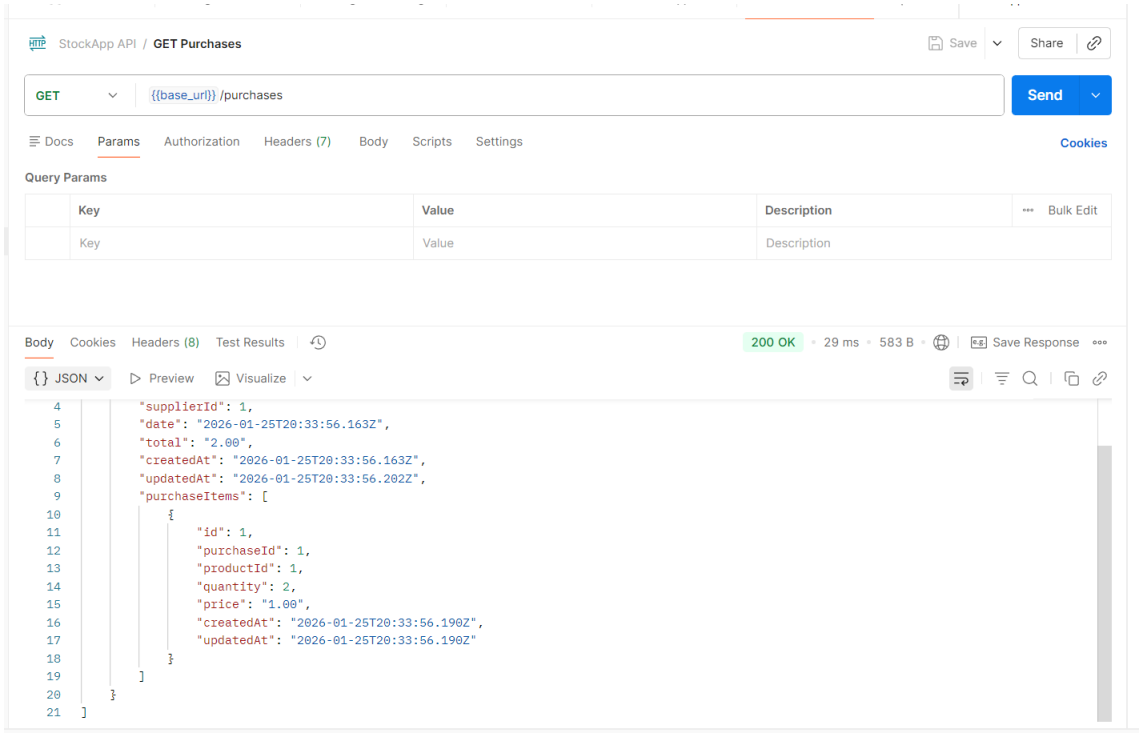

Elaboración propia.



