Desarrollo de un Sistema Integral Móvil para la Gestión de Ventas e Inventarios en Grupo APL Ingeniería LTDA

Daniel Jair Cañate Velasco

Trabajo de Grado para Optar al Título de Ingeniero en Sistemas

Director

Carlos Adolfo Beltrán Castro

Ingeniero en sistemas especialista en telecomunicaciones

Universidad Industrial de Santander

Facultad de Ingenierías Fisiomecánicas

Escuela de Ingeniería de Sistemas e Informática

Ingeniería de sistemas

Bucaramanga

2025

Dedicatoria

Este logro es el reflejo del esfuerzo conjunto de muchas personas que han estado a mi lado en los momentos más importantes de este camino.

Dedico este proyecto, en primer lugar, a mi familia, mi base y motor incondicional: a mi padre Jair Cañate, por ser siempre un ejemplo de dedicación y perseverancia; a mi madre Sandra Velasco, por su apoyo inagotable y por enseñarme el valor del esfuerzo y la honestidad; y a mis hermanas Laura Daniela y Natalia Sofía, quienes con su cariño y alegría han iluminado los días más difíciles y celebrado conmigo cada logro alcanzado.

A mis amigos Carlos Gutiérrez, Camilo Rincón, David Ricardo, Jorge Moreno, Leonardo Rojas, Diego Leal, Álvaro Torres, Jerónimo Arbeláez y Luis Ayala, quienes han sido compañeros de viaje, de estudio y de vida, aportando con su amistad y apoyo momentos de motivación, confianza y compañía incondicional durante toda esta etapa universitaria.

De manera muy especial, a mi novia Valentina Pérez Mora, quien con sus palabras de aliento, ha sido inspiración y fortaleza constante, impulsándome a continuar incluso cuando los desafíos parecían insuperables. Su presencia ha significado más de lo que las palabras podrían expresar.

A todos ustedes, gracias por creer en mí, por caminar conmigo en cada paso y por ser parte esencial de este logro que hoy culmina una etapa, pero abre la puerta a nuevos sueños.

Agradecimientos

Al culminar este proyecto, quiero expresar mi más sincero agradecimiento a todas las personas que hicieron posible que hoy llegue a esta meta.

En primer lugar, agradezco profundamente al profesor Carlos Adolfo Beltrán Castro, quien en su calidad de director de este proyecto, brindó su guía, acompañamiento y valiosas recomendaciones, demostrando compromiso y dedicación en cada etapa del proceso. Su orientación fue clave para el desarrollo técnico y académico de este trabajo.

Agradezco igualmente al profesor Luis Carlos Gómez Flórez, por su ejemplo de excelencia, su enseñanza constante y su apoyo durante toda mi formación. Su aporte ha dejado huella en mi crecimiento profesional y personal.

Extiendo mi gratitud a Grupo APL Ingeniería LTDA, por abrirme las puertas para realizar mi práctica empresarial, permitiéndome aplicar los conocimientos adquiridos y enfrentar retos reales que sin duda enriquecieron mi formación.

No puedo dejar de agradecer a todos los docentes, administrativos y compañeros que, de una u otra forma, formaron parte de mi camino universitario, aportando enseñanzas, apoyo o simplemente un consejo en los momentos que más se necesitó.

Finalmente, a mi familia, a mis amigos y a mi novia, gracias por acompañarme, por sostenerme en los momentos difíciles y por ser el motor que impulsó cada paso que me llevó hasta aquí.

Este proyecto es también suyo.

Tabla de Contenido

Pág.

1. Introducción 14

2. Planteamiento y justificación 15

3.1 Objetivo general 17

3.2 Objetivos específicos 17

4. Marco de referencia 18

4.1 Point of Sale (POS) 18

4.2 Inventory Management (IM) 18

4.3 Arquitectura Cliente - Servidor 19

4.4 Sincronización de Datos en Bases de Datos Móviles 19

4.5 PDA (Personal Digital Assistant) 20

5. Antecedentes del tema 21

5.1 Alegra POS 21

5.2 Siigo POS 21

5.3 Tienda DA 21

5.4 Vendty 22

6. Metodología 22

6.1 Metodología Agile 23

6.1.1 Origen 23

6.1.2 Principios del Manifiesto Agile 24

6.1.3 Beneficios de la Metodología Agile 25

6.1.4 Tipos de métodos Agile 25

6.1.4.1 Extreme Programming (XP). 25

6.1.4.2 Scrum. 26

6.1.4.3 Dynamic Systems Development Method (DSDM). 26

6.1.4.4 Adaptive Software Development (ASD). 27

6.1.4.5 Crystal Methods. 27

6.2 Implementación de metodologías Agile 27

6.2.1 Desarrollo Iterativo e Incremental 27

6.2.2 Gestión de Requisitos y Priorización 28

6.2.3 Adaptabilidad y Aprendizaje Continuo 28

6.2.4 Integración Continua y Calidad del Código 28

6.2.5 Comunicación y Coordinación del Equipo 29

6.2.6 Planificación y Seguimiento del Desarrollo 29

6.3 Potenciales tecnologías a implementar en el sistema POS e inventario móvil 30

6.3.1 Tecnologías de desarrollo móvil 30

6.3.1.1 React Native. 30

6.3.1.2 Kotlin. 30

6.3.1.3 Flutter. 31

6.3.2 Lenguajes de programación para backend 31

6.3.2.1 JavaScript. 31

6.3.2.2 Python. 32

6.3.2.3 Java. 32

6.3.2.4 C#. 32

6.3.3 Construcción de APIS 33

6.3.3.1 Node.js con Express. 33

6.3.3.2 Django. 33

6.3.3.3 ASP.NET Core Web API. 34

6.3.4 Bases de datos 34

6.3.4.1 PostgreSQL. 34

6.3.4.2 Microsoft SQL Server. 35

6.3.4.3 SQLite. 35

6.3.5 Visual Studio Code 36

6.4 Justificación de las tecnologías seleccionadas 37

6.4.1 Flutter 37

6.4.2 ASP.NET Core Web API 37

6.4.3 C# 38

6.4.4 SQL Server 39

6.4.5 SQLite 39

6.4.6 Visual Studio Code (VSC) 40

6.5 Alineación estratégica de las tecnologías seleccionadas 40

7. Arquitectura del Sistema 41

7.1 Modelo de arquitectura Cliente - Servidor 42

7.1.1 Justificación de la elección cliente servidor 42

7.2 Componentes del sistema 44

7.2.1 Cliente (Aplicación Móvil) 44

7.2.2 Servidor (Backend y API REST) 46

7.2.3 Base de Datos y Sincronización 47

7.3 Diagrama de Arquitectura 49

8. Desarrollo del sistema 49

8.1 Introducción al módulo POS 49

8.2 Análisis y comprensión del sistema existente 52

8.3 Selección de librería para escaneo de productos 52

8.4 Implementación de la base de datos SQLite para funcionalidad offline del Sistema POS 54

8.5 Desarrollo de vistas e interfaces de usuario en el sistema POS 87

8.5.1 Diseño módulo POS 87

8.5.1.1 Justificación del Diseño 87

8.5.1.2 Selección de Colores. 87

8.5.1.3 Iconografía. 88

8.5.2 Menú Lateral 89

8.5.3 Crear Factura 91

8.5.4 Crear Pedido 93

8.5.5 Historial 95

8.5.6 Historial Detalle 96

8.5.7 Subir Información 98

8.5.8 Seleccionar Impresora 99

8.5.9 Cerrar Caja 100

8.6 Introducción al módulo de Inventario 101

8.7 Análisis y comprensión del sistema existente 102

8.8 Relación entre el módulo de Inventarios y el módulo POS 102

8.8.1 Autenticación Unificada 102

8.8.2 Infraestructura de Base de Datos Compartida 103

8.8.3 Reutilización de Funcionalidades 103

8.8.5 Función Complementaria dentro del Ecosistema 104

8.9.1 Menú Lateral 105

8.9.2 Inventario 106

8.9.3 Ajuste de inventario 108

8.9.4 Productos en inventario 109

8.9.5 Ventana emergente de agregado de productos 111

8.9.6 Chequeo de precios 112

9. Pruebas y Validaciones 113

10. Conclusiones 115

Referencias Bibliográficas 117

Apéndices 121

Lista de Figuras

Pág.

Figura 1. Flujo donde Flutter interactúa con una base de datos local SQLite y una API REST, la cual a su vez se conecta con una base de datos SQL Server. 49

Figura 2. Estructura de la Clase SqliteDatabase 84

Figura 3. Código para el cierre de la conexión 86

Figura 4. Vista del menú lateral de navegación 89

Figura 5. Vista de creación de facturas 91

Figura 6. Vista de creación de pedidos 93

Figura 7. Vista del historial 95

Figura 8. Vista de la lista de productos 96

Figura 9. Vista del menú de subir información 98

Figura 10. Vista del menú para elegir que impresora usar 99

Figura 11. Vista del menú para cerrar caja 100

Figura 12. Vista del menú lateral desde inventarios 105

Figura 13. Vista de la lista de bodegas 106

Figura 14. Vista del menú de ajuste de inventario 108

Figura 15. Vista del menú de la bodega principal 109

Figura 16. Vista de la ventana agregar producto 111

Figura 17. Vista del menú chequeo de precio 112

Lista de Apéndices

Pág.

Apéndice A. Vista de Login 121

Apéndice B. Vista de Ajustes 122

Apéndice C. Vista de Configuración 124

Apéndice D. Vista de IMEI 125

Apéndice E. Vista de Inicio 126

Apéndice F. Vista de Arqueo de Caja 127

Apéndice G. Vista de Movimientos de Dinero 129

Apéndice H. Vista de Sincronizar Información 130

Apéndice I. Vista de Crear Cliente 131

Apéndice J. Vista de Métodos de Pago 132

Apéndice K. Vista de Información del Equipo 133

Resumen

Título: Desarrollo de un Sistema Integral Móvil para la Gestión de Ventas e Inventarios en Grupo APL Ingeniería LTDA

Autor: Daniel Jair Cañate Velasco

Palabras Clave: POS móvil, Inventarios, Flutter, API REST, SQL Server, SQLite, Facturación, C#.

Descripción: El presente proyecto de grado expone el diseño y desarrollo de un sistema POS móvil integrado a un módulo de gestión de inventarios, desarrollado para Grupo APL Ingeniería LTDA. Ambos módulos operan dentro de una misma aplicación móvil construida en Flutter, permitiendo el acceso diferenciado a cada uno según las credenciales de usuario ingresadas en la pantalla de inicio de sesión.

El sistema POS proporciona funcionalidades esenciales para la operación de puntos de venta, como facturación de productos, creación de pedidos, gestión de movimientos de dinero, arqueo y cierre de caja, así como sincronización manual de datos en condiciones de operación offline. El módulo de Inventarios, por su parte, permite la consulta de stock en tiempo real, la ejecución de ajustes de inventario y el chequeo de precios, garantizando la trazabilidad de existencias.

La arquitectura implementada es de tipo cliente-servidor, con comunicación vía API REST hacia una base de datos central SQL Server, mientras que para operaciones offline se empleó SQLite en el módulo POS.

Se diseñaron interfaces de usuario eficientes y coherentes con la identidad visual corporativa, priorizando la usabilidad en dispositivos PDA industriales.

Para validar el correcto funcionamiento del sistema, se llevaron a cabo pruebas unitarias funcionales y validaciones manuales en dispositivos reales, verificando la correcta sincronización de datos, la integridad de las operaciones comerciales y la consistencia en la gestión de inventarios.

El sistema desarrollado se plantea como una herramienta para la gestión de los procesos comerciales y de inventarios de la empresa, ofreciendo una plataforma tecnológica escalable y alineada con sus requerimientos operativos.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Fisiomecánicas. Escuela de Ingeniería de Sistemas e Informática. Ingeniería de sistemas. Director: Carlos Adolfo Beltrán Castro. Ingeniero en sistemas especialista en telecomunicaciones.

Abstract

Title: Development of a Comprehensive Mobile System for Sales and Inventory Management at Grupo APL Ingeniería LTDA

Author: Daniel Jair Cañate Velasco

Key Words: Mobile POS, Inventories, Flutter, REST API, SQL Server, SQLite, Invoicing, C#.

Description: This graduation project presents the design and development of a mobile POS system integrated with an inventory management module, developed for Grupo APL Ingeniería LTDA. Both modules operate within a single mobile application built in Flutter, allowing differentiated access to each module based on the user credentials entered on the login screen.

The POS system provides essential functionalities for point-of-sale operations, such as product invoicing, order creation, management of cash movements, cash register audits, and end-of-day closures, as well as manual data synchronization under offline operation conditions. The Inventory module, in turn, enables real-time stock consultation, execution of inventory adjustments, and price checking, ensuring traceability of stock levels.

The implemented architecture follows a client-server model, with communication through a REST API to a centralized SQL Server database, while offline operations in the POS module employ SQLite as a local storage solution.

User interfaces were designed to be efficient and consistent with the company's corporate visual identity, prioritizing usability on industrial PDA devices.

To validate the correct functioning of the system, functional unit tests and manual validations were carried out on real devices, verifying correct data synchronization, transactional integrity, and inventory management consistency.

The developed system is intended to serve as a technological tool for managing the company's commercial and inventory processes, offering a scalable platform aligned with its operational requirements.

---

* Degree Work

** Faculty of Physiomechanical Engineering. School of Systems and Computer Engineering. Systems Engineering. Director: Carlos Adolfo Beltrán Castro. Systems Engineer, Specialist in Telecommunications.

1. Introducción

En el comercio minorista actual, la gestión eficiente del proceso de pago es un factor clave en la operatividad de los establecimientos. Las largas filas en las cajas pueden generar frustración en los clientes y afectar la dinámica de atención en los puntos de venta (van Riel, Semeijn, Ribbink, & Bomert-Peters, 2012). Ante este panorama, la implementación de soluciones orientadas a agilizar las transacciones y mejorar la administración de inventarios representa una alternativa viable para los negocios.

Como respuesta a esta necesidad, este proyecto consiste en la contribución al desarrollo de un sistema de punto de venta (POS) móvil con un módulo de gestión de inventarios. A través de la implementación de soluciones adecuadas, se busca estructurar la gestión de la información y proporcionar una herramienta que se integre a los procesos internos de la empresa. Este sistema se basa en una arquitectura cliente-servidor, permitiendo que la aplicación móvil se comunique con un servidor central para gestionar datos en tiempo real, asegurando una sincronización efectiva entre el punto de venta y el inventario.

El sistema incorpora un dispositivo portátil que permite el escaneo de productos y la generación de facturas de manera inmediata. La "caja de fila rápida" móvil está diseñada para complementar las cajas tradicionales, ofreciendo mayor flexibilidad en el proceso de pago, mientras que el módulo de inventarios facilita el control y actualización del stock mediante el uso de una base de datos local sincronizada con un servidor central para garantizar la integridad de los datos.

Como estudiante en prácticas, la participación en este proyecto implica en intervenir en distintas etapas del desarrollo de la aplicación, apoyando la creación de un nuevo código y contribuyendo a la implementación de los módulos del sistema. A lo largo de esta práctica, se aplicara conocimientos en ingeniería de software para comprender la lógica del negocio e integrarla en el sistema, asegurando su funcionamiento conforme a los requerimientos establecidos. La práctica se centrará exclusivamente en el desarrollo y funcionalidad del sistema, dejando abierta la posibilidad de futuras evaluaciones sobre su impacto en la operación.

2. Planteamiento y justificación

En el comercio minorista, los tiempos de espera en el proceso de pago y la eficiencia en la gestión de inventarios son dos factores cruciales que determinan tanto la satisfacción del cliente como la rentabilidad del negocio. La implementación de un sistema de punto de venta (POS) móvil y una gestión de inventarios ágil se presentan como soluciones complementarias que abordan estos dos desafíos, contribuyendo de manera equitativa a mejorar tanto la experiencia del cliente como los procesos internos del establecimiento.

El sistema POS propuesto en este proyecto busca reducir las filas en las cajas tradicionales mediante el uso de dispositivos PDA que permitan realizar el pago de forma rápida y eficiente para clientes con pocas compras. Estudios han mostrado que las largas esperas afectan de manera negativa la percepción de la tienda, disminuyendo la satisfacción y la intención de regreso del cliente (van Riel, Semeijn, Ribbink, & Bomert-Peters, 2012). Además, la presión social percibida

cuando otros clientes esperan en fila incrementa la incomodidad y el deseo de completar el proceso rápidamente (Dahm, Wentzel, Herzog, & Wiecek, 2018, 2018). La introducción de una "caja de fila rápida" mediante el sistema POS móvil responde directamente a esta problemática, permitiendo una atención ágil y minimizando las frustraciones asociadas al tiempo de espera.

Por otro lado, la gestión de inventarios (IM) en tiempo real, integrada al sistema POS, permite una actualización automática y precisa de los productos, eliminando la necesidad de procedimientos manuales y aumentando la eficiencia del negocio. La gestión tradicional de inventarios en algunos negocios requiere que los empleados transporten físicamente productos a un escáner fijo, lo que consume tiempo y recursos. En este sentido, Pillai (2010) destaca que una adecuada IM es esencial para minimizar los costos operativos y asegurar la disponibilidad de productos, lo cual resulta crucial para la continuidad del servicio y la satisfacción del cliente. La incorporación de dispositivos PDA permite realizar el escaneo en cualquier área del establecimiento, agilizando el proceso de verificación y registro de productos.

La implementación conjunta de este sistema POS y un módulo administrativo de IM busca facilitar el flujo de trabajo y fortalecer el control de inventarios, manteniendo una actualización en tiempo real del stock disponible. Esta integración tiene el potencial de ofrecer una experiencia de compra más rápida y sin contratiempos, además de favorecer la operación interna del establecimiento. Con un sistema POS que busca reducir los tiempos de espera y una IM eficiente que contribuye a la disponibilidad de productos, el proyecto tiene el propósito de satisfacer las demandas de conveniencia y rapidez de los consumidores modernos, beneficiando tanto a clientes como al negocio.

3. Objetivos

3.1 Objetivo general

Contribuir al desarrollo de un sistema de punto de venta (POS) móvil con un módulo de gestión de inventarios, con el propósito de proporcionar una herramienta adecuada para la administración de ventas e inventarios dentro de la empresa.

