Arenillas, Felipe

Club Galo: sistema de gestión integral para comercios de venta al público

Tesis para la obtención del título de grado de Ingeniero en Sistemas

Director:

Porrini, Federico Eduardo

Carreño, Ignacio Luciano

UNIVERSIDAD CATÓLICA DE CÓRDOBA
JESUITAS

PROYECTO FINAL

CLUB GALO

SISTEMA DE GESTIÓN INTEGRAL PARA COMERCIOS DE VENTA AL PÚBLICO

Profesores:

- **Federico Eduardo Porrini**
- **Ignacio Luciano Carreño**

Autor:

- **Felipe Arenillas(Ingeniería en sistemas)**

Universidad Católica de Córdoba - V3.1.3 - 05/05/2025

Resumen

Club Galo, una PyME cordobesa dedicada a la venta de productos para mascotas, afrontaba el riesgo de perder competitividad al carecer de un canal digital y depender de procesos manuales para gran parte de su operatoria. El proyecto desarrollado entrega un sistema integral compuesto por tres aplicaciones —desktop, web y PWA— respaldadas por un único backend, orientado a automatizar procesos de ventas, stock, delivery, contabilidad, CRM y marketing.

La solución incorpora autenticación segura con JWT, un gestor de ventas que otorga puntos canjeables por promociones y un motor de promociones configurable, capaz de soportar modalidades como 2×1, descuentos por cantidad, combos y productos de regalo. Además, se implementó un módulo de notificaciones programadas y automáticas que distribuye mensajes a clientes y empleados por segmentos o destinatarios individuales, abarcando alertas de estado de envío, avisos de stock crítico, recordatorios de cumpleaños de mascotas y promociones personalizadas. También se desarrolló un algoritmo de aprendizaje automático que predice cuándo un cliente agotará un producto y genera la notificación en el momento oportuno, con el fin de favorecer la recompra.

Como ampliación de la solución original, el sistema incorpora una capa analítica desacoplada de la operatoria transaccional, orientada a la visualización de indicadores históricos y dashboards estadísticos para soporte a la toma de decisiones. Esta capa se apoya en procesos ETL, un Data Warehouse separado de la base operativa y la herramienta Metabase para la construcción de paneles analíticos. De este modo, el módulo estadístico permite consultar métricas relevantes del negocio, tales como ventas, stock, productos, promociones y desempeño general, manteniendo además la posibilidad de acceder al panel incluso ante contingencias en el backend principal.

En materia de calidad, el proyecto se desarrolló con metodología SCRUM y apoyo de tablero Kanban, priorizando funcionalidades de alto impacto y complementando las pruebas manuales y de aceptación con una capa de pruebas automatizadas end-to-end sobre el backend. En conjunto, la solución propuesta busca incrementar las ventas online, mejorar la fidelización de clientes, reducir tareas manuales y fortalecer la capacidad de análisis del negocio, posicionando a Club Galo como una empresa mejor preparada para enfrentar los desafíos actuales y futuros.

Palabras clave: sistema integral, PyME, dashboards estadísticos, Data Warehouse, ETL, notificaciones inteligentes, promociones, gestión de stock, CRM, PWA, ventas.

Abstract

Club Galo, a Córdoba-based SME specializing in the sale of pet products, faced the risk of losing competitiveness due to the absence of digital channels and its reliance on manual processes for much of its daily operation. The project delivers an integrated system composed of three applications—desktop, web, and PWA—supported by a single backend, aimed at automating sales, inventory, delivery, accounting, CRM, and marketing processes.

The solution includes secure JWT-based authentication, a sales management module that grants points redeemable for promotions, and a configurable promotion engine capable of supporting different formats such as buy-one-get-one offers, quantity discounts, bundles, and gift products. In addition, a scheduled and automatic notification module was implemented to distribute messages to customers and employees by segment or individual recipient, covering shipment status alerts, critical stock warnings, pet birthday reminders, and personalized promotions. A machine learning algorithm was also developed to predict when a customer is likely to run out of a product and to trigger a notification at the appropriate moment in order to encourage repeat purchases.

As an extension of the original solution, the system incorporates an analytical layer decoupled from the transactional operation, aimed at supporting decision-making through historical indicators and statistical dashboards. This layer is based on ETL processes, a Data Warehouse separated from the operational database, and Metabase as the visualization tool for building analytical dashboards. As a result, the statistical module enables the consultation of relevant business metrics such as sales, stock, products, promotions, and overall performance, while also allowing access to the dashboard even in the event of failures affecting the main backend.

From a quality perspective, the project was developed using SCRUM methodology and a Kanban board, prioritizing high-impact features and complementing manual and user acceptance testing with a layer of automated end-to-end tests for the backend. Overall, the proposed solution aims to increase online sales, improve customer loyalty, reduce manual work, and strengthen the company's analytical capabilities, positioning Club Galo as a business better prepared to face current and future challenges.

Presentación del tema

En la última década, el comercio minorista argentino ha experimentado una acelerada digitalización impulsada por el crecimiento del e-commerce y la demanda de experiencias omnicanal. Este fenómeno no se limita a las grandes cadenas: las pequeñas y medianas empresas (PyMEs) que comercializan productos para mascotas y animales de granja —como las forrajerías— necesitan adoptar tecnologías que integren la venta presencial con canales online para mantener su competitividad y fidelizar a una clientela cada vez más conectada.

Club Galo es una forrajería cordobesa que atiende cerca de 2000 clientes al mes con un equipo de cinco empleados. Hasta 2025, su operación dependía de planillas Excel y procesos manuales para registrar ventas, controlar inventario y gestionar la relación con el cliente. Estas prácticas dificultan obtener métricas confiables, provocando sobre-stock o en su defecto falta de stock, además de impedir la evaluación precisa de la rentabilidad de productos y la efectividad de las promociones, que a largo plazo genera pérdida de fidelidad de los clientes.

La presente iniciativa propone un sistema integral multiplataforma —desktop, web y PWA— orientado a cuatro ejes: venta, control de stock, CRM y seguridad. Su adopción permitirá a Club Galo contar con información en tiempo real, optimizar recursos y ofrecer una experiencia de compra moderna, beneficiando tanto a los clientes como al equipo interno.

Glosario

Forrajería: Comercio minorista que vende alimentos, insumos y accesorios para mascotas y animales de granja.

Machine Learning (Aprendizaje automático): Conjunto de técnicas que permiten a un sistema “aprender” patrones a partir de datos; en el proyecto se aplica para predecir el desabastecimiento de productos recurrentes.

Pedido de delivery / Last Mile: Proceso de entrega de un pedido desde el comercio hasta el cliente; el sistema registra y sigue el estado de estas entregas en tiempo real.

Puntos canjeables: Mecanismo de fidelización que otorga puntos por cada compra, los cuales el cliente puede canjear por descuentos o promociones.

Stock crítico: Nivel mínimo de existencias a partir del cual se genera una alerta para reponer el producto antes de que se agote.

CAGR: o Tasa de Crecimiento Anual Compuesta, es una métrica financiera que representa la tasa de crecimiento anual de una inversión durante un período específico, asumiendo que las ganancias se reinvierten cada año.

Dashboard estadístico: Interfaz visual que presenta indicadores, métricas y gráficos de forma sintética para facilitar el monitoreo y análisis del negocio.

KPI: Indicador clave de desempeño utilizado para medir el cumplimiento de objetivos o evaluar el comportamiento de un proceso, área o negocio.

Metabase: Herramienta de inteligencia de negocios y visualización de datos utilizada para construir dashboards y consultar métricas a partir de bases de datos.

Capa analítica: Conjunto de componentes destinados a la consolidación, consulta y visualización de información histórica y estratégica, desacoplados de la operatoria transaccional.

OLTP: Modelo de procesamiento orientado a operaciones transaccionales del sistema, como inserciones, actualizaciones y consultas propias de la operatoria diaria.

OLAP: Modelo de procesamiento orientado al análisis de datos históricos y consolidados, utilizado para consultas analíticas, comparativas y generación de reportes.

Diagnóstico: Situación Actual

Previo al desarrollo del presente proyecto, Club Galo enfrentaba una problemática asociada a la falta de digitalización e integración de sus procesos operativos y comerciales. Como pequeña empresa dedicada a la venta de productos para mascotas y animales de granja, su funcionamiento dependía en gran medida de mecanismos manuales y de herramientas aisladas, insuficientes para acompañar de manera eficiente el volumen de operaciones y las necesidades actuales de sus clientes.

Esta situación generaba dificultades para disponer de información actualizada, confiable y centralizada sobre aspectos fundamentales del negocio, tales como las ventas, el inventario, los pedidos y la relación con los clientes. Al no contar con un sistema que integrara estos procesos, la empresa veía limitada su capacidad para supervisar la operatoria diaria, detectar desvíos, evaluar resultados y tomar decisiones basadas en datos concretos. La información existente se encontraba fragmentada y su actualización dependía de intervenciones manuales, incrementando el riesgo de errores y reduciendo la agilidad de gestión.

A su vez, la ausencia de un canal digital propio y de herramientas orientadas al seguimiento de pedidos y a la fidelización de clientes restringía las posibilidades de crecimiento comercial de Club Galo. En un contexto donde los consumidores valoran cada vez más la compra en línea, la disponibilidad de información inmediata y las experiencias personalizadas, la empresa no contaba con los medios necesarios para fortalecer la relación con sus clientes ni para ofrecer servicios acordes a estas nuevas expectativas.

En consecuencia, el problema central identificado radicaba en la carencia de una solución tecnológica integral capaz de centralizar la información, automatizar tareas operativas, mejorar el control del negocio y brindar soporte a nuevas estrategias comerciales. Esta necesidad justificó el desarrollo de un sistema multiplataforma que permitiera modernizar la gestión de Club Galo, optimizar sus procesos internos y sentar las bases para una operación más eficiente, trazable y orientada al cliente.

Objetivos y Alcance del Proyecto

Objetivo general

Desarrollar e implementar un sistema multiplataforma que optimice la eficiencia operativa y comercial de Club Galo mediante la digitalización integral de los procesos de ventas, inventario y fidelización de clientes.

Objetivos específicos

- Relevar y documentar los procesos operativos y comerciales de Club Galo, identificando las necesidades funcionales y las principales problemáticas asociadas a la gestión de ventas, inventario, entregas y clientes.

- Diseñar una arquitectura de software multiplataforma compuesta por una aplicación de escritorio, una aplicación web y una aplicación web progresiva (PWA), integradas mediante un backend y una base de datos común.
- Desarrollar los módulos funcionales necesarios para la gestión de autenticación y usuarios, productos y stock, ventas, clientes, promociones, notificaciones, entregas a domicilio y administración general del sistema.
- Implementar un canal digital de consulta y compra para clientes, accesible desde la aplicación web y la PWA, que permita visualizar productos, realizar pedidos, gestionar promociones y consultar el estado de las entregas.
- Incorporar mecanismos de fidelización y comunicación con clientes mediante un sistema de puntos, promociones configurables y notificaciones programadas o automáticas.
- Implementar una funcionalidad de predicción de reposición basada en el historial de compras, orientada a generar recordatorios automáticos ante una posible necesidad futura de recompra.
- Integrar servicios externos necesarios para la operación del sistema, incluyendo la gestión de pagos mediante MercadoPago, el almacenamiento de recursos digitales y la generación de reportes operativos exportables.
- Desarrollar una capa analítica desacoplada de la operatoria transaccional, compuesta por procesos ETL, un Data Warehouse y dashboards estadísticos construidos con Metabase, para la consulta de indicadores históricos del negocio.
- Validar el funcionamiento de la solución mediante pruebas manuales, pruebas de aceptación con el usuario y pruebas automatizadas end-to-end sobre los principales endpoints del backend.

Objetivos e indicadores de éxito

Al implantar el sistema se medirán, entre otros:

- **Margen bruto por producto y por categoría** (esperado: visibilidad 100 %).

- **Rotación de inventario** (esperado: reducción de sobre-stock ≥ 30 %).
- **Ventas diarias y ticket promedio** (esperado: incremento de ventas online y fidelización).
- **Tasa de cumplimiento de entrega** (objetivo: ≥ 95 % on-time).

Estas mediciones estarán disponibles en el panel de administración y estadísticas del sistema de escritorio

Supuestos y restricciones

Supuestos:

- Conectividad a Internet estable en el local.
- Uso de navegadores modernos en dispositivos de clientes y empleados.
- Disponibilidad de un PC de escritorio con sistema operativo soportado (Windows, Linux o macOS) para la app desktop.
- Personal motivado y disponible para sesiones de capacitación.

Restricciones:

- Presupuesto limitado al hardware existente; no se adquirirán dispositivos POS adicionales.
- Cronograma inicial de 12 meses para el desarrollo completo.
- Tecnologías circunscritas a JavaScript/Node.js/React y PostgreSQL.
- Cumplimiento de prácticas de seguridad estándar: headers seguros, cifrado TLS y protección de endpoints sensibles mediante JWT.

Metodología y Plan de Trabajo

Marco de desarrollo ágil

El proyecto adoptó **SCRUM** con *sprints* de **dos semanas**. Las ceremonias se mantuvieron ligeras: una reunión de coordinación al final de cada sprint para revisar avances, identificar bloqueos y planificar las tareas siguientes. El flujo diario se gestionó de forma colaborativa, dado que los tres integrantes trabajaron en estrecho contacto.

