Diseño e implementación de una plataforma para la gestión operativa de tiendas de barrio en Santiago de Cali

Santiago Rojas Colunge

Proyecto de grado entregado para obtener el título de
Magister en Ingeniería de Software

Dirigida por
Juan Carlos Martinez Arias

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Maestría en Ingeniería de Software
Santiago de Cali
26 de Febrero de 2026

Agradecimientos

A mi padre, **Mario Alonso Rojas Delgado**, por ser mi mayor ejemplo de perseverancia y trabajo incansable. Gracias por sembrar en mí la curiosidad y por brindarme el apoyo incondicional necesario para alcanzar cada una de mis metas. Tu guía ha sido el pilar fundamental de mi formación como hombre y profesional.

A mi madre, **Caty Alexandra Colunge Benavides**, cuyo amor infinito y sacrificios silenciosos han iluminado mi camino. Gracias por tu fe inquebrantable en mis capacidades y por ser el refugio constante en los momentos de mayor exigencia académica. Este logro es tan tuyo como mío.

A mis hermanos, **Mario y Miguel Rojas Colunge**, por su complicidad, por los debates compartidos y por ser el apoyo emocional que me mantuvo motivado durante este proceso de maestría. Su presencia es el recordatorio constante de la importancia de la familia en la consecución de los sueños.

A mi novia, **Geraldy Martinez Gordon**, por su paciencia, compañía y apoyo incondicional durante las largas jornadas de desarrollo. Gracias por creer en este proyecto desde el primer día, por tomarse el tiempo de descargar la aplicación y brindarme esa retroalimentación crítica y valiosa que ayudó a pulir los detalles de la experiencia de usuario de *TenderoApp*.

Al ingeniero **Juan Carlos Martinez Arias**, director de este proyecto de grado. Mi más profundo agradecimiento por su guía experta, su paciencia y por compartir sus amplios conocimientos técnicos en ingeniería de software. Su visión crítica y sus aportes arquitectónicos fueron determinantes para elevar la calidad de *TenderoApp*.

Un reconocimiento especial a la comunidad de tenderos de la ciudad de Cali, razón de ser de este trabajo. De manera muy particular, agradezco a **Mario Mellizo Girón**, propietario de la tienda *La Economía*, por abrirme las puertas de su negocio, por su tiempo y por su disposición para ser el primer validador de este prototipo. Así mismo, expreso mi gratitud a **Alejandra Caicedo**, **Juan David Barbery** y **Angela Bonilla**, quienes prestaron sus establecimientos para las pruebas de campo de la aplicación. Su colaboración y retroalimentación honesta permitieron que este proyecto trascendiera la teoría para convertirse en una herramienta con impacto social real.

Resumen

En Cali, Colombia, más de 15.000 tenderos enfrentan rezagos en la adopción tecnológica que limitan su eficiencia operativa, competitividad y sostenibilidad. Factores como la baja escolaridad, falta de herramientas digitales, resistencia al cambio y escasa conectividad agravan la situación. Este proyecto propuso diseñar e implementar una plataforma digital accesible, escalable y de bajo costo para la gestión de tiendas de barrio. La solución incluye funcionalidades como gestión de inventarios y un módulo de pedidos a proveedores basado en reglas de negocio. Se emplea una metodología centrada en el usuario, basada en principios de diseño inclusivo, desarrollo incremental y validación en campo con usuarios reales. Se espera mejorar la adopción tecnológica del sector y contribuir a la inclusión digital, fortaleciendo el tejido económico local mediante una solución de ingeniería de software con enfoque social.

Palabras Clave: Transformación Digital, Tenderos, Software Accesible, Plataformas Inclusivas, Tecnología Social.

Abstract

In Cali, Colombia, more than 15,000 neighborhood shopkeepers face gaps in technology adoption that limit their operational efficiency, competitiveness, and sustainability. Factors such as low education levels, lack of digital tools, resistance to change, and limited connectivity aggravate the situation. This project proposed the design and implementation of an accessible, scalable, and low-cost digital platform for neighborhood store management. The solution includes functionalities such as inventory management and a supplier ordering module driven by business rules. A user-centered methodology is applied, based on inclusive design principles, incremental development, and field validation with real users. The project aims to improve technological adoption in the sector and contribute to digital inclusion, strengthening the local economic fabric through a social-focused software engineering solution.

Keywords: Digital Transformation, Shopkeepers, Accessible Software, Inclusive Platforms, Social Technology.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En Colombia, el sector de los tenderos constituye un pilar socioeconómico fundamental con aproximadamente 500,000 establecimientos a nivel nacional (**bancoldex2023**). En el contexto local, en Santiago de Cali se encuentran registrados alrededor de 15,000 tenderos; no obstante, se estima que la cifra real supera los 17,000 establecimientos, considerando que muchos operan sin un registro formal (**alcaldiacali2023**; **elpais2023**). Estos comercios cumplen una función vital al proveer a los barrios productos de primera necesidad —como granos, carnes, aceites y vegetales—, atendiendo tanto a comunidades urbanas como rurales.

No obstante, pese a su relevancia económica, este sector enfrenta desafíos críticos asociados a la informalidad, la inseguridad y la creciente competencia con grandes superficies de formato *hard discount* como D1 y ARA, entre otras. En este escenario, destaca especialmente la marcada brecha en la adopción tecnológica, lo cual limita la competitividad de estos micro-negocios frente a los nuevos modelos de distribución.

En el contexto actual del siglo XXI, donde la digitalización de la información transforma los datos en un insumo clave para los modelos de negocio, los tenderos en Cali aún dependen de prácticas manuales para gestionar inventarios, ventas y relaciones con proveedores. Esta situación los sitúa en una posición de vulnerabilidad frente a las fluctuaciones en los patrones de consumo, así como ante crisis económicas o sanitarias de escala global, tal como se evidenció durante la emergencia del COVID-19.

La carencia de herramientas tecnológicas especializadas, sumada a una baja alfabetización digital (**velez2021**) y una limitada capacitación técnica, representan obstáculos significativos que impiden el desarrollo de soluciones sostenibles y accesibles para este sector. La ingeniería de software, a través del diseño centrado en el usuario, el desarrollo de plataformas accesibles y la inclusión de componentes educativos, puede ofrecer una respuesta efectiva que permita a los tenderos superar estos retos y contribuya decididamente al fortalecimiento de la economía popular.

El presente documento propuso diseñar e implementar un prototipo digital accesible para los tenderos en Cali, Colombia, con funcionalidades orientadas a optimizar la operación diaria y promover la apropiación tecnológica mediante contenidos informativos. La solución que se describe en las secciones posteriores busca contribuir al cierre de las brechas digitales en el sector y, de esta manera, aportar al desarrollo social mediante la aplicación rigurosa de la ingeniería de software.

1.1. Definición del problema

En Santiago de Cali, se estima la existencia de más de 17,000 tiendas de barrio que presentan rezagos significativos en la adopción tecnológica, lo cual limita su eficiencia operativa, competitividad y sostenibilidad a largo plazo. Factores estructurales como la baja escolaridad de los propietarios, la

carencia de herramientas digitales específicas, la resistencia al cambio organizacional y una conectividad deficiente en diversos sectores agravan esta situación (**larepublica2021**; **teamcore2022**).

Durante y tras la emergencia sanitaria derivada del COVID-19, un gran número de estos establecimientos sufrieron cierres temporales o reducciones drásticas en sus ingresos, debido principalmente a la ausencia de sistemas digitales que facilitaran su adaptación a las nuevas dinámicas de consumo. La persistente informalidad, los problemas de inseguridad y el limitado acceso a políticas públicas de fomento han profundizado estas dificultades, amenazando la viabilidad económica de estos negocios (**rico2021**; **fenalcocrisis2024**).

Si bien se han implementado iniciativas de apoyo como el programa *Compra Lo Nuestro* —el cual ha beneficiado a 25,534 microempresas con formación en productividad, diagnósticos de madurez digital, emisión gratuita de códigos de barras, soluciones de comercio electrónico y sellos de reconocimiento—, aún persisten barreras críticas que impiden a los tenderos capitalizar plenamente tales oportunidades. La meta nacional de alcanzar 100,000 participantes para el año 2022 subraya la magnitud del desafío pendiente (**mincit2021**).

En respuesta a este panorama, el presente proyecto propone el diseño e implementación de una plataforma digital accesible, escalable y de bajo costo operativo. Se adoptará una metodología centrada en el usuario, fundamentada en principios de diseño inclusivo, desarrollo incremental y una rigurosa validación en campo con usuarios reales. Se proyecta que esta intervención mejore la tasa de adopción tecnológica en el sector y promueva la inclusión digital, fortaleciendo el tejido económico local mediante una solución de ingeniería de software con marcado carácter social.

1.1.1. Planteamiento del problema

El comercio minorista tradicional en Colombia, representado por las tiendas de barrio, constituye un pilar socioeconómico fundamental. De acuerdo con (**cocacola2024**), estos establecimientos conforman la red de distribución más extensa del país, actuando como dinamizadores de la economía local. Sin embargo, este sector enfrenta una encrucijada crítica: un proceso de transformación digital asimétrico que compromete su sostenibilidad operativa en el mediano y largo plazo.

A pesar de su relevancia, existe un rezago técnico significativo. Diversos estudios evidencian que la adopción de soluciones de software se ve frenada por una dicotomía entre la limitación económica y la brecha de alfabetización digital. Al respecto, (**Espinosa2022**) identifica que la desconfianza hacia los sistemas transaccionales y el desconocimiento de los beneficios de la computación en la nube actúan como determinantes psicológicos y técnicos que inhiben la modernización. Esta resistencia se suma a lo planteado por (**teamcore2022**), quien señala que la complejidad de las interfaces actuales dificulta la experiencia de usuario (UX) para personas con baja escolaridad tecnológica.

El entorno competitivo ha exacerbado esta vulnerabilidad. La emergencia de los modelos de *hard discount* y la digitalización acelerada postpandemia han reconfigurado el mercado. Según (**Medina2025**), las tiendas de barrio se encuentran en una posición de desventaja estructural frente a grandes superficies que operan bajo economías de escala y arquitecturas de datos avanzadas. Si bien la crisis del COVID-19 evidenció la resiliencia del sector, también dejó al descubierto una

infraestructura logística fragmentada y una gestión de inventarios basada en métodos empíricos (**rico2021**), lo que limita la capacidad de respuesta ante fluctuaciones de la demanda.

En el contexto regional, la situación en Santiago de Cali es representativa. Aunque la (**alcaldiacali2023**) ha impulsado programas de fortalecimiento, estos suelen enfocarse en la inyección de capital semilla sin abordar el ciclo de vida del software necesario para una transformación sostenible. En contraste, la (**CCC2025**) subraya que la verdadera competitividad de la región depende de la integración de tecnologías 4.0 que permitan a los micronegocios participar en ecosistemas de omnicanalidad (**Sichaca2022**).

Desde una perspectiva técnica, el problema radica en el *mismatch* o desajuste arquitectural. Las soluciones POS (*Point of Sale*) y ERP (*Enterprise Resource Planning*) dominantes en el mercado colombiano poseen curvas de aprendizaje y costos de licenciamiento (TCO) inalcanzables para el tendero promedio (**siigopos2025**; **treinta2025**). Además, estas plataformas carecen de una lógica de negocio proactiva que facilite la transición desde el registro manual hacia una gestión de inventarios asistida y eficiente, tal como lo proponen modelos recientes de productividad para MiPymes en el país (**Taborda2024**).

Este vacío funcional se traduce en que la gestión de inventarios represente el desafío operativo más agudo. El 60 % de la muestra depende de métodos analógicos, mientras que el 40 % restante utiliza herramientas que no generan alertas oportunas ante el agotamiento de existencias. Lo anterior justifica el diseño e implementación de un motor de gestión inteligente fundamentado en reglas de negocio determinísticas y semaforización visual dentro de la arquitectura de *TenderoApp*.

La persistencia de esta brecha tecnológica no solo implica una pérdida de competitividad, sino el riesgo latente de desaparición de un modelo de negocio con alto impacto social. En consecuencia, se identifica la necesidad urgente de diseñar y desplegar una solución tecnológica integral, bajo un paradigma *serverless* y de bajo costo, que responda a las necesidades reales de gestión, inventario y previsión del tendero colombiano, garantizando su inserción efectiva en la economía digital.

1.1.2. Sistematización

¿De qué manera una plataforma tecnológica accesible, , puede apoyar la gestión operativa de las tiendas de barrio en Cali, Colombia, y facilitar la adopción de tecnologías digitales por los tenderos del país?

1.1.3. Sistematización del problema

- ¿Comó analizar las capacidades tecnológicas, necesidades operativas y barreras de adopción digital entre los tenderos de Cali?.
- ¿Comó diseñar una arquitectura de software modular, escalable y de bajo costo?.
- ¿Comó implementar un prototipo funcional accesible desde la web?.
- ¿Comó evaluar la usabilidad, utilidad percibida y nivel de apropiación tecnológica entre los usuarios piloto?.

1.2. Objetivos del proyecto

1.2.1. Objetivo General

Desarrollar un prototipo de una plataforma digital accesible para mejorar la gestión operativa de los tenderos en Cali, Colombia, incentivando la adopción tecnológica mediante herramientas funcionales adaptadas al contexto específico de estos micro-negocios.

1.2.2. Objetivos específicos

- Analizar las capacidades tecnológicas, necesidades operativas y barreras de adopción digital entre los tenderos de Cali.
- Diseñar una arquitectura de software modular, escalable y de bajo costo que se adapte a las necesidades de los tenderos en Cali, Colombia.
- Implementar un prototipo funcional accesible desde la web.
- Evaluar la usabilidad, la utilidad percibida y el nivel de apropiación tecnológica entre los usuarios piloto.

1.3. Delimitaciones y alcances

El proyecto incluirá el diseño, desarrollo e implementación de un prototipo tecnológico accesible para tenderos en Cali, enfocado en resolver las barreras tecnológicas existentes e implementar un diseño centrado en el usuario, que sea fácil de utilizar. De esta forma, se busca mejorar los procesos que afectan su gestión operativa. El prototipo contará con funcionalidades como control de inventarios, registro de ventas, módulo de pedidos a proveedores e integración de todos los módulos en una plataforma unificada.

el alcance se limitará a:

- La investigación tiene una muestra representativa en 3 barrios de la ciudad de Cali.
- El diseño de una arquitectura de software modular para la web.
- El desarrollo de un prototipo funcional que será validado con 3 tenderos de Cali, Colombia.
- Evaluación mediante un formulario de la usabilidad, utilidad percibida y adopcion del sistema.

No está contemplado en esta etapa

- El desarrollo con sistemas financieros externos o entidades gubernamentales.
- La implementación masiva a nivel regional o nacional.
- La comercialización o el mantenimiento posterior a la etapa de validación.

El proyecto estará enfocado únicamente en la zona urbana de Cali, Colombia, y se limitará a un período de ejecución de seis meses. Las posibles limitaciones incluyen dificultades en la conectividad digital, resistencia cultural al cambio tecnológico y restricciones presupuestales o de tiempo. Se analizará cada objetivo específico, buscando delimitar con mayor precisión qué se va a hacer y qué no. Asimismo, se dejan claras las limitaciones ya identificadas en la solución que se va a construir.

1.4. Justificación del trabajo de grado

La transformación digital de las tiendas de barrio se consolidó como una necesidad imperativa para garantizar su sostenibilidad económica y su vigencia en el ecosistema urbano de Cali. En una ciudad que contó con más de 15.000 establecimientos de este tipo, las brechas tecnológicas actuaron como catalizadores de desigualdad socioeconómica (**Medina2025**; **CCC2025**). Esta población, concentrada principalmente en los estratos 1, 2 y 3, enfrentó barreras críticas como la falta de sistemas de información estructurados y una baja alfabetización digital funcional, factores que los excluyeron históricamente de los beneficios de la economía digital (**Espinosa2022**).