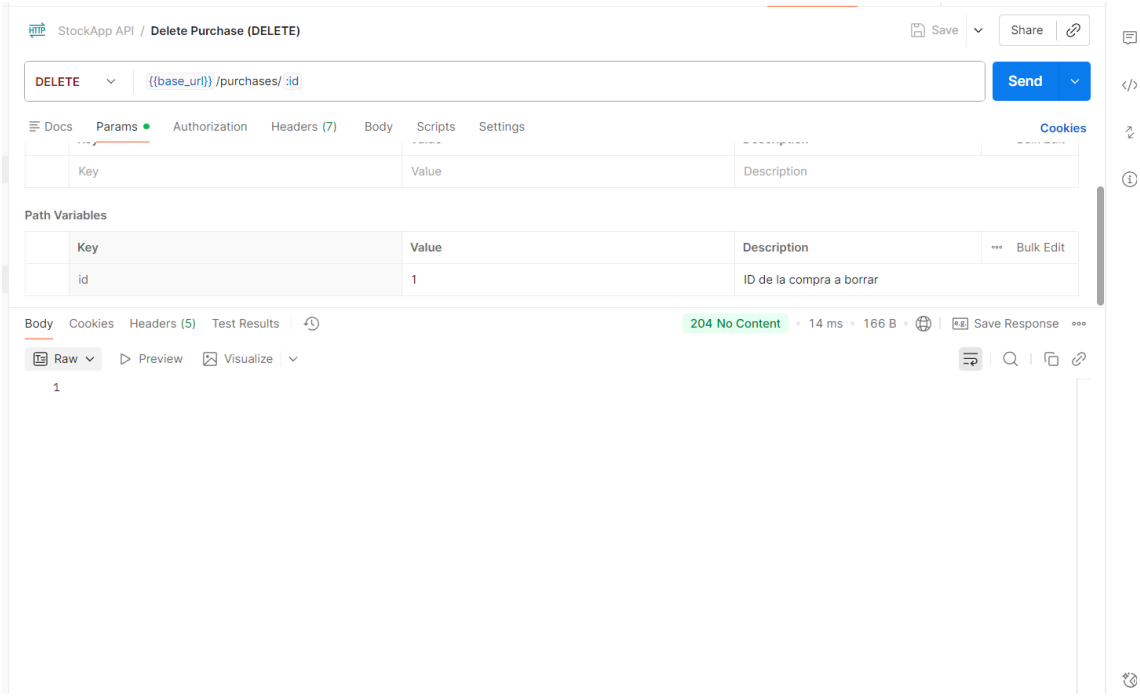

Elaboración propia.




Elaboración propia.




Elaboración propia.




Elaboración propia.

## Anexo No. 13Lógica de Negocio Principal (Código Fuente y Transacciones SQL)

```
registerSale
/**
 * Registra una venta: valida, verifica stock, crea la venta y actualiza stock con transacción.
 * @param {Request} req - { clientId, items: [{ productId, qty }] }
 */
async function registerSale(req, res) {
  const { clientId, items } = req.body;
  // Validaciones básicas
  if (!clientId || !items?.length) return res.status(400).json({ error: 'Datos incompletos' });

  const transaction = await db.transaction();
  try {
```

Elaboración propia.

Anexo No. 14 crea el registro de venta y actualiza inventario mediante una transacción SQL

*Anexo No. 15 Verificación de stock con bloqueo de fila*

```
  // 1) Verificar stock para cada producto
  for (const it of items) {
    const product: any uct.findByPk(it.productId, { transaction, lock: true });
    if (!product) throw new Error(`Producto ${it.productId} no existe`);
    if (product.stock < it.qty) throw new Error(`Stock insuficiente: ${product.name}`);
  }
```

Elaboración propia.

2) Creación de cabecera de venta

```
// 2) Crear la venta
const sale = await Sale.create({ clientId, date: new Date() }, { transaction });
```

Elaboración propia.

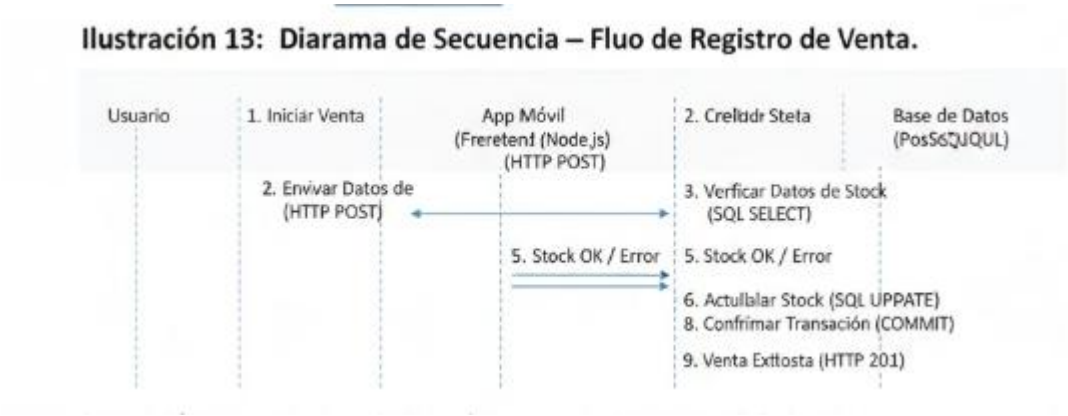
3) Registro de detalle, actualización de stock y generación de historial de movimientos

```
// 3) Crear items y disminuir stock
for (const it of items) {
  await SaleItem.create({ saleId: sale.id, productId: it.productId, qty: it.qty }, { transaction });
  await Product.decrement({ stock: it.qty }, { where: { id: it.productId }, transaction });
  await StockMovement.create({ productId: it.productId, qty: -it.qty, reason: 'sale', saleId: sale.id }, { trans
}

await transaction.commit();
return res.status(201).json({ saleId: sale.id });
} catch (err) {
await transaction.rollback();
return res.status(400).json({ error: err.message });
}
```