Organización del equipo y roles

Rol	Integrante	Responsabilidades
Product Owner / Scrum Master	**Felipe Arenillas**	Interfaz con el cliente, priorización del backlog, facilitación de las reuniones y remoción de impedimentos.
Development Team	**Felipe Arenillas**, **Octavio Garcia**, **Julián Gergolet Albornoz**	Análisis, diseño, codificación y pruebas manuales.
Stakeholders externos	**Dueño de Club Galo**	Define requisitos y valida funcionalidades.

Asesores académicos	**Federico Eduardo Porrini, Ignacio Luciano Carreño, Juan José Vulcano**	Recomendaciones de arquitectura, base de datos y tecnologías.

Herramientas de gestión y soporte

- **Jira** (plantilla SCRUM) con tablero **Kanban** para backlog y seguimiento de tareas.
- **Confluence** para notas internas y decisiones de diseño.
- **GitHub** para control de versiones y **GitHub Actions** como pipeline CI/CD (build, lint, deploy).
- **Swagger/OpenAPI** para documentar la API del backend.

Gestión de riesgos y cambios

- Matriz de riesgo elaborada al inicio y revisada periódicamente

No. de Riesgo	Elemento de la EDT	Tipo de riesgo	Riesgo		Síntoma	Impacto (A/M/B)	Probabilidad (A/M/B)	Evaluación		Respuesta	Responsable de la acción de respuesta
			Fuente	Consecuencia				Valor (1 al 9)	Nivel (A/M/B)		
1	Entregable	Técnico	Falta de práctica del grupo en la tecnología aplicada para la creación de la aplicación mobile	Retraso en el desarrollo general	El equipo se queda hasta muy tarde para cumplir con el Gantt	Alto	Media	6	Alto	El equipo se capacita con cursos de youtube antes de comenzar	Felipe Arenillas
2	Compone	Alcance	Compatibil	El sistema no	Ficha técnica del	Bajo	Alta	3	Medio	Intentar obtener lo	Octavio

	nte		idad del lector del código de barras con el sistema	va a brindar lectura con código de barras	lector del codigo de barras					antes posible información sobre la ficha técnica del lector	Garcia
3	Producto	Cronograma	Cambios en los requerimientos	Falta de definición clara de los requisitos desde el inicio del proyecto, cambios en las necesidades del cliente o del mercado	Desviaciones del plan original, afectando el cronograma y presupuesto del proyecto	Alto	Media	6	Alto	Mantener una comunicación continua y clara con el cliente para revisar y validar los requisitos regularmente. Asegurarse de que los requisitos estén documentados de manera exhaustiva y	Felipe Arenillas

										firmados por el cliente. Establecer un proceso formal para manejar las solicitudes de cambios, incluyendo evaluaciones de impacto y aprobaciones necesarias.	
4	Componente	Técnico	Falla en la integración de los datos entre aplicaciones	Incompatibilidades entre las tecnologías utilizadas, errores en la lógica de integración,	Datos incorrectos o inconsistentes, afectando la operatividad y la toma de decisiones	Alto	Media	6	Alto	Diseñar una arquitectura de software que facilite la integración y sea flexible para cambios. Realizar	Julian Gergolet

				falta de pruebas exhaustivas						pruebas de integración continuas para asegurar que las diferentes partes del sistema funcionen correctamente juntas. Establecer estándares y protocolos claros para el manejo de datos entre las aplicaciones.	

5	Entregable	Técnico	Falta de comprensión del software por parte de los empleados	Falta de capacitación adecuada, interfaces de usuario complicadas, falta de documentación accesible	Baja en la productividad, la satisfacción del cliente y la moral del personal, además de errores operacionales	Alto	Baja	3	Medio	Implementar un programa de capacitación integral y continuo para todos los empleados. Proveer manuales de usuario y videotutoriales accesibles y claros. Diseñar interfaces de usuario que sean intuitivas y fáciles de usar.	Julian Gergolet

Identificación y Mitigación de Riesgos - TP8.xlsx

- Cambios de alcance gestionados mediante re-estimación y repriorización del backlog.

Documentación y comunicación

- **Swagger** alojado en el servidor de integración para la API.
- **Confluence** y repositorios Git para documentación técnica adicional.
- Comunicación diaria mediante **WhatsApp** y llamadas; reuniones presenciales/virtuales quincenales con el cliente.

Marco Teórico

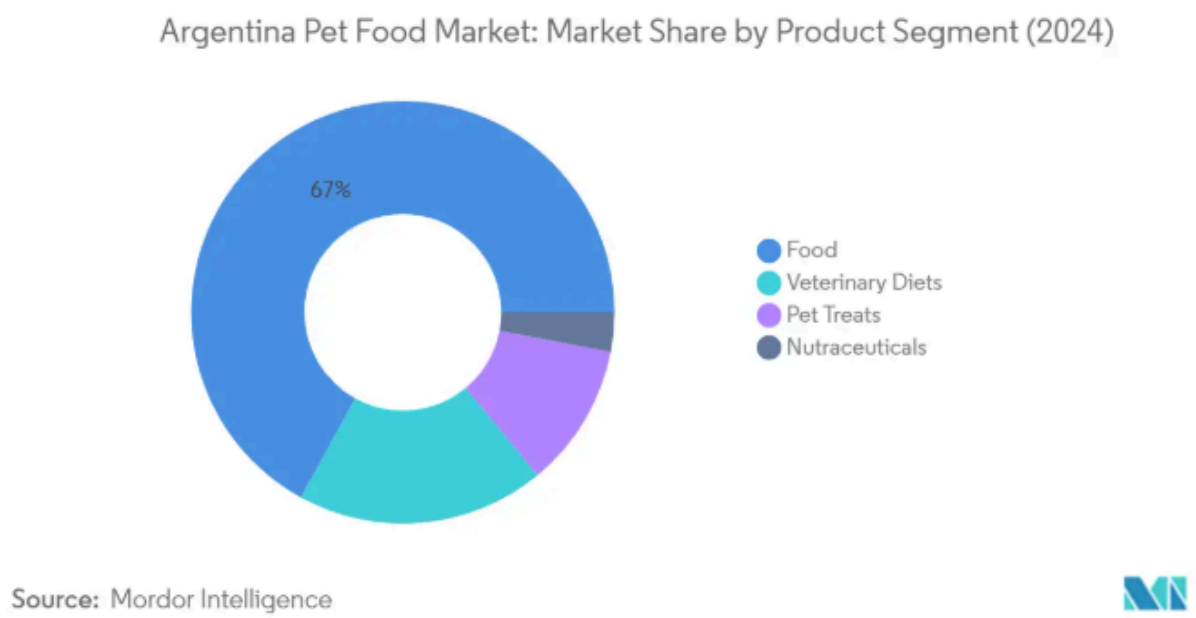
1. Investigación del dominio

1.1 Industria de alimentos y suministros para mascotas y animales de granja en Argentina

Argentina atraviesa un ciclo de fuerte expansión en el cuidado de mascotas. Los estudios realizados por Mordor Intelligence[1], indican que el mercado de alimentos para mascotas pasó de USD 1,3 mil millones en 2023 a un estimado de USD 1,73 mil millones en 2025 y proyecta duplicarse hasta USD 3,65 mil millones en 2030, con una tasa de crecimiento anual compuesta (CAGR - Anual Compound Annual Growth Rate) del 16,1 % para 2025-2030. Ese crecimiento se sustenta en tres factores:

- **Humanización de las mascotas.** El número total de animales de compañía llegó a 46,3 millones en 2022, lo que eleva la demanda de productos premium y fórmulas especializadas (sin cereales, dietas veterinarias, suplementos funcionales).
- **Premiumización.** Las ventas minoristas de alimento seco premium para perros crecieron de USD 38,3 M (2016) a USD 143,4 M (2022), a una CAGR superior al 20 %.
- **Modernización minorista.** Cadenas como Carrefour, Coto o Walmart destinan góndolas exclusivas a la categoría, mientras tiendas especializadas —Puppis, MisPichos— combinan espacios físicos y plataformas on-line con consultas veterinarias remotas.

La demanda también se diversifica: el segmento de golosinas proyecta un alza del 19 % CAGR (2024-2029) y los nutracéuticos ganan participación gracias a la tendencia de bienestar preventivo.



1.2 Tendencia de digitalización y omnicanalidad

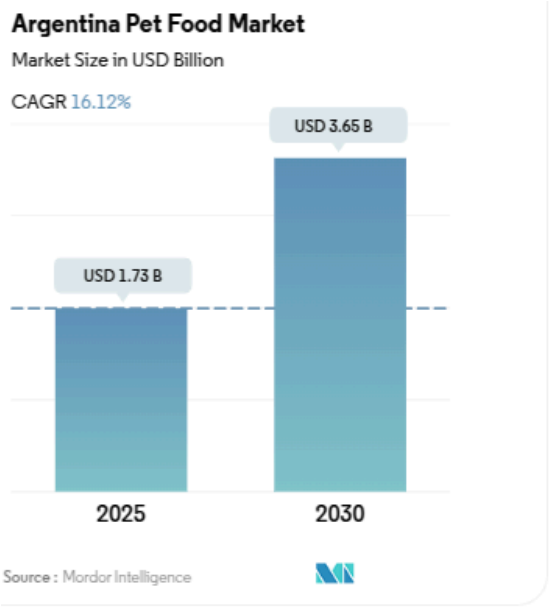
El Informe Anual CACE 2024[2] reporta un incremento interanual del 181 % en la facturación del e-commerce argentino; dentro de este, el rubro mascotas triplicó sus ventas frente a 2021. La incorporación de billeteras digitales (MercadoPago, Modo) y la alta adopción de smartphones impulsan el canal on-line: el valor distribuido por comercio electrónico en alimentos para mascotas pasó de USD 51,8 M (2017) a USD 156,9 M (2022).

Para los retailers se impone una estrategia omnicanal: combinar catálogo on-line, retiro en tienda, entregas programadas y programas de suscripción. Las tiendas especializadas refuerzan la experiencia con asesorías nutricionales virtuales y seguimiento posventa, mientras los grandes supermercados apuestan a la compra “one-stop shop” que integra provisiones del hogar y alimento para mascotas.

1.2.1 Tendencias del mercado

El dinamismo de la “pet economy” argentina se explica por la confluencia de cuatro vectores:

- **Humanización**. Las mascotas pasan a ocupar un rol de compañero familiar y se proyecta que la población total de animales de compañía superará los 49 millones en 2025, con cafés para mascotas, hoteles caninos y servicios de suscripción que refuerzan esa relación[1].
- **Premiumización**. Entre 2016 y 2022 las ventas de alimento seco premium para perros treparon de USD 38,3 M a USD 143,4 M (CAGR 20,7 %). La oferta se especializa en dietas grain-free, ingredientes limitados y fórmulas funcionales con probióticos y Omega-3[1].
- **Nutrición especializada y salud preventiva**. Los segmentos de golosinas funcionales y nutracéuticos proyectan un 19 % CAGR (2024-2029), impulsados por la demanda de suplementos dentales, joint-care y dietas veterinarias[1].
- **Modernización del retail**. Supermercados e hipermercados siguen dominando la góndola (≈ 23 % de las ventas en 2024), pero las cadenas especializadas amplían superficie y servicios digitales; Puppis y MisPichos ofrecen consultas online y recomendaciones personalizadas.



Periodo de estudio	2017 - 2030
Año base para la estimación	2024
Período de datos de pronóstico	2025 - 2030
Tamaño del mercado (2025)	USD 1.73 mil millones
Tamaño del mercado (2030)	USD 3.65 mil millones
TACC (2025 - 2030)	16.12%
Concentración de mercado	Mediana

Principales actores

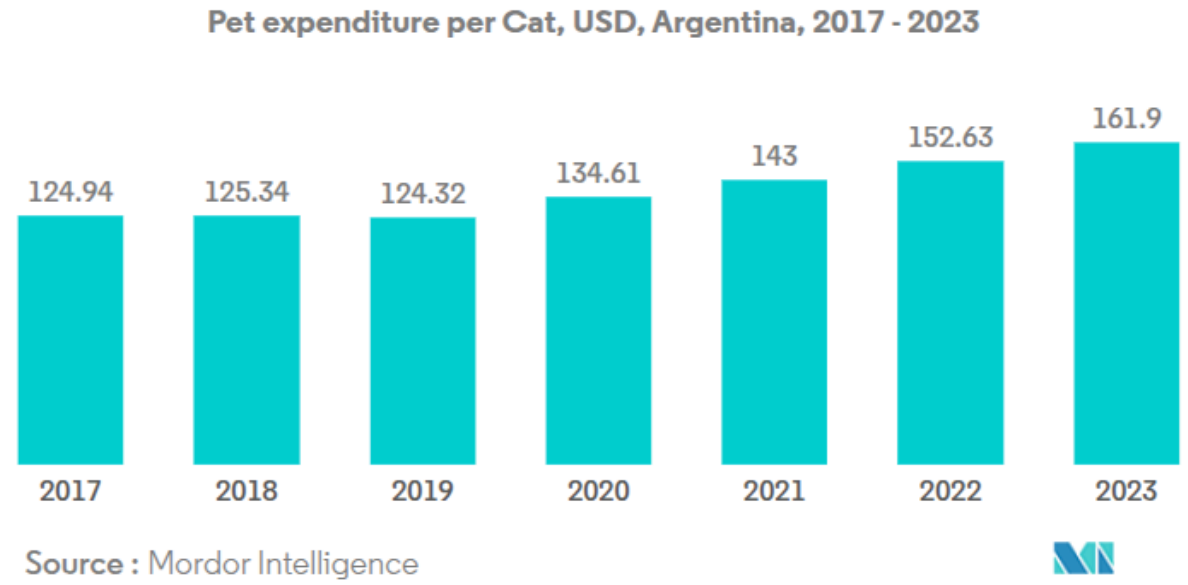
*Descargo de responsabilidad: los jugadores principales están clasificados sin ningún orden en particular