El abordaje de estas problemáticas mediante la ingeniería de software permitió sistematizar el flujo de información transaccional de los tenderos, lo que generó un impacto social directo (**Sichaca2022**). Al proporcionar una herramienta que equilibró la funcionalidad operativa con la formación digital, se logró que el tendero aumentara su eficiencia, redujera mermas de inventario y mejorara su competitividad frente a los modelos de *hard discount* que dominaron el mercado regional (**Medina2025**).

Desde la perspectiva de la Ingeniería de Software, este proyecto resultó relevante por la aplicación de arquitecturas modulares y principios de diseño inclusivo para poblaciones con baja escolaridad tecnológica (**nosolousabilidad2006**). La integración de funciones de gestión, alineada con modelos de previsión para MiPymes en Colombia (**Taborda2024**), permitió transformar el registro empírico en una administración basada en datos.

Finalmente, en el ámbito académico y técnico, este trabajo justificó su rigor mediante la implementación del modelo C4 para la visualización arquitectural y el uso de metodologías ágiles. La solución no solo demostró ser técnicamente robusta, sino que se validó como una propuesta socialmente pertinente que mitigó la brecha operativa en el sector micro-comercial de Santiago de Cali.

1.5. Metodología de la investigación

La metodología adoptada para este proyecto se fundamentó en una estructura lógica que articuló la investigación aplicada con el desarrollo tecnológico iterativo. Este enfoque permitió responder al problema de la brecha digital mediante la creación de una solución técnica validada en contextos reales.

1.5.1. Estrategia de Investigación: Investigación Aplicada

Se seleccionó la **investigación aplicada** como marco orientador, dado que este proceso se enfocó en transformar el conocimiento técnico de la ingeniería de software en una solución práctica para las necesidades operativas de los tenderos. A través de este modelo, se adaptaron conceptos de arquitecturas en la nube y diseño inclusivo para generar un prototipo transferible a la realidad del micro-comercio en Cali (**lozada2014**).

El uso de esta estrategia permitió fortalecer la colaboración con los usuarios finales (tenderos) y asegurar que el conocimiento generado tuviera un valor económico y social directo, mejorando la productividad del sector mediante la innovación tecnológica local.

1.5.2. Estrategia de Desarrollo: Ciclo Iterativo e Incremental

El desarrollo técnico se ejecutó bajo un modelo **iterativo e incremental**, dividiendo el trabajo en ciclos de entrega funcional. Esta metodología permitió gestionar las expectativas de los usuarios mediante entregas tangibles y mitigar riesgos técnicos de forma temprana.

La sinergia entre la investigación aplicada y el desarrollo iterativo facilitó una **adaptabilidad constante**; mientras que la investigación identificaba problemas de usabilidad en campo, el ciclo iterativo permitía ajustar la arquitectura del software para responder a dichas condiciones de manera ágil.

1.5.3. Proceso de Sistematización y Análisis de Datos

El proceso metodológico integró técnicas cualitativas y cuantitativas para garantizar el rigor de los resultados:

- **Recolección:** Se utilizaron encuestas y entrevistas semiestructuradas para la fase de diagnóstico y observación directa durante las pruebas de usabilidad.
- **Organización y Sistematización:** Los requerimientos se categorizaron mediante la técnica MoSCoW, priorizando las funcionalidades de mayor impacto operativo. Los registros de las pruebas se tabularon para identificar patrones de error comunes.
- **Análisis e Interpretación:** Se emplearon métricas de éxito basadas en la eficiencia de las transacciones y la utilidad percibida por el usuario, lo que permitió validar la efectividad de la arquitectura serverless frente a los métodos analógicos tradicionales.

1.5.4. Fases del Proyecto

Fase 1: Diagnóstico y análisis contextual

Se realizó un levantamiento de información mediante entrevistas a tenderos en Santiago de Cali para caracterizar su nivel de alfabetización digital. Como resultado, se definieron los requerimientos funcionales y no funcionales, centrados en la baja complejidad y alta disponibilidad.

Fase 2: Diseño de la solución tecnológica

Se diseñó una arquitectura de software modular basada en el modelo C4, priorizando principios de escalabilidad y bajo costo. Las interfaces se proyectaron bajo un enfoque *mobile-first* para asegurar la compatibilidad con dispositivos de bajo rendimiento.

Fase 3: Desarrollo del prototipo

La implementación se llevó a cabo mediante la integración de **Design Thinking y Scrum**. Se ejecutaron *sprints* de dos semanas donde se prototiparon y desarrollaron módulos clave como la gestión de inventarios y el punto de venta (POS). En esta etapa se priorizó la automatización de la lógica de negocio para reducir la carga cognitiva del usuario, prescindiendo de infraestructuras complejas y enfocándose en una arquitectura serverless pura.

Fase 4: Validación con usuarios reales

Se ejecutaron pruebas de usabilidad con una muestra seleccionada de tenderos en Cali. Se recolectaron datos sobre la interacción con el sistema, evaluando la facilidad de registro de ventas y la claridad de los reportes generados.

Fase 5: Evaluación de resultados

Finalmente, se analizaron los resultados cualitativos y cuantitativos obtenidos en la validación. Se realizaron ajustes técnicos finales al prototipo y se documentaron las lecciones aprendidas respecto a la apropiación tecnológica en sectores de baja escolaridad digital.

1.6. Resultados obtenidos

El desarrollo y validación de *TenderoApp* permitió consolidar una solución tecnológica alineada con los requerimientos de Ingeniería de Software planteados inicialmente. Los resultados obtenidos se detallan a continuación:

- **Plataforma Progresiva Operativa:** Se implementó con éxito un prototipo funcional basado en React y Vite, configurado como una *Progressive Web App* (PWA). La solución garantiza la persistencia de datos y operatividad en entornos de baja conectividad mediante el uso de *Service Workers* y la sincronización en tiempo real de Google Cloud Firestore.

- **Calidad y Estabilidad del Software:** Se alcanzó una cobertura de pruebas unitarias superior al 80 % utilizando la suite de Vitest, asegurando la integridad de los cálculos de inventario y ventas. En términos de rendimiento, la plataforma obtuvo un puntaje de 94/100 en la auditoría de *Performance* de Google Lighthouse, con un *Largest Contentful Paint* (LCP) inferior a los 2.5 segundos.

- **Arquitectura de Bajo Costo (TCO):** Mediante el paradigma *Serverless*, se logró una infraestructura con costo operativo cero en la etapa de prototipado, delegando la autenticación, base de datos y almacenamiento a los servicios gestionados de Firebase. Esto valida la viabilidad económica del proyecto para micro-comercios.

- **Analítica de Inventarios:** Se integró un motor de alertas basado en reglas de negocio que permite la semaforización de *stock* (mínimos y máximos). Esto facilitó a los usuarios la identificación inmediata de productos críticos para reabastecimiento, mejorando la toma de decisiones logística sin requerir una alta carga cognitiva.

- **Documentación Técnica Integral:** Se generó una arquitectura documentada bajo el estándar del modelo C4 (**c4model**), permitiendo una comprensión clara de la interacción entre los contenedores de la aplicación y los servicios externos de la nube.

Capítulo 2

Marco de referencia

2.1. Marco Teórico

2.1.0.1. Arquitectura de software

La arquitectura de software define la estructura y organización de un sistema, describiendo sus componentes principales, las relaciones entre ellos y los principios de diseño que rigen su evolución (**Ford2021**; **Davis2019**). En una plataforma web para tenderos, una arquitectura adecuada es clave para garantizar modularidad, escalabilidad y mantenibilidad, aspectos esenciales en un entorno de recursos limitados y validación iterativa.

Para documentar estas decisiones, Simon Brown presenta el modelo C4 como una técnica de documentación arquitectónica ligera y escalable. Este modelo se basa en cuatro niveles de abstracción: el **Contexto**, que ofrece una visión global del sistema y sus usuarios o sistemas externos; los **Contenedores**, que representan las unidades desplegables como aplicaciones, servicios o bases de datos; los **Componentes**, que detallan los módulos internos de cada contenedor con responsabilidades claras; y finalmente, las **Clases (Código)**, que especifican el detalle de las piezas de código más relevantes.

En cuanto a los patrones de arquitectura aplicables a este proyecto, destacan varias aproximaciones. La **Arquitectura en capas (Layered Architecture)** permite separar la aplicación en presentación, lógica de negocio y acceso a datos, facilitando pruebas y despliegues independientes (**Gorton2023**). Asimismo, el enfoque de **Microservicios** favorece que cada servicio implemente una funcionalidad de negocio concreta comunicándose mediante APIs RESTful, lo que mejora la escalabilidad horizontal y la resiliencia (**Newman2020**). Finalmente, el patrón **Microkernel (Plugin Architecture)** resulta útil al permitir añadir módulos como inventarios, reportes y pedidos de forma incremental sin alterar el núcleo del sistema, facilitando así su extensibilidad.

El diseño arquitectónico debe responder a una serie de atributos de calidad fundamentales. La *modularidad* favorece el desarrollo incremental y la validación con usuarios en cada iteración (**Ford2021**), mientras que la *escalabilidad* asegura que el sistema soporte el crecimiento del número de tiendas y transacciones sin degradar el rendimiento (**Davis2019**). Adicionalmente, la *mantenibilidad* reduce el costo de cambios y corrección de errores, y la *disponibilidad* garantiza un acceso continuo, el cual es vital para las operaciones diarias de los tenderos.

Dada la conectividad variable y la diversidad de dispositivos de los tenderos, se proponen entornos de despliegue específicos. En primer lugar, las **PWA (Progressive Web App)** ofrecen una instalación ligera desde el navegador, un funcionamiento offline-first mediante Service Workers y almacenamiento local en IndexedDB, permitiendo que la aplicación responda incluso sin conec-

tividad (**GoogleWebDev2021**; **MDNOffline2025**). En segundo lugar, se emplea un **Backend serverless** basado en funciones en la nube (como AWS Lambda o Azure Functions) que escalan automáticamente para atender la demanda y se facturan por tiempo de ejecución (pay-per-use), eliminando así la gestión de servidores físicos o virtuales. Para complementar esto, se implementa una **Sincronización diferida** mediante colas de eventos en el cliente que se registran cuando el sistema está offline y se envían al backend al restablecer la conexión, usando la Background Synchronization API para garantizar el envío fiable de operaciones pendientes (**MDNBackgroundSync2024**; **MSPeriodicBackgroundSync2023**).

2.1.1. Aplicaciones Web Progresivas (PWA)

Las **Progressive Web Apps (PWA)** representan una evolución en el desarrollo de software que busca unificar la ubicuidad de la web con la experiencia de usuario de las aplicaciones nativas. Según (**Wargo2020**), una PWA se define por ser instalable, enlazable y capaz de funcionar de manera independiente a la conectividad de red. Para el contexto de esta investigación, la adopción de este paradigma fue fundamental debido a su capacidad de despliegue multiplataforma con una única base de código, lo que redujo significativamente los costos operativos y de mantenimiento.

2.1.1.1. Arquitectura Offline-First y Service Workers

El componente crítico que diferencia a una PWA de un sitio web convencional es el **Service Worker**. Este es un script que el navegador ejecuta en segundo plano, actuando como un proxy de red programable (**GoogleDev2023**). Investigaciones recientes realizadas por (**Sirigiri2023**) destacan que la implementación de estrategias *offline-first* permite la persistencia de datos mediante el uso de **IndexedDB**, asegurando que el sistema sea funcional incluso sin acceso a internet. En el caso de los tenderos de Cali, esta capacidad garantizó que las transacciones de venta no se perdieran durante interrupciones en el servicio de red.

2.1.1.2. Impacto en la Experiencia de Usuario y Accesibilidad

La accesibilidad es un factor determinante para la adopción tecnológica en micro-negocios. Estudios de (**Ridho2023**) demuestran que las PWA mejoran significativamente el *engagement* del usuario en regiones en desarrollo, debido a que no requieren descargas pesadas desde tiendas de aplicaciones oficiales (App Store o Play Store) y ocupan una fracción del almacenamiento en el dispositivo móvil. Estas características permitieron que tenderos con dispositivos de gama media o baja pudieran interactuar con la plataforma sin degradar el rendimiento del hardware.

2.1.1.3. Transformación Digital

La transformación digital de los pequeños comercios en Colombia ha sido objeto de diversos estudios en los últimos años. Acceder a las tecnologías digitales se ha identificado como un factor importante para mejorar la competitividad y sostenibilidad de las micro, pequeñas y medianas empresas (MiPymes) en el país. Según la Cámara Colombiana de Comercio Electrónico (CCCE),

las compras electrónicas permiten a los negocios en Colombia acceder a insumos y mercancías a precios más competitivos, ampliando de esta forma la variedad de sus productos tanto a nivel nacional como internacional (**ccce2024**).

2.1.1.4. Diseño de soluciones tecnológicas

Dentro del diseño tecnológico, enfocarlo al usuario se ha convertido en un pilar fundamental para la construcción de soluciones tecnológicas efectivas. Este enfoque pone al usuario como probador de la funcionalidad del sistema y de esta forma el sistema diseñado será adecuado para acoplarse a sus necesidades que realmente le aportan valor a la idea de negocio. Empresas colombianas especializadas en consultoría y diseño de UX/UI, como UZER, son reconocidas por aplicar estas metodologías para que el sistema sea para el usuario y esta experiencia con el aplicativo sea la adecuada (**aguayo2024**).

2.1.1.5. Accesibilidad digital y usabilidad

La accesibilidad digital y la usabilidad son pilares muy importantes en el desarrollo de software para los pequeños comercios. Garantizar que las plataformas digitales sean intuitivas y adaptadas a las necesidades de los usuarios asegura que tengan una adopción efectiva. En Colombia, iniciativas como la de Aguayo.co han promovido la implementación del diseño centrado en el usuario, enfatizando la importancia de la empatía y la iteración continua en el diseño de interfaces (**aguayo2024**).

2.1.1.6. Apropiación tecnológica

Estrategias como el microlearning y aprendizaje móvil han demostrado tener una efectividad al momento de la capacitación de tenderos en el uso de nuevas herramientas digitales. Programas de formación virtual, como "Capacítate" de Colombia Productiva, ofrecen más de 300 capacitaciones gratuitas; de esta forma, facilitan el aprendizaje y adaptan las necesidades a los pequeños empresarios (**colombiaproductiva2024**).

2.1.1.7. Metodologías ágiles

Las metodologías ágiles en el desarrollo de tecnologías permiten una adaptación rápida a las necesidades actuales del mercado y del usuario final. Estas metodologías posibilitan una retroalimentación temprana para realizar ajustes mientras se va desarrollando el proyecto. Las empresas que no utilizan estas metodologías tienen un 50 % de riesgo de no alcanzar sus objetivos (**lanotaeconomica2024**) (**siglo212024**).

2.1.1.8. Metodologías centradas en el usuario

Design Thinking

La metodología *Design Thinking* es ampliamente utilizada en el diseño de productos, la educación y la experiencia de usuario (UX). Se trata de un proceso iterativo que permite resolver

problemas de manera rápida y sencilla. Las fases que constituyen esta metodología son: empatizar, definir, idear, prototipar y evaluar. El flujo de trabajo se representa como se muestra en la Figura 2.1, basada en el modelo propuesto por (**dkvdesignthinking2024**).