3.2 Objetivos específicos

- Comprender y analizar la lógica de negocio, para facilitar su correcta integración y funcionalidad dentro de la aplicación.
- Apoyar en el diseño de interfaces, asegurando que el sistema POS y de inventarios proporcione una experiencia de usuario accesible y eficiente en dispositivos móviles.
- Contribuir al desarrollo del módulo de gestión de inventarios, asistiendo en la implementación de funcionalidades que permitan mantener el control actualizado de los productos y garantizar una administración adecuada del stock.
- Realizar tareas relacionadas con la administración de bases de datos, asegurando su correcta estructura, almacenamiento y sincronización para la gestión eficiente de la información dentro del sistema.
- Participar en la integración del sistema bajo una arquitectura eficiente, facilitando la comunicación entre la aplicación móvil y la base de datos, garantizando un flujo de datos constante y confiable.
- Organizar y planificar el desarrollo del sistema de manera que se permita ajustar y mejorar continuamente el producto final.

4. Marco de referencia

4.1 Point of Sale (POS)

Un sistema POS (Point of Sale o Punto de Venta) es una solución tecnológica diseñada para gestionar y registrar transacciones comerciales de forma digital, reemplazando métodos convencionales como el papeleo. Este sistema permite a las Micro, Pequeñas y Medianas Empresas (MSMEs) optimizar sus procesos de venta, disminuir errores humanos y mejorar la eficiencia operativa (Sidhunata et al., 2023).

El sistema POS ofrece funcionalidades que incluyen el registro preciso de transacciones, la gestión de información sobre productos y servicios, y el acceso a datos relevantes para supervisar el rendimiento empresarial (Sidhunata et al., 2023). Según los autores, "la digitalización del proceso de grabación de transacciones de ventas puede mejorar la competitividad y el rendimiento empresarial" (p. 879). Además, se destaca que "los sistemas de gestión de punto de venta deben optimizarse para promover la innovación y mejorar el rendimiento empresarial" (p. 876).

4.2 Inventory Management (IM)

La gestión de inventario es el proceso de orquestar el flujo de mercancías a través de una empresa en un ciclo continuo de pedidos, almacenamiento, producción, venta y reposición. La gestión del inventario se realiza generalmente en dos niveles: gestión agregada de inventario y ubicación de almacenamiento y gestión de inventario a nivel de artículo (Hearson, 2024).

4.3 Arquitectura Cliente - Servidor

La arquitectura cliente-servidor es un modelo de comunicación en red utilizado en el diseño de sistemas informáticos, donde se establece una distinción clara entre los roles de los componentes de una red. En este modelo, el cliente es el dispositivo o programa que realiza solicitudes de recursos o servicios, y el servidor es el sistema que proporciona esos servicios o recursos. El cliente envía peticiones al servidor, quien procesa esas solicitudes y envía de vuelta la respuesta solicitada. Este patrón es fundamental en la mayoría de las aplicaciones modernas, donde los clientes pueden ser dispositivos como computadoras personales, teléfonos inteligentes o incluso otros servidores, y los servidores proporcionan servicios como bases de datos, acceso a archivos o servicios web. La comunicación en la arquitectura cliente-servidor generalmente se realiza mediante protocolos de red bien establecidos, como HTTP, FTP o TCP/IP. Una característica clave de esta arquitectura es la centralización del procesamiento y almacenamiento de datos en el servidor, lo que permite la administración más eficiente de los recursos y facilita el mantenimiento, ya que las actualizaciones o cambios en el servidor no requieren modificaciones en todos los clientes. Además, este modelo facilita la escalabilidad, permitiendo que varios clientes interactúen con un servidor o con una red de servidores (Tanenbaum & Wetherall, 2011; Comer, 2018).

4.4 Sincronización de Datos en Bases de Datos Móviles

La sincronización de datos es el proceso de mantener la coherencia y actualización de la información entre múltiples dispositivos o sistemas. Este procedimiento garantiza que todos los usuarios y aplicaciones accedan a la misma versión de los datos, independientemente del dispositivo o plataforma que utilicen. Es especialmente crucial cuando diversos equipos o aplicaciones requieren acceso simultáneo a la misma información, asegurando que cualquier

modificación se refleje en todos los sistemas en tiempo real. La sincronización de datos ayuda a preservar la integridad de la información, minimiza el riesgo de pérdida o duplicación de datos y facilita la colaboración y la toma de decisiones informadas (Oracle, 2024).

4.5 PDA (Personal Digital Assistant)

Una PDA (Personal Digital Assistant) es un dispositivo portátil que combina funciones de organización personal —como agenda, calendario, notas, contactos— con capacidades de computación móvil. Tradicionalmente diseñadas para asistir en la gestión de información personal, las PDA evolucionaron integrando funcionalidades más avanzadas como conectividad inalámbrica, sincronización de datos con servidores centrales, lectura de códigos de barras, gestión de inventarios y ejecución de aplicaciones empresariales específicas (Turban, Volonino, & Wood, 2015).

Actualmente, las PDA industriales, utilizadas en entornos corporativos como logística, distribución y ventas en campo, incorporan características robustas tales como resistencia a golpes, humedad y polvo, pantallas táctiles optimizadas para exteriores y módulos de conectividad como Wi-Fi, Bluetooth o datos móviles (Grewal, Roggeveen, & Nordfält, 2017). Además, permiten la ejecución de sistemas especializados que requieren movilidad, garantizando la recolección de datos en tiempo real y la sincronización directa con infraestructuras empresariales.

En el contexto de este proyecto, las PDA funcionan como dispositivos de trabajo esenciales para la ejecución de operaciones en el sistema POS y de inventarios, ofreciendo versatilidad, portabilidad y confiabilidad en la captura y gestión de información comercial y logística.

5. Antecedentes del tema

5.1 Alegra POS

Alegra POS es un software colombiano que permite simplificar las tareas relacionadas con la venta, facturación y el cobro de productos o servicios en un local, además de funcionar sin la necesidad de tener que descargar un programa para puntos de venta. Alegra es una solución en la nube la cual es accesible desde un dispositivo móvil facilitando el acceso del personal en cualquier lugar del establecimiento. (Alegra, n.d.)

5.2 Siigo POS

Siigo POS es una extensión del software contable Siigo, ampliamente utilizado en Colombia su módulo POS está diseñado para permitir a los negocios registrar ventas y controlar inventarios. Además, ofrece la ventaja de estar conectado directamente a las finanzas y contabilidad de la empresa.

Esto permite a la empresa generar informes de forma centralizada, lo que ahorra tiempo y mejora la precisión en la gestión administrativa (Siigo, n.d.).

5.3 Tienda DA

TiendaDA es una solución diseñada para pequeñas empresas locales y tiendas de conveniencia que buscan modernizar sus procesos sin requerir una infraestructura tecnológica complicada. En contraste con otros sistemas POS, TiendaDA tiene características particulares para registrar ventas a crédito y administrar cuentas por cobrar, lo cual es una práctica habitual en las tiendas de barrio en Colombia. Asimismo, les da la opción a los dueños de visualizar informes simples sobre las ventas y el inventario, proporcionándoles una perspectiva rápida y fácil del

desempeño de su tienda sin características avanzadas que pudieran ser superfluas para empresas de pequeña escala (Tiendada, n.d.).

5.4 Vendty

Es una plataforma POS que brinda una solución más amplia y flexible para negocios de tamaño pequeño y mediano, como tiendas minoristas y cafeterías. Contrario a otros sistemas POS posibilita la administración de varias sucursales y dispone de herramientas para retener clientes, estrategias de mercadeo y un sistema de analytics avanzado que examina las ventas y la efectividad de los productos en cada establecimiento. Además, esta plataforma incluye características de comercio electrónico que posibilitan a las empresas ampliar sus actividades a través de ventas en Internet y vincular su stock físico con su tienda en línea (Vendty, n.d.).

6. Metodología

El desarrollo del sistema se basa en una combinación de diversas metodologías ágiles, seleccionadas estratégicamente para adaptarse a las necesidades del proyecto. En lugar de adherirse a un solo marco, se han integrado principios y prácticas de Extreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD) y Crystal Methods para optimizar el proceso de desarrollo, priorizando la calidad del software, la adaptabilidad y la entrega incremental de valor.

La elección de este enfoque híbrido responde a la necesidad de contar con un marco metodológico que combine la flexibilidad y rapidez de adaptación de Agile con estrategias

específicas para garantizar calidad, priorización de requisitos y comunicación efectiva. No todas las metodologías ágiles cubren de manera integral los desafíos del proyecto, por lo que adoptar una única metodología resultaría insuficiente para abordar los distintos aspectos del desarrollo del sistema.

6.1 Metodología Agile

La metodología Agile es un enfoque de desarrollo de software diseñado para adaptarse a entornos de cambio rápido y cumplir con ciclos de productos más cortos, característico de la economía digital. Este método promueve la flexibilidad y agilidad, optimizando procesos mediante iteraciones continuas, desarrollo de prototipos y plantillas para facilitar la adaptación a requisitos cambiantes durante el ciclo de desarrollo (Al-Saqqa, Sawalha, & Abdelnabi, 2020; Livermore, 2007).

6.1.1 Origen

Agile surgió en respuesta a las limitaciones de los métodos tradicionales como el enfoque en cascada, que resultaba lento y rígido. Fue formalizado en 2001 con el **Manifiesto Agile**, un documento que establece los valores y principios clave para una gestión de proyectos más dinámica, priorizando la entrega de valor al cliente y la capacidad de respuesta rápida ante cambios. Este manifiesto destaca cuatro valores principales:

1. **Individuos e interacciones sobre procesos y herramientas**: La comunicación entre las personas es más importante que seguir procesos rígidos o depender únicamente de herramientas específicas.

2. **Software funcionando sobre documentación extensiva**: Prioriza la creación de un producto funcional que se pueda probar en lugar de dedicar tiempo a documentación detallada.
3. **Colaboración con el cliente sobre negociación contractual**: Se enfoca en trabajar de cerca con el cliente durante todo el desarrollo, permitiendo adaptaciones constantes a sus necesidades.
4. **Respuesta ante el cambio sobre seguir un plan**: El objetivo es adaptarse al cambio en lugar de seguir un plan fijo, lo que permite al equipo ajustar sus objetivos según el aprendizaje y feedback continuo (Al-Saqqa et al., 2020).

6.1.2 Principios del Manifiesto Agile

De los valores principales de Agile derivan 12 principios que guían dicha metodología:

1. **Satisfacción del cliente** mediante la entrega continua de software funcional.
2. **Bienvenida a los cambios en los requisitos** incluso en etapas tardías del proyecto.
3. **Entregas frecuentes** de software, preferiblemente en periodos cortos.
4. **Colaboración constante entre el equipo y las partes interesadas**.
5. **Motivación y apoyo al equipo** para asegurar un ambiente de trabajo óptimo.
6. **Comunicación cara a cara** como método más efectivo de transmisión de información.
7. **El software funcionando** como medida principal de progreso.
8. **Desarrollo sostenible** mediante un ritmo constante de trabajo.
9. **Excelencia técnica y buen diseño** para mantener la agilidad del equipo.
10. **Simplicidad**, evitando la realización de tareas innecesarias.
11. **Equipos autoorganizados** que aprovechan sus conocimientos para adaptarse y mejorar.

12. **Reflexión periódica del equipo** para ajustar procesos y mejorar continuamente (Livermore, 2007; Al-Saqqa et al., 2020).

6.1.3 Beneficios de la Metodología Agile

La metodología Agile proporciona beneficios destacables, entre ellos:

- **Flexibilidad y capacidad de adaptación**: Permite modificar y ajustar el proyecto en cualquier momento sin comprometer su integridad, lo cual es vital en entornos con requisitos cambiantes.
- **Entrega continua de valor**: Gracias a sus iteraciones cortas y entregas frecuentes, el cliente recibe un producto funcional desde las primeras fases del proyecto.
- **Colaboración y comunicación enriquecida**: La estructura Agile fomenta una comunicación constante entre el equipo y el cliente, asegurando que el producto final esté alineado con las expectativas del cliente.

Sin embargo, implementar Agile también presenta ciertos desafíos. Por ejemplo, la falta de documentación extensiva puede ser un obstáculo para algunos proyectos que requieren trazabilidad. Además, la participación activa y constante del cliente puede ser una demanda difícil de cumplir en algunos contextos (Livermore, 2007).

6.1.4 Tipos de métodos Agile

6.1.4.1 Extreme Programming (XP).

Extreme Programming (XP) es un método ágil que se centra en mejorar la calidad del software y la capacidad de respuesta del equipo a los cambios en los requisitos del cliente. XP

promueve la entrega frecuente de pequeñas versiones de software, lo que permite obtener retroalimentación temprana y continua del cliente. Algunas de las prácticas clave en XP incluyen la programación en pareja (pair programming), el desarrollo dirigido por pruebas (test-driven development), y la integración continua (continuous integration). Estas prácticas tienen como objetivo minimizar errores y mejorar la colaboración dentro del equipo de desarrollo (Strode, 2006).

6.1.4.2 Scrum.

Scrum es un marco de trabajo orientado a la gestión de proyectos que organiza el trabajo en iteraciones llamadas sprints, que suelen durar entre dos y cuatro semanas. Durante cada sprint, el equipo de desarrollo trabaja en un conjunto de tareas priorizadas por el Product Owner. Scrum promueve reuniones diarias, llamadas "daily stand-ups", para sincronizar el progreso y discutir los desafíos. Este método enfatiza la adaptabilidad y la entrega incremental, y se estructura en roles específicos: el Product Owner, el Scrum Master y el equipo de desarrollo, cada uno con responsabilidades definidas que facilitan la autoorganización del equipo (Strode, 2006).

6.1.4.3 Dynamic Systems Development Method (DSDM).

DSDM es un método ágil que surgió como una adaptación del desarrollo rápido de aplicaciones (RAD) y está orientado a proyectos de desarrollo de software a gran escala. Este método define principios y prácticas que permiten a los equipos ajustar su enfoque en función de los requisitos del proyecto y los cambios que surgen. DSDM se centra en la participación activa del usuario y en la entrega incremental, utilizando ciclos de desarrollo iterativos. Además, DSDM impone un control estricto de tiempo y recursos, lo cual lo hace adecuado para proyectos con plazos fijos (Strode, 2006).

6.1.4.4 Adaptive Software Development (ASD).

El método de desarrollo adaptativo de software (ASD) se enfoca en adaptarse a entornos de alta incertidumbre. ASD considera el desarrollo de software como un proceso continuo de especulación, colaboración y aprendizaje, en lugar de seguir una planificación estricta. Los equipos de ASD suelen responder de manera rápida a cambios repentinos en los requisitos y se organizan para aprender de cada iteración del desarrollo. Este enfoque resulta útil en proyectos donde los requisitos evolucionan constantemente y la velocidad de respuesta es crucial (Strode, 2006).

6.1.4.5 Crystal Methods.

Los métodos Crystal son un conjunto de metodologías ágiles que incluyen varias versiones (como Crystal Clear, Crystal Yellow, Crystal Orange), cada una adaptada al tamaño del equipo y la criticidad del proyecto. Este enfoque promueve la colaboración cercana y la comunicación entre los miembros del equipo, ajustando sus prácticas en función del contexto del proyecto. Crystal destaca por su flexibilidad y su enfoque en crear un ambiente de trabajo que favorezca la comunicación y la adaptabilidad, lo cual lo convierte en una buena opción para proyectos de diversa escala y complejidad (Strode, 2006)

6.2 Implementación de metodologías Agile

6.2.1 Desarrollo Iterativo e Incremental

El sistema se desarrollará en iteraciones cortas, permitiendo la entrega progresiva de funcionalidades en pequeños incrementos. Cada iteración incluirá fases de planificación, desarrollo, integración y revisión, asegurando la mejora continua del producto.

6.2.2 Gestión de Requisitos y Priorización

Siguiendo el enfoque de DSDM, se empleará la técnica de priorización MoSCoW para clasificar los requisitos en cuatro niveles:

- Must-have (Imprescindibles): Funcionalidades esenciales para el funcionamiento del sistema POS y el módulo de inventarios.
- Should-have (Importantes): Características que mejoran la experiencia del usuario, pero cuya ausencia no compromete la funcionalidad principal.
- Could-have (Opcionales): Elementos adicionales que pueden ser incorporados si el tiempo y los recursos lo permiten.
- Won't-have (No esenciales en esta fase): Funcionalidades que quedan fuera del alcance actual del proyecto.

6.2.3 Adaptabilidad y Aprendizaje Continuo

Siguiendo los principios de ASD, el proceso de desarrollo será flexible ante cambios en los requisitos o necesidades del negocio. Se fomentará la experimentación y la iteración para mejorar continuamente la solución y responder de manera efectiva a las condiciones cambiantes del entorno.

6.2.4 Integración Continua y Calidad del Código

Se adoptarán prácticas de Extreme Programming (XP) para garantizar la estabilidad del sistema y la correcta integración de los módulos desarrollados. Entre las principales prácticas adoptadas se incluyen:

- Integración continua: Las funcionalidades desarrolladas se integrarán de manera progresiva, asegurando que el sistema se mantenga estable a lo largo del proceso de desarrollo.
- Revisión constante del código: Se fomentará la validación frecuente de las funcionalidades implementadas para detectar posibles mejoras o errores en fases tempranas del desarrollo.

6.2.5 Comunicación y Coordinación del Equipo

De acuerdo con Crystal Methods, se priorizará una comunicación eficiente dentro del equipo de desarrollo, asegurando que las decisiones se tomen con información clara y precisa. Para ello, se implementarán:

- Interacciones directas y reuniones periódicas para discutir avances y resolver problemas de manera ágil.
- Registro estructurado de cambios y mejoras utilizando herramientas colaborativas, reduciendo la documentación innecesaria sin comprometer la trazabilidad del proyecto.

6.2.6 Planificación y Seguimiento del Desarrollo

El proceso de desarrollo seguirá principios de Scrum, con iteraciones cortas y reuniones de seguimiento regulares para evaluar el progreso del proyecto. Se establecerán reuniones de control técnico para:

- Revisar el estado actual del desarrollo.
- Ajustar prioridades y planificar nuevas iteraciones.
- Evaluar la integración de funcionalidades y su alineación con los objetivos del proyecto.

6.3 Potenciales tecnologías a implementar en el sistema POS e inventario móvil

En el desarrollo del sistema de punto de venta (POS) y gestión de inventarios, se explorarán diversas tecnologías que pueden ser utilizadas para cada uno de los componentes del sistema. A continuación, se presentan las principales tecnologías evaluadas, que incluyen opciones para el desarrollo móvil, bases de datos y arquitecturas de backend, las cuales serán seleccionadas según las necesidades del proyecto.