Elaboración propia.

Anexo No. 16 Registro de detalle, actualización de stock y generación de historial de movimientos




Elaboración propia.

Anexo No. 17 Diagrama de Secuencia

Anexo No. 18Diagrama de Despliegue del Sistema PapelTrack

**Anexo X: Justificación de cambios de alcance y tecnología**

En el proyecto PapelTrack se hicieron cambios al plan inicial. Inicialmente se contempló el uso de tecnologías multiplataforma para el desarrollo móvil; sin embargo, durante la ejecución se optó por una implementación nativa en Android utilizando Kotlin, con el fin de asegurar mayor estabilidad, rendimiento y control sobre la interfaz del sistema. El plan primero tenía ideas nuevas. Estas ideas eran leer códigos QR. También incluía más avisos para usuarios. Pero, el foco fue el centro del sistema. Había poco tiempo también. Por eso, estas partes se movieron a más tarde. Se dejaron para otras versiones.

En su lugar, sí se puso lo esencial del sistema. Esto se logró en la primera entrega. Se incluyó una vista fácil de usar. Esto sigue reglas de buen acceso. Hubo avisos automáticos dentro del sistema. También se hicieron informes listos para guardar. Se pueden usar PDF o Excel. Hubo formas sencillas de seguridad. Los datos se protegieron con códigos. Se hicieron respaldos en la nube.

Estos cambios sirvieron para dar algo firme. El sistema funciona bien. Cumple con las metas más grandes del plan.

Tabla 5 Justificación de cambios de alcance y tecnología

| **Aspecto** | **Plan aprobado** | **Implementación final** | **Justificación** |
|---|---|---|---|
| Escaneo QR / barras | Funcionalidad incluida para agilizar el registro y consulta de productos. | No implementada en la versión inicial. | Se priorizó el desarrollo y validación de los módulos centrales del sistema. La funcionalidad fue postergada por alcance y tiempo, sin afectar el objetivo general del proyecto. |
| Impacto en el sistema | Mejora en el ingreso de datos. | No afecta la operación del sistema. | El sistema mantiene su funcionamiento completo mediante registro manual estructurado. |
| Proyección futura | Prevista para la versión inicial. | Planificada para versiones futuras. | La arquitectura permite su incorporación posterior sin cambios estructurales significativos. |

```
/**
 * Utilidades para mejorar la accesibilidad según WCAG 2.1
 * Proporciona funciones helper para hacer la app accesible para usuarios con discapacidades
 */
object AccessibilityHelper {

    /**
     * Verifica si el modo de accesibilidad está activo
     */
    fun isAccessibilityEnabled(context: Context): Boolean {
        val am = context.getSystemService(Context.ACCESSIBILITY_SERVICE) as AccessibilityManager
        return am.isEnabled && am.isTouchExplorationEnabled
    }

    /**
     * Configura content description para una vista con información de producto
     */
    fun setProductAccessibility(view: View, name: String, price: Double, stock: Int) {
        view.contentDescription = "Producto: $name. Precio: $${String.format("%.2f", price)}. Stock disponible
        ViewCompat.setImportantForAccessibility(view, ViewCompat.IMPORTANT_FOR_ACCESSIBILITY_YES)
    }

    /**
     * Configura content description para una imagen
     */
    fun setImageAccessibility(imageView: ImageView, description: String) {
        imageView.contentDescription = description
        ViewCompat.setImportantForAccessibility(imageView, ViewCompat.IMPORTANT_FOR_ACCESSIBILITY_YES)
    }

    /**
```