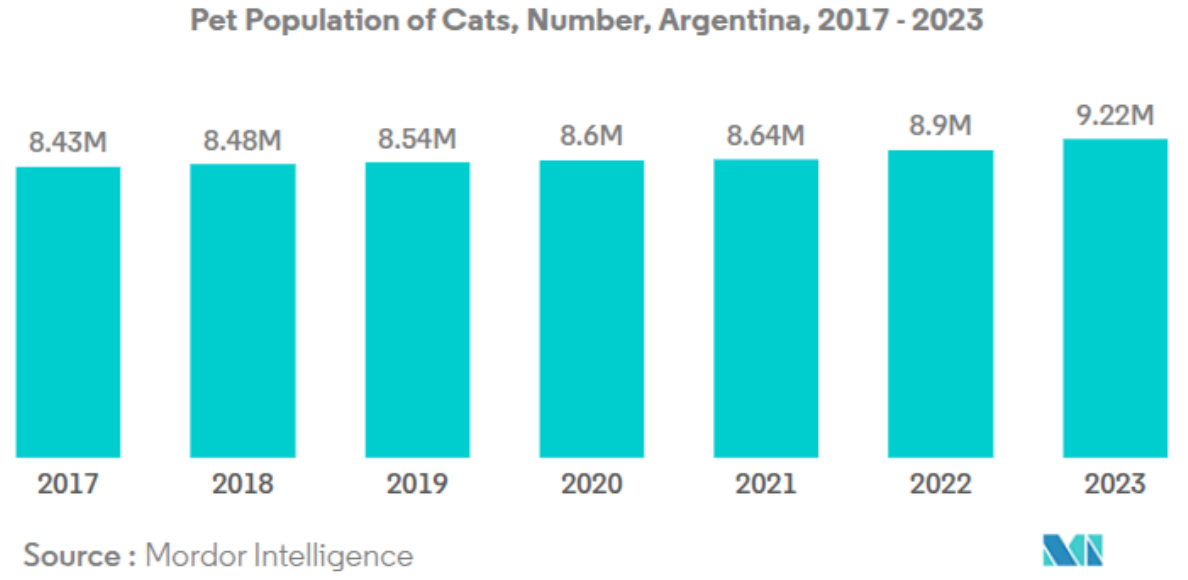
1.2.2 Suba del consumo premium y gasto por mascota

El gasto total en alimento para mascotas creció 19,8 % entre 2019 y 2022. Por especie, el desembolso en perros subió 34,4 %, en gatos 22,1 % y en "otras mascotas" 12,9 % en el mismo período[1]. El ticket anual promedio alcanzó USD 428 por animal (6,4 % por debajo del gasto en Brasil), impulsado por:

- Mayor aceptación/preferencia por productos premium y super-premium.
- Preferencia por raciones secas "croquetas" —67 % de share— por conveniencia de almacenaje, complementadas por sobres húmedos de alta palatabilidad.
- Creciente adopción de dietas veterinarias para gestionar diabetes, alergias o peso.

Una métrica elocuente es el consumo per cápita de alimento balanceado, que llegó a 49,1 kg/persona en 2023, el más alto de Sudamérica[3].





1.2.3 Canal on-line: crecimiento y características

Aunque en 2022 el 86 % de las ventas se concretó aún en tienda física, el canal digital pasó de USD 51,8 M (2017) a USD 156,9 M (2022) y se proyecta un 19 % CAGR entre 2024 y 2029[1]. Los motores de ese salto son:

- Internet (85 %) y smartphones (≈ 78 % de la población).
- Integración de opciones de pago “one-tap” vía billeteras tipo MercadoPago.
- Logística de última milla con entrega en 24-48 h en zonas urbanas.
- Incentivos: descuentos exclusivos on-line, puntos de fidelidad y envíos gratis.

Para las tiendas especializadas el e-commerce representa una doble vía: amplía el alcance geográfico y, gracias a los datos transaccionales, habilita programas de suscripción y reposición programada (auto-ship) que mejoran la recurrencia de compra.

1.3 Procedimientos actuales relevados en Club Galo

Como parte del análisis previo al desarrollo de la solución, se realizó un relevamiento de los procedimientos utilizados por Club Galo para gestionar su operatoria diaria. Este relevamiento permitió identificar el modo en que se administraban el inventario, las ventas, las entregas a domicilio y la relación con los clientes antes de la incorporación del sistema propuesto. Por lo tanto, las prácticas descriptas en esta sección corresponden específicamente a la organización analizada y no pretenden caracterizar de manera general a la totalidad de las forrajerías o comercios del rubro.

En relación con el inventario, Club Galo realizaba el recuento de productos manualmente, registrando inicialmente la información en una libreta y transcribiéndola posteriormente a una planilla Excel. El procedimiento implicaba

controlar tanto el stock disponible en depósito como los productos ubicados en el frente de venta. Además, las unidades vendidas o reservadas para delivery debían actualizarse manualmente. Según lo relevado, un conteo exhaustivo demandaba una jornada completa de trabajo de un empleado, con apoyo ocasional del cajero, y la información obtenida podía quedar desactualizada rápidamente ante nuevas operaciones.

Las ventas se concretaban de manera presencial o mediante pedidos recibidos por teléfono y WhatsApp. Los precios de los productos se administraban individualmente en una planilla, requiriendo actualizaciones manuales ante modificaciones de costos o precios finales. Esta modalidad dificultaba determinar con precisión el margen obtenido por producto, consultar información histórica de las transacciones y analizar qué artículos presentaban mayor o menor rotación.

En cuanto a las entregas a domicilio, los pedidos se coordinaban mediante mensajes o llamadas telefónicas. No existía una herramienta específica para registrar y consultar el estado de cada envío, por lo que la empresa dependía de la comunicación del cadete para conocer si un pedido había sido entregado o si se habían producido demoras. Esta situación limitaba la trazabilidad del proceso y reducía la capacidad de brindar información oportuna al cliente.

Por otra parte, Club Galo no disponía de un registro estructurado de clientes, mascotas, preferencias de compra o frecuencia de consumo. Tampoco contaba con un sistema de puntos ni con mecanismos formales para administrar promociones orientadas a la fidelización. En consecuencia, la empresa no podía identificar patrones de recompra ni desarrollar acciones personalizadas para fortalecer la relación comercial con sus clientes.

Los procedimientos relevados mostraban una operatoria dependiente de tareas manuales y de información distribuida en medios no integrados. Las principales consecuencias identificadas se sintetizan en la siguiente tabla:

Área	Consecuencia identificada	Evidencia relevada en Club Galo
Inventario	Horas hombre desperdiciadas y probabilidad de error alta.	Un conteo exhaustivo toma un día laboral completo y suele quedar desactualizado al día siguiente.
Stock	Sobre-stock en productos de baja rotación y faltantes en artículos clave.	Se identifican pallets con mercadería estacionada; se venden sustitutos sin medir ventas perdidas.
Pricing	Margen incierto por variaciones de costos y precios finales.	El dueño actualiza precios “a ojo” y no sabe la ganancia exacta por producto.
Delivery	Falta de visibilidad del pedido y potencial insatisfacción del cliente.	Solo se sabe si hubo demoras cuando el cadete lo reporta.
Fidelización	Dificultad para retener clientes y generar recompra.	No se registran datos de clientes, puntos ni promociones.

En función de este relevamiento, se identificó la necesidad de incorporar una solución tecnológica que centralizara la información operativa, disminuyera la dependencia de registros manuales y brindara herramientas para el seguimiento de ventas, stock, entregas y clientes. Estos procedimientos constituyeron el punto de partida para la definición de los requerimientos funcionales y para el diseño de la solución propuesta en el presente proyecto.

1.4 Marco regulatorio

La normativa que rige la fabricación, comercialización y distribución de alimentos para animales en la Argentina se articula principalmente a través del Servicio Nacional de Sanidad y Calidad Agroalimentaria (SENASA). Dos resoluciones enmarcan el escenario actual:

- **Resolución 594/2015** —fue la primera Norma Técnica integral para alimentos destinados a la alimentación animal. Exigió que todo producto balanceado, suplemento, núcleo vitamínico, premezcla o aditivo se inscribiera en el Registro de Productos para Alimentación Animal. Además, impuso la figura del *director técnico* profesional, buenas prácticas de manufactura, planes de retiro de lote y requisitos de rotulado que incluyen número de registro, fecha de vencimiento y composición garantizada. La trazabilidad se limitaba a conservar documentación física de compra-venta por un mínimo de dos años.
- **Resolución 1415/2024** —actualiza y consolida el marco anterior. Mantiene la obligatoriedad de registrar productos y establecimientos, pero digitaliza gran parte del flujo: los formularios de inscripción y renovación ahora se gestionan en línea y se incorpora el Sistema Nacional de Trazabilidad de Productos Veterinarios. La nueva norma obliga a que cada envase lleve un código QR que enlaza al registro SENASA para verificar autenticidad y facilitar retiros en caso de incumplimiento. También amplía las sanciones ante falta de rotulado o venta de productos vencidos y exige declarar cambios de fórmula con antelación (Boletín Oficial, 04-dic-2024).

Estas disposiciones impactan directamente en el diseño de un sistema, la adopción de un software de gestión integral debe contribuir no solo a la eficiencia operativa sino también al cumplimiento normativo y a la inocuidad alimentaria.

2. Investigación de soluciones existentes

El mercado argentino ofrece dos grandes clases de herramientas para comercios minoristas de mascotas y forrajerías: los ERP / POS generalistas —habitualmente SaaS o instalables on-premise— y los marketplaces de delivery que brindan un escaparate propio a cambio de una comisión por venta.

En el primer grupo conviven productos maduros, como Contabilium o Wynges, que resuelven facturación, stock y reportes, con plataformas globales focalizadas en e-commerce como Shopify y Tiendanube, donde prevalecen plantillas prediseñadas y módulos que se activan bajo suscripción. En el segundo grupo encontramos a PedidosYa y Rappi apuesta a la logística de última milla; facilita la exposición a un público amplio, pero no da control sobre la marca ni acceso al historial del cliente y aplica aranceles que van del diez al treinta y cinco por ciento.

Un repaso funcional muestra que ninguno de los competidores integra, en una única oferta, una aplicación de escritorio para operación interna, un frontend web/PWA propio, un módulo CRM especializado en mascotas y un motor predictivo de reposición. Las soluciones ERP pueden personalizarse, aunque el costo de consultoría crece con cada ajuste; los marketplaces, por su parte, no permiten personalización ni acceso directo a métricas de fidelidad.

Plataforma	Tipo	Personalización	Predicción de reposición	Multi-plataforma (Desktop/Web/Mobile)	Gestión CRM	Costo inicial
Contabilium	SaaS argentino	Media (módulos)	✖	Web	Básico (datos de clientes)	Medio (licencias)
Odoo Community	ERP open source	Alta (modular)	✖ (addons de terceros)	Web / mobile app	✔ (módulo CRM)	Bajo (licencia) / Alto (customización)
Shopify	SaaS global	Baja (plantillas)	✖	Web / mobile app	Básico	Bajo (suscripción + apps)

Tiendanube	SaaS LATAM	Media	✖	Web	Limitado	Bajo
PedidosYa	Marketplace de delivery	Nula (catálogo propio de la plataforma)	✖	Web / mobile app	✖	Sin cuota; comisión 10-30 % por venta
Rappi	Marketplace de delivery	Nula	✖	Web / mobile app	✖	Sin cuota; comisión 15-35 % por venta
Fudo	SaaS argentino (POS-Rest)	Media (módulos)	✖	Web / mobile app	Básico	Bajo (suscripción mensual)

Wynges	SaaS argentino (ERP PyME)	Media (módulos)	✖	Web	Módulo opcional	Medio
Calipso	ERP argentino (on-prem / cloud)	Media (módulos)	✖	Desktop / Web / mobile app	✔	Alto (licencia + consultoría)
Atera	SaaS global (IT MSP)	Baja (orientado a soporte)	✖	Desktop / Web	✖	Bajo (suscripción)

Pedix	SaaS LATAM (tienda on-line + WhatsApp Business)	Baja-Media (plantillas)	✖	Web / PWA (catálogo compartible por link)	Limitado (datos de cliente y pedido)	Bajo (suscripción mensual)

La tabla muestra que ninguna plataforma comercial integra de serie un motor predictivo de reposición ni una aplicación de escritorio para operación interna.

3. Investigación tecnológica

3.1 Bases de datos

Las bases de datos constituyen un componente central en los sistemas de gestión, ya que permiten almacenar, relacionar y consultar información necesaria para la operación diaria, como productos, clientes, ventas, pedidos, movimientos de stock y promociones. En aplicaciones comerciales, resulta especialmente relevante que el motor seleccionado garantice integridad de la información y permita ejecutar operaciones transaccionales de manera confiable.