6.3.1 Tecnologías de desarrollo móvil

6.3.1.1 React Native.

React Native es un framework de desarrollo de aplicaciones móviles de código abierto creado por Meta Platforms (anteriormente Facebook). Permite a los desarrolladores construir aplicaciones móviles utilizando JavaScript y el framework React, facilitando el desarrollo de aplicaciones nativas para plataformas como Android e iOS con una sola base de código. Una característica destacada de React Native es su capacidad de "hot reload", que permite a los desarrolladores ver los cambios en tiempo real durante el proceso de desarrollo, mejorando la eficiencia y la experiencia de desarrollo. (Meta Platforms, n.d.)

6.3.1.2 Kotlin.

Kotlin es un lenguaje de programación de tipo estático y multiplataforma desarrollado por JetBrains. Es completamente interoperable con Java y se utiliza principalmente para el desarrollo de aplicaciones Android. Kotlin ofrece características modernas como la seguridad de nulidad y funciones de extensión, que simplifican el proceso de desarrollo y reducen la posibilidad de

errores. Además, Kotlin Multiplatform permite compartir código entre plataformas, facilitando el desarrollo de aplicaciones para Android e iOS. (JetBrains, n.d.)

6.3.1.3 Flutter.

Flutter es un kit de herramientas de interfaz de usuario de código abierto creado por Google, diseñado para desarrollar aplicaciones nativas compiladas para dispositivos móviles, web y de escritorio a partir de una única base de código. La principal ventaja de Flutter radica en su capacidad para construir interfaces de usuario altamente personalizables y de alto rendimiento, aprovechando su propio motor gráfico, lo cual permite una experiencia visual coherente y rápida en todas las plataformas (Flutter, n.d.).

Una de las características distintivas de Flutter es su uso del lenguaje de programación Dart, optimizado para proporcionar tiempos de ejecución rápidos y soportar compilación tanto en modo nativo como just-in-time (JIT). Gracias a esto, los desarrolladores pueden disfrutar de una experiencia de desarrollo interactiva y eficiente, con funciones como "hot reload", que permite ver los cambios en la aplicación al instante sin necesidad de recompilaciones completas (Flutter, n.d.).

6.3.2 Lenguajes de programación para backend

6.3.2.1 JavaScript.

JavaScript es un lenguaje de programación interpretado y orientado a objetos que se utiliza principalmente para crear páginas web dinámicas. Originalmente desarrollado por Netscape, este lenguaje se ejecuta en el navegador web y es una parte integral de las tecnologías fundamentales de la web, como HTML y CSS. JavaScript permite la manipulación de contenido en tiempo real, creación de interactividad, y es utilizado tanto en el desarrollo frontend como backend (a través de

Node.js). Su capacidad de funcionar en múltiples plataformas sin modificaciones lo convierte en un pilar en la creación de aplicaciones web modernas(Flanagan, 2020).

6.3.2.2 Python.

Python es un lenguaje de programación de alto nivel que enfatiza la legibilidad del código y la facilidad de uso. Fue creado por Guido van Rossum y lanzado en 1991. Python es ampliamente utilizado en áreas como desarrollo web, análisis de datos, inteligencia artificial, ciencia de datos y automatización debido a su sintaxis clara y sus poderosas bibliotecas. Su comunidad activa y la continua expansión de bibliotecas hacen de Python una de las opciones más populares para los desarrolladores de software en diversos campos (Van Rossum & Drake, 2009).

6.3.2.3 Java.

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases, desarrollado por James Gosling y Mike Sheridan en 1991, y lanzado oficialmente por Sun Microsystems en 1995. Una de sus características clave es la portabilidad, que permite ejecutar aplicaciones en cualquier dispositivo que tenga instalada la Java Virtual Machine (JVM), lo que ha dado lugar al lema "Write Once, Run Anywhere" (WORA). Java sigue siendo un lenguaje ampliamente utilizado para aplicaciones empresariales, móviles y sistemas integrados debido a su estabilidad y escalabilidad (Gosling, Joy, & Steele, 2000).

6.3.2.4 C#.

C# es un lenguaje de programación orientado a objetos y de propósito general, desarrollado por Microsoft como parte de su plataforma .NET. Diseñado para ser simple, moderno y versátil, C# permite a los desarrolladores crear aplicaciones de una amplia gama de tipos, desde

aplicaciones web y móviles hasta software de escritorio y servicios en la nube. Su sintaxis es similar a la de otros lenguajes de la familia C, como C++ y Java, lo que facilita su aprendizaje para programadores con experiencia en esos lenguajes (BillWagner, 2024)

C# destaca por su capacidad de trabajar en un entorno de desarrollo robusto, ofreciendo características avanzadas como la administración automática de memoria, el manejo de excepciones y la capacidad de trabajar con componentes reutilizables. Estas características permiten a los desarrolladores crear software seguro, escalable y eficiente (BillWagner, 2024).

6.3.3 Construcción de APIS

6.3.3.1 Node.js con Express.

Node.js es un entorno de ejecución de JavaScript basado en el motor V8 de Google Chrome. Está diseñado para facilitar la construcción de aplicaciones rápidas, escalables y con alto rendimiento, especialmente en entornos que requieren manejar múltiples conexiones simultáneas. Express, por su parte, es un framework web para Node.js que simplifica la construcción de aplicaciones web y APIs, proporcionando un conjunto de herramientas que facilita la configuración de rutas, middleware y la gestión de peticiones HTTP. Esta combinación es particularmente útil para aplicaciones que requieren velocidad, escalabilidad y capacidad de manejo eficiente de múltiples solicitudes(Cantrell & Pugh, 2017).

6.3.3.2 Django.

Django es un framework de desarrollo web de alto nivel escrito en Python, que fomenta el desarrollo rápido de aplicaciones web seguras y escalables. Su principal ventaja radica en su filosofía de "reutilizar y no reinventar", al ofrecer una serie de herramientas y componentes

preconfigurados que permiten a los desarrolladores centrarse en los aspectos específicos del proyecto. Entre sus características destacadas se encuentran la administración automática de bases de datos, un sistema de autenticación robusto, y medidas de seguridad integradas para proteger contra vulnerabilidades comunes. Django está especialmente diseñado para facilitar el desarrollo de aplicaciones web complejas y con altos requisitos de seguridad (Holovaty & Kaplan, 2009).

6.3.3.3 ASP.NET Core Web API.

Es un framework basado en ASP.NET Core que permite la creación de APIs RESTful (Interfaz de Programación de Aplicaciones) para facilitar la comunicación entre sistemas a través de la web. Utiliza el protocolo HTTP y los métodos estándar como GET, POST, PUT y DELETE para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los recursos expuestos. Este framework es multiplataforma, lo que permite su ejecución en Windows, macOS y Linux gracias al uso de .NET Core. ASP.NET Core Web API está diseñado para ser altamente escalable, eficiente y seguro, con soporte para autenticación, autorización y versionado de APIs. Además, incluye herramientas como Swagger para la documentación automática, facilitando el desarrollo y mantenimiento de APIs en arquitecturas modernas, como aplicaciones basadas en microservicios. (Microsoft, 2025).

6.3.4 Bases de datos

6.3.4.1 PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional de código abierto. Es reconocido por su capacidad para manejar grandes volúmenes de datos y soportar transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). PostgreSQL permite la manipulación de datos complejos a través de un sistema de extensiones que permiten definir tipos

de datos personalizados, lo que lo convierte en una opción preferida para proyectos que requieren flexibilidad y rendimiento. También es conocido por su cumplimiento con los estándares SQL y su alta capacidad de personalización (Hernández & González, 2018).

6.3.4.2 Microsoft SQL Server.

SQL Server es un sistema de gestión de bases de datos relacionales desarrollado por Microsoft, diseñado para gestionar y almacenar grandes volúmenes de datos de forma segura y eficiente.SQL Server permite consultas complejas y manipulación de datos utilizando el lenguaje de consulta estructurado (SQL), el lenguaje estándar para interactuar con bases de datos relacionales.Este sistema es ampliamente utilizado en aplicaciones empresariales, tanto locales como en la nube, para gestionar datos críticos de manera confiable y con alto rendimiento (rwestMSFT & MikeRayMSFT, 2024).

Server SQL proporciona una variedad de herramientas avanzadas, como seguridad mejorada, replicación, análisis de datos y soporte de inteligencia empresarial, lo que permite a las organizaciones gestionar y obtener valor de sus datos.También incluye funciones de alta disponibilidad y recuperación ante desastres, lo que garantiza continuidad y confiabilidad en entornos críticos (rwestMSFT & MikeRayMSFT, 2024).

6.3.4.3 SQLite.

SQLite es una biblioteca de software que proporciona un sistema de administración de bases de datos relacional, autocontenido, sin servidor y de configuración mínima. A diferencia de otros sistemas de bases de datos que requieren un proceso de servidor separado, SQLite se integra directamente en la aplicación que lo utiliza, almacenando toda la base de datos en un único archivo de disco. Esto hace que SQLite sea extremadamente ligero y fácil de implementar, siendo ideal

para aplicaciones móviles, dispositivos embebidos y sistemas de almacenamiento local (SQLite, n.d.).

Además, SQLite es compatible con SQL y permite realizar operaciones estándar de bases de datos como consultas, inserciones y actualizaciones de manera eficiente. Al ser de código abierto, SQLite es ampliamente utilizado y adaptable para diversas aplicaciones, desde proyectos personales hasta sistemas comerciales (SQLite, n.d.).

6.3.5 Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente ligero, pero potente, desarrollado por Microsoft. Es un entorno de desarrollo integrado (IDE) gratuito y de código abierto, diseñado para facilitar la escritura, depuración y gestión de código en diversos lenguajes de programación. VS Code es altamente personalizable a través de extensiones que permiten añadir soporte para lenguajes adicionales, herramientas de desarrollo, y frameworks. Se destaca por su compatibilidad con una amplia gama de lenguajes, como JavaScript, Python, C#, Java, PHP, y muchos más. Además, proporciona características avanzadas como la depuración integrada, control de versiones con Git, y la capacidad de trabajar de manera remota mediante su integración con contenedores Docker o conexiones SSH. Su ligereza y capacidad de personalización lo convierten en una herramienta muy popular entre los desarrolladores, especialmente aquellos que buscan un editor rápido y flexible para proyectos pequeños o grandes (Microsoft, n.d.).

6.4 Justificación de las tecnologías seleccionadas

6.4.1 Flutter

Flutter fue seleccionado como framework de desarrollo móvil debido a su capacidad de construir aplicaciones altamente eficientes y con interfaces de usuario modernas utilizando un único lenguaje de programación (Dart).

Su motor de renderizado propio permite un control completo sobre cada píxel de la pantalla, garantizando un rendimiento óptimo incluso en dispositivos especializados como terminales PDA basadas en Android, donde se prioriza la velocidad y la eficiencia en entornos de ventas rápidas.

Además, Flutter proporciona una arquitectura altamente modular basada en widgets reutilizables, lo que facilita el mantenimiento, la escalabilidad y el desarrollo ágil de nuevas funcionalidades del sistema POS.

La elección de Flutter responde también a la necesidad de construir soluciones adaptativas capaces de trabajar en resoluciones de pantalla diversas sin sacrificar la experiencia del usuario, requisito crítico en el ámbito de dispositivos móviles industriales

6.4.2 ASP.NET Core Web API

ASP.NET Core Web API fue elegido como tecnología para el desarrollo del backend del sistema debido a su naturaleza multiplataforma, su alto rendimiento y su capacidad para construir servicios web RESTful robustos y seguros.

Esta tecnología permite implementar APIs ligeras y escalables, esenciales para manejar eficientemente la comunicación entre las aplicaciones móviles y los servidores centrales, incluso en entornos con limitaciones de ancho de banda o infraestructura variable.

ASP.NET Core proporciona además integraciones avanzadas para manejo de autenticación, autorización, encriptación de datos y validación de modelos, lo cual garantiza un alto nivel de seguridad en las operaciones de transmisión de datos entre clientes y servidor. Su compatibilidad nativa con arquitecturas de microservicios y su eficiencia en escenarios de alta concurrencia aseguran un crecimiento sostenible del sistema en el futuro.

6.4.3 C#

C# fue el lenguaje de programación elegido para el desarrollo de las capas de negocio y de acceso a datos del backend debido a su madurez, expresividad y sólida integración con el ecosistema .NET.

C# ofrece características modernas como programación asíncrona nativa, manejo eficiente de excepciones, y una sintaxis clara que favorece la mantenibilidad del código a largo plazo. Además, proporciona potentes herramientas para la conexión segura a bases de datos, consumo de APIs externas y gestión de la lógica de negocio compleja, todo dentro de un marco que prioriza la robustez y la seguridad de las aplicaciones empresariales.

La utilización de C# garantiza además una fácil interoperabilidad entre distintas capas del sistema y un soporte extensivo por parte de la comunidad técnica y académica.

6.4.4 SQL Server

SQL Server fue adoptado como base de datos central en el servidor debido a su capacidad para gestionar grandes volúmenes de información de manera segura, eficiente y escalable.

Esta plataforma ofrece funcionalidades avanzadas como replicación de datos, procedimientos almacenados, transacciones ACID, y herramientas de monitoreo que permiten garantizar la consistencia y disponibilidad de los datos críticos en ambientes empresariales.

SQL Server asegura también una integración fluida con servicios de backup, recuperación ante desastres y reportes analíticos, aspectos fundamentales para la operación continua de un sistema POS que maneja información sensible como ventas, cierres de caja, arqueos y datos de clientes.

Su soporte a estándares de seguridad como el cifrado de datos en reposo y en tránsito refuerzan la elección como repositorio de datos principal.

6.4.5 SQLite

SQLite fue implementado como solución de almacenamiento local en el dispositivo móvil para permitir el funcionamiento del sistema en modo offline, asegurando la continuidad de las operaciones aún en ausencia de conectividad.

SQLite destaca por su ligereza, fiabilidad y facilidad de implementación dentro de aplicaciones móviles, permitiendo manejar de manera eficiente transacciones locales como la creación de ventas, pedidos, registros de movimientos de dinero y clientes nuevos, sin requerir de infraestructura adicional.

La estructura embebida de SQLite elimina la necesidad de configuración externa, optimizando los tiempos de instalación y operación en ambientes móviles, y garantizando además la integridad de los datos mediante soporte nativo a transacciones seguras.

Posteriormente, cuando la conectividad está disponible, el sistema permite la sincronización manual de los datos almacenados localmente hacia el servidor central.

6.4.6 Visual Studio Code (VSC)

Visual Studio Code (VSC) fue utilizado como entorno principal de desarrollo de la aplicación móvil debido a su versatilidad, ligereza y extensibilidad.

VSC proporciona un conjunto amplio de extensiones que facilitan la programación en Flutter y Dart, como asistentes de depuración, autocompletado inteligente (IntelliSense), control de versiones integrado con Git, y emuladores de dispositivos móviles, optimizando así el flujo de trabajo de desarrollo ágil.

Además, su bajo consumo de recursos en comparación con entornos más pesados permite un desarrollo rápido y fluido en equipos de hardware medio, característica relevante en ambientes de desarrollo empresarial.

Su adopción garantiza también compatibilidad con prácticas de programación modernas, integración continua (CI), y soporte activo de una amplia comunidad de desarrolladores.

6.5 Alineación estratégica de las tecnologías seleccionadas

Adicionalmente, la selección de las tecnologías mencionadas se encuentra alineada con la estrategia tecnológica vigente del Grupo APL Ingeniería LTDA.

Dicha alineación no solo favorece la compatibilidad e interoperabilidad con los sistemas y plataformas existentes dentro de la organización, sino que también facilita la integración de nuevos desarrollos y módulos complementarios de forma ordenada y coherente.

Esta decisión estratégica permite optimizar los procesos de despliegue, soporte técnico y mantenimiento a largo plazo, minimizando los costos asociados a la adopción de nuevas tecnologías y garantizando la continuidad operativa del ecosistema tecnológico empresarial.

La estandarización en el uso de herramientas consolidadas refuerza, además, la sostenibilidad del proyecto en el tiempo y permite aprovechar de manera eficiente los recursos humanos y técnicos actualmente disponibles en la empresa.

7. Arquitectura del Sistema

El sistema se basa en una arquitectura cliente-servidor, en la cual la aplicación móvil actúa como cliente, comunicándose con un servidor central a través de una API REST. Esta estructura permite gestionar transacciones y sincronizar datos de manera controlada, garantizando la integridad, disponibilidad y seguridad de la información.

El sistema adopta un enfoque híbrido de operación, diferenciando los módulos de inventario y de punto de venta (POS) en cuanto al tratamiento de la conectividad:

- En el módulo de inventarios, cuando existe conexión a Internet, las consultas se realizan directamente en tiempo real a la base de datos central a través de la API REST, asegurando información actualizada.
- En el módulo POS, las operaciones como ventas, pedidos, movimientos de dinero, arqueos y cierres de caja se almacenan inicialmente de manera local en una base de datos SQLite. Posteriormente, el usuario debe ejecutar manualmente la sincronización de esta información mediante la funcionalidad "Subir Información" para consolidar los datos en la base de datos central.

Este diseño asegura la operatividad continua del sistema, incluso en entornos con conectividad limitada o intermitente, optimizando la experiencia del usuario y minimizando el riesgo de pérdida de información.

7.1 Modelo de arquitectura Cliente - Servidor

La arquitectura cliente-servidor es una de las más utilizadas en sistemas de punto de venta (POS) y gestión de inventarios, ya que permite que múltiples dispositivos accedan a una base de datos centralizada sin comprometer la integridad de la información. En este modelo, la aplicación móvil actúa como cliente, enviando y recibiendo datos mediante peticiones HTTP a una API desarrollada en ASP.NET Core, la cual gestiona las transacciones y almacena la información en una base de datos centralizada en SQL Server.

7.1.1 Justificación de la elección cliente servidor

- **Centralización de la Información**

  La arquitectura cliente-servidor permite que los datos se almacenen y gestionen en un servidor centralizado, lo que garantiza la consistencia y actualización de la información.

Esta centralización facilita la administración y el mantenimiento de la base de datos, asegurando que todos los clientes accedan a una fuente única de información veraz (IEEE, 2022).