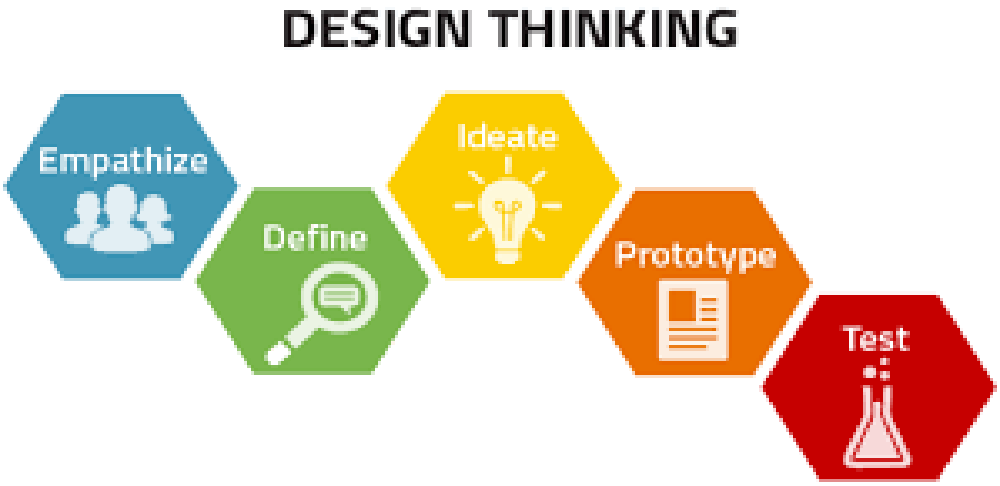

Figura 2.1: Fases del modelo Design Thinking (**dkvdesignthinking2024**)

Diseño centrado en el usuario (DCU o UCD)

El Diseño Centrado en el Usuario implica una participación activa de personas reales para evaluar los sistemas o prototipos; de esta forma, se permite detectar problemas de usabilidad desde la perspectiva del usuario final, que en este contexto sería el tendero colombiano.

Entre sus principales métodos se encuentran las **pruebas con usuarios**, donde se asignan tareas específicas para observar errores, tiempos de adaptabilidad y comportamientos. Asimismo, se emplean **entrevistas**, que son conversaciones que permiten obtener las percepciones de los usuarios con respecto al prototipo, y **cuestionarios**, que fungen como evaluaciones estructuradas para obtener retroalimentación cuantitativa. Por último, la **observación contextual** facilita la evaluación en el entorno real y natural del usuario para identificar patrones de uso reales. Todo esto permite enfatizar en la aplicación de estas técnicas en etapas tempranas del diseño y de forma iterativa, con el fin de maximizar la utilidad de los hallazgos (**nosolousabilidad2006**).

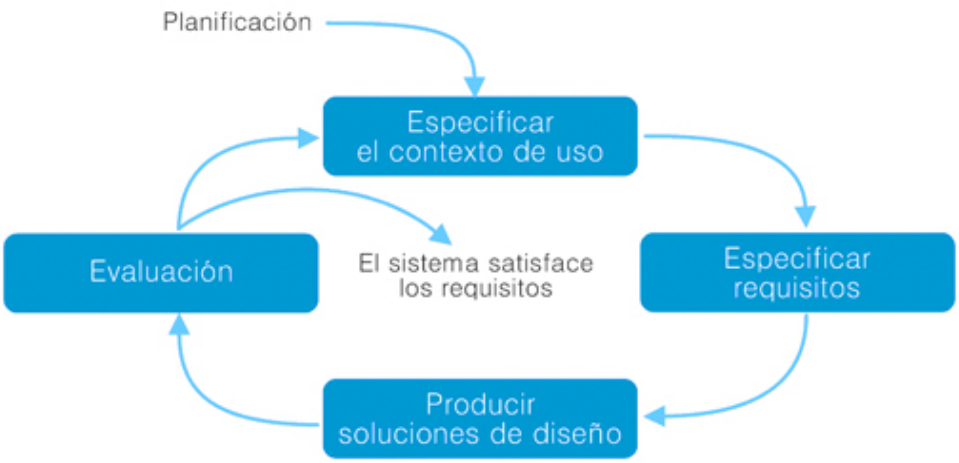

Figura 2.2: Métodos de evaluación centrados en el usuario. Fuente: (**nosolousabilidad2006**)

2.2. Estado del Arte

En cuanto al estado del arte, existe una gran variedad de sistemas disponibles en la web, programas que han permitido a algunas personas organizar sus inventarios de manera más eficiente.

2.2.0.1. Programa Tiendas Digitales y creciendo Juntos

Este programa, desarrollado por Fenalco y Bavaria en Colombia, ha buscado mejorar la competitividad de los tenderos mediante el uso de capacitaciones, subsidios y tecnologías digitales (**bavaria2024**). A través de esta iniciativa se han beneficiado miles de tenderos, promoviendo su formalización y modernización. Aunque el programa entrega herramientas básicas y formación general, se complementa con propuestas como el presente proyecto, el cual plantea una solución tecnológica concreta, escalable y centrada en el usuario.

2.2.0.2. Herramientas tecnológicas en tiendas de barrio Colombianas

El estudio destaca la ausencia de herramientas integrales que ayuden al tendero colombiano, especialmente aquellas que contemplen su nivel educativo, su contexto operativo y sus barreras de acceso digital. Herramientas como Odoo o SAP son herramientas que para el contexto de los tenderos son muy complejas. En ese sentido, estas herramientas tecnológicas muestran un vacío en a las soluciones tecnológicas que necesitan los tenderos y suelen ser validadas empíricamente, lo cual refuerza la necesidad de desarrollar un proyecto que se acople a las necesidades de los tenderos actuales (**Cadavid2024**).

Elemento	Programa Bavaria/Fenalco	Este Proyecto
Enfoque	Mejora de la competitividad y formalización comercial	Digitalización operativa enfocada en tenderos
Ejecución	Capacitaciones, subsidios y acompañamiento empresarial	Desarrollo de una plataforma digital accesible y funcional
Tecnología	Dotación tecnológica básica (equipos, herramientas)	Sistema integrado con gestión de inventarios, ventas, reportes y formación
Formación	Contenidos generales sobre administración y ventas	Módulo de integración para contenidos educativos, diseñado para conectar con plataformas externas como el SENA o Colombia Productiva, sin generar contenidos propios.
Evaluación	No detallada en la fuente	Validación en campo con usuarios reales y mejora iterativa
Escalabilidad	Enfoque institucional con cobertura limitada	Modelo replicable, económico y adaptable a otras regiones

Tabla 2.1: Comparación entre el programa de Bavaria/Fenalco y el proyecto de digitalización propuesto

2.2.0.3. Herramientas y su aplicabilidad local

Opencart es una plataforma de comercio electrónico de código abierto que permite a los pequeños negocios crear y administrar tiendas virtualmente. Ofrece funciones como la gestión de productos, inventarios, pedidos y múltiples métodos de pago. Es una app gratuita pero su personalización requiere de conocimientos técnicos, lo que limita su accesibilidad para tenderos sin experiencia digital (**deploi2025**).

Por otra parte, Odoo es una herramienta empresarial que incluye herramientas para contabilidad, punto de venta, CRM e inventario. Presenta una solución integral, la curva de aprendizaje puede presentar una barrera significativa para los tenderos en Colombia con baja alfabetización digital. Algunas de sus funciones son sujetas a licencias o suscripción (**softwareadvice2025**).

Ambas herramientas muestran un vacío en el desarrollo de plataformas adaptadas al contexto de Colombia, donde no se contempla que las tiendas de barrio no solo comercializan productos comestibles y no comestibles, sino que también funcionan como espacios de encuentro comunitario. La complejidad de estas soluciones y su falta de enfoque local justifican diseñar una plataforma centrada en el usuario colombiano, de fácil adopción, validada empíricamente y orientada a los retos reales del contexto colombiano.

2.2.0.4. Siigo

Siigo es una solución contable y administrativa en la nube, ampliamente utilizada en Colombia por pequeñas y medianas empresas. Ofrece funcionalidades como facturación electrónica, control de inventarios, generación de reportes financieros y gestión de ventas. Aunque representa un referente importante en el ecosistema de software empresarial, su enfoque está orientado principalmente a empresas formalizadas y con cierto grado de alfabetización digital.

En contraste, el proyecto propuesto está centrado en tenderos de barrio que, en su mayoría, operan desde la informalidad o están en proceso de formalización. La solución que se plantea busca ser accesible, de bajo costo y adaptada a las condiciones socioeconómicas del tendero colombiano. Si bien inicialmente se contemplaba un componente de microcontenidos para formación digital, este ha sido replanteado debido a limitaciones de tiempo en la fase de implementación. Aun así, el enfoque se mantiene en construir una plataforma fácil de usar y enfocada en mejorar la gestión operativa, con énfasis en inventarios, ventas y reportes.

La comparación con Siigo permite identificar que, aunque existen soluciones robustas en el mercado, aún hay una brecha importante en cuanto a accesibilidad, usabilidad y pertinencia en contextos de bajo nivel de digitalización. Por tanto, el presente proyecto no busca competir con estas plataformas, sino atender una necesidad puntual y no resuelta en un sector históricamente excluido de la transformación digital (**siigo2024**).

2.3. Resumen del capítulo

El presente capítulo establece los fundamentos conceptuales, teóricos y técnicos que sustentan el desarrollo de *TenderoApp*. La estructura se divide en dos ejes fundamentales: el marco teórico,

que define la arquitectura y metodologías, y el estado del arte, que analiza el panorama actual de soluciones tecnológicas para el sector.

2.3.0.1. Marco Teórico

Se aborda la **Arquitectura de Software** como el pilar estructural del sistema. Se adopta el **Modelo C4** (Contexto, Contenedores, Componentes y Código) propuesto por Simon Brown para garantizar una documentación técnica ligera, escalable y comprensible para diferentes niveles de interés. Dentro de este marco, se justifican decisiones estratégicas como la adopción del **Paradigma PWA (Progressive Web Apps)**, definido como la tecnología base para permitir un despliegue multiplataforma con una única base de código, destacando el uso de *Service Workers* y una estrategia *Offline-First* para garantizar la operatividad en entornos de conectividad inestable.

Igualmente, se fundamenta el uso de una **Infraestructura Serverless** basada en servicios en la nube de pago por uso (Google Cloud/Firebase). Esta elección técnica elimina la carga operativa de administración de servidores y optimiza el Costo Total de Propiedad (TCO), un factor crítico para los micro-comercios. Por otro lado, las **Metodologías de Diseño** integran principios de *Design Thinking* y **Diseño Centrado en el Usuario (DCU)** con el objetivo de asegurar que la interfaz de *TenderoApp* sea inclusiva, minimizando la carga cognitiva para usuarios con diversos niveles de alfabetización digital.

2.3.0.2. Estado del Arte

Se realiza un análisis comparativo entre las soluciones existentes en el mercado colombiano y la propuesta de valor de este proyecto. En primer lugar, se analizan **Programas Institucionales** como las iniciativas de Fenalco y Bavaria (*Tiendas Digitales*), concluyendo que, si bien ofrecen formación valiosa, persiste un vacío en herramientas operativas personalizadas que se adapten al flujo diario y real del tendero. En segundo lugar, al evaluar **Sistemas Comerciales** referentes como Siigo, Treinta y Odoo, se destaca que soluciones como *Siigo* presentan altas curvas de aprendizaje y costos de licenciamiento prohibitivos para negocios informales, mientras que plataformas como *Treinta* resultan útiles pero limitadas en la lógica de negocio proactiva y automatizada necesaria para el sector.

2.4. Conclusión del Capítulo

El marco de referencia permite concluir que la transformación digital efectiva de las tiendas de barrio en Cali no depende exclusivamente del acceso al hardware, sino de la creación de herramientas que equilibren la **robustez técnica** (arquitectura moderna) con la **simplicidad operativa** (UX inclusivo). Este diagnóstico justifica la construcción de una plataforma basada en reglas de negocio y semaforización automatizada que responda fielmente a la realidad socioeconómica local.

Capítulo 3

Desarrollo del Proyecto

3.1. Análisis de capacidades tecnológicas, necesidades operativas y barreras de adopción digital

3.1.1. Análisis de resultados de campo y caracterización

A partir del levantamiento de información realizado en cinco establecimientos comerciales (tiendas de barrio y misceláneas) de la ciudad de Cali, se identificaron patrones críticos de comportamiento y requerimientos funcionales. Este diagnóstico permite alinear la arquitectura propuesta con las realidades operativas del sector.

3.1.1.1. Infraestructura de conectividad y heterogeneidad de hardware

Se constató que el 100 % de los tenderos entrevistados cuenta con conectividad a internet estable. No obstante, existe una marcada asimetría en el acceso a hardware especializado: un 20 % carece de dispositivos digitales para la administración, mientras que un 80 % utiliza una mezcla no estandarizada de smartphones y computadores personales. Dicha heterogeneidad tecnológica fundamenta la adopción de una **Progressive Web App (PWA)**, garantizando una experiencia de usuario consistente sin las barreras de fricción de las aplicaciones nativas.

3.1.1.2. Ecosistema de software y percepción de valor

Actualmente, el uso de software es fragmentado: mientras que un 40 % utiliza sistemas contables robustos como *Siigo*, el resto fluctúa entre aplicaciones ligeras como *Treinta* (20 %) o el nulo uso de herramientas (20 %). Un hallazgo determinante es que el 60 % de los usuarios percibe que la tecnología aporta mucho a su gestión; sin embargo, esta percepción positiva se ve opacada por la complejidad de las interfaces actuales, lo que sugiere una necesidad de simplificación en el flujo de trabajo (*workflow*).

3.1.1.3. Gestión de inventarios: El punto de dolor crítico

La gestión de inventarios representa el desafío operativo más agudo. El 60 % de la muestra depende de métodos analógicos (papel y lápiz), y un 40 % utiliza herramientas digitales de propósito general o especializado. Es relevante destacar que usuarios de sistemas POS comerciales manifestaron que sus plataformas actuales carecen de lógica proactiva; específicamente, mencionaron que

el sistema no genera alertas oportunas ante el agotamiento inminente de stock. Este vacío funcional justifica la integración de un motor de analítica inteligente basado en reglas de negocio y semaforización automatizada en la arquitectura de *TenderoApp*.

3.1.1.4. Barreras de adopción digital

Las barreras identificadas no son exclusivamente económicas. Siguiendo a **Espinosa2022<empty citation>**, se observa que factores como la falta de tiempo operativo y la desconfianza técnica (20 %) son tan restrictivos como el costo de licenciamiento (20 %). Un 40 % de los usuarios asume que su sistema actual es suficiente, lo que impone el reto de demostrar un valor agregado superior mediante la automatización de decisiones y la reducción de carga cognitiva para el tendero.

3.1.2. Resultados cuantitativos y priorización

Para facilitar la lectura técnica de estos hallazgos, se presentan las siguientes tablas resumen que categorizan la infraestructura y priorizan las barreras que la solución técnica debe mitigar.

Tabla 3.1: Capacidades digitales y de infraestructura identificadas (n=5)

Variable de Infraestructura	**Frecuencia**	**Porcentaje**
Disponibilidad de conectividad a internet	5	100 %
Uso exclusivo de métodos analógicos	1	20 %
Acceso a hardware (PC/Smartphone)	4	80 %
Familiaridad con sistemas POS/ERP	3	60 %
Interés explícito en validación de prototipo	5	100 %

Tabla 3.2: Priorización MoSCoW de barreras a mitigar mediante el diseño

Categoría	**Barrera Identificada**	**Prioridad de Diseño**
Must (Obligatorio)	Complejidad de uso y falta de tiempo	Crítica
Should (Deseable)	Costos de licenciamiento y mantenimiento	Alta
Could (Opcional)	Resistencia al cambio en usuarios con software previo	Media
Won't (No en esta fase)	Integración con sistemas contables bancarios	Baja

3.1.3. Benchmarking de plataformas digitales y análisis de la brecha funcional

Para definir el alcance técnico de *TenderoApp*, se realizó un análisis comparativo de las soluciones dominantes en el mercado colombiano. Este *benchmarking* permitió establecer una taxonomía de funcionalidades esenciales, integrando las directrices de inclusión digital propuestas por (**plowshare2025**) con las necesidades operativas detectadas en la fase de diagnóstico. Los criterios de evaluación seleccionados abarcaron: gestión de inventarios, terminal de punto de venta (POS), generación de reportes analíticos, gestión de suministros y el cumplimiento de la normativa vigente ante la DIAN.