Entre los motores relacionales más difundidos se encuentran MySQL, MariaDB y PostgreSQL. Estas alternativas permiten representar información estructurada mediante tablas y relaciones, y ofrecen mecanismos para mantener la consistencia de los datos durante operaciones de alta, modificación y consulta. PostgreSQL, además, incorpora soporte para estructuras semiestructuradas mediante el tipo de dato JSONB, que permite almacenar atributos flexibles y realizar consultas sobre ellos sin abandonar el modelo relacional.

Otra alternativa posible para almacenar información con estructuras variables consiste en utilizar bases de datos documentales, como MongoDB. Este enfoque puede resultar conveniente en sistemas donde predomina información altamente flexible o documentos con estructuras heterogéneas. Sin embargo, cuando una solución combina datos comerciales fuertemente relacionados con atributos complementarios semiestructurados, también es posible resolver ambos tipos de necesidad dentro de un motor relacional que incluya soporte documental.

De acuerdo con los análisis comparativos consultados, PostgreSQL presenta ventajas relevantes en operaciones que utilizan JSONB e índices especializados para consultas sobre documentos, mientras que MySQL y PostgreSQL ofrecen desempeños competitivos en cargas transaccionales tradicionales[4] [5] [6]. Estas características permiten comprender por qué los motores relacionales modernos continúan siendo alternativas adecuadas para sistemas de gestión que requieren consistencia transaccional y, al mismo tiempo, cierto grado de flexibilidad en el almacenamiento de información.

3.2 Backend

Para la construcción de servicios backend existen diferentes frameworks y entornos de desarrollo que permiten exponer funcionalidades, gestionar solicitudes, interactuar con bases de datos y organizar la lógica de negocio de una aplicación. Entre las alternativas analizadas se encuentran Django, NestJS y Node.js junto con Express, cuyas características se describen a continuación.

3.2.1 Django

Django es un framework web de código abierto desarrollado en Python que incorpora, dentro de una misma solución, múltiples funcionalidades para la construcción de aplicaciones backend. Entre sus componentes se encuentran un ORM para acceso a datos, mecanismos de autenticación, un panel de administración y herramientas para la gestión de rutas, formularios y seguridad[18].

Su enfoque se caracteriza por ofrecer una estructura integral desde el inicio del desarrollo, reduciendo la necesidad de incorporar múltiples bibliotecas externas para resolver funcionalidades habituales. Esta característica puede resultar conveniente en proyectos que requieran contar rápidamente con módulos administrativos, gestión de usuarios y persistencia de datos dentro de un ecosistema unificado.

3.2.2 NodeJS

Node.js, lanzado en 2009, introdujo el modelo de E/S no bloqueante sobre el motor V8 y rápidamente se convirtió en la base de innumerables backend —más de 2,1 millones de proyectos en GitHub y el 42 % de las respuestas del Stack Overflow Developer Survey 2023 afirman usarlo en producción. Express, aparecido en 2010, aportó un routing minimalista y, desde entonces, ha acumulado más de 60.000 estrellas y 27 millones de descargas semanales en npm.

Su madurez, estabilidad y amplia comunidad se traducen en documentación abundante, una gran variedad de bibliotecas y soluciones disponibles para casos de uso frecuentes en aplicaciones web. Express, por su enfoque minimalista, permite estructurar rutas y servicios con un grado elevado de flexibilidad, dejando al equipo de desarrollo la selección de bibliotecas complementarias según las necesidades de cada aplicación.

3.2.3 NestJS

NestJS —publicado en 2017— adopta una arquitectura inspirada en Angular, basada en decoradores y TypeScript por defecto. Sus estrellas en GitHub superan las 63.000, señal de una adopción creciente, pero las consultas en foros y la literatura técnica siguen siendo muy inferiores a las de Express.

NestJS ofrece una estructura orientada a aplicaciones backend modulares, incorporando conceptos como módulos, providers, controladores y guards[16]. Su utilización junto con TypeScript favorece la definición explícita de tipos y la organización de servicios en proyectos que requieren una arquitectura más estructurada. A su vez, este enfoque supone familiaridad con los patrones y convenciones propios del framework.

3.2.4 Comparación NodeJS (ExpressJS) vs NestJS

De acuerdo con la comparación detallada de Radixweb (2024) «NestJS vs ExpressJS»[7], NestJS ofrece ventajas tangibles en estructura, inyección de dependencias y soporte nativo para pruebas, logrando incluso un rendimiento levemente superior cuando se compila a código nativo. Sin embargo, la misma fuente destaca que la curva de aprendizaje es más pronunciada: NestJS exige comprender conceptos como modules, providers y guards, además de abrazar por completo TypeScript para acceder a la mayoría de ejemplos oficiales.

3.3 Frontend

El frontend constituye la capa de interacción directa entre el usuario y el sistema, encargándose de presentar la información, capturar acciones y comunicar las operaciones requeridas al backend. En aplicaciones web modernas, esta capa suele desarrollarse mediante bibliotecas o frameworks basados en componentes, que permiten construir interfaces reutilizables, dinámicas y adaptables a distintos tamaños de pantalla.

Entre las alternativas más utilizadas se encuentran React, Angular y Vue. React propone un enfoque basado en componentes y una amplia disponibilidad de bibliotecas complementarias; Angular ofrece un framework más integral, con múltiples herramientas incorporadas para estructurar aplicaciones; y Vue presenta una alternativa progresiva, orientada a facilitar la adopción incremental y la construcción de interfaces reactivas. La elección entre estas tecnologías depende del alcance del proyecto, la experiencia del equipo, la complejidad esperada de la aplicación y las necesidades de mantenimiento.

Además de la biblioteca o framework utilizado para construir la interfaz, resulta necesario considerar las herramientas empleadas durante el desarrollo y para la generación de los artefactos de producción. Entre estas se encuentran servidores de desarrollo y bundlers, como Vite, así como frameworks web basados en React, como Next.js, que incorporan sus propios mecanismos de compilación. Aspectos como el tiempo de arranque, la recarga durante el desarrollo y el proceso de build pueden influir en la productividad del equipo y en la organización de un proyecto frontend.

3.4 Plataforma mobile – PWA vs. React Native vs. Flutter

Para ofrecer acceso desde dispositivos móviles, existen distintas alternativas tecnológicas, entre las que se destacan las aplicaciones nativas o multiplataforma y las aplicaciones web progresivas. Cada enfoque presenta características particulares en términos de experiencia de usuario, reutilización de código, acceso a capacidades del dispositivo, distribución y mantenimiento.

Flutter es un framework multiplataforma que permite construir aplicaciones para distintos sistemas operativos a partir de una base de código común, utilizando el lenguaje Dart. Su enfoque facilita la generación de interfaces con comportamiento

consistente entre plataformas y permite compilar aplicaciones distribuidas mediante tiendas oficiales.

React Native constituye otra alternativa para el desarrollo móvil multiplataforma. Al basarse en JavaScript y en un modelo de componentes relacionado con React, permite reutilizar conocimientos y parte de la lógica utilizada en aplicaciones web. Al igual que otras soluciones móviles empaquetadas, su distribución habitualmente requiere generar builds y publicarlos en las tiendas correspondientes.

Por otra parte, una Progressive Web App (PWA) es una aplicación web que incorpora capacidades orientadas a mejorar la experiencia en dispositivos móviles, tales como instalación desde el navegador, ejecución en pantalla completa, almacenamiento en caché y, según el entorno disponible, notificaciones o acceso a determinadas funciones del dispositivo. A diferencia de las aplicaciones publicadas en tiendas, una PWA puede distribuirse directamente a través de una URL y actualizarse mediante el despliegue de la aplicación web.

En consecuencia, la selección entre Flutter, React Native y PWA depende de factores como las capacidades nativas requeridas, la estrategia de distribución, la experiencia técnica del equipo, la posibilidad de reutilizar componentes existentes y los costos asociados al mantenimiento y publicación de la aplicación.

3.5 Inteligencia de negocio – Algoritmo de reposición

Con el objetivo de mejorar la experiencia del cliente y aumentar la recurrencia de compra, se busca generar notificaciones automáticas de reposición para productos adquiridos de manera periódica. La lógica detrás de esta funcionalidad radica en anticiparse a la necesidad del cliente y ofrecer el producto en el momento oportuno, fortaleciendo así la relación del cliente y mejorando la percepción del servicio.

El enfoque propuesto consiste en registrar cada compra realizada por un cliente, calcular el intervalo promedio entre adquisiciones de un mismo ítem y, con base en dicho promedio, programar un recordatorio unos días antes de la fecha estimada de

reposición. La finalidad es que el sistema aprenda de manera progresiva los hábitos de consumo asociados a cada cliente-producto, y pueda emitir alertas que reduzcan la probabilidad de quiebre de stock en el momento de mayor necesidad.

3.5.1 Algoritmo de reposición en el backend

Implementar esta lógica directamente en la capa de backend, implica que tras cada compra, el servicio correspondiente deba calcular el nuevo promedio y generar una tarea programada para emitir la notificación anticipada. Sin embargo, esta aproximación presenta varias limitaciones.

En primer lugar, implica que cada operación de compra ejecute procesos adicionales en el servidor, como cálculos, actualizaciones en la base de datos y reprogramación de tareas, lo que podría convertirse en un cuello de botella ante un crecimiento en el volumen de transacciones. Toda la carga de procesamiento recae sobre el backend, comprometiendo su rendimiento.

En segundo lugar, la programación de notificaciones depende de herramientas externas al sistema de gestión de base de datos (como cron jobs o colas de mensajes), lo cual introducía nuevas dependencias, riesgos y puntos de falla. Ante eventuales fallos (fallo en la ejecución de un job, o reinicio del servidor), se pierde la fiabilidad del sistema de alertas, afectando la experiencia del usuario.

En tercer lugar, mantener la lógica de cálculo en el backend obliga a duplicar tareas entre el código y el motor SQL. Muchas de estas operaciones, cómo calcular promedios móviles, almacenar históricos o generar fechas futuras, se realizan más naturalmente dentro de la base de datos. La necesidad de traducir cada operación entre dos capas diferentes aumenta la posibilidad de errores y dificulta el mantenimiento del sistema.

3.5.2 Algoritmo de reposición en la base de datos

Trasladar el algoritmo de reposición al motor de base de datos, implementando mediante procedimientos almacenados y triggers en PL/pgSQL (lenguaje nativo de PostgreSQL) conlleva varias ventajas.

Procesar los datos directamente en la base permite que las consultas de compras anteriores y el cálculo de promedios se realicen en la misma transacción que la inserción de la nueva compra. Esto evita latencias innecesarias entre capas y garantiza que los datos utilizados sean siempre consistentes y actualizados.

Mediante el uso de triggers, cada vez que se registra una compra se activa automáticamente un procedimiento almacenado que recupera el historial del cliente-producto, calcula la nueva frecuencia estimada, actualiza los registros de seguimiento y programa la fecha de notificación correspondiente. De esta forma, con la ayuda de una tarea programada, es posible agendar las diferentes notificaciones.

Centralizar la lógica en la base de datos también facilita el mantenimiento: cualquier modificación en la fórmula de cálculo, los días de anticipación o condiciones específicas (como ajustes por temporada o promociones especiales) puede realizarse en un único punto, sin necesidad de volver a desplegar el backend o modificar artefactos externos.

Por último, al ejecutarse dentro de la misma transacción que registra la compra, se asegura la atomicidad de todo el proceso: si ocurre un error, tanto la compra como los cálculos asociados y la programación de la notificación son revertidos de forma conjunta, garantizando la integridad de los datos y la coherencia del sistema.

3.6 Pasarelas de pago

Las pasarelas de pago permiten procesar transacciones electrónicas mediante proveedores especializados, evitando que una aplicación comercial deba gestionar directamente la totalidad del flujo de cobro y los datos sensibles asociados a los medios de pago. En soluciones de comercio electrónico, su evaluación suele considerar aspectos como los métodos aceptados, la experiencia de uso, la integración técnica disponible, los costos por operación y los mecanismos ofrecidos para la confirmación del pago.

Entre las alternativas disponibles en el contexto argentino se encuentran MercadoPago y Pagos360. Ambas ofrecen mecanismos para integrar cobros digitales a aplicaciones comerciales, aunque presentan diferencias en sus modalidades de alta, métodos aceptados, experiencia de pago móvil, documentación disponible y estructura de costos. La siguiente comparación resume las principales características relevadas para cada alternativa.

3.6.1 MercadoPago vs. Pagos360

Característica	MercadoPago Checkout Pro	Pagos360
Cuenta / alta	Se habilita con la misma cuenta de MercadoLibre; trámite 100 % on-line[12]	Alta sujeta a validación bancaria y contrato escrito[13]
Métodos aceptados	Tarjeta, saldo MP, efectivo en Rapipago/PagoFácil, billeteras	Tarjetas y débito on-line; no posee billetera propietaria

One-tap mobile	✔ Pago biométrico directo en la app si el usuario la tiene instalada	✖ Deriva al formulario de tarjeta
SDKs & docs	Documentación actualizada mensualmente; librerías JS, iOS, Android	API REST y Webhooks; ejemplos limitados
Tarifas (mayo 2025)	6,39 % + IVA por transacción con tarjeta < ARS 50 k	6 % + IVA + costos fijos de alta
Reputación usuario	Integra puntos y promociones de MercadoLibre	Sin programa equivalente

3.7 Infraestructura

3.7.1 Comparación de proveedores cloud

La infraestructura de despliegue de una solución de software puede apoyarse en diferentes modelos de servicio en la nube. Entre ellos se encuentran las plataformas como servicio (PaaS), que permiten desplegar aplicaciones delegando gran parte de la administración de servidores, y las alternativas de infraestructura como servicio (IaaS), que ofrecen mayor flexibilidad de configuración a cambio de una mayor responsabilidad operativa.