- **Seguridad Mejorada**

  Al concentrar la gestión de datos en un servidor, es posible implementar medidas de seguridad más robustas, como firewalls y controles de acceso, protegiendo la información contra accesos no autorizados. Esto es esencial en sistemas que manejan datos sensibles, como transacciones financieras y registros de inventario (IEEE, 2022).

- **Escalabilidad y Mantenimiento Eficiente**

  La arquitectura cliente-servidor facilita la escalabilidad del sistema, permitiendo agregar o actualizar componentes sin interrumpir el funcionamiento general. Por ejemplo, se pueden mejorar las capacidades del servidor para manejar un mayor número de clientes o incrementar el volumen de datos sin afectar a las aplicaciones cliente. Además, las actualizaciones y el mantenimiento se realizan centralmente en el servidor, simplificando la gestión y reduciendo costos operativos (IEEE, 2022).

- **Integración y Compatibilidad**

  Este modelo arquitectónico facilita la integración con otros sistemas y tecnologías, permitiendo que diferentes plataformas y dispositivos interactúen de manera eficiente. Esto es especialmente beneficioso en entornos empresariales donde coexisten múltiples

sistemas que requieren compartir información y funcionalidades (IEEE, 2022).

- **Rendimiento Optimizado**

  Al delegar tareas específicas al servidor (como consultas complejas a la base de datos) y otras al cliente (como la presentación de la interfaz de usuario), se optimiza el uso de recursos y se mejora el rendimiento general del sistema. Esta distribución de responsabilidades permite una respuesta más rápida a las solicitudes de los usuarios y una experiencia más fluida (IEEE, 2022).

- **Gestión Eficiente del Inventario**

  En el contexto de la gestión de inventarios, una arquitectura cliente-servidor permite la trazabilidad en tiempo real de las existencias, facilitando la toma de decisiones informadas sobre compras y ventas. La centralización de datos permite un seguimiento preciso de las existencias y una gestión más eficiente de los almacenes (IEEE, 2022).

7.2 Componentes del sistema

El sistema se divide en tres componentes principales, cada uno con una función específica dentro del flujo de trabajo del sistema POS y gestión de inventarios. Estos componentes interactúan entre sí utilizando una arquitectura cliente-servidor, garantizando que los datos sean accesibles de manera centralizada y en tiempo real.

7.2.1 Cliente (Aplicación Móvil)

Tecnología utilizada: Flutter (Dart).

Rol en el sistema: Interfaz de usuario para los módulos de punto de venta (POS) y gestión de inventarios.

Funciones clave:

- Escaneo de productos mediante la cámara del dispositivo.
- Generación de facturas y pedidos en formato digital.
- Registro de nuevos clientes y movimientos de dinero.
- Consulta de inventarios en línea.
- Almacenamiento local de datos de transacciones mediante SQLite.
- Sincronización manual de los datos transaccionales utilizando la funcionalidad "Subir Información".

Descripción:

La aplicación móvil constituye la herramienta principal de operación para los usuarios en los puntos de venta.

En el módulo de inventarios, cuando existe conectividad disponible, las consultas sobre productos y existencias se realizan en tiempo real a través de la API REST, garantizando información actualizada directamente desde la base de datos central.

En cambio, en el módulo POS (ventas, pedidos, movimientos de dinero, arqueo y cierre de caja), todas las operaciones se almacenan inicialmente de forma local en una base de datos SQLite, independientemente de la disponibilidad de conexión.

Posteriormente, el usuario debe realizar la sincronización manual de estos datos mediante la funcionalidad "Subir Información" para consolidarlos en la base de datos central.

Esta estrategia asegura que el sistema POS pueda seguir operando de forma ininterrumpida aún en entornos con conectividad limitada o inestable.

7.2.2 Servidor (Backend y API REST)

Tecnología utilizada: ASP.NET Core Web API (C#).

Rol en el sistema: Procesamiento de datos, autenticación de usuarios y gestión de transacciones.

Funciones clave:

- Manejo de autenticación y control de accesos de usuarios.
- Procesamiento de operaciones de venta y pedidos.
- Gestión y sincronización de inventarios y clientes.
- Exposición de endpoints RESTful para comunicación con los clientes móviles.
- Almacenamiento persistente en base de datos centralizada.

Descripción:

El servidor central funciona como intermediario entre los dispositivos móviles y la base de datos SQL Server.

Su principal función es gestionar la lógica de negocio, validar transacciones, administrar la autenticación de usuarios y procesar las solicitudes recibidas de las aplicaciones móviles. Mediante la API REST, el servidor permite operaciones seguras como consultas de inventario,

actualizaciones de datos de clientes y registros de ventas, retornando respuestas estructuradas en formato JSON para asegurar compatibilidad y eficiencia en la transferencia de datos.

7.2.3 Base de Datos y Sincronización

Tecnologías utilizadas: SQL Server y SQLite.

Rol en el sistema: Almacenamiento y gestión de datos transaccionales.

Descripción:

El sistema utiliza una estrategia de gestión de datos diferenciada según el módulo operativo:

- **SQL Server**:

  Opera como base de datos centralizada en el servidor, alojando la información consolidada del negocio: productos, inventarios, ventas, clientes y movimientos de caja. El módulo de inventarios permite consultas en tiempo real a SQL Server mediante la API REST, garantizando disponibilidad de información actualizada cuando la conexión a Internet está disponible.

- **SQLite**:

  Actúa como base de datos local embebida en los dispositivos móviles.

  En el módulo POS, todas las operaciones transaccionales (ventas, pedidos, registros de clientes, movimientos de dinero, arqueos y cierres de caja) se almacenan inicialmente en SQLite, independientemente del estado de la conectividad.

Modelo de sincronización:

Los datos almacenados localmente en SQLite deben ser sincronizados manualmente mediante la funcionalidad "Subir Información".

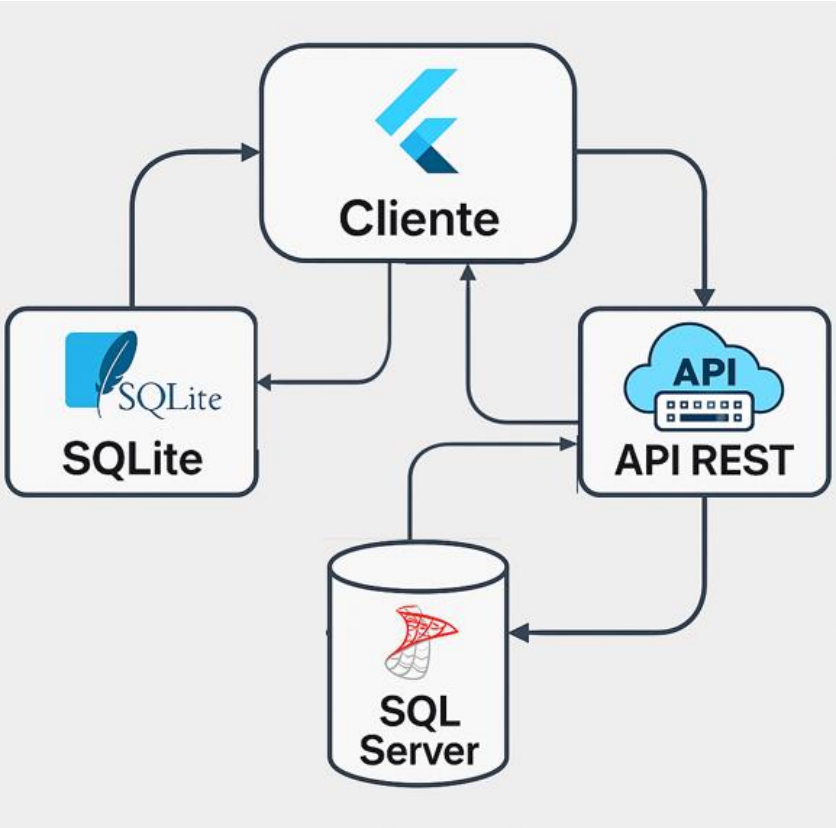
Esta sincronización asegura que todas las operaciones realizadas offline sean transmitidas posteriormente al servidor para su consolidación en SQL Server, preservando así la integridad y la disponibilidad de los datos a nivel central.

Esta arquitectura híbrida permite garantizar tanto la operatividad offline del sistema POS como el acceso en línea a datos críticos de inventario.

7.3 Diagrama de Arquitectura

Figura 1

Flujo donde Flutter interactúa con una base de datos local SQLite y una API REST, la cual a su vez se conecta con una base de datos SQL Server.



Nota. Flujo de interacción entre Flutter, SQLite, API REST y SQL Server.

8. Desarrollo del sistema

8.1 Introducción al módulo POS

El módulo de Punto de Venta (POS) es una parte esencial del sistema, diseñado para facilitar y agilizar el proceso de facturación y cobro en establecimientos comerciales. Su función

principal es permitir la gestión de transacciones de venta, integrando funcionalidades como el escaneo de productos, la generación de facturas digitales y la sincronización con el inventario. Además, el sistema permite la impresión de facturas a través de conexión Bluetooth, facilitando la entrega de comprobantes físicos a los clientes.

Este módulo está diseñado para facilitar la operatividad en los puntos de venta, contribuyendo a una organización más eficiente de las transacciones y a una gestión estructurada de la información comercial.

Estado del Módulo Antes de la Intervención

Antes de la implementación de mejoras en el sistema, el módulo POS ya había sido parcialmente desarrollado, con los siguientes aspectos implementados:

- Interfaz gráfica básica: Existía un diseño preliminar de la aplicación en Flutter, con algunas pantallas funcionales.
- Conexión con el backend: Se había implementado una API en ASP.NET Core, permitiendo la consulta a la base de datos central en SQL Server.
- Base de datos central: La información se almacenaba en SQL Server, con una estructura de tablas definidas.
- Sincronización con SQLite: Aún no se había implementado una sincronización con bases de datos locales en dispositivos móviles, lo que generaba dependencia constante de la conexión al servidor.
- Funcionalidades iniciales: Algunas operaciones básicas, como el login , creación de usuarios, sincronización de datos de SQL Server, movimientos de dinero y arqueos ya estaban implementadas.

Objetivo de la Intervención

Dado el estado previo del sistema, la intervención se centró en comprender y mejorar la estructura del sistema con el fin de facilitar la implementación de nuevas funcionalidades del módulo. Los objetivos principales de esta fase incluyen:

- Analizar la lógica de negocio existente para comprender los procesos de venta y facturación.
- Revisar y entender la estructura de la base de datos, asegurando una correcta integración con la aplicación móvil.
- Seleccionar una librería eficiente para el escaneo de productos, garantizando la mejor compatibilidad y rendimiento dentro del entorno Flutter.
- Desarrollar y optimizar las vistas en Flutter, asegurando que reflejen adecuadamente la lógica de negocio y los procesos del sistema.
- Implementar la sincronización de datos con SQLite, permitiendo que la aplicación pueda funcionar sin conexión constante al servidor, mejorando la disponibilidad de los datos en escenarios de conectividad limitada.
- Realizar pruebas unitarias para verificar el correcto funcionamiento del módulo y su integración con los demás componentes del sistema.

Esta intervención permitió estructurar el desarrollo de manera más organizada y mejorar la cohesión del sistema POS con el resto de la arquitectura del software, asegurando su funcionamiento en distintos entornos operativos.

8.2 Análisis y comprensión del sistema existente

8.3 Selección de librería para escaneo de productos

El escaneo de códigos de barras es esencial en los sistemas de punto de venta (POS), ya que permite la identificación rápida y precisa de los productos, facilitando el proceso de facturación y la actualización automática del inventario. La eficiencia en el escaneo depende de varios factores, incluyendo el rendimiento del hardware utilizado y la eficacia del software de escaneo implementado, especialmente cuando se emplean dispositivos especializados como las PDAs.

Análisis de Opciones Disponibles en Flutter

Para la implementación del escaneo de códigos en el sistema POS desarrollado con Flutter, se evaluaron diversas librerías disponibles en pub.dev, el repositorio oficial de paquetes para Flutter. A continuación, se presentan las principales opciones consideradas:

- flutter_barcode_scanner

  Esta librería permite la lectura de códigos de barras 1D y 2D y es conocida por su simplicidad y facilidad de uso. Está diseñada para integrar rápidamente la funcionalidad de escaneo de códigos en aplicaciones Flutter. Sin embargo, presenta limitaciones en el rendimiento en condiciones de iluminación subóptimas, lo que puede afectar su eficacia en entornos con poca luz (Gangadhare, 2022).

- qr_code_scanner

  Esta librería está especializada en la lectura de códigos QR y proporciona un excelente rendimiento y precisión. Sin embargo, su soporte para otros tipos de códigos, como los

códigos de barras estándar (EAN, UPC), es limitado. Está bien documentada y es fácil de integrar, pero su uso es más adecuado para aplicaciones que requieren exclusivamente escaneo de códigos QR (JuliusCanute, 2023).

- zxing_flutter

  Basada en la tecnología ZXing (Zebra Crossing), esta librería es compatible con varios tipos de códigos de barras y QR. La librería proporciona un alto nivel de precisión y está diseñada para aplicaciones que requieren un alto rendimiento. Sin embargo, su integración con Flutter es más compleja en comparación con otras librerías, lo que puede aumentar el tiempo de implementación en proyectos con tiempos ajustados (Khoren, 2025).

- mobile_scanner

  mobile_scanner es la librería seleccionada para el proyecto debido a su alto rendimiento y facilidad de integración con Flutter. Esta librería aprovecha la API nativa de la cámara para Android e iOS, lo que permite un escaneo rápido y eficiente de códigos 1D y 2D. Además, se adapta bien al hardware especializado de los dispositivos PDA, asegurando un buen rendimiento incluso en entornos con condiciones de luz difíciles (Steenbakker, 2025).

Criterios de Selección

La elección de la librería más adecuada se basó en los siguientes criterios:

- Compatibilidad con Flutter y dispositivos PDA: Se priorizó la librería que mejor aprovechara el hardware especializado en el escaneo de códigos en dispositivos PDA, como mobile_scanner.

- Rendimiento en condiciones de iluminación variadas: La capacidad para mantener un buen rendimiento incluso en condiciones de baja luz fue fundamental para asegurar la funcionalidad en diversos entornos comerciales.
- Facilidad de integración: Se consideró la simplicidad de integración con el sistema POS y la documentación disponible, lo que facilitó la adopción de mobile_scanner sin complicaciones adicionales.
- Soporte para múltiples formatos de códigos: mobile_scanner destacó por su capacidad para manejar tanto códigos de barras tradicionales como códigos QR, lo que aumentó su flexibilidad.
- Consumo de recursos y estabilidad: La librería mobile_scanner fue preferida por su eficiencia en el uso de recursos, lo que permitió un escaneo fluido sin afectar el rendimiento general de la aplicación.

Librería Elegida y Justificación

La librería seleccionada para el sistema POS fue mobile_scanner, debido a su excelente rendimiento en la detección de códigos y su compatibilidad con dispositivos PDA. Su integración con Flutter fue directa y bien soportada, lo que permitió incorporar rápidamente la funcionalidad de escaneo al sistema POS. Además, su capacidad para manejar diversos tipos de códigos y su bajo consumo de recursos la hicieron la opción ideal para un sistema que necesita ser eficiente y rápido en un entorno comercial.

8.4 Implementación de la base de datos SQLite para funcionalidad offline del Sistema POS

1. ConfigurationTable

Propósito: Esta tabla almacena la configuración de conexión, específicamente la IP y el puerto para conectar al servidor.

Campos:

- Id: Identificador único de la configuración (autoincrementable). Esto se usa para distinguir las distintas configuraciones si alguna vez se añaden más configuraciones al sistema.

- Ip: La dirección IP que el sistema debe usar para conectarse al servidor.

- Port: El puerto en el que el servidor está escuchando para las conexiones.

Uso:

- Esta tabla es útil cuando se necesita gestionar configuraciones de red de manera centralizada y es común para sistemas distribuidos, donde la IP y el puerto del servidor pueden cambiar según el entorno.

2. ImpresoraTable

Propósito: Esta tabla almacena las impresoras configuradas para el sistema.

Campos:

- Id: Identificador único (autoincrementable) de la impresora.

- Name: Nombre de la impresora, lo que facilita su identificación dentro del sistema.

- Address: Dirección o identificador de la impresora, que podría ser una dirección IP o una dirección de puerto dependiendo de cómo se conecte la impresora.

Uso:

- Esta tabla es utilizada por el sistema POS para hacer referencia a una impresora cuando sea necesario realizar una impresión de una factura o recibo.

3. ClientesNuevosTable

Propósito: Almacena información detallada sobre nuevos clientes que se registran en el sistema.

Campos:

- Id: Identificador único de cliente (autoincrementable).

- NoDocumento: Número de documento de identidad del cliente (cédula, pasaporte, etc.).

- Procesada: Indicador que señala si la información del cliente ha sido procesada (1 para procesado, 0 para no procesado).

- TipoDeDocumentoId: Tipo de documento utilizado para la identificación del cliente (cédula, pasaporte, etc.).

- NoIdentificacion: Número de identificación del cliente.

- NombreCompleto: Nombre completo del cliente.

- PrimerNombre, PrimerApellido, SegundoNombre, SegundoApellido: Componentes del nombre y apellido del cliente.

- Direccion: Dirección física del cliente.

- Telefono, Telefono3: Números de teléfono (teléfono fijo, móvil, etc.).

- Email: Correo electrónico del cliente.

- Genero: Género del cliente (por ejemplo, 0 para masculino, 1 para femenino).

- FechaNacimiento: Fecha de nacimiento del cliente.

- PaisId, CiudadId, DepartamentoId, BarrioId: Identificadores para la ubicación geográfica del cliente.

- TipoPersona: Identifica si el cliente es persona natural o jurídica.

- RegimenId: Identificador del régimen fiscal del cliente.
- TipoDeContribuyenteId: Tipo de contribuyente según las leyes fiscales.
- CiudadDeExpedicionId: Ciudad en la que se expidió el documento del cliente.
- RutaId: Identificador de la ruta de distribución o envío.
- Alias: Alias o nombre alternativo para referirse al cliente.
- EquipoId: Identificador del equipo asociado al cliente.
- UsuarioId: Identificador del usuario asociado al cliente.
- DigitoDeChequeo: Dígito de verificación del cliente (seguridad).
- RazonSocial: Razón social del cliente si es persona jurídica.
- Representante: Nombre del representante legal, si es persona jurídica.
- FechaDeSistema: Fecha en que el cliente fue registrado en el sistema.

- TipoListaPreciosId: Identificador del tipo de lista de precios que se aplica al cliente.