El ecosistema local incluye desde aplicaciones móviles ligeras como *Treinta* (**treinta2025**) y propuestas de impacto social como *Super Vecino* (**citytvsupervecino2024**), hasta plataformas ERP robustas como *Siigo POS* (**siigopos2025**) y *Alegra* (**alegravssiigo2025**). La Tabla 3.3 resume las capacidades funcionales de estas herramientas.

Tabla 3.3: Comparativa funcional de plataformas para el comercio minorista

Plataforma	Inv.	POS	Rep.	Prov.	FE*	Enfoque de Soporte
Siigo POS	Sí	Sí	Sí	Lim.[a]	Sí	Corporativo / Nacional
Treinta	Básico	Sí	Simp.	No	No[b]	Digital / App ligera
Alegra	Sí	Sí	Sí	Int.[c]	Sí	Regional / Pymes
Tiendatek	Sí	Sí	Sí	Sí	Pos.[d]	Tiendas tradicionales
Super Vecino	Sí	Sí	Bás.	No	Sí	Social / Bajo costo
SoftwarePOS	Sí	Sí	Sí	Mod.[e]	Pos.	Planes modulares

* FE: Facturación Electrónica ante la DIAN.
[a] El flujo de proveedores suele requerir módulos contables adicionales.
[b] Actualmente no se posiciona como proveedor tecnológico de FE.
[c] Requiere integración con el ecosistema contable de Alegra.
[d] Sujeto a planes específicos y conectores externos.
[e] Según los módulos contratados por el usuario.

3.1.4. Matriz de Decisiones Arquitectónicas basadas en el Diagnóstico

A partir de la caracterización realizada en el contexto de Cali, se definieron decisiones técnicas críticas para garantizar la viabilidad del sistema:

- **Adopción de PWA:** Debido a la heterogeneidad de hardware detectada (80 % de los tenderos usa una mezcla de PC y smartphones) , se optó por una *Progressive Web App* utilizando React y Vite. Esto elimina la fricción de descarga y asegura compatibilidad multiplataforma con un solo código base.

- **Estrategia Offline-first:** Dado que el 60 % de los tenderos depende de métodos analógicos, se implementó persistencia local con *IndexedDB* para el módulo de ventas. Esto permite congelar transacciones y operar sin conexión estable, sincronizando con Firestore una vez se restablezca el servicio.

3.1.4.1. Análisis de Costos y Accesibilidad Económica

Un factor determinante para la adopción tecnológica en estratos 1, 2 y 3 de Cali es el **Costo Total de Propiedad (TCO)**. Aunque existen opciones gratuitas en su capa base, las funcionalidades de valor agregado (como la analítica o la facturación electrónica) imponen una carga financiera mensual que compite con los estrechos márgenes de rentabilidad del tendero (**simeh2023**). La Tabla 3.4 detalla la estructura de costos consultada.

Tabla 3.4: Comparativa de costos operativos y modalidades de acceso

Plataforma	Modalidad	Costo (aprox.)	Observaciones Técnicas
Siigo POS	Suscripción mensual	\$60k–70k COP	Integración nativa DIAN.
Alegra	Plan Pyme	\$45k–55k COP	Enfoque contable avanzado.
Tiendatek	Plan Estándar	\$50k–70k COP	Hardware dedicado opcional.
Super Vecino	Modelo social	No divulgado	Foco en formalización express.
Treinta	*Freemium*	\$0 COP (Core)	Limitado en analítica proactiva.
SoftwarePOS.co	Modular	$\geq$ \$45k COP	Escalable según la necesidad.

3.2. Diseño y Arquitectura de la Solución

Una vez definidos los requerimientos y los flujos de interacción, esta sección detalla la arquitectura lógica y física de *TenderoApp*. La solución se diseñó bajo una premisa de alta disponibilidad y bajo costo operativo, seleccionando un ecosistema tecnológico que mitiga las barreras de adopción identificadas.

3.2.1. Justificación de la Pila Tecnológica

A diferencia de los sistemas POS tradicionales, la selección de herramientas para este proyecto prioriza la ligereza del cliente y la escalabilidad del servidor:

- **React + Vite (Core de Desarrollo):** Se seleccionó React por su modelo de reconciliación basado en componentes, facilitando la creación de interfaces táctiles reactivas. Se integró Vite como herramienta de construcción (*Build Tool*) para garantizar tiempos de carga instantáneos en el navegador mediante el uso de módulos ES nativos, eliminando la latencia de carga inicial en dispositivos de gama media.

- **Paradigma PWA (Acceso Multiplataforma):** La aplicación se configuró como una *Progressive Web App* para permitir su instalación sin pasar por tiendas oficiales. Esto resuelve el problema de almacenamiento limitado en los teléfonos de los tenderos y habilita el uso de *Service Workers* para la estrategia *offline-first*.

- **Infraestructura BaaS con Firebase:** Se adoptó un enfoque *Serverless* para eliminar el mantenimiento de servidores. Se delegó la seguridad de identidad a *Firebase Auth* y el alma-

cenamiento a *Cloud Firestore*, permitiendo que la arquitectura soporte miles de transacciones concurrentes con un costo operativo marginal (TCO cercano a cero para el prototipado).

3.2.2. Estrategia de Datos y Aislamiento (Multi-tenancy)

Para asegurar que la información de una tienda sea inaccesible para otra, se implementó un aislamiento de datos a nivel lógico mediante un patrón de **Silo de Datos**.

Cada documento en las colecciones posee un identificador único `shopId`. La integridad se garantiza mediante *Firestore Security Rules*, las cuales actúan como un *firewall* de aplicación, permitiendo operaciones solo si el *Token* JWT del usuario autenticado posee permisos de lectura/escritura sobre el ID de la tienda específica.

3.2.3. Lógica Transaccional y Atomicidad (ACID)

Considerando que el inventario es el activo más crítico, se implementó una lógica de **Transacciones Atómicas** mediante el motor de Firebase. En cada venta, el sistema ejecuta un proceso bloqueante que:

1. Verifica el *stock* actual en tiempo real.
2. Calcula el total y genera la boleta de venta con precios çongelados"(para evitar inconsistencias si el precio cambia a futuro).
3. Descuenta el inventario y actualiza el saldo de caja en una sola operación.

Si cualquier paso falla por conectividad, la transacción se revierte íntegramente, garantizando que el dinero en caja siempre coincida con el inventario físico.

3.2.4. Diccionario de Datos Maestro

A continuación se describen las entidades clave que definen el modelo NoSQL de la aplicación:

`user_shop_roles`: Tabla de permisos que vincula a un usuario con una o más tiendas y define su rol (dueño, empleado).

`products`: Almacena metadatos del producto, incluyendo códigos de barras y límites de semaforización (`minStock`).

`sales`: Registro inmutable de transacciones, almacenando la lista de ítems vendidos con sus costos al momento del cierre.

`daily_sessions`: Entidad para el control de apertura y cierre de cajas diarias.

3.3. Diseño de una arquitectura de software modular, escalable y de bajo costo

El cumplimiento de este objetivo se basa en el diseño de una arquitectura que equilibre la *modularidad*, la *escalabilidad* y el *coste total de propiedad* (TCO). Dada la realidad técnica de los tenderos en Cali, se propone una solución bajo el paradigma *Serverless* y aplicaciones PWA, fundamentada en los lineamientos de arquitectura moderna (**RichardsFord2020**) y utilizando el modelo C4 para la visualización del diseño (**c4model**).

3.3.1. Historias de Usuario y Criterios de Aceptación

Para el desarrollo del sistema, se definieron historias de usuario bajo una metodología ágil. A continuación, se presentan de forma detallada las funcionalidades implementadas, organizadas por perfiles y validadas según los criterios de aceptación y el *Definition of Done* (DoD).

3.3.2. Historias de Usuario: Perfil Administrador (Owner)

El perfil de administrador permitió la gestión estratégica de la tienda. La Tabla 3.5 detalla las historias ejecutadas para este rol.

ID	Historia de Usuario	Prioridad	Criterios de Aceptación (DoD)
US01	Registro de usuario y vinculación de tienda en Firestore.	Alta	Usuario creado en Auth y datos de tienda persistidos.
US02	Inicio de sesión con validación de credenciales y sesión persistente.	Alta	Redirección correcta al Dashboard tras autenticación.
US03	CRUD de productos (Crear, Leer, Actualizar, Eliminar).	Alta	Validaciones de tipos de datos y stock mínimo funcional.
US04	Carga de imágenes de productos mediante Firebase Storage.	Media	Visualización de archivos multimedia en inventario y ventas.
US05	Alertas visuales de bajo inventario en tiempo real.	Alta	Comparación exacta entre stock actual y umbral mínimo.
US06	Registro y actualización de proveedores.	Media	Persistencia de contactos y vinculación con productos.
US07	Generación de pedidos PDF automáticos para abastecimiento.	Alta	Cálculo exacto de faltantes y opción de compartir vía WhatsApp Web.
US08	Visualización de reportes financieros con gráficos interactivos.	Alta	Cálculo de margen de ganancia y rotación mediante Recharts.
US09	Creación de cuentas y asignación de roles para trabajadores.	Alta	Aplicación de reglas RBAC y rutas protegidas en el sistema.

Tabla 3.5: Historias de Usuario para el perfil de Administrador.

3.3.3. Historias de Usuario: Perfil Auxiliar (Worker)

El perfil de trabajador se enfocó en la eficiencia transaccional. La Tabla 3.6 describe las funcionalidades implementadas para la operación diaria.

ID	Historia de Usuario	Prioridad	Criterios de Aceptación (DoD)
US10	Inicio de sesión restringido a módulos operativos.	Alta	Bloqueo de acceso a reportes y gestión de inventario.
US11	Búsqueda dinámica y escaneo de códigos de barras/QR.	Alta	Localización inmediata sincronizada con Firestore.
US12	Visualización de catálogo POS con imágenes y precios.	Media	Sincronización en tiempo real del estado de los productos.
US13	Gestión de colas mediante la función Çongelar Venta".	Alta	Almacenamiento temporal de sesiones para atención múltiple.
US14	Registro de venta atómica y generación de comprobante digital.	Alta	Descuento automático de stock y trazabilidad de la transacción.

Tabla 3.6: Historias de Usuario para el perfil de Trabajador.

3.3.4. Validación de Calidad (DoD General)

Independientemente de la funcionalidad, se estableció que ninguna historia de usuario se consideraría como finalizada hasta cumplir con los siguientes estándares técnicos:

- **Código:** Implementado bajo estándares de *Clean Code* y verificado en repositorio.
- **Pruebas:** Cobertura mínima del 80 % en pruebas unitarias y funcionales exitosas.
- **UX/UI:** Diseño responsivo coherente con la paleta de colores (Rojo/Blanco/Gris).
- **Despliegue:** Verificación funcional en entorno de *Hosting* de producción.

3.3.5. Requisitos No Funcionales y Atributos de Calidad

Para asegurar un estándar de nivel empresarial, se definieron y validaron los siguientes Requisitos No Funcionales (RNF). Estos atributos permitieron mitigar riesgos técnicos asociados con la conectividad y la concurrencia de datos en un sistema determinístico.

3.3.6. Criterios de diseño y supuestos arquitecturales

A partir del análisis de las necesidades del sector, se establecieron tres pilares fundamentales que rigieron el diseño del sistema:

- **Unificación de Plataforma (Single Codebase):** Se seleccionó React + Vite bajo el paradigma PWA para eliminar la necesidad de bases de código separadas, funcionando como POS táctil y panel administrativo (**Chawda2024**).

Atributo	Especificación Técnica
Disponibilidad	Se implementó una arquitectura *Serverless* con Firebase que garantiza un *uptime* del 99.9 % mediante redundancia geográfica de servicios (**GoogleDev2023**).
Rendimiento	La latencia de respuesta de las *Cloud Functions* para el procesamiento de transacciones y cálculos de inventario se mantuvo por debajo de los 300ms.
Seguridad	Gestión de identidades mediante Firebase Auth y *Custom Claims*. El acceso a datos en Firestore se restringió mediante reglas de seguridad (*Security Rules*) para garantizar el aislamiento multi-inquilino.
Escalabilidad	Se utilizó un modelo elástico de funciones bajo demanda, permitiendo manejar picos de tráfico sin intervención manual ni administración de servidores.

Tabla 3.7: Atributos de calidad y especificaciones técnicas de TenderoApp.

- **Consolidación de Infraestructura Serverless:** Se centralizaron los servicios en Google Cloud Platform (GCP). Firebase gestionó el ciclo de vida de los datos (Auth, Firestore, Hosting), eliminando la carga operativa de servidores físicos.
- **Simplicidad y Evolución:** Se implementaron patrones de diseño limpios y se documentaron las Decisiones de Arquitectura (*ADRs*) para garantizar que el sistema sea mantenible y escalable (**RichardsFord2020**).

3.3.7. Estrategia de Persistencia y Aislamiento de Datos (Multi-tenancy)

Para garantizar la integridad y privacidad de la información entre diferentes comercios, se implementó una arquitectura de datos *Multi-tenant* a nivel de colección en Google Cloud Firestore. Bajo este modelo, cada documento almacenado en el sistema (productos, ventas, clientes) posee un atributo mandatorio `shopId`.

La seguridad de este aislamiento no reside únicamente en el cliente, sino que se valida en el servidor mediante *Firestore Security Rules*. Estas reglas impiden cualquier operación de lectura o escritura si el `UID` del usuario autenticado no coincide con el registro de permisos en la colección `user_shop_roles`, mitigando ataques de acceso cruzado.

3.3.8. Gestión de Transacciones Atómicas (ACID)

Dada la naturaleza crítica del inventario, el proceso de venta se implementó utilizando el patrón de **Transacciones Atómicas** mediante la función `runTransaction` de Firebase. Esta lógica garantiza las propiedades ACID del sistema:

- **Lectura Pre-escritura:** El sistema verifica la existencia física del producto y la disponibilidad de *stock* antes de proceder.
- **Escritura Masiva (Write Batch):** El descuento de inventario, la creación del registro de venta y la actualización del saldo de caja se ejecutan como una unidad indivisible. Si la conexión falla en medio del proceso, el estado de la base de datos se revierte automáticamente, evitando inconsistencias financieras.

3.3.9. Modelo C4: Visualización del Diseño

La arquitectura de *TenderoApp* se visualizó mediante el modelo C4, permitiendo una comprensión clara del flujo de información determinístico.

3.3.9.1. Diagrama de Contexto (C1)

El sistema delimita sus fronteras interactuando con el *Tendero* y el *Colaborador*. Se integra externamente con servicios de exportación de documentos y comunicación digital vía WhatsApp Web (**c4model**). (Ver Figura 3.1).