Para comparar estas alternativas pueden considerarse criterios como el costo inicial, la facilidad de despliegue, la capacidad de escalamiento, la administración requerida y el grado de flexibilidad ofrecido. La Tabla 2 presenta una comparación general de

proveedores y modalidades relevantes para aplicaciones web desplegadas en la nube.

Proveedor	Modalidad	Ventajas principales	Desventajas principales	Costo inicial*
Heroku	PaaS	Despliegue “git push”, ecosistema Buildpacks, documentación sencilla, escalar con *dynos*	Límites de CPU en planes gratuitos, coste por duración de proceso	USD 0 (Free Tier) / USD 5 mes (Hobby)
Railway	PaaS	Interfaz muy simple, bases de datos incluidas en minutos, precios por uso	Comunidad más pequeña, planes con límites estrictos de horas	USD 0 / USD 5 mes

Render	PaaS	Auto-deploy con Git, incluye CDN y dominio SSL, precios competitivos	Tiempo de *cold start* alto en planes gratuitos	USD 0 / USD 7 mes
Vercel	PaaS orientado a Frontend	SSR/ISR optimizado para Next.js, CDN global	Backends limitados (serverless), coste rápido al crecer	USD 0 / USD 20 mes (Pro)
AWS (Elastic Beanstalk + servicios)	IaaS/PaaS	Máxima flexibilidad, zona local en São Paulo, amplia oferta	Configuración compleja, curva de aprendizaje pronunciada, facturación difícil de predecir	USD 0 (Free Tier 12 meses) / variable

Google Cloud (App Engine + servicios)	IaaS/PaaS	Herramientas de IA, facturación por segundos, buenas cuotas gratuitas	Curva de aprendizaje, menos guías en castellano, precios en USD	USD 0 (Free Tier 12 meses) / variable
Microsoft Azure (App Service)	IaaS/PaaS	Integración con ecosistema MS, SLA 99,95 %	Panel menos intuitivo, documentación dispersa	USD 0 (Free Tier) / variable

** Precios consultados en mayo 2025 para el plan más económico capaz de ejecutar un contenedor Node.js de pruebas*

Las plataformas analizadas presentan distintos compromisos entre flexibilidad, costo y esfuerzo de administración. Las alternativas IaaS o aquellas apoyadas en ecosistemas de servicios cloud amplios permiten configurar arquitecturas con mayor nivel de personalización y escalabilidad, aunque requieren una gestión más detallada de infraestructura, seguridad y costos. Por su parte, las plataformas PaaS tienden a simplificar el proceso de despliegue y operación de aplicaciones, delegando parte de la administración técnica al proveedor.

En consecuencia, la elección de una plataforma de despliegue depende de factores como el tamaño de la solución, el volumen esperado de uso, las capacidades técnicas disponibles para administrarla, los costos operativos aceptables y la necesidad de personalización de la infraestructura.

3.8 Dashboards estadísticos y herramientas de analítica visual

3.8.1 Concepto de dashboard estadístico

Un dashboard estadístico es una herramienta de visualización de información que permite presentar, de forma sintética e intuitiva, un conjunto de indicadores relevantes para el análisis y seguimiento de un proceso, área o negocio. Su objetivo principal es transformar datos en información útil para la toma de decisiones, facilitando la interpretación rápida de métricas, tendencias, comparaciones y comportamientos a lo largo del tiempo.

A diferencia de los reportes tradicionales, que suelen consistir en tablas extensas o listados estáticos de datos, un dashboard organiza la información mediante gráficos, tarjetas de indicadores, filtros y elementos visuales que permiten identificar patrones, desvíos y oportunidades de mejora con mayor claridad. De esta manera, el usuario no solo accede a los datos, sino que también puede analizarlos en contexto y obtener una visión general del estado de una operación o del desempeño de un conjunto de variables.

En el ámbito empresarial, los dashboards estadísticos suelen utilizarse para monitorear indicadores clave de desempeño (KPIs), tales como ventas, rentabilidad, stock, comportamiento de clientes, productividad o evolución de pedidos. Su valor

radica en que concentran información dispersa en una única interfaz visual, permitiendo un seguimiento más eficiente y favoreciendo decisiones más ágiles, fundamentadas y consistentes.

En el contexto del presente proyecto, el dashboard estadístico se entiende como una capa de apoyo a la gestión, orientada a consolidar información histórica y operativa para su consulta visual por parte de los responsables del negocio. Esto permite complementar la operatoria diaria del sistema con una perspectiva analítica, facilitando el control de la actividad comercial y el seguimiento de métricas relevantes para la administración.

3.8.2 Breve historia y evolución

Los dashboards estadísticos tienen su origen en la necesidad de resumir grandes volúmenes de información en formatos comprensibles para la supervisión y la toma de decisiones. En sus etapas iniciales, esta función era cubierta por reportes impresos, planillas de cálculo y cuadros de mando elaborados de forma manual, en los que los responsables de una organización analizaban indicadores a partir de tablas y resúmenes periódicos. Si bien estos mecanismos permitían cierto control de la información, presentaban limitaciones importantes en términos de actualización, flexibilidad y rapidez de análisis.

Con el avance de los sistemas de información empresariales y las bases de datos, surgieron herramientas capaces de automatizar la generación de reportes y de centralizar información proveniente de distintas áreas del negocio. En este contexto comenzó a consolidarse el concepto de cuadro de mando o dashboard como una interfaz que no solo mostraba datos, sino que los organizaba visualmente para facilitar su interpretación. Esta evolución estuvo acompañada por el crecimiento de la inteligencia de negocios (Business Intelligence), disciplina que impulsó el uso de indicadores clave de desempeño y de modelos de análisis orientados al soporte gerencial.

Posteriormente, la aparición de soluciones de visualización interactiva permitió un cambio significativo en la forma de consumir la información. Los dashboards dejaron de ser simples resúmenes estáticos y pasaron a incorporar filtros, segmentaciones, comparaciones temporales y gráficos dinámicos, posibilitando que distintos usuarios pudieran explorar los datos según sus necesidades. Esta transformación también estuvo vinculada al desarrollo de arquitecturas analíticas más robustas, como los Data Warehouse y los procesos ETL, que hicieron posible consolidar información histórica y consultarla de manera más eficiente.

En la actualidad, los dashboards forman parte de una evolución más amplia en la gestión de datos dentro de las organizaciones. Ya no se los considera únicamente una herramienta de reporte, sino un componente estratégico para el monitoreo del negocio, la detección de tendencias y el apoyo a decisiones basadas en evidencia. Su evolución ha estado marcada por una transición desde reportes manuales y estáticos hacia plataformas visuales, interactivas y conectadas con estructuras analíticas cada vez más especializadas.

3.8.3 Estado actual

En la actualidad, los dashboards estadísticos ocupan un lugar central dentro de los sistemas de apoyo a la toma de decisiones en organizaciones de distintos tamaños y rubros. Su utilización se ha extendido debido a la necesidad de contar con información accesible, actualizada y presentada de manera comprensible, especialmente en contextos donde los datos operativos crecen de forma constante y resulta inviable analizarlos manualmente. En este escenario, los dashboards permiten sintetizar grandes volúmenes de información y convertirlos en una visión clara del desempeño de un negocio o proceso.

Uno de los rasgos distintivos del estado actual de estas herramientas es su fuerte orientación a la interacción y al autoservicio analítico. A diferencia de los reportes tradicionales, los dashboards modernos permiten aplicar filtros, cambiar rangos temporales, segmentar información y explorar distintas dimensiones de análisis sin necesidad de generar nuevos informes manualmente. Esto posibilita que usuarios no técnicos puedan acceder a métricas relevantes y comprender rápidamente el comportamiento de variables como ventas, stock, rentabilidad, productividad o evolución de clientes.

Otra característica importante del contexto actual es la integración de los dashboards con arquitecturas de datos más especializadas. En muchos casos, estas herramientas se apoyan en procesos ETL y en estructuras como Data Warehouse, que permiten consolidar información histórica proveniente de sistemas transaccionales. De este modo, los dashboards ya no dependen exclusivamente de consultas directas sobre bases operativas, sino que se alimentan de capas analíticas diseñadas específicamente para mejorar la consistencia, el rendimiento y la calidad de la información visualizada.

Asimismo, el mercado actual ofrece una amplia variedad de herramientas para la construcción de dashboards, con enfoques que van desde soluciones orientadas a la inteligencia de negocios y el análisis comercial hasta plataformas más enfocadas

en monitoreo técnico, observabilidad o series temporales. Esta diversidad ha facilitado la adopción de dashboards en diferentes contextos, permitiendo seleccionar la alternativa más adecuada según las necesidades funcionales, el nivel técnico del equipo, el tiempo de desarrollo disponible y el tipo de información que se desea analizar.

En este marco, los dashboards estadísticos se consolidan hoy como un componente habitual en la gestión moderna de la información, ya que permiten complementar la operatoria diaria de los sistemas con una perspectiva analítica y gerencial. Su valor actual no reside únicamente en mostrar datos, sino en hacerlo de manera estructurada, visual y orientada a la acción, contribuyendo a una gestión más eficiente y fundamentada.

3.8.4 Herramientas populares de BI

En la actualidad existen diversas herramientas orientadas a la construcción de dashboards y visualizaciones analíticas, cada una con características, enfoques y públicos objetivo diferentes. La elección de una de estas soluciones depende de factores como el tipo de información a analizar, el perfil de los usuarios, la facilidad de integración con las fuentes de datos, el tiempo de desarrollo disponible y el nivel de personalización requerido.

Entre las herramientas más difundidas se encuentra Metabase, una plataforma de código abierto orientada a la exploración de datos y a la construcción de dashboards de negocio de manera rápida e intuitiva. Su principal fortaleza radica en permitir la conexión con bases de datos relacionales y la creación de visualizaciones sin necesidad de desarrollar una interfaz analítica completa desde cero. Además, ofrece una experiencia accesible para usuarios no técnicos, lo que la vuelve especialmente útil en contextos donde se busca facilitar la consulta de indicadores de gestión.

Otra herramienta ampliamente conocida es Grafana, utilizada principalmente para visualización de métricas, monitoreo y observabilidad. Si bien permite construir dashboards potentes y altamente configurables, su uso suele estar más asociado al seguimiento de infraestructura, sistemas, servicios y series temporales, siendo especialmente popular en entornos DevOps y de monitoreo técnico. No obstante, también puede emplearse para ciertos escenarios analíticos, dependiendo de la naturaleza de los datos y de los objetivos perseguidos.

También se destacan soluciones como Power BI, desarrollada por Microsoft, y Tableau, ambas muy utilizadas en entornos corporativos por su capacidad de integración, variedad de visualizaciones y funciones avanzadas de análisis. Estas herramientas ofrecen un alto nivel de madurez y prestaciones, aunque en muchos casos implican licenciamiento, costos de adopción o una complejidad mayor en comparación con alternativas más livianas. Asimismo, puede mencionarse Looker Studio, orientada a la elaboración de reportes y dashboards conectados a múltiples fuentes de datos, con un enfoque accesible y ampliamente adoptado en ecosistemas vinculados a Google.

Finalmente, otra alternativa posible consiste en desarrollar dashboards de forma personalizada mediante bibliotecas de visualización en JavaScript, como Chart.js, Recharts o D3.js. Este enfoque brinda un mayor nivel de control sobre la interfaz y el comportamiento de los gráficos, pero también demanda más tiempo de desarrollo, mayor esfuerzo de mantenimiento y la implementación manual de numerosos aspectos que las herramientas especializadas ya resuelven, tales como filtros, consultas, configuración de visualizaciones y administración general del entorno analítico.

En consecuencia, el panorama actual de herramientas para dashboards es amplio y diverso. Algunas soluciones se orientan principalmente a la inteligencia de negocios y al análisis comercial, mientras que otras se enfocan más en monitoreo técnico o en desarrollos altamente personalizados. Por ello, la selección de una herramienta adecuada debe responder al contexto particular del proyecto, a las necesidades de información del negocio y a las restricciones de tiempo y complejidad de implementación.

3.8.5 Comparación de enfoques

Al momento de incorporar dashboards estadísticos a un sistema, existen distintos enfoques posibles para su implementación. La elección entre ellos depende de múltiples factores, tales como el tipo de información a visualizar, el perfil de los usuarios, el tiempo de desarrollo disponible, la necesidad de personalización y el esfuerzo de mantenimiento que se está dispuesto a asumir. En términos generales, pueden identificarse tres enfoques principales: el uso de herramientas de inteligencia de negocios ya consolidadas, la utilización de plataformas orientadas al monitoreo técnico y el desarrollo de dashboards a medida mediante código.

El primer enfoque consiste en adoptar una herramienta especializada en analítica de negocio, como Metabase, Power BI o Tableau. Estas plataformas ofrecen un

conjunto de funcionalidades ya resueltas para la construcción de dashboards, tales como conexión con bases de datos, creación de consultas, generación de visualizaciones, aplicación de filtros y organización de indicadores en paneles reutilizables. Su principal ventaja es que reducen de manera significativa el tiempo de desarrollo y facilitan la explotación de información por parte de usuarios no técnicos. Además, permiten centrar el esfuerzo del proyecto en la calidad de los datos, el diseño de indicadores y la utilidad de los tableros, en lugar de invertir una gran cantidad de tiempo en el desarrollo de una interfaz analítica desde cero.