- TipoClienteId: Tipo de cliente (puede ser frecuente, preferencial, etc.).

- CodigoActividad: Código de la actividad económica del cliente.

- RegimenFiscalId: Identificador del régimen fiscal aplicable al cliente.

- ResponsabilidadTributariaId: Identificador de la responsabilidad tributaria del cliente.

Uso:

- Esta tabla es esencial para almacenar toda la información relacionada con clientes nuevos, lo que permite realizar un seguimiento eficiente de sus datos y facilitar transacciones futuras.

4. IMEITable

Propósito: Guarda los números IMEI de los dispositivos móviles.

Campos:

- Id: Identificador único del registro (autoincrementable).

- IMEI: El número IMEI único del dispositivo móvil.

Uso:

- Esta tabla es útil para registrar los dispositivos móviles en el sistema y garantizar que la aplicación sea utilizada únicamente por dispositivos del personal autorizado.

5. SalidasDeMercanciaTerceroTable

Propósito: Registra las salidas de mercancía realizadas a terceros (por ejemplo, a clientes o distribuidores).

Campos:

- SalidaId: Identificador único de la salida de mercancía.
- NoDocumento: Número de documento asociado con la salida de mercancía.
- Procesada: Estado de procesamiento (1 si se ha procesado, 0 si no).
- TipoDeDocumentoId: Tipo de documento (por ejemplo, factura, nota de crédito).
- NoIdentificacion: Número de identificación del tercero.
- CiudadDeExpedicionId: Ciudad de expedición.
- NombreCompleto: Nombre completo del tercero.

- PrimerNombre, PrimerApellido, SegundoNombre, SegundoApellido: Componentes del nombre y apellido.

- Alias: Alias o nombre alternativo del tercero.

- Direccion: Dirección de envío.

- Telefono, Telefono3: Números de teléfono del tercero.

- Email: Correo electrónico del tercero.

- Genero: Género (0 para masculino, 1 para femenino).

- FechaNacimiento: Fecha de nacimiento (opcional).

- PaisId, DepartamentoId, CiudadId, BarrioId: Identificadores para la ubicación.

- RutaId: Ruta de distribución asociada.

- TipoPersona: Tipo de persona (natural o jurídica).

- RegimenId: Regimen fiscal asociado.

- TipoDeContribuyenteId: Tipo de contribuyente.

- EquipoId: Identificador del equipo asociado.

- UsuarioId: Usuario que realizó la transacción.

- DigitoDeChequeo: Dígito de verificación (si aplica).

- RazonSocial: Razón social en caso de ser persona jurídica.

- Representante: Representante legal si aplica.

- RegimenFiscalId: Identificador del régimen fiscal.

- ResponsabilidadTributariaId: Responsabilidad tributaria.

Uso:

- Esta tabla se utiliza para registrar las salidas de mercancías realizadas hacia terceros y es clave para el control de inventarios y la gestión de envíos.

6. SalidasDeMercanciaTable

Propósito: Almacena las salidas de mercancía en general, como ventas, distribuciones o devoluciones. Es una tabla central para controlar los movimientos de inventario.

Campos:

- SalidaId: Identificador único de la salida de mercancía.
- NoDocumento: Número de documento de la salida.
- Procesada: Estado de procesamiento de la salida.
- EmpresaId, BodegaId, CentroDeCostoId: Identificadores de la empresa, bodega y centro de costo.
- VendedorId, MotivoId, DomiciliarioId: Identificadores del vendedor, motivo de la salida y domiciliario.
- NoCotizacion: Número de cotización relacionada con la salida.
- FechaDeSalida: Fecha en la que se realizó la salida.
- HoraDeSalida: Hora en que se efectuó la salida.

- TotalSalida, TotalUtilidad, TotalCancelo, TotalPuntos, TotalAhorro, TotalDescuento: Totales relacionados con la salida, utilidad, pago y puntos.

- TotalDomicilioTercero, TotalCredito: Totales de costos relacionados con el domicilio de terceros y créditos.

- Resolucion: Resolución de la transacción.

- Observacion: Observaciones adicionales sobre la salida.

- EquipoId, UsuarioId: Identificadores del equipo y usuario asociados.

- FechaDeSistema: Fecha en la que el sistema registró la salida.

- EstadoApartado, EstadoCotizacion: Estado de la salida relacionada con el apartado y cotización.

- TipoDevolucion, ModoDevolucion, ModoEnSa: Información relacionada con devoluciones y modos de entrega.

- TipoDocumento: Tipo de documento asociado (por ejemplo, factura, remisión).

- Discriminator: Campo utilizado para identificar el tipo de salida.

- NoFacturaDevolucion: Número de factura en caso de devolución.

- EstadoPedido, RutaPedidoId: Estado del pedido y su ruta asociada.

Uso:

- Esta tabla es la base para registrar las salidas de mercancía, calcular los totales de ventas y gestionar aspectos como devoluciones, cotizaciones y facturación.

7. SalidasDeMercanciaDetalleTable

Propósito: Almacena los detalles de cada salida de mercancía. Esto incluye información sobre los productos específicos involucrados en la transacción, como el precio, la cantidad y otros detalles relacionados con cada ítem.

Campos:

- NoDocumento: Número de documento asociado a la salida.

- Id: Identificador único del detalle de la salida.

- ProductoId: Identificador del producto.

- CodigoAlterno: Código alternativo del producto (si existe).

- DescripcionLarga: Descripción larga del producto.
- DescripcionCorta: Descripción corta del producto.
- Embalaje: Tipo de embalaje del producto.
- IvaVentaId, IvaCompraId: Identificadores del IVA para la venta y la compra.
- ImpoConsumo: Valor del impuesto al consumo asociado.
- PrecioCosto: Precio de costo del producto.
- PrecioPublico: Precio de venta pública del producto.
- PrecioPublicoReal: Precio real de venta del producto.
- CantidadEntrada, CantidadSalida: Cantidades de entrada y salida del producto.
- BasePuntos: Puntos base asociados a la venta del producto.
- NoPuntos: Número de puntos asociados.

- TotalPuntos: Total de puntos acumulados.

- D1, D2, D3, D4, D5: Campos adicionales para datos de control o atributos específicos.

- TotalRegistro: Total registrado para el detalle.

- NoRemision, NoApartado: Número de remisión y apartado asociados (si aplican).

- Observacion: Observaciones sobre el detalle de la salida.

- CanalId, ListaId, ZonaId, EventoId, VendedorId: Identificadores asociados a los canales de venta, lista de precios, zona, evento y vendedor.

- TipoDocumento: Tipo de documento relacionado.

- TarifaIvaVenta, TarifaIvaCompra: Tarifa de IVA aplicada tanto para venta como para compra.

- PrecioVenta: Precio de venta del producto.

- AplicaPuntos: Indica si el producto aplica para acumular puntos.

- CantidadDpc: Cantidad de unidades de producto por contenedor (si aplica).

- DescuentoProducto: Descuento aplicado al producto.

- Ahorro: Ahorro asociado al producto.

- MostrarDescuento: Indica si el descuento debe mostrarse.

- VenderXPeso, VenderXFraccion: Indicadores de si el producto se vende por peso o fracción.

- NoManejaInventario: Indica si el producto no maneja inventario.

- EsConjunto, TieneLote, TieneSerial, EsServicio, EsProduccion, EsAncheta, EsConcesion, EsObsequio: Indicadores de características especiales del producto (conjunto, lote, serial, servicio, producción, etc.).

- PerteneceAsociacion, Productoweb, EsBolsa, Interno: Indicadores relacionados con la pertenencia a asociaciones, si es un producto web, si es una bolsa, o si es interno.

- Fecha: Fecha del detalle de la salida.

- UsuarioId, EquipoId: Identificadores del usuario y equipo asociados a la salida.

- TipoLiquidacionId: Identificador de la liquidación asociada.

- NoCotizacion, NoPedido, NoFacturaTemporal: Número de cotización, pedido o factura temporal asociados.

- AplicaGrupoDeCosto, NoAplicaRedondeo: Indicadores de aplicación de grupos de costo y redondeo.

- EsInsumo: Indica si el producto es un insumo.

- Id1, Id2: Campos adicionales para identificadores asociados.

- SalidaId: Identificador de la salida de mercancía asociada.

- TipoListaPreciosId: Identificador del tipo de lista de precios aplicada.

- EsKit: Indica si el producto es un kit.

- ImpuestoId: Identificador del impuesto aplicado al producto.

- TarifaIvaImpuesto: Tarifa del IVA asociada al impuesto.

Uso:

- Esta tabla es esencial para registrar los detalles de los productos involucrados en cada salida de mercancía. Permite realizar un seguimiento más granular de cada ítem, sus precios, descuentos y otros atributos.

8. SalidasDeMercanciaComboTable

Propósito: Almacena información relacionada con las combinaciones de productos o combos dentro de las salidas de mercancía.

Campos:

- SalidaId: Identificador único de la salida de mercancía.

- Id: Identificador único del combo.

- ProductoComboId: Identificador del combo de productos.

- NoDocumento: Número de documento relacionado con la salida.

- ProductoId: Identificador del producto asociado al combo.

- ValorIvaVenta, PrefijoIvaVenta, ValorIvaCompra, PrefijoIvaCompra: Valores y prefijos relacionados con el IVA de venta y compra.

- PrecioCosto, PrecioCostoPromedio: Precios asociados al costo de los productos en el combo.

- PrecioPublico: Precio de venta del combo.

- Cantidad: Cantidad de combos incluidos en la salida.

Uso:

- Esta tabla se utiliza para registrar los productos que se agrupan en combos dentro de las salidas de mercancía. Permite calcular el valor total de un combo, considerando los productos que lo componen.

9. SalidasDeMercanciaAuditoriaTable

Propósito: Esta tabla almacena los registros de auditoría de las salidas de mercancía. Está diseñada para seguir los cambios y procesos que se aplican a cada salida, permitiendo el seguimiento detallado de cualquier modificación realizada en los documentos o productos asociados a las salidas.

Campos:

- SalidaId: Identificador único de la salida de mercancía.

- AuditoriaId: Identificador único de la auditoría.

- NoDocumento: Número de documento de la salida de mercancía que fue auditado.

- ProductoId: Identificador del producto relacionado con la auditoría.

- Id: Identificador del registro específico de auditoría.

- Proceso: Identificador del proceso realizado (puede estar relacionado con el tipo de cambio que se hizo).

- TipoDocumento: Tipo de documento relacionado con la auditoría (por ejemplo, factura, remisión).

- ModificadorFactura: Indica si hubo algún cambio en la factura relacionada con la salida de mercancía.

- ValorAnterior: Valor anterior antes de que se realizara alguna modificación o cambio.

- ValorActual: Valor actualizado después del cambio.

- Observacion: Notas o comentarios adicionales sobre la auditoría.

- Fecha: Fecha en que se registró la auditoría.

- UsuarioId: Identificador del usuario que realizó la auditoría.

- EquipoId: Identificador del equipo que utilizó el usuario para realizar la auditoría.

- Procesada: Estado que indica si la auditoría fue procesada (1 para procesada, 0 para no procesada).

- NoDocumentoAuditoria: Número de documento específico de la auditoría.

10. PermisosAuditoriaTable

Propósito: Almacena los registros de auditoría relacionados con los permisos de los usuarios, indicando qué permisos han sido asignados, modificados o auditados en el sistema.

Campos:

- AuditoriaId: Identificador único de la auditoría.

- EquipoId: Identificador del equipo donde se registró la auditoría.

- MenuId: Identificador del menú al cual pertenece el permiso.

- SubMenuId: Identificador del submenú asociado al menú.

- PermisoId: Identificador del permiso relacionado.

- UsuarioId: Identificador del usuario que tiene el permiso.

- AutorizaId: Identificador del usuario que autoriza el permiso.

- EmpresaId: Identificador de la empresa asociada al permiso.

- SucursalId: Identificador de la sucursal asociada al permiso.

- Fecha: Fecha de la auditoría.

- FechaDeSistema: Fecha del sistema en que se registró el evento.

- Procesada: Estado que indica si el evento ha sido procesado.

Uso:

- Esta tabla es útil para llevar un registro detallado de los cambios en los permisos de los usuarios, permitiendo realizar auditorías sobre quién, cuándo y qué permisos fueron modificados o asignados.

11. TesoreriaTable

Propósito: Almacena los registros relacionados con las transacciones financieras de tesorería, como pagos, abonos, valores, y sus respectivas auditorías y procesamientos.

Campos:

- NoDocumento: Número de documento asociado a la transacción.
- Id: Identificador único de la transacción.
- TipoDocumento: Tipo de documento (por ejemplo, factura, recibo, etc.).
- FechaDocumento: Fecha del documento.
- PersonaId: Identificador de la persona asociada con la transacción.
- TipoDePago: Tipo de pago asociado (efectivo, tarjeta, cheque, etc.).
- Valor: Valor de la transacción.
- TipoNaturaleza: Tipo de naturaleza de la transacción (activo, pasivo, etc.).
- EmpresaId: Identificador de la empresa asociada.
- EquipoId: Identificador del equipo utilizado.
- UsuarioId: Identificador del usuario que realiza la transacción.

- FechaDeSistema: Fecha del sistema cuando se registró la transacción.

- MotivoValorId: Identificador del motivo del valor.

- EntidadId: Identificador de la entidad financiera.

- Autorizacion: Autorización relacionada con la transacción.

- Iva, BaseDevIva, Porcentaje: Valores relacionados con el IVA de la transacción.

- EntidadBancoId: Identificador del banco asociado.

Uso:

- Esta tabla permite realizar un seguimiento detallado de las transacciones financieras realizadas por la tesorería, facilitando la auditoría y gestión de los pagos y valores asociados a la empresa.

12. EntradaSalidaDineroTable

Propósito: Esta tabla maneja los registros de las entradas y salidas de dinero dentro del sistema. Está diseñada para registrar todos los movimientos de dinero que se realicen, ya sea por pagos, cobros, ingresos, egresos, etc. Estos registros son clave para llevar el control financiero.

Campos:

- NoDocumento: Número de documento asociado a la transacción.

- Valor: Monto de dinero involucrado en la transacción.

- Observacion: Notas o comentarios sobre la transacción.

- UsuarioId: Identificador del usuario que realizó la transacción.

- Fecha: Fecha en la que se registró la transacción.

- EquipoId: Identificador del equipo utilizado para registrar la transacción.

- TipoEntradaSalidaDeDinero: Tipo de operación (entrada o salida de dinero).

- MotivoId: Motivo de la transacción (por ejemplo, pago de deuda, ingreso por ventas, etc.).

- Consecutivo: Consecutivo único para el registro de las transacciones.

- NoDocumentoEntrada: Si es una entrada, este campo contiene el número de documento de la entrada asociada.

- Procesada: Indica si la transacción ha sido procesada (1 = procesada, 0 = no procesada).

- ClienteId: Identificador del cliente relacionado con la transacción.

- AutorizaId: Identificador del usuario que autorizó la transacción.

Uso:

- Esta tabla es esencial para registrar todos los movimientos de dinero que ocurren dentro del sistema, desde pagos hasta ingresos por ventas, y se utiliza para el control financiero y la auditoría de las transacciones de dinero.

13. ArqueoCajasTable

Propósito: La tabla ArqueoCajas se utiliza para llevar el control de las operaciones relacionadas con el arqueo de las cajas. Aquí se registran los detalles de cada arqueo realizado, incluyendo el total de ventas, el número de documentos procesados, y otras métricas financieras importantes.

Campos:

- ArqueoId: Identificador único del arqueo realizado.

- UsuarioId: Identificador del usuario que realiza el arqueo.

- BodegaId: Identificador de la bodega asociada al arqueo.

- EmpresaId: Identificador de la empresa relacionada con el arqueo.
- SucursalId: Identificador de la sucursal donde se realizó el arqueo.
- EquipoId: Identificador del equipo utilizado.
- Fecha: Fecha del arqueo.
- NoClientesNuevos: Número de clientes nuevos registrados durante el arqueo.
- NoClientesRegistrados: Número de clientes registrados en el sistema durante el arqueo.
- NoPedidos: Número de pedidos procesados.
- NoFacturas: Número de facturas emitidas.
- NoFacturasTemporales: Número de facturas temporales generadas.
- NoCotizaciones: Número de cotizaciones realizadas.
- NoApartados: Número de apartados realizados.

- NoDevoluciones: Número de devoluciones procesadas.
- TotalVentas: Total de ventas realizadas.
- TotalPuntos: Total de puntos acumulados en las ventas.
- TotalUtilidad: Total de utilidad generada en el período de arqueo.
- TotalRemisiones: Total de remisiones procesadas.
- TotalApartados: Total de apartados realizados.
- TotalDevoluciones: Total de devoluciones registradas.
- TotalEntradas: Total de entradas registradas en el arqueo.
- TotalSalidas: Total de salidas registradas en el arqueo.
- NoRemisiones: Número de remisiones procesadas.
- EEfectivoVentas: Total de ventas en efectivo.

- EEfectivoAbonoApartado: Total de abonos en efectivo de apartados.
- EEfectivoAbonoCartera: Total de abonos en efectivo de cartera.
- EEfectivoAbonoDomicilio: Total de abonos en efectivo de domicilios.
- EEfectivoEntrada: Total de entradas en efectivo.
- SEfectivoDevolucion: Total de devoluciones en efectivo.
- SEfectivoVueltos: Total de vueltos en efectivo.
- SEfectivoPrestamos: Total de préstamos en efectivo.
- SEfectivoGastos: Total de gastos en efectivo.
- SEfectivoSalida: Total de salidas en efectivo.
- CreditoCartera: Total de crédito en cartera.
- CreditoDomicilios: Total de créditos de domicilios.
- CreditoPrestamos: Total de crédito de préstamos.

Uso:

- La tabla ArqueoCajas es vital para mantener un control detallado sobre las actividades diarias de ventas, entradas, salidas, pagos y cobros, brindando así un historial financiero claro para la auditoría.

14. EdicionPedidosTable

Propósito: Esta tabla está diseñada para registrar los cambios o ediciones en los pedidos. Cada vez que un pedido se edita, ya sea por modificación de productos, precios o cantidades, este registro se guarda para tener un historial de las modificaciones.

Campos:

- Id: Identificador único de la edición de pedido.
- SalidaIdAnterior: Identificador del pedido anterior (antes de la modificación).
- SalidaIdNuevo: Identificador del pedido después de la modificación.
- UsuarioId: Identificador del usuario que realizó la edición.
- FechaEdicion: Fecha en que se realizó la edición del pedido.