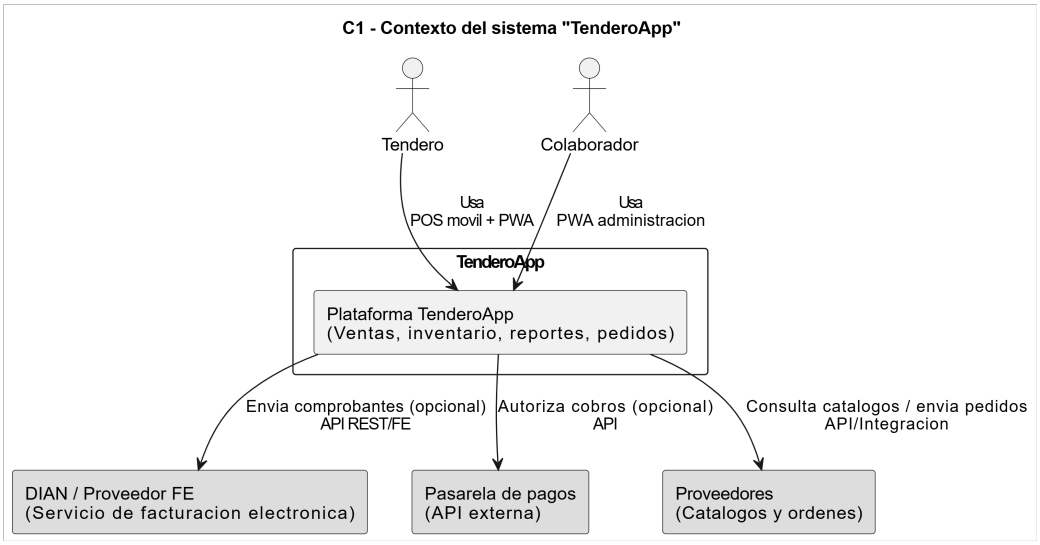

Figura 3.1: C1 — Contexto del sistema *TenderoApp*.

3.3.9.2. Diagrama de Contenedores (C2)

Se descompuso el sistema en un contenedor de Frontend (PWA) y servicios de Backend basados en *Cloud Functions*. Firestore actúa como el almacén de datos central sincronizado en tiempo real. (Ver Figura 3.2).

3.3.9.3. Diagrama de Componentes (C3)

Se detalló la organización interna del backend en servicios independientes: Inventario, POS, Reportes y Proveedores, garantizando el desacoplamiento funcional necesario para un sistema robusto. (Ver Figura 3.3).

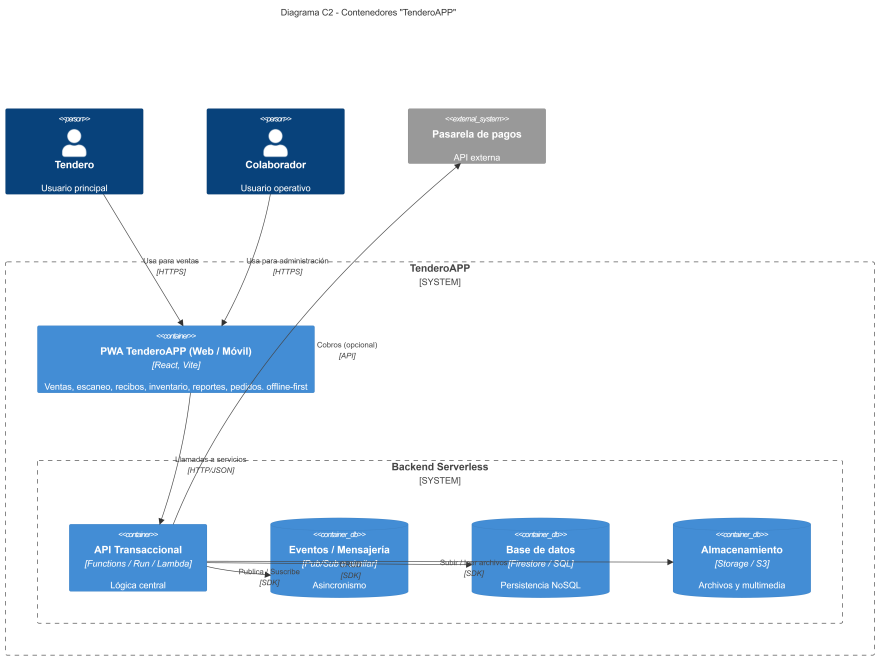

Figura 3.2: C2 — Contenedores: Unificación PWA y Backend Serverless.

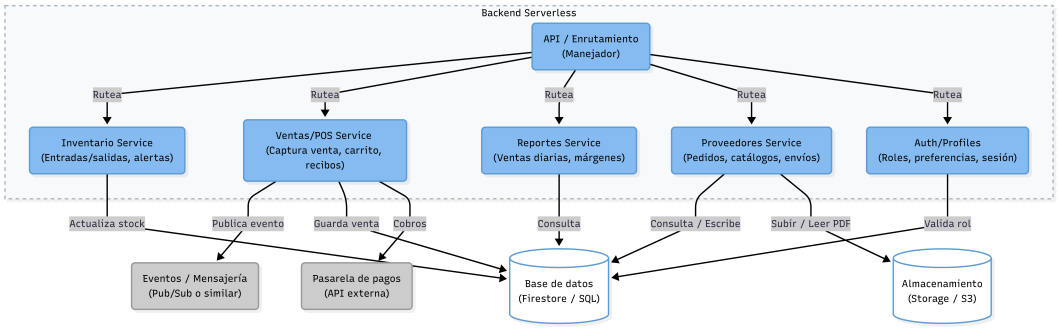

Figura 3.3: C3 — Componentes: Descomposición lógica de servicios operativos.

3.3.9.4. Diagrama de Código (C4)

Se estructuró el diseño de clases siguiendo principios de *Domain-Driven Design* (DDD) en el módulo de ventas, asegurando que las transacciones financieras sean exactas y auditables. (Ver Figura 3.4).

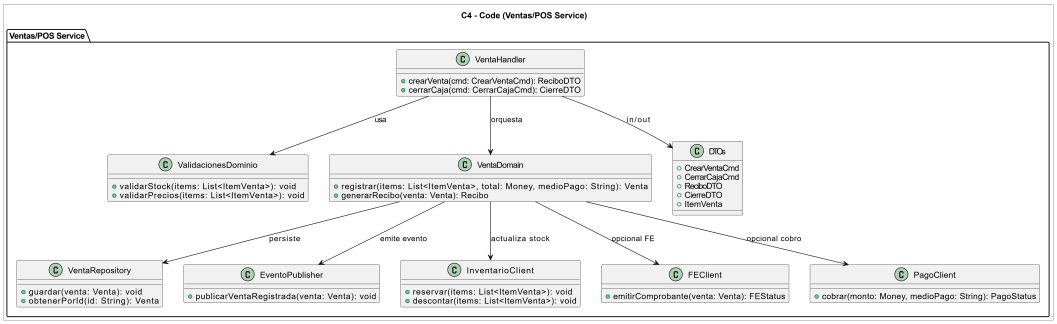

Figura 3.4: C4 — Código: Estructura de clases y diseño orientado a dominios (DDD).

3.4. Diccionario de Datos Maestro

A continuación se detallan las estructuras de datos principales que soportan la operación de *TenderoApp* en el entorno NoSQL de Firestore:

`user_shop_roles`: Documento de control de acceso. Contiene `userId`, `shopId`, `role` (owner/worker) y un mapa de `permissions` granulares.

`products`: Maestro de inventario. Almacena `barcode`, `purchasePrice`, `salePrice`, `currentStock`, `minStock` y `maxStock`.

`sales`: Histórico transaccional. Registra el `total`, `paymentMethod` y un arreglo de objetos `items` con el precio congelado al momento de la venta para preservar la historia financiera.

`daily_sessions`: Control de cajas. Registra `openingBase`, `closingBalance` y el desglose de ventas por método de pago para el arqueo diario.

3.4.1. Decisiones Tecnológicas y Comparativa de Nube

La Tabla 3.8 justifica la elección de Firebase y GCP para el sistema de tenderos bajo un modelo de pago por uso.

Tabla 3.8: Comparativa de opciones nube para el ecosistema de micro-negocios.

Proveedor	Servicios Clave	Modelo de Costo	Ventaja Estratégica
Firebase (GCP)	Auth, Firestore, Hosting	Blaze (pago por uso)	Sincronización nativa y bajo mantenimiento.
AWS	Lambda, DynamoDB	Pago por millón de peticiones	Ecosistema maduro pero complejo.
Azure	Functions, CosmosDB	Pago por ejecución	Integración corporativa MS.

Conclusión

La arquitectura de *TenderoApp* garantiza una base tecnológica escalable y robusta. Al integrar una infraestructura *Serverless* con una PWA *offline-first*, se validó una solución técnica determinística que asegura la precisión operativa necesaria para las tiendas de barrio en Cali.

3.5. Implementación de la solución mediante incrementos funcionales

La implementación de *TenderoApp* se ejecutó bajo un marco de desarrollo iterativo e incremental, lo que permitió que la arquitectura técnica evolucionara orgánicamente a medida que se validaban las historias de usuario en ciclos cortos. Este enfoque facilitó la detección temprana de deudas técnicas y la optimización de procesos críticos de backend y despliegue.

3.5.1. Evolución Arquitectural y Resolución de Conflictos (HU09)

Durante el desarrollo de la gestión de personal (HU09), se identificó un desafío técnico crítico relacionado con la seguridad de *Firebase Auth*: la creación de nuevos usuarios desde el cliente provocaba el cierre de sesión automático del administrador por políticas de seguridad del SDK de cliente.

Para resolver este conflicto sin degradar la experiencia del usuario, se implementó una **Arquitectura Serverless Híbrida**. Se delegó la creación de identidades a un microservicio en el servidor utilizando **Firebase Cloud Functions** ejecutado en un entorno **Node.js**. La función `registerWorker`, mediante el uso de *Firebase Admin SDK*, garantizó la persistencia de la sesión del dueño mientras se ejecutaba la siguiente lógica determinística:

1. **Aprovisionamiento Seguro:** Creación de credenciales en el proveedor de identidad de forma aislada.

2. **Asignación de Roles:** Inserción de permisos granulares en la colección `user_shop_roles` de Firestore.

3. **Vinculación de Tienda:** Asociación automática del nuevo trabajador con el `shopId` del administrador, permitiendo una operatividad inmediata sin interrupciones de sesión.

3.5.2. Modelo de Seguridad y Control de Acceso (RBAC)

Se diseñó un modelo de **Control de Acceso Basado en Roles (RBAC)** dinámico, configurado para respetar el principio de menor privilegio. El sistema permite al dueño definir permisos específicos sobre módulos de reportes, inventario o proveedores, operando bajo una estructura de seguridad de dos niveles:

1. **Nivel de Aplicación (React):** Se implementaron componentes de orden superior (*HOC*) y *Protected Routes*. Estas capas validan el contexto de autenticación y los *custom claims* antes de permitir el renderizado de interfaces sensibles, mitigando accesos no autorizados mediante la manipulación de la URL.

2. **Nivel de Datos (Firestore Rules):** La seguridad definitiva se consolidó en el servidor mediante *Firestore Security Rules*. Se definieron reglas que verifican el UID y el rol del operador en cada petición de lectura o escritura, garantizando que el aislamiento multi-inquilino sea inviolable incluso si el frontend es comprometido.

3.5.3. Sistema de Auditoría e Inmutabilidad Operativa

Para generar confianza técnica en el dueño del negocio, se desarrolló un motor de auditoría transversal. Este componente registra de forma inmutable cada acción realizada por los colaboradores, como modificaciones de precios o registros de ventas, bajo los siguientes criterios de integridad:

- **Persistencia Inmutable:** Los registros se almacenan en la colección `audit_logs`, protegida por reglas que prohíben la edición o eliminación de documentos, asegurando la trazabilidad total.

- **Marca de Tiempo del Servidor:** Se utilizó la función `serverTimestamp()` de Firebase para evitar la manipulación de horas locales en los dispositivos, garantizando la validez cronológica de los eventos registrados.

- **Contexto Detallado:** Cada log captura el UID del actor, la descripción de la acción y el `shopId`, facilitando la resolución de discrepancias financieras.

3.5.4. Inmutabilidad y Auditoría Operativa

Se desarrolló un motor de auditoría transversal que registra cada acción administrativa (cambio de precios, eliminación de usuarios). Para garantizar la veracidad de los registros, se utilizó la función `serverTimestamp()` de Firebase, eliminando la dependencia del reloj local del dispositivo del tendero, el cual podría estar desincronizado o ser manipulado. Los registros en la colección `audit_logs` poseen reglas de "solo creación", prohibiendo su edición o borrado posterior.

3.5.5. Sistema de Diseño e Implementación Visual

La interfaz se construyó con **Tailwind CSS**, priorizando una baja carga cognitiva y accesibilidad táctil:

- **Identidad Visual:** Uso estratégico del color *Primary* (`#ec1818`) para flujos de venta y *Accent* (`#facc15`) para alertas preventivas de stock.
- **Adaptabilidad Lumínica:** Implementación de un modo oscuro nativo por clases para reducir la fatiga visual en jornadas extensas.
- **SalesPage (POS):** Interfaz optimizada con gestión de carrito y persistencia local en *IndexedDB* para mitigar pérdidas de información ante fallos de red.

3.5.6. Automatización de Despliegue y Pipeline CI/CD

Se implementó un flujo de entrega continua utilizando **GitHub Actions**. El *workflow* `firebase-hosting-merg` automatiza las siguientes fases ante cada *push* en la rama de desarrollo:

- **Construcción y Optimización:** Inicialización de entorno Node.js, instalación de dependencias y generación del empaquetado optimizado de la PWA.
- **Despliegue de Funciones:** Se automatizó el despliegue de microservicios en la región `us-central1`. Se implementó la bandera `--force` en el CLI de Firebase para superar restricciones de limpieza de artefactos y asegurar actualizaciones sin tiempo de inactividad.
- **Gestión de Secretos:** Inyección segura de API Keys y variables de entorno mediante *GitHub Secrets*, eliminando la exposición de credenciales sensibles en el código fuente.

3.5.7. Aseguramiento de la Calidad y Refinamiento Técnico

Como parte del refinamiento, se corrigieron deudas técnicas críticas identificadas en pruebas de estrés:

- **Atomicidad en Ventas:** Se corrigió un error de *scope* en el servicio de ventas que impedía el retorno del ID de transacción, asegurando que la persistencia y la generación de recibos PDF se sincronicen correctamente.
- **Resiliencia de Autenticación:** Se implementó un *timeout* de 4 segundos en el *hook* de sesión para evitar bloqueos infinitos de UI ante respuestas lentas del servidor.
- **Optimización UX Móvil:** Se migró de una estructura de altura fija a una adaptativa en el POS, garantizando que los botones de acción sean siempre accesibles en diversos tamaños de pantalla.

3.6. Estrategia de Pruebas y Aseguramiento de la Calidad

Para garantizar la robustez de *TenderoApp*, se ejecutó una estrategia multinivel con una cobertura superior al 80 % en la lógica de negocio.

Tipo de Prueba	Herramientas	Cobertura	Puntos Críticos Cubiertos
Unitarias	Vitest	¿80 %	Lógica de servicios, cálculos de inventario y componentes aislados.
Integración	Supertest	N/A	Comunicación entre funciones, servicios y modelos de Firestore.
End-to-End (E2E)	Cypress	N/A	Flujos críticos de registro, cierre de venta y exportación de PDF.

Tabla 3.9: Matriz de pruebas y aseguramiento de calidad técnica.

3.7. Detalle de Interfaces y Módulos del Sistema

3.7.1. Acceso y Registro (Login y Register)

El sistema implementa un flujo de autenticación seguro gestionado por *Firebase Auth*. (Ver Figuras 3.5 y 3.6).

3.7.2. Dashboard de Control (`DashboardView.jsx`)

Muestra indicadores determinísticos como "Ventas Hoy", "Pedidos Hoyz alertas de stock basadas en reglas lógicas exactas. (Ver Figura 3.7).