Un segundo enfoque corresponde al uso de herramientas orientadas principalmente al monitoreo técnico y la observabilidad, entre las que se destaca Grafana. Este tipo de plataformas resulta especialmente adecuado para el seguimiento de métricas de infraestructura, servicios, rendimiento y series temporales, siendo ampliamente utilizado en contextos de administración de sistemas y entornos DevOps. Si bien estas herramientas también permiten construir dashboards visuales, su orientación principal no suele estar centrada en la analítica de negocio tradicional, sino en el monitoreo operativo y técnico. Por ello, su adecuación depende en gran medida del tipo de información que se pretenda analizar y del perfil de uso esperado dentro de la organización.

El tercer enfoque consiste en desarrollar dashboards personalizados mediante bibliotecas de visualización en lenguajes como JavaScript. Esta alternativa brinda un mayor control sobre el diseño, la experiencia de usuario y el comportamiento específico de cada visualización. Sin embargo, también implica una mayor complejidad de implementación, dado que requiere desarrollar manualmente no solo los gráficos, sino también la lógica de consultas, filtros, interacción, mantenimiento, seguridad y evolución del módulo analítico. En proyectos con alcance acotado o con restricciones de tiempo, este enfoque puede resultar menos conveniente, ya que incrementa considerablemente el esfuerzo de desarrollo y de mantenimiento futuro.

En consecuencia, cada enfoque presenta ventajas y limitaciones. Las herramientas de inteligencia de negocios priorizan rapidez de implementación, facilidad de uso y orientación al análisis gerencial; las plataformas de monitoreo técnico se destacan en escenarios de observabilidad y series temporales; y los desarrollos a medida ofrecen máxima flexibilidad, aunque a costa de mayor complejidad y esfuerzo. Por ello, la selección del enfoque más adecuado debe responder a las necesidades concretas del proyecto y a los objetivos perseguidos por la solución propuesta.

4 Seguridad y cumplimiento

4.1 Políticas de seguridad según nivel de exposición de un sistema

Se distinguen tres “anillos” de exposición:

Anillo	Contexto típico	Riesgos predominantes	Controles recomendados (OWASP, NIST SP-800-63)
Interno (LAN)	ERP on-premise, inventario sin datos personales	Movimiento lateral, abuso de privilegios	Autenticación basada en red, contraseñas robustas, registro de auditoría centralizado
Semi-público	Portales con usuarios autenticados pero sin pagos	Robo de sesión, XSS, CSRF	JWT de sesión corta, cabeceras de seguridad, rate-limit, 2FA opcional
Público crítico	E-commerce, banca, API abiertas	Intercepción, fraude, robo de tarjeta	JWT + refresh, cookies httpOnly, cifrado TLS 1.3, WAF/CDN, cumplimiento PCI-DSS

4.2 Gestión de sesión

- **Cookies de sesión tradicional** son adecuadas en entornos internos—se almacenan en servidor y expiran al cerrar el navegador.

- **JWT** reemplaza el estado del servidor en sistemas semi-públicos: cada solicitud transporta sus credenciales firmadas; los tokens pueden vivir pocos minutos para reducir la ventana de explotación (OWASP[14] Cheat-Sheet, 2024).
- **JWT + refresh** se reserva a plataformas públicas con alto tráfico y necesidad de UX fluida; el *refresh-token* (rotado) renueva la sesión sin repetir login y permite invalidación global tras cambios de clave (Auth0[15] white-paper, 2023).

4.3 Almacenamiento de contraseñas

El consenso actual —NIST SP-800-63-B, 5.1.1.2— es usar bcrypt, scrypt o Argon2 con *salt* individual y ≥ 10 iteraciones para contrarrestar ataques de fuerza bruta en GPU. Hash simples (SHA-256) o rainbow tables se consideran obsoletos.

4.4 Cabeceras y políticas de navegador

Una aplicación expuesta a Internet debe fijar al menos:

- **Strict-Transport-Security** (HSTS) con max-age ≥ 6 meses para forzar HTTPS.
- **Content-Security-Policy** (CSP “default-src 'self'”) y prohibición de unsafe-inline para bloquear *XSS*.
- **X-Frame-Options: DENY** para impedir *clickjacking*.

4.5 Protección de API y tasas de petición

Para sistemas semi-públicos basta limitar < 300 peticiones por 5 min por IP; los sistemas de pago suelen endurecer a < 30 peticiones por min en endpoints críticos (OWASP API Security Top 10, 2023).

4.5.1 Manejo de datos sensibles

- **Tokenización / delegación**. PCI-DSS recomienda desviar el flujo completo de tarjeta a un PSP (Payment Service Provider). Plataformas como MercadoPago o Stripe devuelven un payment_token y descargan al comercio del cumplimiento PCI.
- **Encriptación en reposo**. Las copias de seguridad con datos personales (nombre, domicilio) deben cifrarse con AES-256 y rotar la clave al menos una vez al año (ISO 27002).

Propuesta de Solución

1 Alcance Funcional

La solución propuesta contempla la digitalización integral de los procesos operativos del negocio, abarcando la gestión de productos, stock, ventas, promociones, deliveries, clientes, empleados y administración general del sistema. A su vez, la versión final del proyecto incorpora una capa analítica complementaria orientada a la visualización de indicadores clave de desempeño (KPIs) y al análisis histórico de la información del negocio.

Esta ampliación permite no solo asistir la operatoria diaria mediante funcionalidades transaccionales, sino también brindar soporte a la toma de decisiones a través de dashboards estadísticos construidos sobre información consolidada. Para ello, se integran procesos de extracción, transformación y carga de datos (ETL), una estructura analítica orientada a la consulta y una herramienta de visualización que facilita el seguimiento de métricas relevantes desde la aplicación de escritorio.

De este modo, el alcance funcional del sistema no se limita únicamente a la ejecución de operaciones del negocio, sino que también incorpora capacidades de análisis, monitoreo y consulta gerencial, complementando la gestión operativa con una perspectiva estadística e histórica.

1.1 Requerimientos

1.1.1 Requerimientos Funcionales

ID	Prioridad	Descripción
RF-0001	Alta	El empleado inicia sesión con credenciales válidas.
RF-0002	Alta	El empleado crea, modifica y elimina productos así como sus existencias.
RF-0003	Alta	El empleado registra una venta y emite comprobante.
RF-0004	Media	El empleado configura promociones y define a quién se notifican.
RF-0005	Alta	El empleado asigna entregas y actualiza su estado en tiempo real.
RF-0006	Media	El empleado consulta el historial de acciones para auditoría.
RF-0007	Alta	El propietario gestiona altas / bajas de empleados y permisos.

RF-0008	Media	El propietario visualiza estadísticas operativas e indicadores históricos de ventas, productos, stock y desempeño mediante un panel analítico.
RF-0009	Alta	El cliente se registra, inicia sesión y administra datos personales y de mascotas.
RF-0010	Alta	El cliente consulta catálogo, compra on-line y solicita delivery.
RF-0011	Media	El cliente revisa historial de compras y estado de pedidos.
RF-0012	Media	El cliente acumula puntos y los canjea por promociones.
RF-0013	Alta	El delivery inicia sesión, ve su cronograma y actualiza cada entrega.
RF-0014	Media	El sistema envía recordatorios de reposición predictiva al cliente.
RF-0015	Alta	El sistema procesa pagos a través de MercadoPago con flujo *one-tap*.

RF-0016	Alta	El sistema ejecuta procesos ETL periódicos para extraer, transformar y consolidar datos operativos en una estructura analítica orientada a la consulta de indicadores y KPIs.
RF-0017	Media	El propietario consulta KPIs y comparativas históricas por período, categoría, producto y desempeño desde la aplicación de escritorio.
RF-0018	Alta	El sistema permite la consulta del panel analítico aun cuando el backend principal no se encuentre disponible, mediante una infraestructura desacoplada de la operatoria transaccional.