Uso:

- La tabla es crucial para realizar un seguimiento de los pedidos que han sido modificados, lo que es importante para la auditoría, la gestión de cambios y el control de los pedidos en el sistema.

Creación de tablas si No Existen

La creación de las tablas si no existen se maneja mediante la instrucción SQL CREATE TABLE IF NOT EXISTS, que está presente en cada una de las tablas definidas en el método createTableIfNotExistQuery. Este comando se asegura de que, al intentar crear una tabla, si ya existe no se vuelva a crear, evitando errores de duplicación.

Cada clase que implementa la interfaz DatabaseTable tiene definida una propiedad createTableIfNotExistQuery, que proporciona el SQL necesario para crear la tabla. Esto garantiza que en cada instancia de la base de datos se intentará crear todas las tablas especificadas en el constructor de la base de datos.

Estructura de la Clase SqliteDatabase

La clase SqliteDatabase maneja la lógica principal para crear y abrir la base de datos, como se muestra en el siguiente fragmento:

Figura 2

Estructura de la Clase SqliteDatabase

```
Future<Database> openDb() async {
  return _lock.synchronized(() async {
    await requestWritePermisses();
    String pathAPLFolder = await createAPLFolder();
    _database = await openDatabase(join(pathAPLFolder, databaseName),
        onCreate: (db, version) async {
      for (var table in _tables) {
        await db.execute(table.createTableIfNotExistQuery);
      }
    }, onUpgrade: (db, oldVersion, newVersion) async {
      for (var table in _tables) {
        await db.execute(table.createTableIfNotExistQuery);
      }
    }, version: 15);
    return _database!;
  });
}
```

Nota. Estructura general de la clase SqliteDatabase para gestión de datos locales.

Descripción del proceso de creación de la base de datos SQLite:

1. **Solicitud de Permisos:**

    - La función requestWritePermisses() asegura que la aplicación tiene permisos para escribir en el almacenamiento del dispositivo, que es necesario para manipular la base de datos.

2. **Creación de la Carpeta para la Base de Datos:**

- createAPLFolder() verifica si la carpeta donde se almacenará la base de datos existe. Si no, la crea. Este es un paso necesario para gestionar la base de datos en el dispositivo.

3. **Apertura de la Base de Datos:**

- openDatabase() abre la base de datos, si no existe se crea automáticamente.

- La función onCreate se ejecuta solo la primera vez que la base de datos es creada, y dentro de esta, se crean todas las tablas definidas en el constructor (usando el método createTableIfNotExistQuery de cada tabla).

- Si la base de datos ya existe pero se actualiza (en el caso de que la versión de la base de datos cambie), se ejecuta la función onUpgrade y las tablas se verifican y actualizan.

4. **Ejecutando las Sentencias SQL:**

- Para cada tabla en _tables, se ejecuta la sentencia SQL de creación (createTableIfNotExistQuery). Esto garantiza que todas las tablas necesarias estén presentes en la base de datos.

5. **Versión de la Base de Datos:**

    - El parámetro version: 15 especifica la versión de la base de datos, lo cual es importante para gestionar actualizaciones de la estructura de la base de datos (por ejemplo, cuando se agrega nuevas tablas o columnas).

Cierre de la Conexión:

Figura 3

Código para el cierre de la conexión

```
Future<void> closeDb() async {
  if (_database != null && _database!.isOpen) {
    await _database!.close();
    _database = null;
    logger.i("Conexión a la base de datos cerrada exitosamente.");
  } else {
    logger.w("La base de datos ya está cerrada o no se ha abierto.");
  }
}
```

Nota. Código implementado para el cierre seguro de la conexión en SQLite.

- La base de datos se cierra utilizando el método close().
- Si la base de datos ya está cerrada o no se ha abierto, se muestra un mensaje de advertencia.

8.5 Desarrollo de vistas e interfaces de usuario en el sistema POS

8.5.1 Diseño módulo POS

Antes de detallar cada una de las vistas construidas, es importante resaltar que el diseño visual de la aplicación móvil se realizó siguiendo principios de usabilidad, consistencia visual y adecuación al entorno corporativo. A continuación se describen los aspectos principales:

8.5.1.1 Justificación del Diseño

La aplicación fue diseñada bajo los principios de diseño minimalista y enfoque en la funcionalidad, orientado a garantizar rapidez de operación, claridad en la visualización de información y facilidad de navegación para los usuarios en ambientes de alta rotación como los puntos de venta.

Se buscó priorizar la accesibilidad visual, utilizando jerarquías claras de información y evitando sobrecargar la interfaz con elementos innecesarios, en línea con las mejores prácticas recomendadas por Material Design.

8.5.1.2 Selección de Colores.

La paleta de colores empleada en la aplicación se centra principalmente en tonos de azul oscuro (Color(0xFF052964)) combinados con fondos claros. Esta elección responde a:

- **Identidad Corporativa:** El azul oscuro refuerza la identidad visual de la empresa Grupo APL Ingeniería LTDA.
- **Confianza y Profesionalismo:** El color azul, según estudios de psicología del color, transmite sensaciones de seguridad, confianza y profesionalismo, atributos esenciales en soluciones de gestión comercial.

- **Contraste y Legibilidad:** La combinación de azul oscuro con fondos blancos o muy claros asegura un contraste visual óptimo que facilita la lectura y reduce la fatiga ocular.

8.5.1.3 Iconografía.

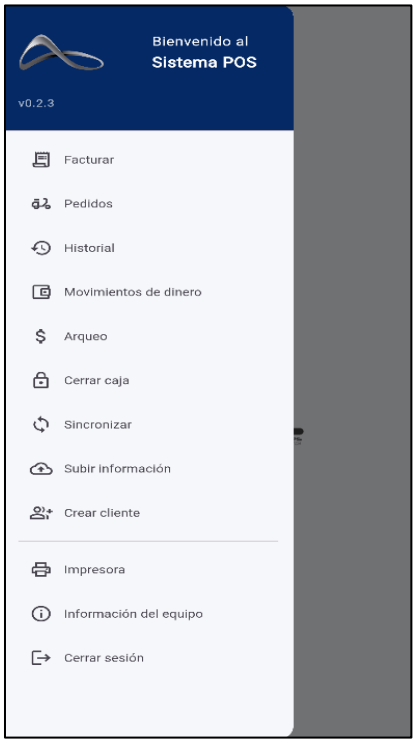
La selección de iconografía se basó en el uso de íconos estandarizados provistos por la librería de Material Icons de Google, garantizando:

- **Reconocimiento Universal:** Facilitan una comprensión rápida y universal de las funciones representadas.
- **Consistencia Visual:** Se mantuvo un estilo uniforme, utilizando siempre versiones outline para alinear el lenguaje gráfico.
- **Optimización en Dispositivos PDA:** Los íconos fueron seleccionados considerando su visibilidad y claridad en dispositivos móviles de pantallas pequeñas, típicos en entornos de punto de venta.

8.5.2 Menú Lateral

Figura 4

Vista del menú lateral de navegación



Nota. Menú lateral principal del sistema POS móvil.

Descripción funcional:

El menú lateral de navegación constituye el componente principal de interacción para la transición entre los diferentes módulos de la aplicación POS móvil. Este menú fue diseñado para ser accesible desde cualquier pantalla mediante el gesto de deslizamiento lateral o el icono de menú en el AppBar.

Cada opción dentro del menú representa un módulo operativo del sistema, organizado de manera jerárquica y lógica para optimizar el flujo de trabajo del usuario en el punto de venta.

Se incorporaron elementos gráficos como el logotipo corporativo de Grupo APL Ingeniería LTDA y mensajes de bienvenida personalizados, reforzando la identidad de marca en todo momento.

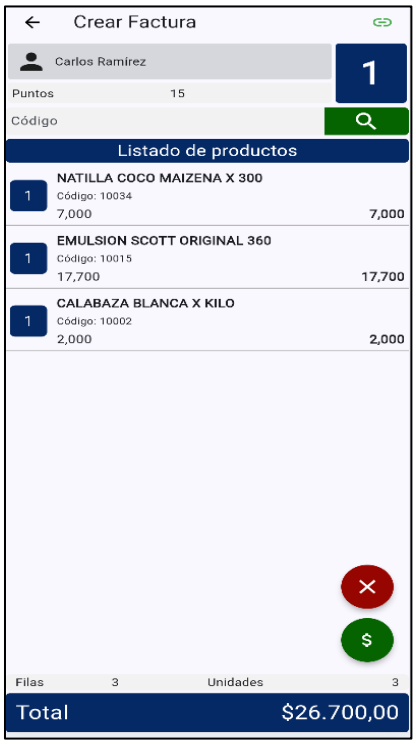
Alcance de la vista:

- Acceso rápido y estructurado a los módulos principales: Facturación, Pedidos, Historial, Movimientos de dinero, Arqueo de caja, Cierre de caja, Configuraciones, Impresoras, Subir Información, entre otros.
- Implementación de una experiencia de usuario coherente en toda la aplicación.
- Personalización estética basada en la identidad visual de la empresa.
- Diseño responsive adaptado a diferentes resoluciones de dispositivos móviles industriales.

8.5.3 Crear Factura

Figura 5

Vista de creación de facturas



Nota. Vista inicial de creación de facturas en el sistema POS.

Descripción funcional:

La vista de creación de facturas permite al usuario registrar ventas de productos de forma ágil y eficiente. A través de esta pantalla, se agregan productos al documento de venta, se especifican cantidades, se verifican precios unitarios y se calculan subtotales y totales automáticamente.

La facturación permite también asociar las ventas a clientes específicos, integrando funcionalidades de programas de fidelización (puntos de cliente) si están disponibles.

Una vez conformada la factura, al presionar el botón verde de acción, el sistema redirige al módulo de Métodos de Pago, donde se registra el pago del cliente. Solo después del pago exitoso, se habilita la opción de imprimir el comprobante de venta.

Funciones principales:

- Agregar productos mediante código de barras (scanner) o búsqueda manual.
- Ingreso de cantidad mediante ventana emergente que aparece después de seleccionar el producto.
- Visualización dinámica de productos agregados, cantidades, precios unitarios, subtotal y total acumulado.
- Botón rojo para cancelar la operación (anular venta).
- Botón verde para continuar hacia la pantalla de Métodos de Pago.

Alcance de la vista:

- Registro de ventas en modo offline (SQLite) para posterior sincronización.
- Cálculo automático del total de la venta y resumen de unidades.
- Asociación opcional de facturas a clientes registrados.
- Visualización de totales en tiempo real, con formato monetario local.
- Inclusión de flujo de impresión tras la generación de la factura.

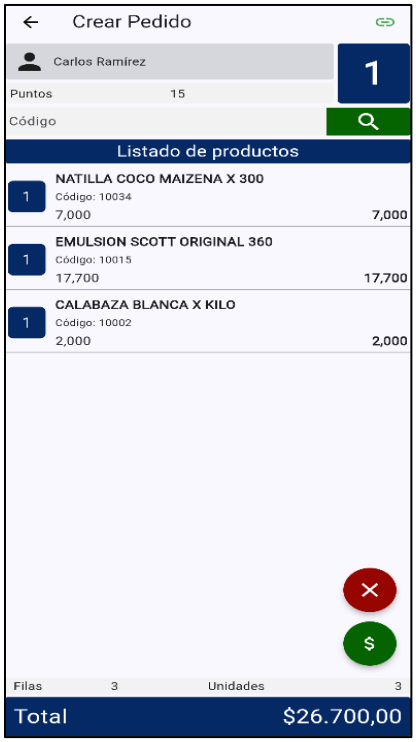
Importancia dentro del sistema:

Constituye el núcleo transaccional del POS, permitiendo que las ventas sean registradas en cualquier condición de conectividad, asegurando la operatividad continua del comercio.

8.5.4 Crear Pedido

Figura 6

Vista de creación de pedidos



Nota. Vista inicial de creación de pedidos en el sistema POS.

Descripción funcional:

La vista de creación de pedidos permite a los usuarios registrar solicitudes de productos de clientes. A diferencia de la facturación, la creación de pedidos no genera un compromiso financiero inmediato, sino que representa una orden de solicitud formalizada.

La interfaz replica elementos de la vista de facturación para mantener la consistencia visual, pero modifica flujos de acción como el almacenamiento temporal y la posterior facturación o confirmación del pedido.

Alcance de la vista:

- Captura de pedidos aún sin pago efectuado.
- Registro de múltiples productos con cantidades específicas.
- Cálculo del subtotal y resumen de productos agregados.
- Preparación de pedidos para sincronización manual posterior.

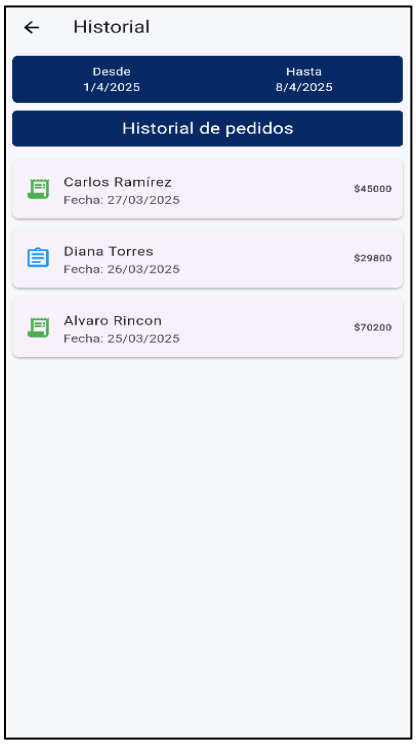
Importancia dentro del sistema:

Permite que el negocio registre solicitudes de clientes que no se convierten inmediatamente en ventas, soportando flujos operativos más flexibles como pedidos anticipados, cotizaciones o preventas.

8.5.5 Historial

Figura 7

Vista del historial



Nota. Pantalla de historial de transacciones (facturas y pedidos) en el sistema POS.

Descripción funcional:

La pantalla de Historial centraliza la visualización de todas las operaciones registradas en el dispositivo móvil (Facturas, Pedidos). Se ofrecen funcionalidades de filtrado por fecha, diferenciación visual entre tipos de documentos (facturas y pedidos) y navegación directa al detalle de cada operación.

Cada registro muestra información clave como fecha de operación, tipo de documento, número identificador y valor total.

Alcance de la vista:

- Consulta rápida de todas las transacciones realizadas.

- Acceso al detalle completo de cualquier pedido o factura.
- Posibilidad de realizar reimpresiones o revisiones de registros históricos.

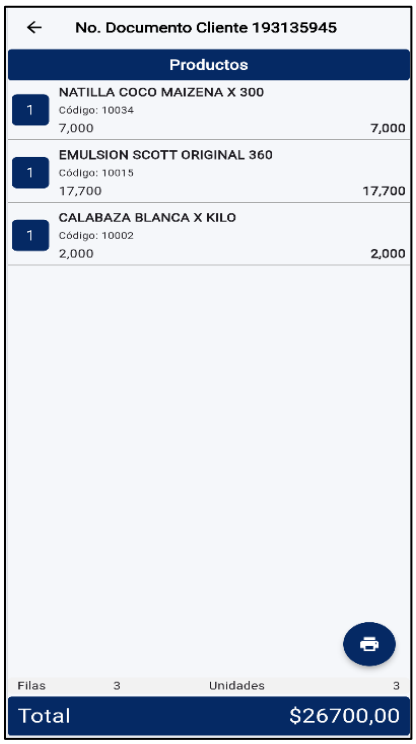
Importancia dentro del sistema:

Garantiza la trazabilidad de las operaciones comerciales realizadas en el POS, siendo un recurso clave para auditorías, cierres de caja y atención a clientes.

8.5.6 Historial Detalle

Figura 8

Vista de la lista de productos



Nota. Vista de listado de productos en inventario en el módulo de ventas.

Descripción funcional:

Permite visualizar en profundidad cada documento registrado, mostrando una lista detallada de todos los productos involucrados, cantidades, precios unitarios y totales parciales.

Incorpora también funcionalidades de impresión de documento y, en caso de que sea un pedido y no una factura, se visualizara un botón encima del botón de impresión para continuar con el proceso de pago del pedido.

Alcance de la vista:

- Visualización detallada de cada producto registrado en un pedido o factura.
- Opción de imprimir comprobantes en impresoras portátiles.
- Control de integridad documental al acceder a información histórica detallada.
- Opción de continuar con proceso de pago en caso de ser una visualización de un pedido.

Importancia dentro del sistema:

Permite a los usuarios validar rápidamente la composición exacta de cada transacción, asegurando transparencia y minimizando errores administrativos.

8.5.7 Subir Información

Figura 9

Vista del menú de subir información

Nota. Pantalla de sincronización manual de información pendiente.

Descripción funcional:

La pantalla de Subir Información actúa como el punto de control de sincronización de los datos almacenados localmente en SQLite con la base de datos central en SQL Server. Gestiona manualmente la carga de información, permitiendo trabajar offline sin riesgo de pérdida de datos.

Alcance de la vista:

- Mostrar listado de operaciones no sincronizadas.
- Permitir la sincronización completa.
- Confirmación visual de éxito o error de subida.

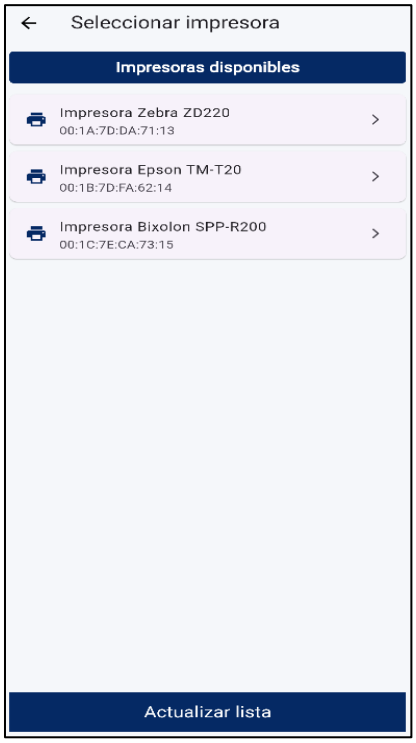
Importancia dentro del sistema:

Es el mecanismo fundamental para mantener la coherencia de datos en ambientes con conectividad intermitente, habilitando la continuidad operativa sin depender de conexión permanente.

8.5.8 Seleccionar Impresora

Figura 10

Vista del menú para elegir que impresora usar



Nota. Vista para la selección y configuración de impresoras Bluetooth.