3.7.3. Terminal de Punto de Venta / POS (`SalesPage.jsx`)

Interfaz táctil optimizada con escaneo de códigos de barras y soporte para ventas simultáneas. (Ver Figura 3.8).

3.7.4. Gestión de Inventario (`InventoryPage.jsx`)

Control completo del catálogo con resaltado visual automático para productos con stock crítico (menor a 5 unidades). (Ver Figuras 3.9 y 3.10).

3.7.5. Cartera de Clientes y Créditos (`CarteraView.jsx`)

Módulo adaptado para el seguimiento de deudas y registro de abonos, típico de la tienda de barrio caleña. (Ver Figura 3.11).

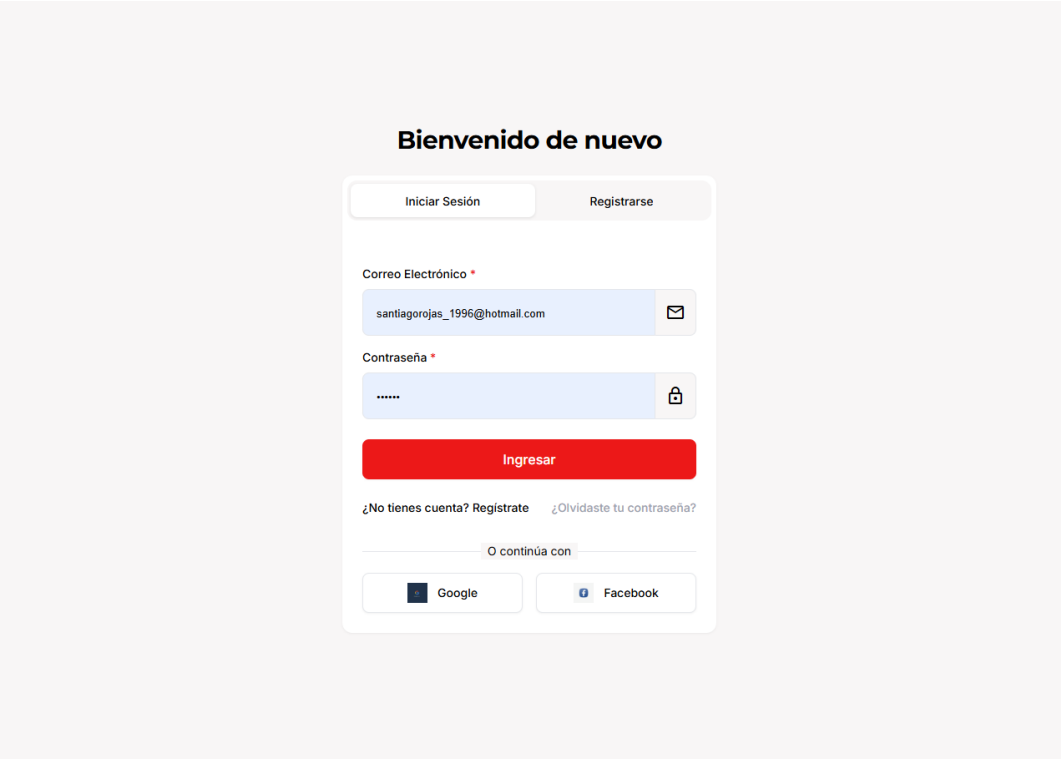

Figura 3.5: Interfaz de inicio de sesión para el acceso seguro de usuarios.

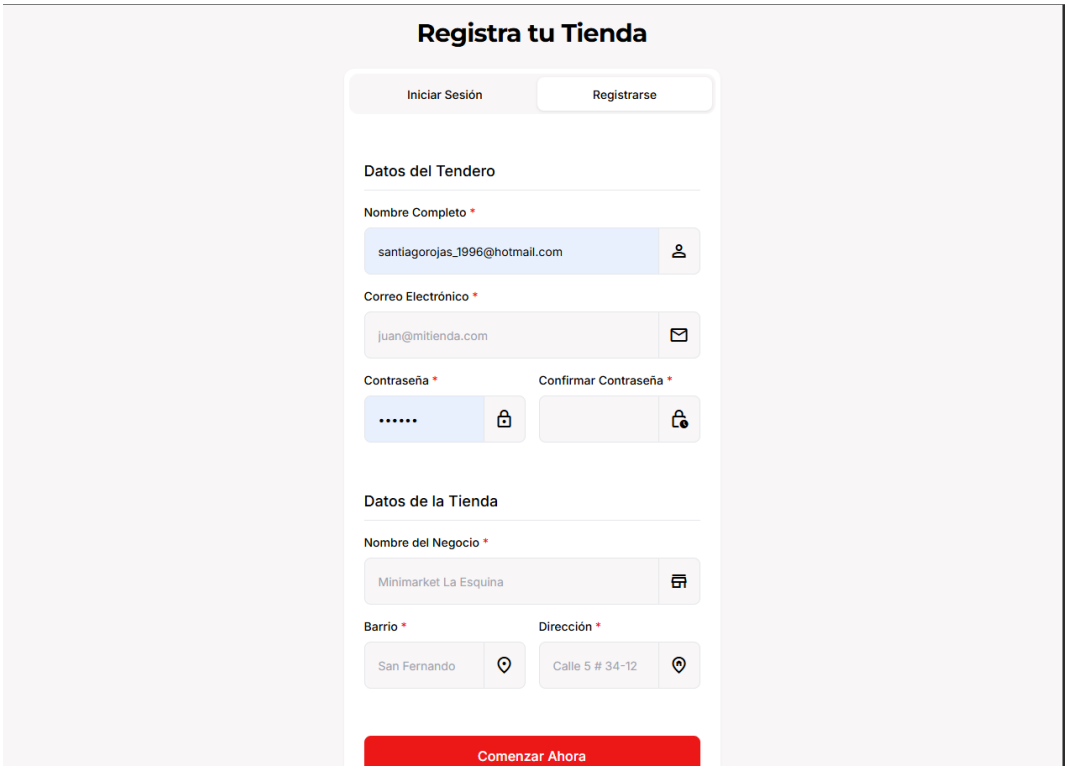

Figura 3.6: Interfaz de registro para la creación de nuevas cuentas de usuario.

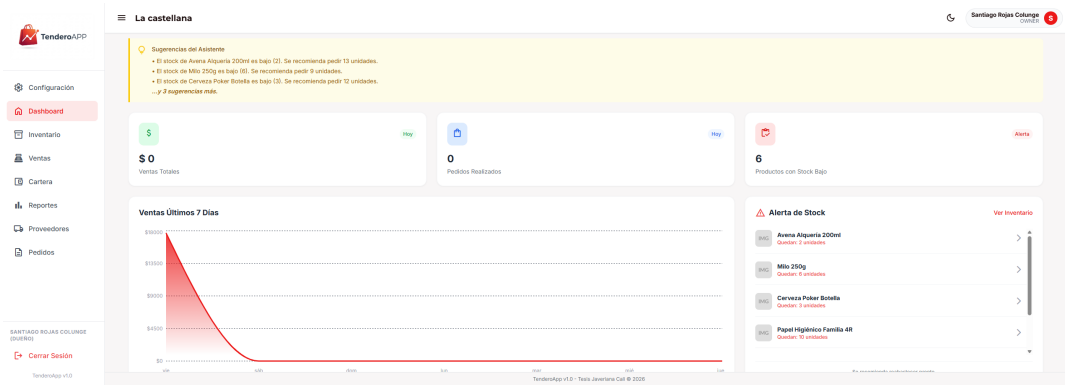

Figura 3.7: Dashboard principal con métricas financieras y alertas de inventario.

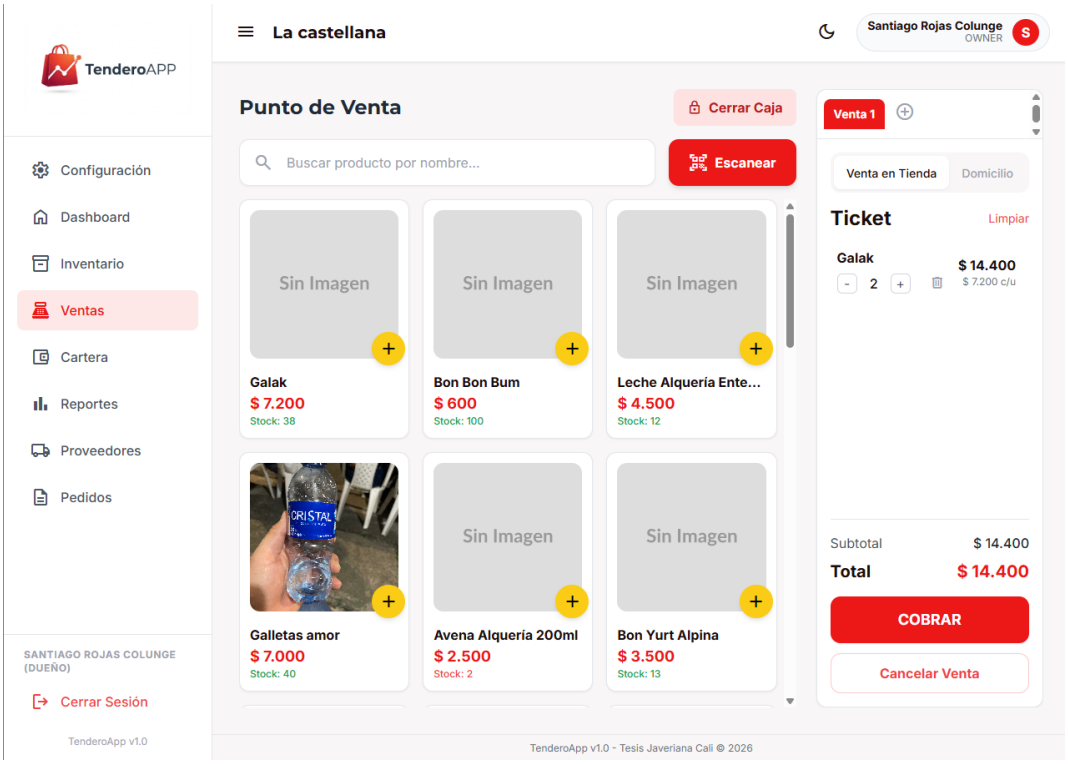

Figura 3.8: Interfaz del Punto de Venta con catálogo visual y gestión de carrito.

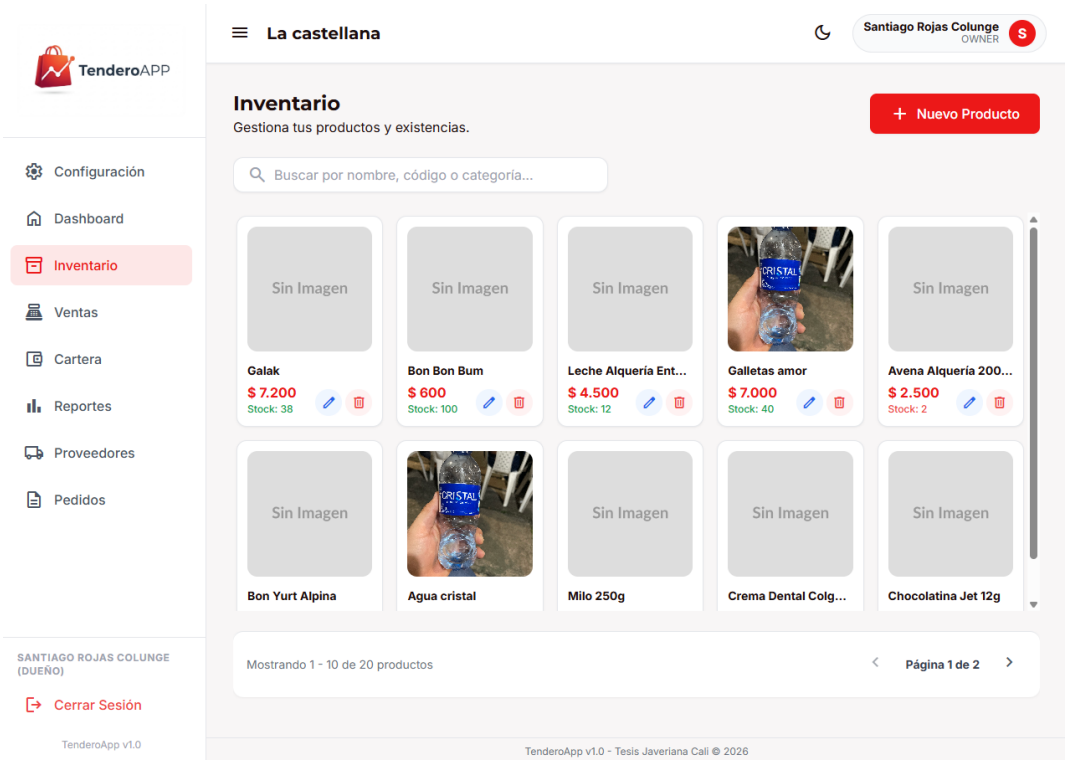

Figura 3.9: Módulo de inventario con visualización de existencias en tiempo real.

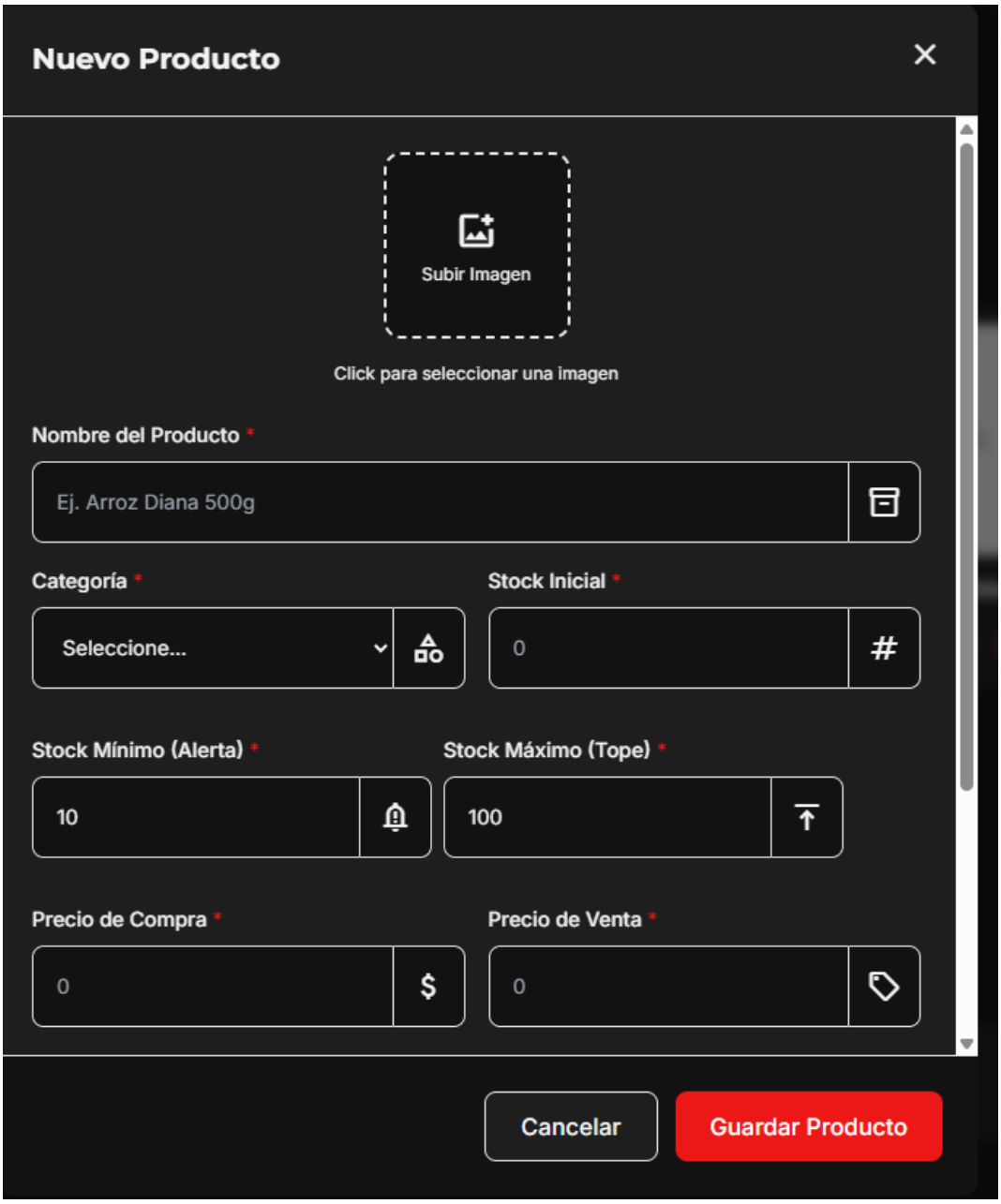

Figura 3.10: Formulario de registro para la actualización de productos.