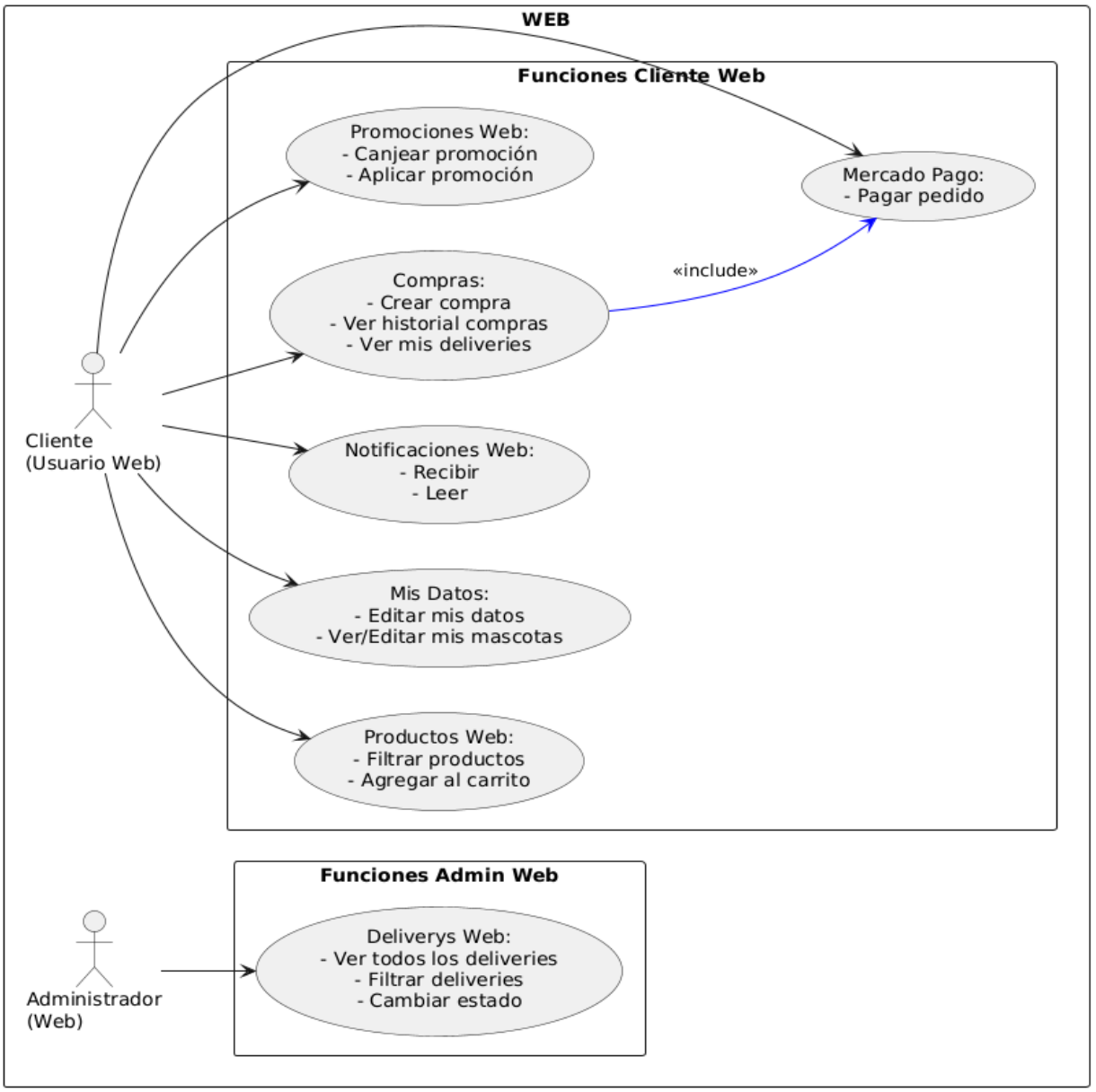
1.1.2 Requerimientos No Funcionales

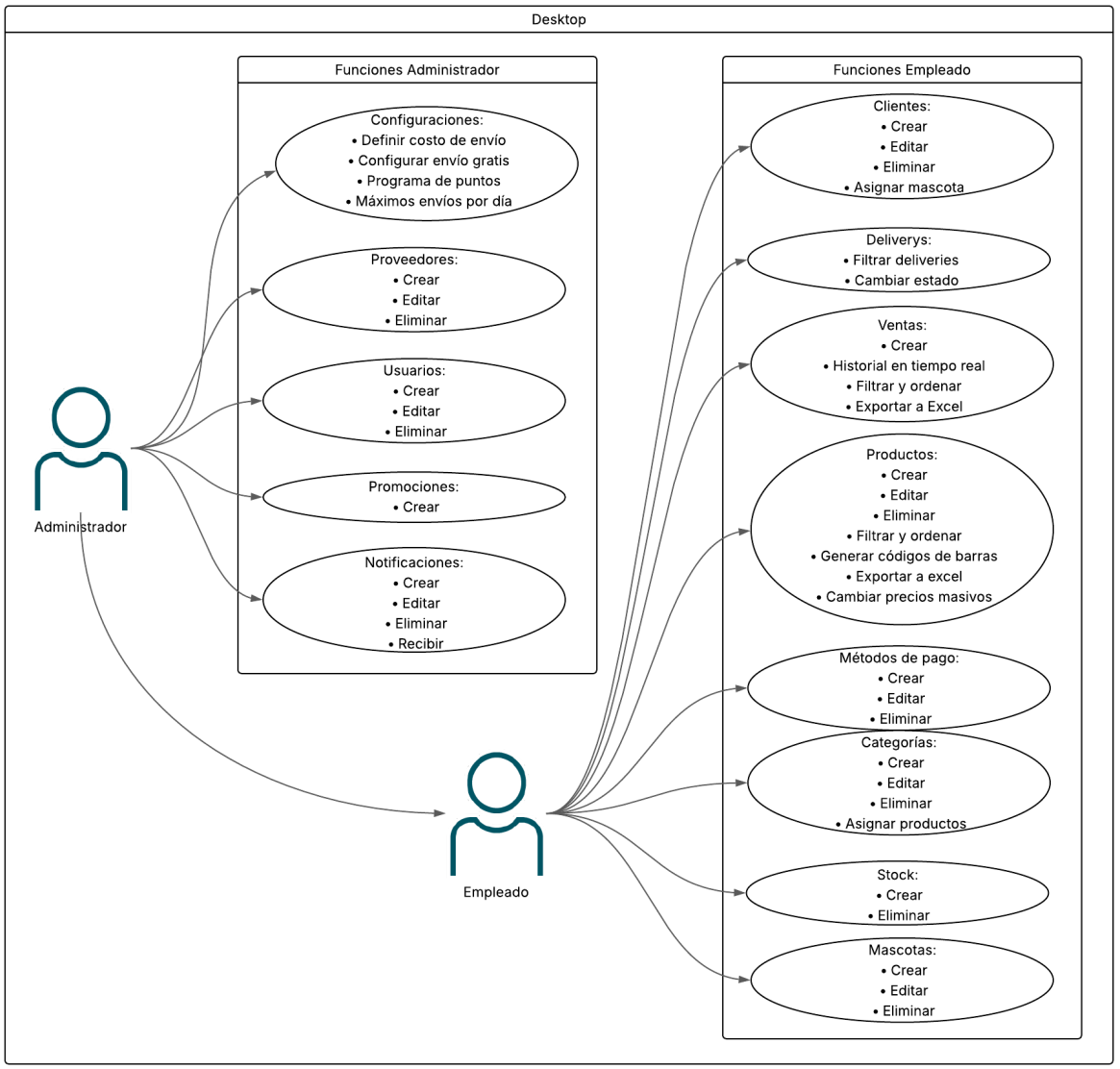
Categoría	ID	Prioridad	Especificación
Rendimiento	RNF-PERF-01	Alta	95 % de las peticiones deben responder en ≤ 300 ms con 50 usuarios concurrentes.
Disponibilidad	RNF-DISP-01	Alta	Uptime ≥ 99 %.

Seguridad	RNF-SEC-01	Alta	Autenticación JWT (access 2,5 min; refresh 7 días) y calificación A+ en *securityheaders.com*.
Seguridad	RNF-SEC-02	Alta	Contraseñas en bcrypt (salt único) y datos de pago delegados a MercadoPago.
Mantenibilidad	RNF-MAN-01	Media	Código con ESLint + Prettier; cobertura mínima de pruebas manuales documentada e incorporación de pruebas automatizadas end-to-end para endpoints del backend.
Integridad de datos	RNF-DAT-01	Alta	La información consolidada en la capa analítica debe mantener consistencia con los datos de la base operativa luego de cada ejecución del proceso ETL.
Actualización analítica	RNF-ANA-01	Media	Los indicadores y tableros analíticos deben actualizarse con una periodicidad programada acorde a las necesidades del negocio.
Disponibilidad	RNF-DISP-02	Alta	El módulo analítico debe poder consultarse de manera independiente del

			backend principal, utilizando una arquitectura desacoplada y una fuente de datos analítica separada.
Seguridad	RNF-SEC-03	Media	El acceso al panel analítico debe gestionarse mediante un mecanismo de autenticación y autorización independiente del backend principal.

1.2 Diagramas de casos de uso





- *Casos de uso en detalle*:

Caso de Uso	Crear promoción
Actor	ADMINISTRADOR
Descripción	Permite al administrador crear y configurar las promociones disponibles para canjear que verán los clientes

Flujo Alternativo	1. Dirigirse a la pestaña de “Administrador” en la navbar 2. Luego al apartado de “Promociones” 3. Presionar el botón “+” de abajo a la derecha 4. Configurar título y descripción de la promoción 5. Configurar fecha de inicio y fecha de fin (disponibilidad de la promo) 6. Seleccionar el costo de puntos, los días de validez indicar si incluye delivery gratuito 7. Seleccionar qué tipo de promoción (6 tipos disponibles) y configurarlos en la sección de la derecha que se abre automáticamente
Precondición	1. Tener conexión a internet 2. Estar logueado como administrador.

1.3 Historias de usuario clave

Historia	Descripción	Criterios de aceptación (Gherkin)
HU-01 Venta en mostrador	Como empleado, quiero buscar un producto por código de barras y cerrar la venta para cobrar rápidamente al cliente.	Dado que estoy en la pantalla Ventas, Cuando escaneo un código válido, Entonces el ítem aparece en el carrito y Cuando presiono «Cobrar», Entonces se registra la venta y se imprime el comprobante.

HU-02 Reposición predictiva	Como cliente, quiero recibir un aviso cuando mi mascota esté a punto de quedarse sin alimento, para comprar a tiempo.	Dado que tengo historial de compras, cuando se aproxime la fecha estimada de reposición, Entonces el sistema me envía una notificación push con un acceso directo al producto.
HU-03 Pago one-tap	Como cliente móvil, quiero pagar con MercadoPago sin volver a ingresar mis datos, para finalizar la compra en segundos.	Dado que la app de MercadoPago está instalada, Cuando elijo MercadoPago y confirmo, Entonces se abre la app, autorizo con biometría y Entonces vuelvo al sitio con estado «Pago aprobado».
HU-04 Canje de promoción	Como cliente, quiero canjear mis puntos por un descuento, para aprovechar mis compras anteriores.	Dado que tengo 300 puntos, Cuando agrego una promoción de 200 puntos al carrito, Entonces el precio total refleja el descuento y los puntos se descuentan de mi cuenta.
HU-05 Notificación masiva	Como administrador, quiero enviar una campaña a todos los clientes, para anunciar una nueva línea premium.	Dado que selecciono segmento «Clientes», Cuando redacto el mensaje y programo la fecha, Entonces el sistema genera una notificación.

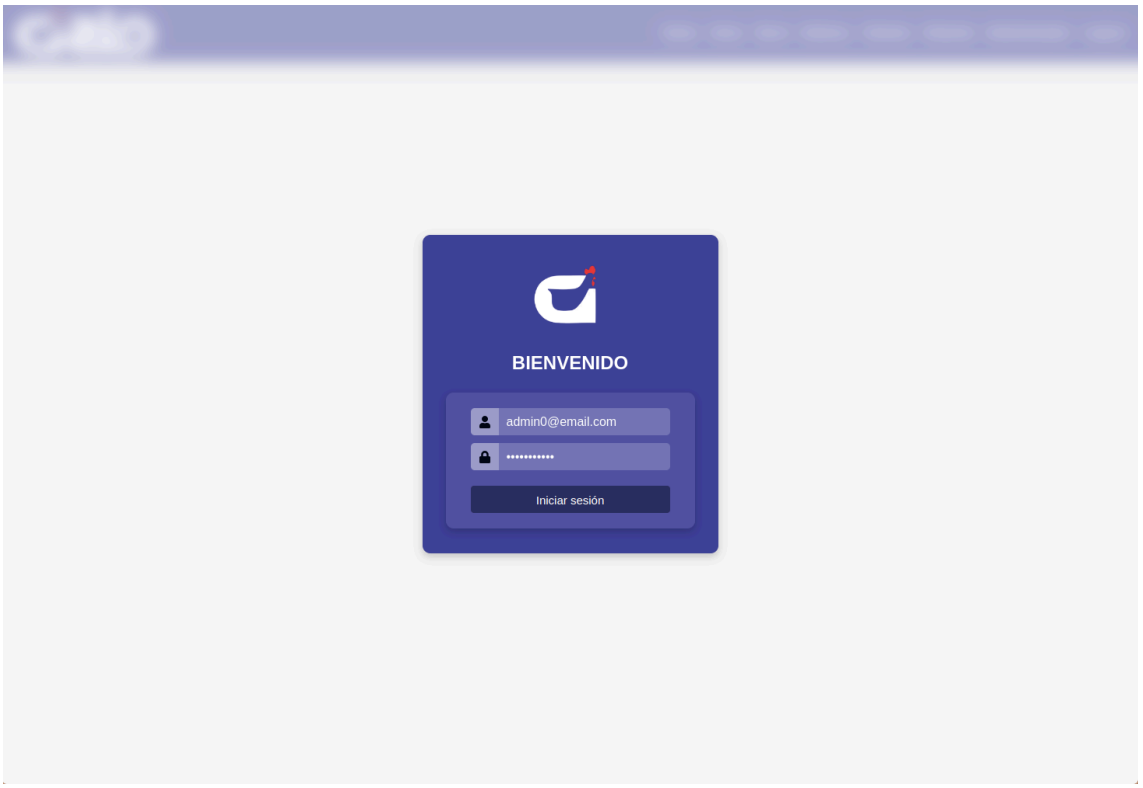
HU-06 Entrega en ruta	Como empleado, quiero marcar un pedido como «entregado» desde mi móvil, para que el cliente reciba la confirmación y el sistema registre la hora.	Dado que veo el listado de entregas asignadas, Cuando selecciono «Pedido #123» y pulso «Entregado», Entonces el estado cambia a «Completo» y el cliente recibe la notificación correspondiente.

2 Diseño

2.1 Pantallas

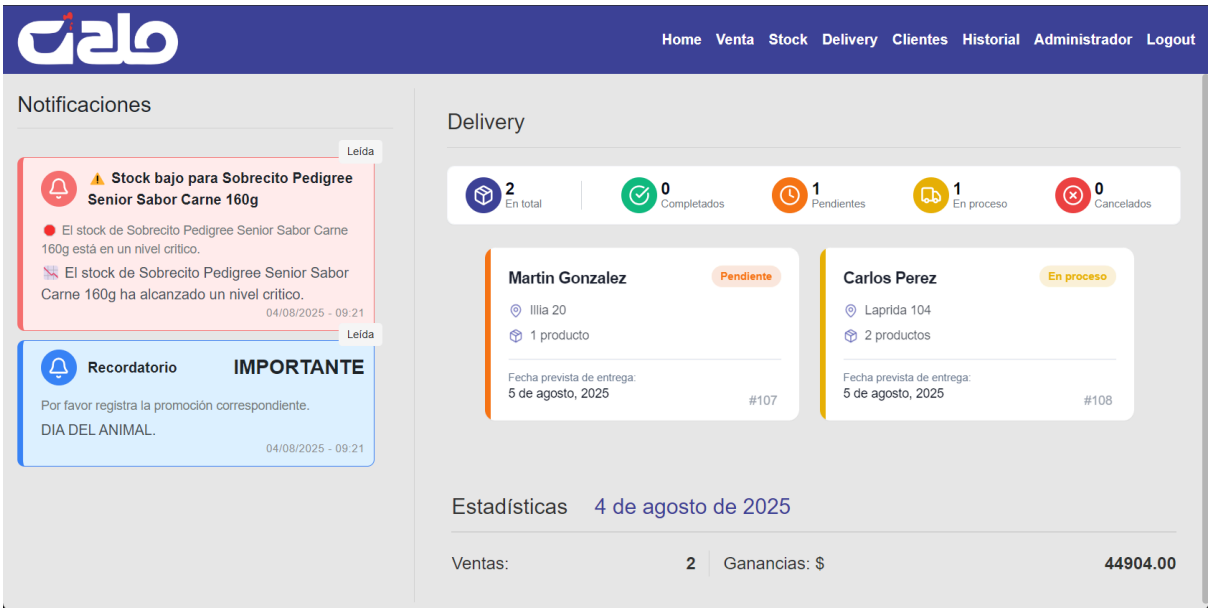
2.1.1 Desktop app

Pantalla de login:



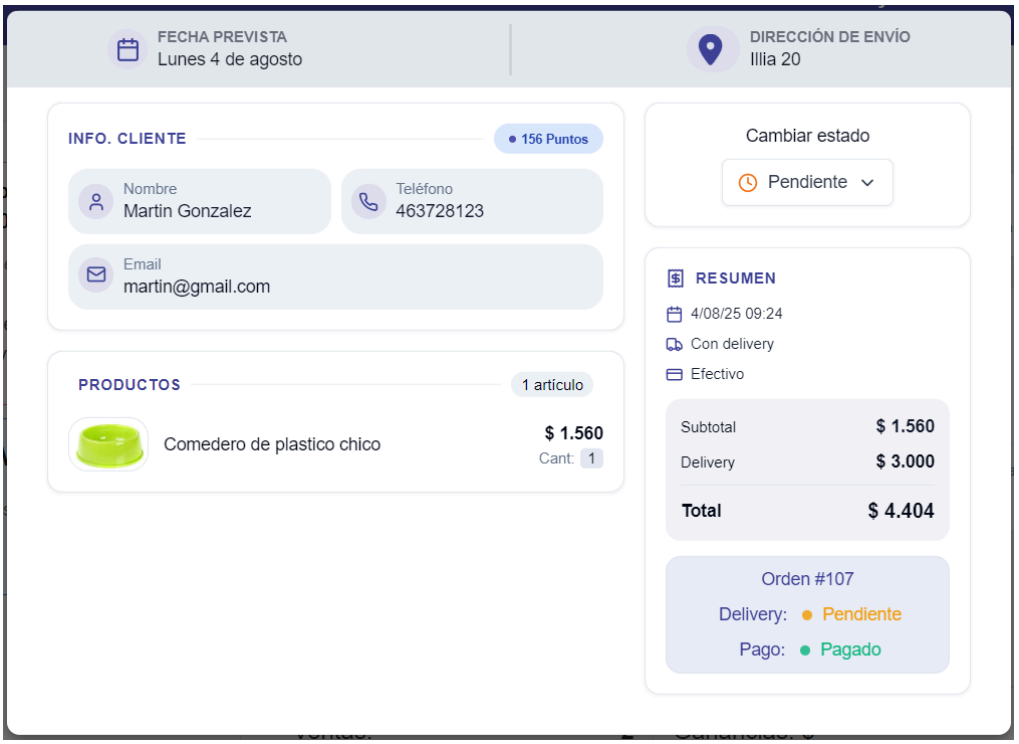
La pantalla de inicio de sesión fue diseñada como punto de acceso único al sistema, permitiendo validar la identidad del usuario antes de habilitar las funcionalidades disponibles. Su incorporación responde a la necesidad de controlar el acceso a la información y restringir las operaciones según el rol asignado, diferenciando permisos para empleados y administradores.