Descripción funcional:

Permite escanear dispositivos Bluetooth cercanos y seleccionar una impresora predeterminada para operaciones de impresión de facturas o pedidos.

Alcance de la vista:

- Escanear impresoras Bluetooth disponibles.
- Conectar y configurar una impresora predeterminada.
- Optimizar procesos de impresión en campo.

Importancia dentro del sistema:

Reduce tiempos de operación en el punto de venta, permitiendo emisión inmediata de documentos impresos, especialmente útil en servicios de venta rápida.

8.5.9 Cerrar Caja

Figura 11

Vista del menú para cerrar caja

Nota. Vista para realizar el proceso de cierre de caja diario.

Descripción funcional:

Facilita el cierre diario de operaciones de caja, presentando un resumen financiero completo basado en la información registrada durante la jornada.

Muestra las ventas realizadas, ingresos por forma de pago, salidas de efectivo, y permite el registro de observaciones asociadas al turno (cerrar caja).

Alcance de la vista:

- Cierre formal de operaciones del turno.
- Consolidación de ingresos, egresos y efectivo final.
- Generación de datos para arqueo y conciliación de caja.

Importancia dentro del sistema:

Cierra el ciclo financiero diario del punto de venta, asegurando control interno y facilitando la generación de reportes para supervisión administrativa.

8.6 Introducción al módulo de Inventario

El módulo de Inventarios fue una extensión desarrollada para complementar las funcionalidades del sistema POS de la empresa Grupo APL Ingeniería LTDA. A diferencia del módulo POS, que está centrado en las transacciones de venta y facturación, el módulo de Inventarios está dedicado a la gestión operativa de los productos en los establecimientos, permitiendo controlar existencias, verificar saldos, y realizar chequeos físicos en tiempo real.

Su propósito principal es ofrecer una herramienta ágil, sencilla y confiable que permita a los usuarios realizar operaciones relacionadas con inventarios directamente desde dispositivos móviles PDA, asegurando la actualización inmediata de la base de datos central en SQL Server.

Este módulo mantiene la misma estética visual, paleta de colores y lineamientos de diseño definidos para la aplicación POS, garantizando una experiencia de usuario homogénea e intuitiva entre los dos entornos.

8.7 Análisis y comprensión del sistema existente

Antes del desarrollo de este módulo, no existía en la aplicación móvil una sección orientada a la gestión de inventarios. Los procesos de inventario se realizaban de forma manual o a través de sistemas externos, lo cual dificultaba la actualización rápida de existencias y aumentaba el riesgo de errores.

La creación de este módulo responde entonces a una necesidad identificada dentro de la empresa: administrar la gestión de inventarios mediante un sistema accesible, móvil y conectado directamente al ecosistema centralizado de datos.

8.8 Relación entre el módulo de Inventarios y el módulo POS

El módulo de Inventarios, aunque representa una sección independiente en cuanto a su funcionalidad, no opera de manera aislada dentro de la aplicación móvil. Su diseño y estructura responden a un esquema de integración plena con el ecosistema general, compartiendo recursos, principios de diseño y bases tecnológicas con el módulo de Punto de Venta (POS).

8.8.1 Autenticación Unificada

El acceso al módulo de Inventarios se gestiona desde la misma pantalla de login utilizada en el módulo POS. Dependiendo del tipo de usuario que inicia sesión, el sistema redirige automáticamente a la sección correspondiente (POS o Inventarios). Esta estrategia garantiza un control de usuarios centralizado y eficiente, evitando redundancias y fortaleciendo los procesos de seguridad.

8.8.2 Infraestructura de Base de Datos Compartida

Tanto el módulo de Inventarios como el de POS comparten la base de datos central en SQL Server, lo que permite:

- Mantener actualizaciones en tiempo real sobre el estado de existencias.
- Asegurar que las operaciones realizadas desde ambos módulos impacten sobre la misma fuente de información, evitando inconsistencias o duplicaciones.

Cabe resaltar que, mientras el POS puede operar de forma offline utilizando una base local SQLite (para cubrir situaciones de falta de conectividad), el módulo de Inventarios trabaja únicamente en línea, ya que las operaciones de inventario exigen precisión y actualización inmediata para garantizar la integridad del stock.

8.8.3 Reutilización de Funcionalidades

En términos de componentes funcionales, el módulo de Inventarios reutiliza varias funcionalidades implementadas en POS, optimizando así los recursos de desarrollo. Entre ellas se destacan:

- Selección de impresora: Para dispositivos que requieren impresión de reportes o etiquetas.
- Acceso a la información técnica del dispositivo: Como la lectura del IMEI, útil en tareas de soporte y rastreo de dispositivos móviles.

Esta reutilización también facilita la curva de aprendizaje de los usuarios al mantener consistencia en los comportamientos de la aplicación.

8.8.4 Uniformidad de Diseño y Experiencia de Usuario

El módulo de Inventarios mantiene la misma línea gráfica adoptada para el módulo POS:

- Uso de los mismos tonos corporativos.
- Estilo de íconos estandarizado.

- Organización de menús y botones siguiendo el mismo patrón de navegación.

Esto garantiza que, a pesar de pertenecer a diferentes módulos funcionales, el usuario perciba la aplicación como un todo cohesivo y no como sistemas separados.

8.8.5 Función Complementaria dentro del Ecosistema

Si bien el POS y el Inventario comparten una base tecnológica, sus funciones son complementarias:

- El módulo POS gestiona el flujo de venta, la facturación y el movimiento de caja.
- El módulo de Inventarios garantiza la precisión física y digital del stock de productos, asegurando que la información sobre existencias esté siempre alineada con la realidad operativa.

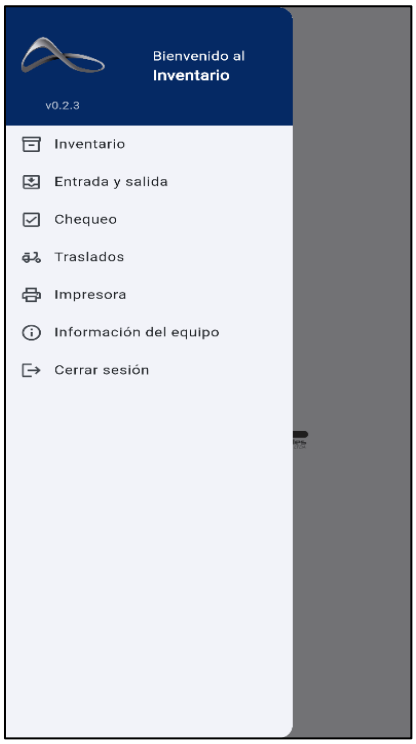
Esta complementariedad refuerza el objetivo estratégico de la aplicación de cubrir el ciclo completo de venta y control de mercancías, mejorando así la gestión empresarial en su conjunto.

8.9 Desarrollo de vistas e interfaces de usuario en el módulo de Inventario

8.9.1 Menú Lateral

Figura 12

Vista del menú lateral desde inventarios



Nota. Menú lateral de navegación del módulo de Inventarios.

Descripción funcional:

El menú lateral en el módulo de Inventarios permite la navegación rápida entre las principales funcionalidades relacionadas con la gestión de stock y auditoría de productos. Fue diseñado siguiendo los mismos principios estéticos y de experiencia de usuario aplicados en el módulo POS, garantizando una transición intuitiva para los usuarios.

Funciones principales:

- Acceso a la vista de Inventario para consulta de productos.

- Acceso a la vista de Entrada y Salida de mercancía.
- Acceso a la funcionalidad de Chequeo de productos.
- Acceso a la configuración de Impresoras y a la Información del equipo.
- Opción de Cerrar sesión de forma segura.

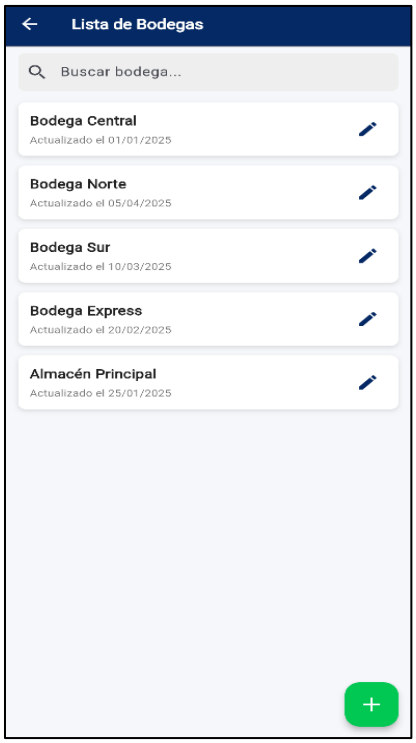
Alcance de la vista:

- Agrupar de manera lógica y accesible todas las funcionalidades del módulo de Inventarios.
- Mantener una experiencia de navegación uniforme entre módulos.

8.9.2 Inventario

Figura 13

Vista de la lista de bodegas



Nota. Pantalla principal de listado de bodegas disponibles.

Descripción funcional:

La pantalla de Inventario presenta la lista de bodegas disponibles en el sistema, permitiendo su gestión visual y funcional. Cada bodega muestra su nombre y fecha de última actualización, además de ofrecer accesos para editar su información o agregar nuevas bodegas mediante el botón flotante. Facilita la organización de los espacios físicos de almacenamiento dentro del flujo de inventarios.

Alcance de la vista:

- Visualización general de todas las bodegas registradas.
- Edición de nombre o datos de una bodega existente.
- Creación de nuevas bodegas.
- Búsqueda rápida de bodegas específicas.

Importancia dentro del sistema:

- Establece la estructura de almacenamiento necesaria para la gestión de inventarios.
- Facilita el control y la trazabilidad de los espacios donde se encuentran los productos.
- Permite segmentar el inventario por ubicaciones, mejorando la eficiencia operativa.

8.9.3 Ajuste de inventario

Figura 14

Vista del menú de ajuste de inventario

Nota. Menú para registrar motivo y observaciones de ajuste de inventario.

Descripción funcional:

La pantalla de Ajuste de Inventario permite iniciar un proceso de ajuste sobre una bodega determinada. El usuario debe indicar el motivo del ajuste (por ejemplo, corrección de stock) y agregar observaciones que documenten la acción. Esta funcionalidad asegura que cada movimiento de inventario quede debidamente registrado.

Alcance de la vista:

- Selección del motivo del ajuste.
- Registro de observaciones o comentarios adicionales.

- Creación formal de un nuevo proceso de ajuste de inventario.

Importancia dentro del sistema:

- Aporta trazabilidad y justificación a los movimientos internos de inventario.
- Reduce errores de manipulación o conteo no documentado.
- Fortalece los controles de calidad y auditoría de existencias.

8.9.4 Productos en inventario

Figura 15

Vista del menú de la bodega principal

Nota. Vista de productos dentro de la bodega principal del inventario.

Descripción funcional:

Esta pantalla muestra el listado de productos almacenados en una bodega específica, con su existencia actual y el conteo nuevo realizado por el usuario. Permite agregar productos no

registrados previamente y realizar búsquedas rápidas dentro del inventario de la bodega. Incluye indicadores de totales de productos y conteos parciales.

Alcance de la vista:

- Visualización del stock actual y conteos nuevos de productos.
- Adición de productos al inventario manualmente o con código de barras.
- Búsqueda de productos específicos por nombre o código.
- Gestión de ajustes parciales o totales de stock.

Importancia dentro del sistema:

- Garantiza la actualización precisa de las existencias físicas versus las registradas.
- Facilita el control periódico de inventarios.
- Mejora la exactitud de reportes y la toma de decisiones basada en stock real.

8.9.5 Ventana emergente de agregado de productos

Figura 16

Vista de la ventana agregar producto

Nota. Ventana emergente para agregar productos nuevos al inventario.

Descripción funcional:

Cuando el usuario desea agregar un producto, se despliega una ventana emergente que permite registrar el código del producto y su cantidad correspondiente. Se puede ingresar manualmente , el escaneo unitario o utilizar el modo de escaneo continuo, pensado para agilizar la carga en inventarios de gran volumen .

Alcance de la vista:

- Ingreso manual de productos por código y cantidad.
- Escaneo rápido de productos mediante cámara o lector de código de barras.
- Agregación automática de múltiples productos sin necesidad de cerrar la ventana.

Importancia dentro del sistema:

- Optimiza el proceso de registro de productos durante el conteo físico.
- Minimiza los errores de captura de información.
- Agiliza las operaciones de grandes cargas o ajustes masivos de stock.

8.9.6 Chequeo de precios

Figura 17

Vista del menú chequeo de precio

Nota. Pantalla de chequeo de precio y creación de habladores para productos.

Descripción funcional:

La pantalla de Chequeo de Precios permite consultar de manera rápida la información de productos en inventario, como su existencia actual y su precio de venta. Funciona como una

herramienta ágil para validaciones en la bodega. Además, esta vista permite la creación, consulta e impresión de habladores, que son etiquetas visuales utilizadas para informar al cliente sobre precios, promociones o características de los productos en exhibición.

Alcance de la vista:

- Consulta inmediata de existencias y precios de productos mediante búsqueda por código en determinada selección de bodega.
- Creación de nuevos habladores personalizados.
- Visualización de habladores ya creados.
- Impresión directa de habladores para etiquetado de productos en tienda.

Importancia dentro del sistema:

- Facilita el control y verificación rápida de inventarios desde cualquier dispositivo móvil.
- Mejora la experiencia de compra de los clientes al permitir identificar precios fácilmente a través de habladores.
- Agiliza procesos de etiquetado y actualización de precios en puntos de venta o almacenes.
- Minimiza errores humanos en la comunicación de precios al público.

9. Pruebas y Validaciones

Durante el proceso de desarrollo del sistema POS móvil y del módulo de Inventarios, se implementaron pruebas unitarias funcionales y validaciones de integración utilizando las herramientas nativas de Flutter, principalmente a través del paquete flutter_test.

Estas pruebas permitieron comprobar el correcto funcionamiento de las funciones críticas desarrolladas, asegurando la integridad de los flujos de datos, la navegación entre pantallas, el manejo de estados internos, la persistencia en bases de datos locales (SQLite) y la comunicación con la API REST.

Objetivo de las pruebas

- Validar la integridad de datos capturados en formularios (facturación, pedidos, ajustes de inventario).
- Verificar la correcta navegación entre módulos POS e Inventario.
- Comprobar la persistencia y sincronización de datos en SQLite y SQL Server.
- Validar flujos de operaciones de escaneo de productos y gestión de inventarios.

Metodología utilizada

Se realizaron pruebas unitarias sobre funciones específicas de la lógica de negocio mediante la librería flutter_test, diseñando casos de prueba que simularon entradas válidas e inválidas y verificando respuestas esperadas.

Además, se llevaron a cabo pruebas funcionales manuales utilizando dispositivos PDA reales para validar la experiencia del usuario final, simular escenarios de operación en línea y offline, y comprobar el comportamiento en condiciones de conectividad variable.

Resultados

Gracias a la implementación de pruebas unitarias y funcionales, fue posible detectar y corregir inconsistencias en etapas tempranas del desarrollo. El sistema mostró una operación estable, robusta y segura al cierre de las validaciones, cumpliendo con los objetivos de calidad y confiabilidad planteados inicialmente.

10. Conclusiones

El desarrollo del Sistema Integral Móvil para la Gestión de Ventas e Inventarios en Grupo APL Ingeniería LTDA representa un avance significativo en la modernización de los procesos comerciales de la organización, alineándose a las exigencias contemporáneas de eficiencia, movilidad y trazabilidad en entornos empresariales dinámicos.

A lo largo del proyecto se abordó de manera integral el diseño, implementación y validación de un sistema POS móvil complementado con un módulo de gestión de inventarios. La adopción de una arquitectura cliente-servidor híbrida, combinando operaciones offline mediante SQLite y sincronización con un servidor central basado en SQL Server, garantiza tanto la continuidad operativa en escenarios de conectividad limitada como la consistencia de la información a nivel corporativo. Esta solución proporciona a la empresa herramientas prácticas para la atención ágil de clientes, el control eficiente del stock, la administración de movimientos de dinero y la generación de informes estratégicos.

La elección de tecnologías como Flutter, ASP.NET Core Web API y Visual Studio Code, además de la correcta implementación de metodologías ágiles adaptadas (XP, Scrum, DSDM, ASD y Crystal), permitió no solo una construcción técnica sólida del sistema, sino también una respuesta rápida a cambios, asegurando un producto funcional, escalable y adaptable a las necesidades del negocio.

El desarrollo de interfaces de usuario enfocadas en la simplicidad, el profesionalismo y la coherencia visual fortaleció la experiencia de uso, minimizando tiempos de capacitación y errores operativos. La inclusión de vistas especializadas como el historial transaccional, la gestión de métodos de pago, el control de impresoras Bluetooth y el chequeo rápido de precios —con la

posibilidad de creación e impresión de habladores— evidencia el compromiso con una solución que abarca las necesidades del punto de venta moderno en su totalidad.

Finalmente, la ejecución de este proyecto no solo aportó valor tangible a la operación de Grupo APL Ingeniería LTDA, sino que también representó un espacio de aprendizaje y aplicación real de conocimientos en ingeniería de software, fortaleciendo competencias en análisis de sistemas, diseño de arquitecturas, desarrollo móvil, gestión de bases de datos y control de versiones. Se evidencia, entonces, la capacidad de integrar soluciones tecnológicas a problemas empresariales concretos, contribuyendo al crecimiento organizacional y sentando las bases para futuras innovaciones tecnológicas dentro de la empresa.

Referencias Bibliográficas

Alegra. (n.d.). *POS y Software de Facturación para Punto de Venta*. https://www.alegra.com/colombia/pos/

Al-Saqqa, S., Sawalha, S., & Abdelnabi, H. (2020). Agile Software Development: Methodologies and Trends. *International Journal of Interactive Mobile Technologies*, 14, 246-270.

BillWagner. (2024, May 8). *Overview - A tour of C#*. https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview

Cantrell, M., & Pugh, M. (2017). *Node.js in Practice*. Manning Publications.

Comer, D. E. (2018). *Redes de computadoras y sistemas distribuidos* (5.ª ed.). Pearson Educación.