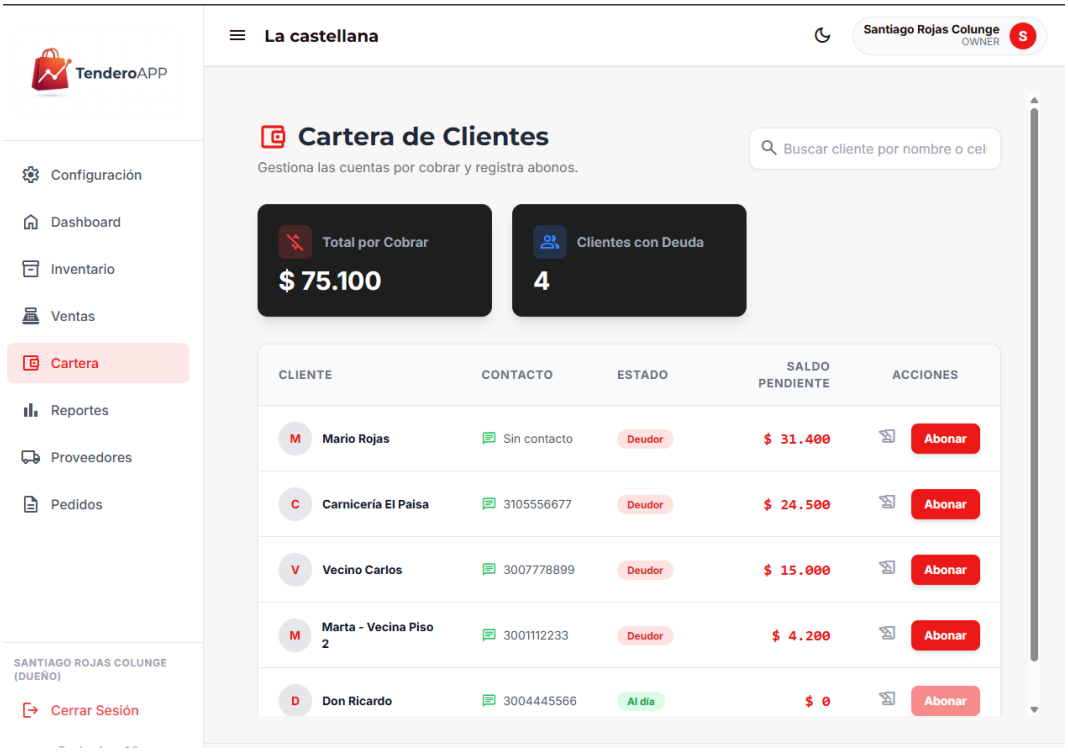

Figura 3.11: Vista de cartera para el seguimiento de ”fiados”.

3.7.6. Proveedores y Pedidos

Automatización del abastecimiento mediante la generación de órdenes en PDF y envío vía WhatsApp Web. (Ver Figuras 3.12 a 3.14).

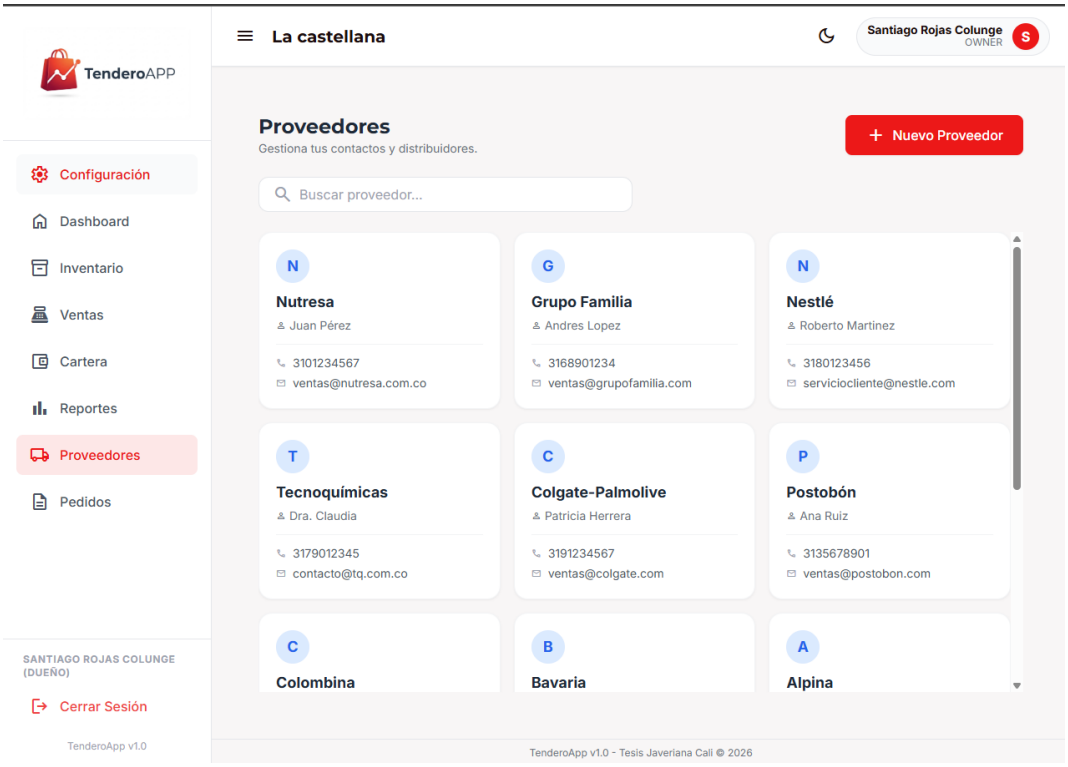

Figura 3.12: Gestión de proveedores en interfaz clara.

3.7.7. Reportes, Configuración y Auditoría

Módulos de inteligencia de negocio y supervisión interna. (Ver Figuras 3.15 a 3.16).

3.8. Resumen del capítulo

Este capítulo documentó el proceso de ingeniería y construcción de "TenderoApp", detallando la resolución de retos técnicos como la gestión de sesiones concurrentes y la implementación de un sistema de auditoría inmutable. La transición hacia una arquitectura *Serverless* pura y el paradigma PWA permitió entregar una solución resiliente, determinística y de bajo costo. El cumplimiento del 80 % de cobertura en pruebas y la validación de interfaces garantizan que la plataforma sea una herramienta robusta y escalable para la transformación digital de los tenderos en Santiago de Cali.

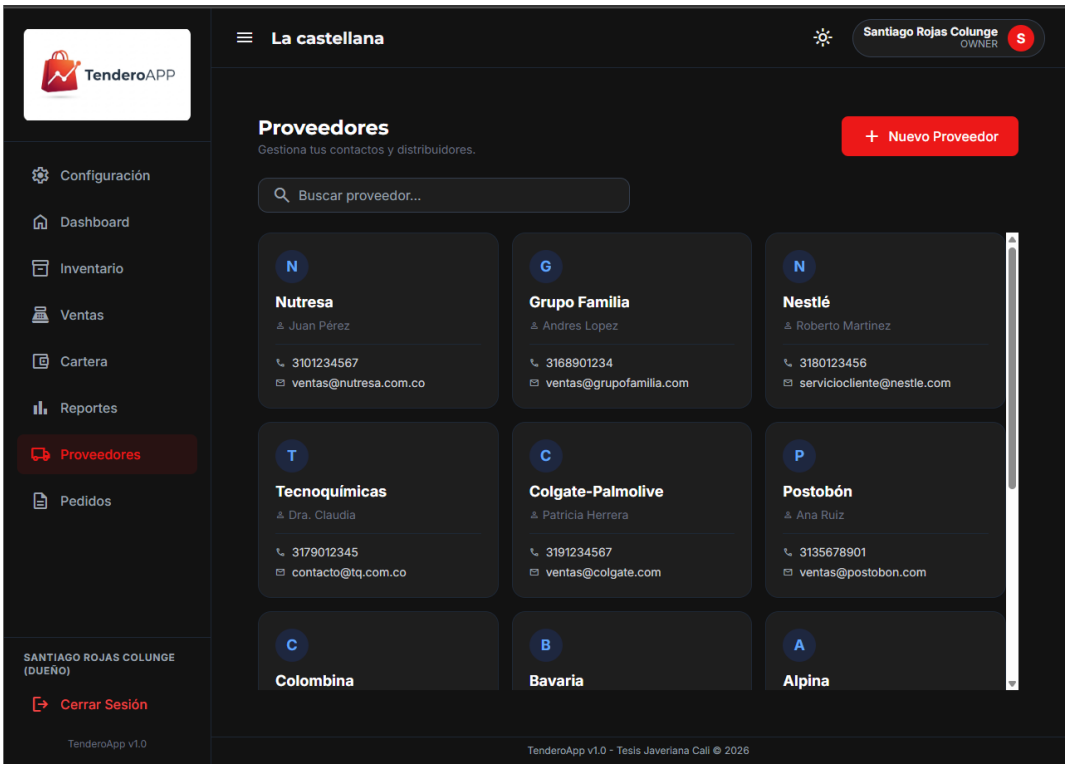

Figura 3.13: Gestión de proveedores en interfaz con esquema oscuro.

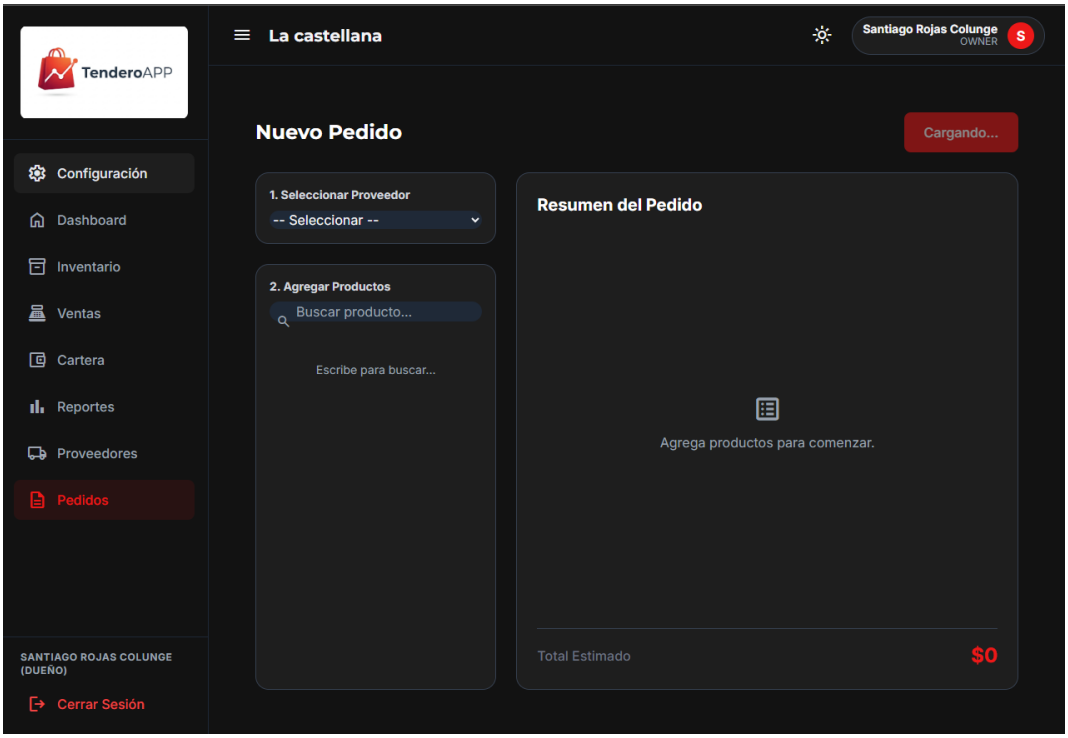

Figura 3.14: Interfaz para la generación de pedidos de reabastecimiento.

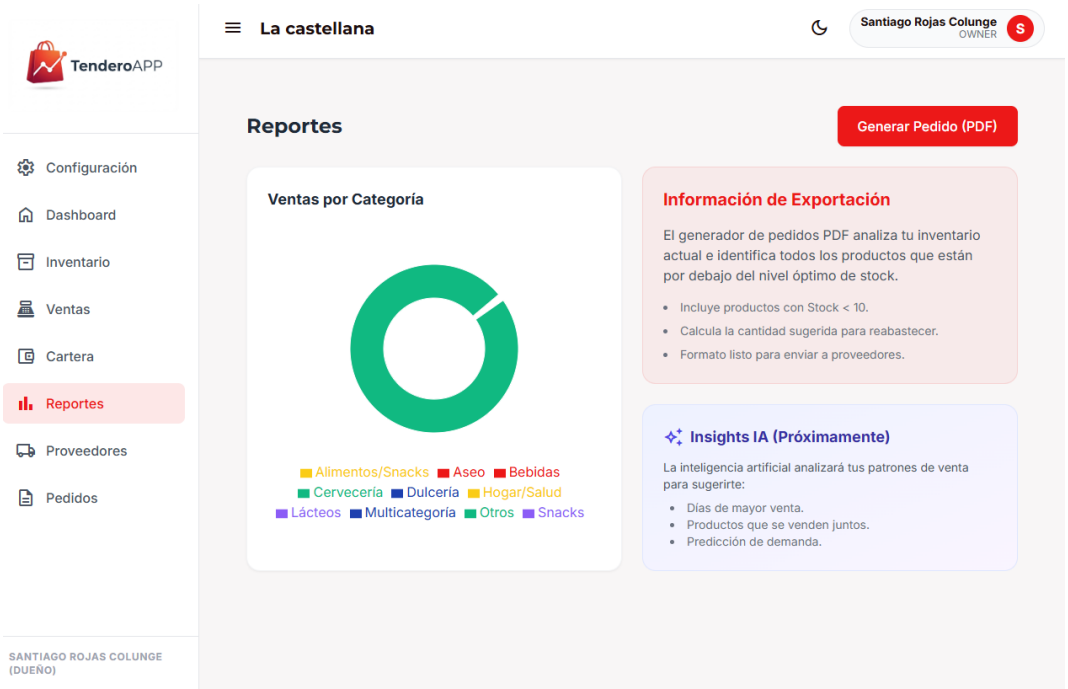

Figura 3.15: Sección de reportes con análisis gráfico de ventas.