Anexo No. 19 Accesibilidad WCAG 2.1 y Notificaciones

```
import { Notification } from "../models/index.js";
import { Product } from "../models/index.js";
import { Op } from "sequelize";

export const getNotifications = async (req, res) => {
  try {
    // Primero, verificar y crear notificaciones de stock bajo
    await checkAllLowStockProducts();

    const notifications = await Notification.findAll({
      order: [['createdAt', 'DESC']],
      limit: 50
    });
    res.json(notifications);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

export const markAsRead = async (req, res) => {
  try {
    const { id } = req.params;
    const notification = await Notification.findByPk(id);
    if (!notification) return res.status(404).json({ error: "Not found" });
    notification.read = true;
    await notification.save();
    res.json({ message: "Marked as read" });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```

```
/**
 * Genera un reporte PDF de ventas
 * @return Ruta del archivo generado
 */
fun generateSalesReport(context: Context, sales: List<Sale>, startDate: String?, endDate: String?): String
    val timestamp = SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(Date())
    val fileName = "Ventas_$timestamp.pdf"
    val file = File(getReportsDirectory(context), fileName)

    val pdfWriter = PdfWriter(file)
    val pdfDocument = PdfDocument(pdfWriter)
    val document = Document(pdfDocument)

    // Título
    val title = Paragraph("Reporte de Ventas")
        .setFontSize(20f)
        .setBold()
```

Anexo No. 20 REPORTES CSV/EXCEL

```
/**
 * Genera un reporte PDF de ventas
 * @return Ruta del archivo generado
 */
fun generateSalesReport(context: Context, sales: List<Sale>, startDate: String?, endDate: String?): String
    val timestamp = SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(Date())
    val fileName = "Ventas_$timestamp.pdf"
    val file = File(getReportsDirectory(context), fileName)

    val pdfWriter = PdfWriter(file)
    val pdfDocument = PdfDocument(pdfWriter)
    val document = Document(pdfDocument)

    // Título
    val title = Paragraph("Reporte de Ventas")
        .setFontSize(20f)
        .setBold()
        .setTextAlignment(TextAlignment.CENTER)
    document.add(title)

    // Período
    if (startDate != null && endDate != null) {
        val period = Paragraph("Período: $startDate - $endDate")
            .setFontSize(12f)
            .setTextAlignment(TextAlignment.CENTER)
```

```
/**
 * Genera un reporte CSV de ventas
 * @return Ruta del archivo generado
 */
fun generateSalesReport(context: Context, sales: List<Sale>, startDate: String?, endDate: String?): String
    val timestamp = SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(Date())
    val fileName = "Ventas_$timestamp.csv"
    val file = File(getReportsDirectory(context), fileName)

    FileWriter(file).use { writer ->
        // Información del reporte
        writer.append("Reporte de Ventas\n")
        if (startDate != null && endDate != null) {
            writer.append("Período: $startDate - $endDate\n")
        }
        writer.append("Generado: ${SimpleDateFormat("dd/MM/yyyy HH:mm", Locale.getDefault()).format(Date()
        writer.append("\n")

        // Encabezados
        writer.append("ID,Fecha,Cliente ID,Total,Estado\n")

        // Datos
        var totalRevenue = 0.0
        sales.forEach { sale ->
            writer.append(sale.id?.toString() ?: "")
            writer.append(CSV_SEPARATOR)
            writer.append(escapeCsvValue(sale.date ?: ""))
            writer.append(CSV_SEPARATOR)
            writer.append(sale.clientId?.toString() ?: "")
```