Dahm, M., Wentzel, D., Herzog, W., & Wiecek, A. (2018). Breathing down your neck! *Journal of Retailing*, *94*(2), 217–230. https://doi.org/10.1016/j.jretai.2018.04.002

Flanagan, D. (2020). *JavaScript: The Definitive Guide* (7.ª ed.). O'Reilly Media

Flutter. (n.d.). *FAQ*. https://docn.dlutter.dev/resources/faq

Gangadhare, A. (2022). *flutter_barcode_scanner*. Recuperado de https://pub.dev/packages/flutter_barcode_scanner

Gosling, J., Joy, B., & Steele, G. (2000). *The Java™ Programming Language* (3.ª ed.). Addison-Wesley.

Grewal, D., Roggeveen, A. L., & Nordfält, J. (2017). The Future of Retailing. *Journal of Retailing*, 93(2), 168-181. https://doi.org/10.1016/j.jretai.2016.12.008

Hearson, J. (2024, 6 junio). *Lean construction: How to Improve Your Scheduling Process*. https://www.oracle.com/scm/inventory-management/what-is-inventory-management/

Hernández, J., & González, A. (2018). *Introducción a PostgreSQL: Un sistema de gestión de bases de datos relacionales avanzado*. Editorial Académica Española.

Holovaty, A., & Kaplan, J. (2009). *The Definitive Guide to Django: Web Development Done Right*. Apress.

IEEE. (2022). *Arquitectura Cliente-Servidor en sistemas distribuidos*. IEEE Region 9. Recuperado de https://r9.ieee.org/wp-content/uploads/2022/01/NoticIEEEro_54.pdf

JuliusCanute. (2023). *qr_code_scanner*. Recuperado de https://pub.dev/packages/qr_code_scanner

Khoren. (2025). *flutter_zxing*. Recuperado de https://pub.dev/packages/flutter_zxing

Livermore, J. A. (2007). Factors that impact implementing an agile software development methodology. *Proceedings 2007 IEEE SoutheastCon*, 82-86. https://doi.org/10.1109/SECON.2007.342860

Meta Platforms. (n.d.). *React Native*. Recuperado de https://reactnative.dev/docs/getting-started

Microsoft. (2025). *Introduction to Web API in ASP.NET Core*. Microsoft Docs. Recuperado de https://learn.microsoft.com/en-us/aspnet/core/web-api/

Microsoft. (n.d.). *Visual Studio Code*. Recuperado el 28 de febrero de 2025, de https://code.visualstudio.com/

Oracle. (2024). *¿Qué es la sincronización de datos?* Oracle Colombia. Recuperado el 28 de febrero de 2025, de https://www.oracle.com/co/integration/data-synchronization/

Pillai, R. N. (2010). Inventory management performance in machine tool SMEs: What factors do influence them? *Journal of Industrial Engineering and Management, 3(3), 542-560.* https://hdl.handle.net/10419/188437

rwestMSFT, & MikeRayMSFT.(2024, February 14). *What is SQL Server? - SQL Server*. Microsoft Learn. https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16

Sidhunata, B. M., Gabbatha, M. K., Susilo, N. A. N., Buu Sada, P. M. L., Farabi, B. D., Piolo, S., & Singgalen, Y. A. (2023). Point of Sales (POS) System Design using Design Thinking Framework for Motorcycle Workshop. *Journal of Information Systems and Informatics (Palembang.Online), 5*(3), 874–886. https://doi.org/10.51519/journalisi.v5i3.515

Siigo. (n.d.). *Sistema POS Siigo | Software fácil y rápido*. Siigo.com. https://www.siigo.com/sistema-pos/

SQLite. (n.d.). *About SQLite*. https://www.sqlite.org/about.html

Steenbakker, J. (2025). *mobile_scanner*. Recuperado de https://pub.dev/packages/mobile_scanner

Strode, D. E. (2006). Agile methods: a comparative analysis. In *Proceedings of the 19th annual conference of the national advisory committee on computing qualifications, NACCQ* (Vol. 6, pp. 257-264).

Tanenbaum, A. S., & Wetherall, D. J. (2011). *Redes de computadoras* (5.ª ed.). Pearson Educación.

Tiendada. (n.d.). *Crea tu tienda virtual - Empieza GRATIS*. https://tiendada.com/

Turban, E., Volonino, L., & Wood, G. (2015). *Information Technology for Management: Digital Strategies for Insight, Action, and Sustainable Performance* (10th ed.). John Wiley & Sons.

van Riel, A.C.R., Semeijn, J., Ribbink, D. and Bomert-Peters, Y. (2012), "Waiting for service at the checkout: Negative emotional responses, store image and overall satisfaction", *Journal of Service Management*, Vol. 23 No. 2, pp. 144-169. https://doi.org/10.1108/09564231211226097

Van Rossum, G., & Drake, F. L. (2009). *The Python Language Reference Manual* (3.ª ed.). Network Theory Ltd.

Vendty. (n.d.). *Software punto de venta para restaurantes y comercios*. https://www.vendty.com/

Apéndices

Apéndice A. Vista de Login

Nota. Pantalla de inicio de sesión del aplicativo móvil.

Descripción general:

La pantalla de inicio de sesión constituye la puerta de entrada principal al sistema POS. Su diseño es minimalista y claro, facilitando el acceso rápido al sistema mediante el ingreso de la contraseña asignada al usuario.

Funciones principales:

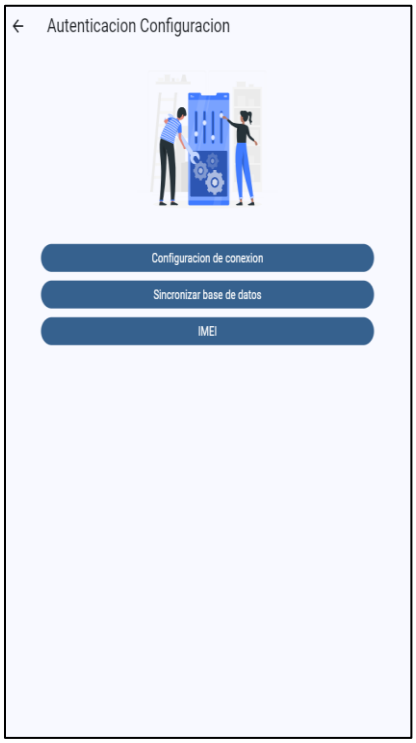
- Ingreso de credenciales para autenticar al usuario.
- Validación de acceso al sistema POS.
- Presentación de la versión actual de la aplicación.

- Acceso directo a la configuración mediante el ícono de ajustes.

Relación con otras vistas o procesos:

- Permite acceder al menú principal si el inicio de sesión es exitoso.
- Si el usuario requiere configurar el dispositivo (por ejemplo, conexión o IMEI), puede hacerlo desde el ícono de ajustes disponible en esta misma vista antes de iniciar sesión.

Apéndice B. Vista de Ajustes



Nota. Vista del menú de ajustes generales de la aplicación.

Descripción general:

Esta vista proporciona funcionalidades de configuración básica y utilidades esenciales del dispositivo, que son críticas para el funcionamiento inicial o la recuperación ante fallas de conexión.

Funciones principales:

- Configuración de conexión (IP y puerto de servidor).
- Sincronización manual de la base de datos local (SQLite) con el servidor (SQL Server).
- Visualización y obtención del IMEI del dispositivo.

Relación con otras vistas o procesos:

- La correcta configuración de conexión permite que la aplicación pueda interactuar con la API REST.
- La sincronización inicial es vital para cargar productos, clientes y demás datos antes de operar offline.
- La obtención del IMEI es necesaria para registrar el dispositivo en el sistema de seguridad de la empresa.

Apéndice C. Vista de Configuración

Nota. Vista de configuración de IP y puerto del servidor API REST.

Descripción general:

La pantalla de configuración ofrece un acceso rápido y sencillo para establecer los parámetros de comunicación entre el cliente (Flutter) y la API REST que conecta con el servidor central.

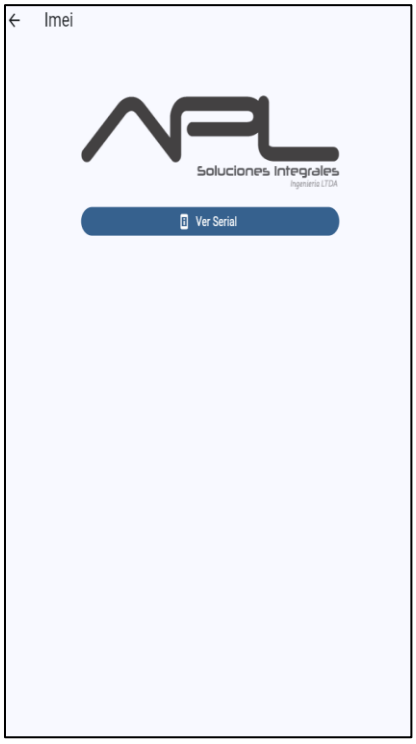
Funciones principales:

- Ingreso de la dirección IP del servidor API.
- Ingreso del puerto de conexión.
- Guardado seguro de los datos para mantener la persistencia de la configuración.

Relación con otras vistas o procesos:

- El correcto ingreso de IP y puerto es necesario para realizar sincronizaciones y transacciones posteriores.
- Permite que la aplicación pueda conectarse de forma automática en siguientes sesiones si los datos fueron guardados correctamente.

Apéndice D. Vista de IMEI



Nota. Pantalla de visualización del IMEI del dispositivo móvil.

Descripción general:

Esta vista permite la visualización del IMEI o número de serie único del dispositivo PDA, esencial para su registro, trazabilidad y control en el sistema empresarial.

Funciones principales:

- Mostrar el IMEI del dispositivo.

- Facilitar la verificación rápida del dispositivo registrado en el sistema.

Relación con otras vistas o procesos:

- El IMEI se utiliza para validar dispositivos permitidos en la plataforma POS.
- Ayuda en la gestión de inventarios de equipos y en la seguridad del acceso al sistema.

Apéndice E. Vista de Inicio

Nota. Pantalla de inicio después del acceso exitoso al sistema POS.

Descripción general:

Es la primera pantalla visible tras el inicio de sesión exitoso. Muestra únicamente el logotipo de la empresa, reforzando la identidad visual y proporcionando un entorno limpio y organizado.

Funciones principales:

- Actúa como pantalla de bienvenida.
- Permite el acceso al menú lateral para navegar a las demás funciones del sistema.

Relación con otras vistas o procesos:

- Desde aquí se despliega el menú lateral que dirige a todas las funcionalidades del POS.
- Refuerza el branding corporativo mientras se mantiene un flujo simple de navegación.

Apéndice F. Vista de Arqueo de Caja

Nota. Vista de resumen de arqueo de caja al cierre de jornada.

Descripción general:

La vista de arqueo de caja proporciona un resumen detallado de los movimientos financieros y operativos de la jornada. Está orientada a la consulta de datos y no a la edición, asegurando la trazabilidad de las operaciones.

Funciones principales:

- Visualización de ventas por cajero (cantidad de facturas y valor total).
- Visualización de métodos de pago utilizados (efectivo, tarjeta, transferencia).
- Visualización de impuestos recaudados.
- Visualización de entradas y salidas de efectivo.
- Reporte de totales consolidados.

Relación con otras vistas o procesos:

- Ayuda a la validación diaria de la caja.
- Facilita la conciliación entre ventas realizadas y efectivo disponible.
- No se edita ningún dato desde esta vista, solo se consulta información generada previamente.

Apéndice G. Vista de Movimientos de Dinero

Nota. Vista para el registro de movimientos de dinero no asociados a ventas.

Descripción general:

Esta vista permite registrar entradas y salidas de dinero que no corresponden a ventas normales (por ejemplo, ingresos extraordinarios).

Funciones principales:

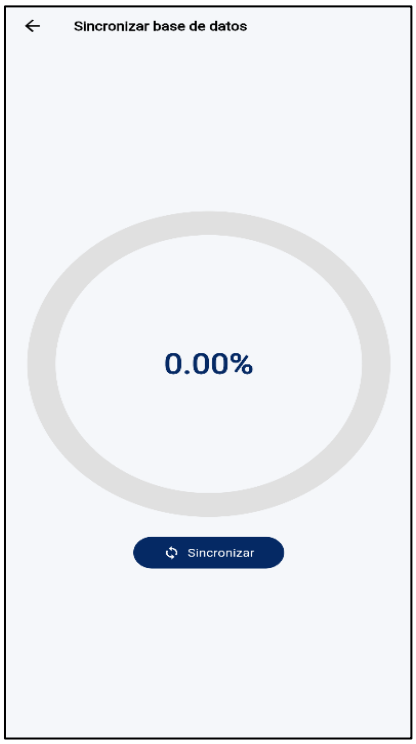
- Registro de movimientos de tipo ingreso o egreso.
- Registro de una "razón" específica del movimiento.
- Visualización del historial de movimientos registrados en la misma jornada.
- Posibilidad de corregir entradas antes de la sincronización.

Relación con otras vistas o procesos:

- Alimenta los datos visibles en las secciones de arqueo de caja y cierre de caja.

- Sirve para mantener control financiero de ingresos y egresos ajenos a facturación.

Apéndice H. Vista de Sincronizar Información



Nota. Pantalla de sincronización de catálogos maestros del sistema.

Descripción general:

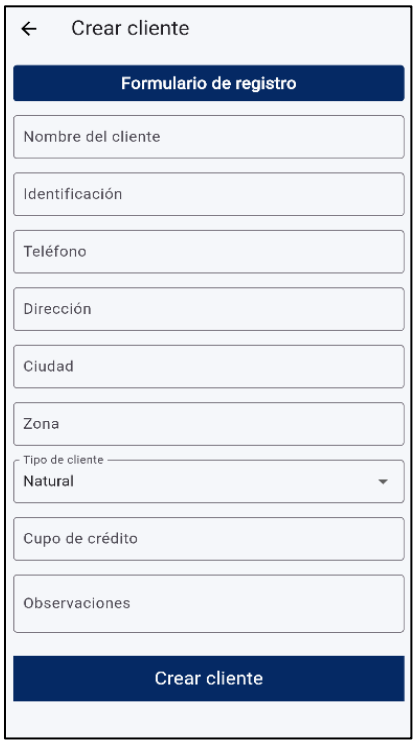
Esta pantalla permite actualizar la información local (almacenada en SQLite) descargándola directamente desde el servidor central (SQL Server) a través de la API REST.

- Funciones principales:
- Visualización de porcentaje de progreso de sincronización de datos.
- Iniciar manualmente el proceso de descarga de datos actualizados.
- Confirmación visual cuando la base de datos local se ha actualizado correctamente.

Relación con otras vistas o procesos:

- Es necesaria para que el dispositivo mantenga su catálogo de productos, clientes, precios, y demás datos actualizados.
- No sube información generada localmente (facturas, pedidos, movimientos de dinero); para eso existe otra vista ("Subir información").
- Permite mantener la operación offline del sistema en óptimas condiciones al refrescar periódicamente la base de datos local.

Apéndice I. Vista de Crear Cliente



Nota. Formulario de creación de nuevos clientes en el sistema.

Descripción general:

Facilita el registro de nuevos clientes directamente desde el dispositivo, incluso en modalidad offline.

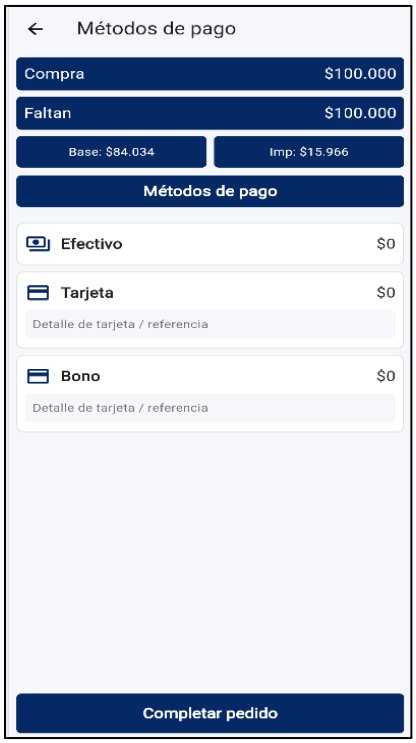
Funciones principales:

- Captura de información básica de cliente: nombre, identificación, dirección, contacto.
- Registro inmediato en la base de datos SQLite.
- Posterior sincronización hacia el servidor en la vista "Subir Información".

Relación con otras vistas o procesos:

- Permite ampliar la base de clientes de manera ágil en campo o en puntos de venta móviles.
- Garantiza que clientes nuevos estén disponibles para facturación apenas se registren.

Apéndice J. Vista de Métodos de Pago



Nota. Vista de asignación de métodos de pago a las transacciones.

Descripción general:

Permite gestionar los métodos de pago utilizados en las facturas o pedidos generados.

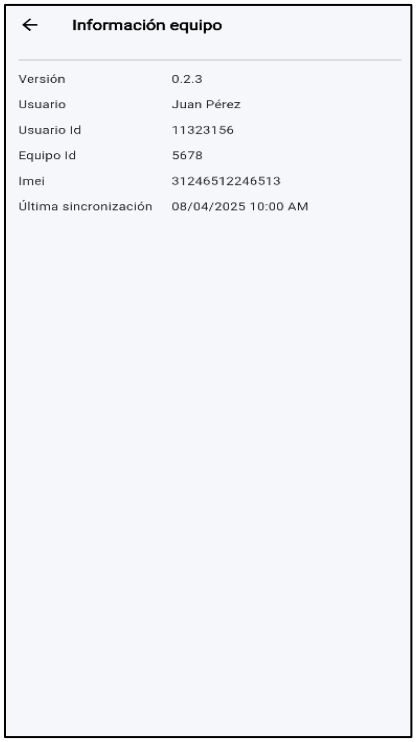
Funciones principales:

- Registro de pagos en efectivo, tarjeta o bono.
- Permitir pagos combinados (múltiples métodos por una misma factura).
- Validación de montos registrados contra el total de la venta.

Relación con otras vistas o procesos:

- Es un paso obligatorio en el flujo de facturación y pedido.
- Alimenta los reportes financieros de arqueo y cierre de caja.

Apéndice K. Vista de Información del Equipo



Nota. Pantalla de información técnica del dispositivo móvil y del usuario.

Descripción general:

Proporciona información técnica relevante del dispositivo en uso.

Funciones principales:

- Muestra versión de la aplicación.
- Identificación del usuario logueado.
- IMEI o número de serial del dispositivo.
- Fecha de última sincronización de datos.

Relación con otras vistas o procesos:

- Permite a los administradores validar configuraciones y diagnósticos.
- Facilita el soporte técnico al tener todos los datos del equipo de forma accesible.