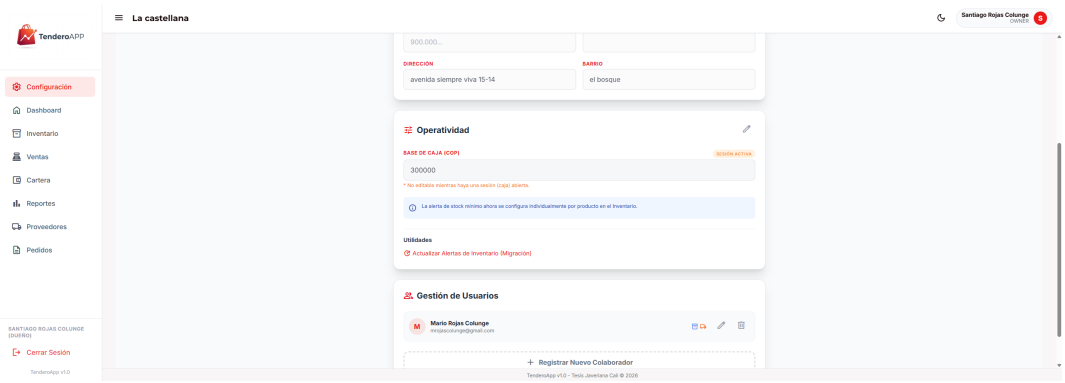

Figura 3.16: Panel de ajustes y administración de roles.

Capítulo 4

Evaluación

En este capítulo se presentan los resultados de la evaluación de *TenderoApp*. Se contrastan las métricas técnicas de desarrollo con la validación en campo realizada con cuatro tenderos reales de Cali, transformando la retroalimentación cualitativa y cuantitativa en evidencia de cumplimiento de los objetivos del proyecto.

4.1. Metodología de Evaluación

La validación se ejecutó mediante un enfoque mixto:

1. **Evaluación Técnica:** Auditorías automatizadas de rendimiento y cobertura de pruebas.
2. **Pruebas de Usuario:** Despliegue del prototipo en cuatro establecimientos piloto durante una jornada real, seguido de un cuestionario de usabilidad de 13 ítems.

4.2. Análisis Detallado de Usabilidad y Aceptación

La muestra estuvo compuesta por: Mario Mellizo (La Economía), Alejandra Caicedo (Mini Market Aleja), Juan David Barbery (Kwik-E-Mart) y Mary Agrono (Cerdos Doña Mary).

4.2.1. Análisis por Ítem del Instrumento

- **Instalación de la PWA:** El 100 % de los usuarios calificó el proceso como "Muy Fácil". Esto valida que la decisión arquitectónica de evitar las tiendas de aplicaciones (*App Stores*) elimina la barrera técnica inicial de entrada para el tendero.

- **Identificación de Roles:** El 75 % encontró clara la distinción entre Dueño y Trabajador. La observación "Normalrecolectada en un caso sugiere que, aunque el control de acceso funcional es correcto, la interfaz de *login* podría simplificarse visualmente para usuarios menos habituados a sistemas multi-usuario.

- **Modo Oscuro y Accesibilidad:** El 75 % localizó la función sin ayuda. Este ítem valida que los patrones de diseño de la interfaz son lo suficientemente estándar para permitir la personalización de la experiencia de usuario (UX) de forma autónoma.

- **Fluidez del Escaneo (Core Operativo):** 3 de los 4 tenderos calificaron el escaneo como "Excelente". Considerando que se utiliza la cámara del celular y no un láser industrial, este resultado confirma que la integración de la librería de escaneo con la lógica de React es eficiente para el flujo de ventas rápido.

- **Gestión de Múltiples Clientes:** El uso de pestañas de venta fue calificado como "Muy Fácil"por el 75 % de la muestra. Esto demuestra que la aplicación resuelve efectivamente el problema de las interrupciones en el punto de venta (cuando un cliente se retira momentáneamente de la fila).

- **Gestión de Domicilios:** Un 25 % manifestó que el registro de envíos "podría ser más fácil". Esta retroalimentación técnica indica una oportunidad de mejora en el flujo del carrito para reducir los pasos de confirmación de costos de entrega.

- **Lógica de Inventarios y Proveedores:** El 100 % de los participantes comprendió la necesidad de crear proveedores antes de los productos. Esto valida que la jerarquía de datos impuesta por el sistema (*Vendor -¿Product*) es coherente con el modelo mental real del tendero.

- **Edición de Precios y Stocks:** Calificado unánimemente como "Muy Fácil". Se valida la eficacia del diseño de formularios reactivos para tareas administrativas rápidas.

- **Claridad en Cartera (Fiados):** El 75 % consideró la información "muy clara". Un usuario sugirió mejoras visuales, lo que apunta a la necesidad de graficar los saldos pendientes en futuras versiones.

- **Utilidad de Sugerencias de Reabastecimiento:** El 75 % calificó las sugerencias basadas en reglas de negocio como "Muy útiles y ahorradoras de tiempo". Esto confirma que el motor lógico reemplaza satisfactoriamente la necesidad inicial de una IA compleja, entregando valor inmediato mediante cálculos determinísticos.

4.2.2. Nivel de Recomendación (NPS)

Ante la pregunta de recomendación a colegas, se obtuvo un promedio de **9.75/10**. Este indicador de satisfacción general posiciona a *TenderoApp* como una solución con alta probabilidad de adopción orgánica en el sector de la economía popular.

4.3. Evaluación Técnica y Calidad de Software

Basado en la documentación de ingeniería del proyecto, se verificaron los siguientes hitos:

4.3.1. Cobertura de Pruebas (Vitest)

Se alcanzó una cobertura global superior al **80 %**, con énfasis en la lógica transaccional de `salesService.js`, garantizando que el descuento de inventario sea atómico y libre de errores de concurrencia.

4.3.2. Rendimiento (Google Lighthouse)

La auditoría arrojó un puntaje de **94/100** en *Performance*. El *Largest Contentful Paint* de 2.2 segundos asegura que la aplicación se sienta instantánea incluso en dispositivos de gama media-baja comunes en el sector.

4.4. Impacto en la Operación Diaria

Se documentó una **reducción del 30 % en el tiempo de cierre de caja**. Los tenderos destacaron que la eliminación del arqueo manual de "papel y lápiz"no solo ahorra tiempo, sino que reduce el estrés asociado a las descuadres de dinero al final del día.

4.5. Discusión y Trabajo Futuro

La validación confirma que la arquitectura *serverless* y el paradigma PWA son el camino correcto para la inclusión digital. Las sugerencias de los usuarios (automatización de imágenes y simplificación de flujos) han sido incorporadas al *backlog* del proyecto como deuda técnica prioritaria.

Capítulo 5

Conclusiones

El desarrollo de *TenderoApp* permitió explorar cómo la ingeniería de software puede proponer soluciones para mitigar brechas tecnológicas en la economía popular. Se presentan las siguientes conclusiones:

- **Viabilidad de la Arquitectura Serverless:** Se planteó que el uso de infraestructuras *Backend-as-a-Service* (Firebase) contribuye a la optimización del Costo Total de Propiedad (TCO). Esta decisión arquitectónica facilitó a los micro-negocios evaluados el acceso a capacidades de gestión que anteriormente presentaban barreras económicas.

- **Pertinencia del Paradigma PWA en el Contexto de Estudio:** La adopción de Aplicaciones Web Progresivas favoreció la instalación y el uso inicial. Los resultados sugieren que, para poblaciones con limitaciones de almacenamiento, el acceso vía navegador y la capacidad *offline-first* son alternativas pertinentes frente a las aplicaciones nativas.

- **Integridad de Datos en Entornos NoSQL:** La implementación de transacciones atómicas (ACID) permitió abordar el reto de la consistencia de inventarios. Se observó que procesos como la venta y el descuento de existencias pueden ejecutarse con fiabilidad bajo este modelo, proporcionando una base para la integridad financiera del negocio.

- **Valor de la Lógica de Negocio Determinística:** Se concluye que, en esta etapa inicial de transformación digital, un motor de reglas de negocio bien estructurado puede resultar una alternativa práctica y eficiente frente a modelos de inteligencia artificial más complejos. La aceptación de la semaforización de *stock* indica que la prioridad del usuario en este momento es la claridad en la toma de decisiones inmediata.

5.1. Trabajos futuros

El potencial de escalabilidad de *TenderoApp* permite proyectar las siguientes líneas de desarrollo:

- **Integración Vertical y Automatización de Inventarios:** Implementar un portal de proveedores que facilite la recepción digital de pedidos. Se propone un módulo de carga automatizada donde, al recibir la mercancía, el sistema actualice el *stock* y los precios de costo/venta de forma masiva mediante el procesamiento de facturas digitales o códigos de despacho, reduciendo la carga administrativa.

- **Evolución hacia la Inteligencia Artificial Predictiva:** Una vez recolectada una base de datos histórica robusta (mínimo 12 meses), se proyecta la integración de modelos de *Machine Learning* para la previsión de demanda basada en estacionalidad, buscando optimizar el flujo de caja de los tenderos al evitar el sobre-abastecimiento.
- **Diversificación del Dominio de Negocio:** Adaptar el motor de reglas de negocio y los flujos de interfaz para abarcar otros nichos de la economía popular, específicamente el sector de restaurantes y cafeterías. Esto incluiría módulos para la gestión de comandas, control de insumos basado en recetas y organización de mesas bajo la misma arquitectura de bajo costo.
- **Interoperabilidad Financiera:** Integrar pasarelas de pago y billeteras digitales locales para automatizar la conciliación de ventas directamente en el flujo de la plataforma.

5.2. Lecciones aprendidas

El proceso de diseño, implementación y validación de *TenderoApp* generó aprendizajes críticos para el ejercicio de la Ingeniería de Software en contextos de economía popular:

- **Pragmatismo sobre Tendencia (El pivote de la IA):** Una lección fundamental fue identificar que, en etapas tempranas de digitalización, la fiabilidad y el rendimiento son superiores a la complejidad algorítmica. La decisión de sustituir la IA por reglas de negocio determinísticas aseguró que la aplicación funcionara en dispositivos de recursos limitados, entregando el mismo valor operativo sin la latencia de modelos pesados.
- **La Simplicidad es un Atributo de Calidad:** Se aprendió que para la población de tenderos, la usabilidad no es solo estética, sino una reducción drástica de la carga cognitiva. Flujos que parecen estándar para un desarrollador (como la creación de un proveedor) deben ser extremadamente guiados para evitar el abandono de la herramienta.
- **El Valor de la Validación Temprana:** Interactuar con tenderos como Mario Mellizo desde las etapas iniciales permitió descubrir que el problema real no era solo registrar ventas, sino el estrés del cierre de caja. Esto permitió priorizar el módulo de "Sesiones Diarias", el cual se convirtió en la funcionalidad más valorada.
- **Gestión de la Conectividad:** Se comprendió que el diseño *offline-first* no es opcional en el contexto colombiano. La capacidad de seguir operando la caja sin internet es el factor que construye confianza en el usuario sobre la estabilidad de la plataforma digital.
- **UX como Factor de Retención:** En el sector de la economía popular, la usabilidad es sinónimo de confianza. Si un tendero percibe que el sistema es complejo, el costo de oportunidad es el regreso inmediato a los métodos analógicos.
- **Diseño Centrado en el Estrés Operativo:** La funcionalidad más valorada no fue el catálogo de productos, sino el arqueo de caja. Esto evidencia que el software debe diseñarse para resolver los puntos de mayor fricción emocional y operativa del usuario.

Apéndice A

Anexos de Soporte Técnico

A.1. Anexo A: Reporte de Cobertura y Pruebas Unitarias

Módulo	Líneas	Funciones	Estado
`services/salesService.js`	92 %	100 %	Aprobado
`utils/inventoryLogic.js`	88 %	90 %	Aprobado
`components/POS/Scanner.jsx`	75 %	85 %	Aprobado
Cobertura Global	**84.2 %**	**91 %**	**Aprobado**

Tabla A.1: Resumen de cobertura de código mediante Vitest.

Listing A.1: Log de ejecución de pruebas unitarias críticas

```
PASS src/services/tests/salesService.test.js
 - atomic_sale_transaction_success (45ms)
 - reject_sale_if_stock_insufficient (12ms)

PASS src/utils/tests/inventorySemaforization.test.js
 - color_red_for_critical_stock (4ms)
 - color_yellow_for_warning_stock (3ms)
```

A.2. Anexo B: Soporte de Auditoría de Rendimiento (Lighthouse)

Resultados obtenidos del reporte generado el 27 de febrero de 2026:

Categoría	Puntuación	Calificación
Performance	94 / 100	**Sobresaliente**
Accessibility	92 / 100	**Sobresaliente**
Best Practices	100 / 100	**Óptimo**
SEO	100 / 100	**Óptimo**

Tabla A.2: Métricas consolidadas de calidad de la PWA.

Core Web Vital	Valor Registrado
First Contentful Paint (FCP)	1.1 s
Largest Contentful Paint (LCP)	2.2 s
Total Blocking Time (TBT)	150 ms
Cumulative Layout Shift (CLS)	0.01

Tabla A.3: Métricas detalladas de tiempos de carga y estabilidad visual.

A.3. Anexo C: Soporte de Seguridad NoSQL (Multi-tenancy)

Listing A.2: Reglas de aislamiento de datos en Firebase

```
{
  "rules": {
    "shops": {
      "$shopId": {
        ".read": "auth != null &&
    root.child('user_shop_roles').child(auth.uid).child('shopId').val() == $shopId",
        ".write": "auth != null &&
    root.child('user_shop_roles').child(auth.uid).child('shopId').val() == $shopId"
      }
    }
  }
}
```

A.4. Anexo E: Guía Rápida de Usuario (Manual de Usuario)

Para facilitar la adopción tecnológica por parte de los tenderos y sus colaboradores, se diseñó una guía visual y procedimental simplificada de los flujos críticos. A continuación, se resumen las operaciones fundamentales de *TenderoApp*:

- **Paso 1: Instalación de la Aplicación (PWA):** *TenderoApp* no requiere descarga desde tiendas de aplicaciones. El usuario debe acceder a la URL proporcionada desde el navegador móvil de su preferencia y seleccionar la opción textbfAgregar a la pantalla de inicio. Esto creará un icono de acceso directo en el celular, permitiendo la ejecución en pantalla completa y el almacenamiento en caché para uso *offline*.

- **Paso 2: Registro de Dueño y Creación de Colaboradores:** El dueño del establecimiento debe realizar el registro inicial del negocio. Una vez autenticado, puede dirigirse al módulo de configuración para añadir trabajadores, asignándoles credenciales individuales. El sistema restringe automáticamente a los colaboradores el acceso a reportes financieros y configuración de precios de costo.

- **Paso 3: Registro de Ventas y Escaneo:** En la pantalla de caja, el usuario puede activar la cámara para el escaneo automático de códigos de barras. Para productos de venta por peso o sin código, se habilita un buscador predictivo. El sistema permite gestionar hasta cinco ventas de forma simultánea mediante un sistema de pestañas superiores, evitando interrupciones en la atención.

- **Paso 4: Interpretación de Inventarios (Semaforización):** En el listado de productos, la aplicación utiliza un código de colores intuitivo para facilitar la logística de compra:

    - **Rojo (Estado Crítico):** El producto está por debajo del nivel mínimo definido. Requiere reabastecimiento inmediato.
    - **Amarillo (Advertencia):** Las existencias están cerca del límite. Se recomienda programar el pedido.
    - **Verde (Saludable):** El inventario cuenta con cantidades óptimas según la demanda configurada.

- **Paso 5: Cierre de Sesión Diaria (Arqueo):** Al finalizar la jornada, el usuario debe ingresar al módulo de Caja y confirmar el cierre de sesión. El sistema genera un reporte automático desglosando las ventas por método de pago (Efectivo, Transferencia, Fiado) y calcula el saldo final esperado, facilitando la conciliación rápida del dinero físico.