Pantalla de inicio:

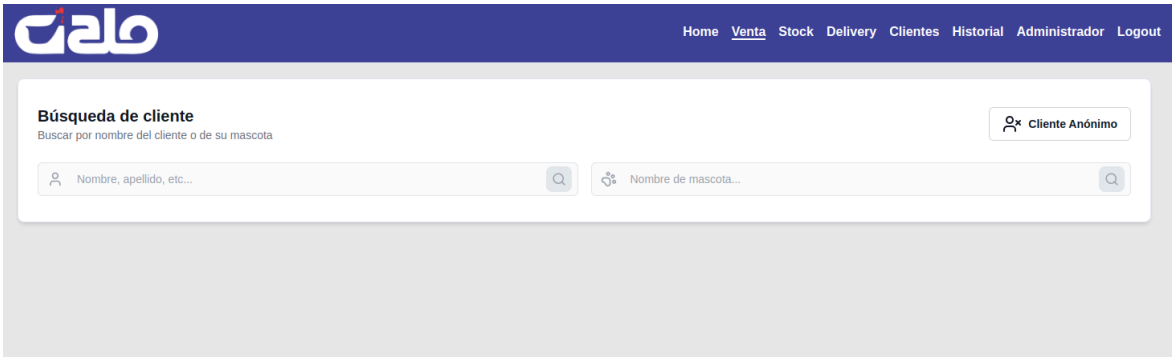


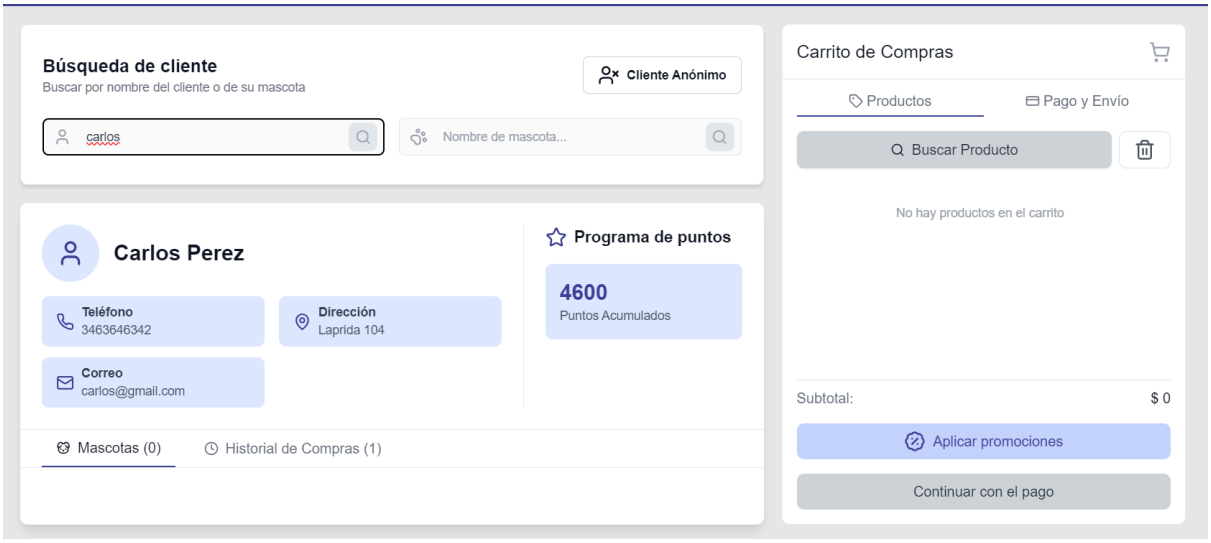
La pantalla de inicio fue concebida como un panel de resumen operativo para los usuarios internos. Integra información relevante sobre notificaciones, entregas y métricas generales, permitiendo acceder rápidamente al estado diario del negocio. Esta organización busca reducir la dispersión de información y facilitar la supervisión de eventos importantes, como cambios de stock, entregas pendientes o avisos generados por el sistema.

Pantalla de inicio > modal de delivery:



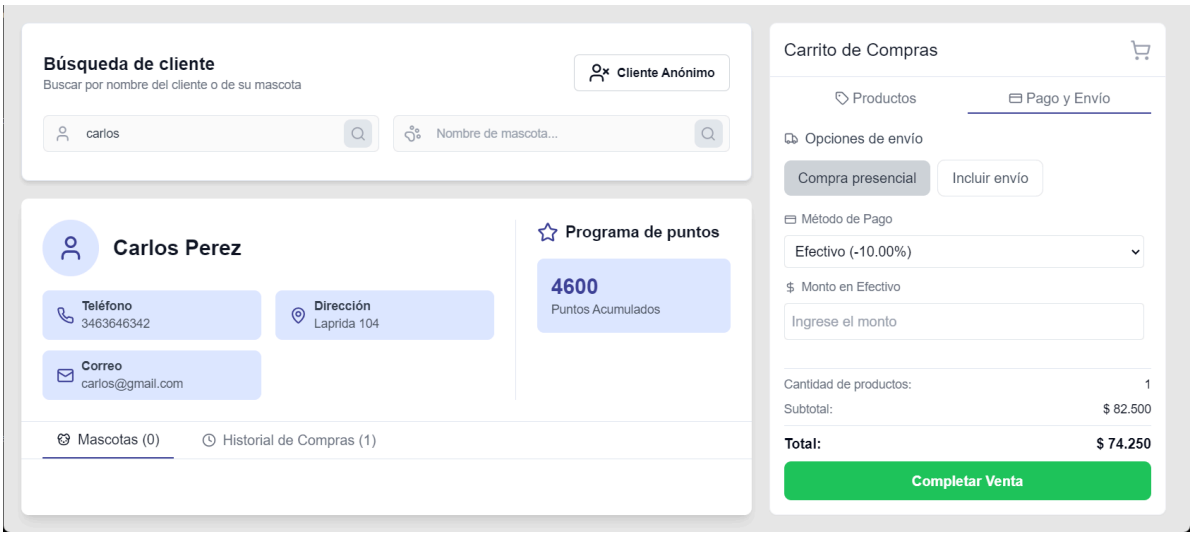
Pantalla de ventas:



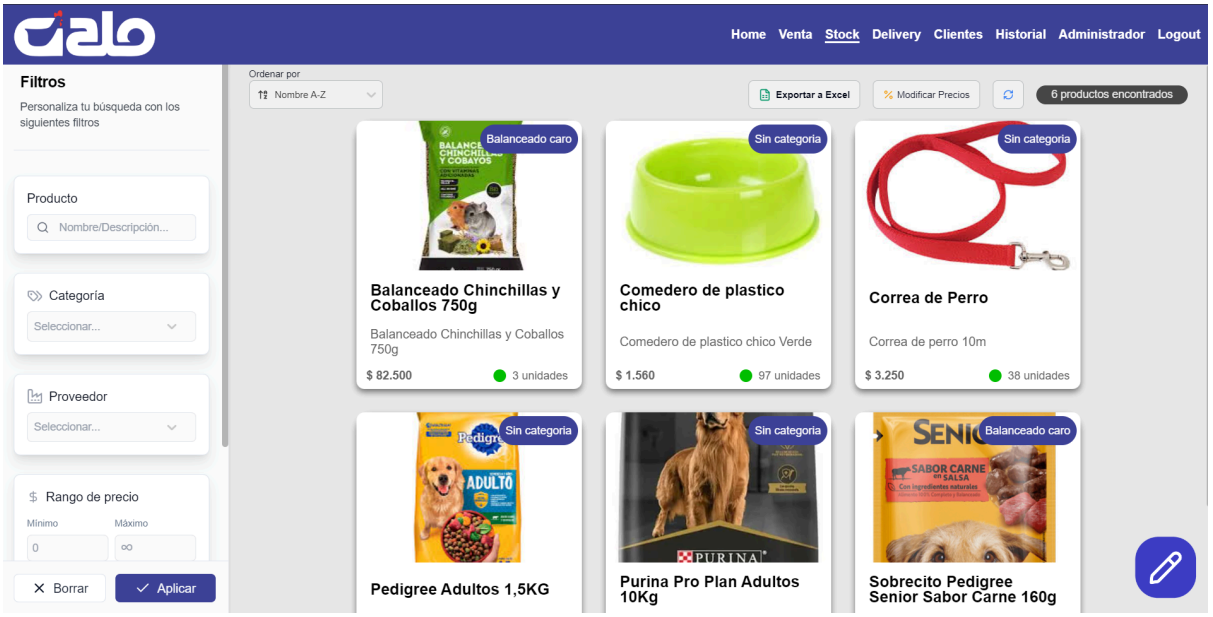


La pantalla de ventas centraliza el proceso de registración de operaciones comerciales, integrando la selección del cliente, la búsqueda de productos y la composición del carrito en una misma interfaz. El diseño contempla tanto la carga manual de productos como la lectura mediante código de barras, con el objetivo de agilizar la atención en mostrador, reducir errores de carga y mantener actualizada la información asociada a ventas, stock y clientes.

Pantalla de ventas > modal búsqueda de productos:



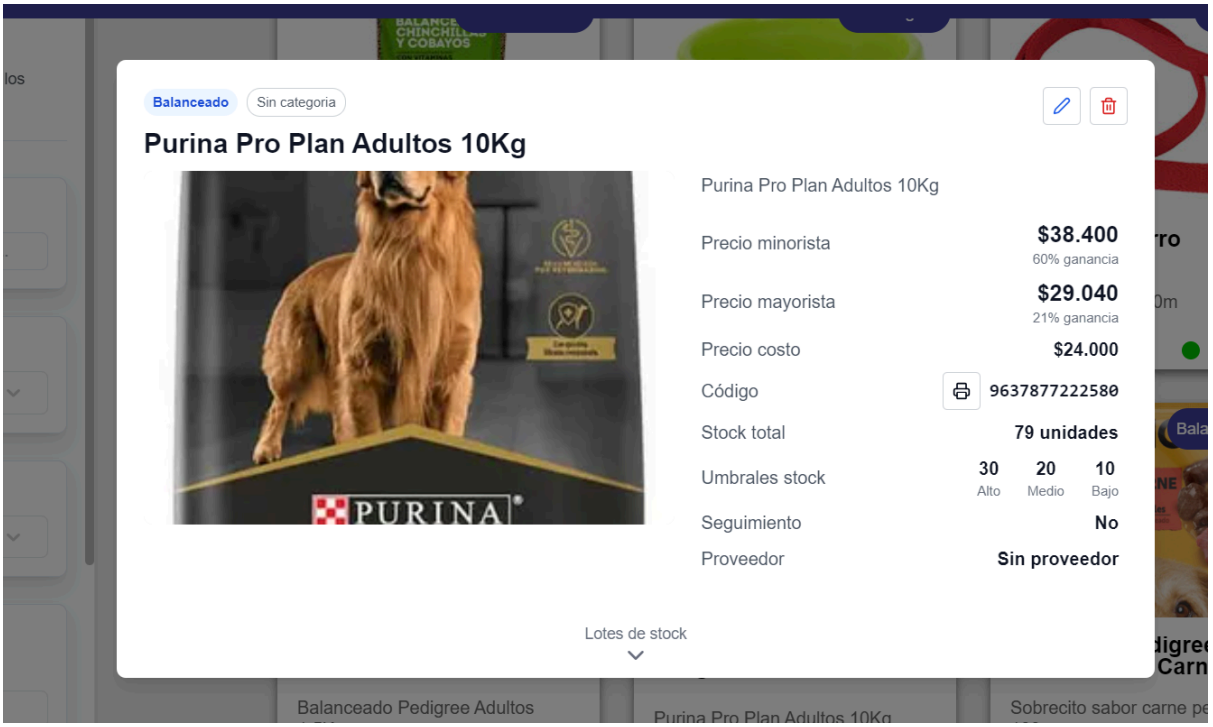
Pantalla de stock:



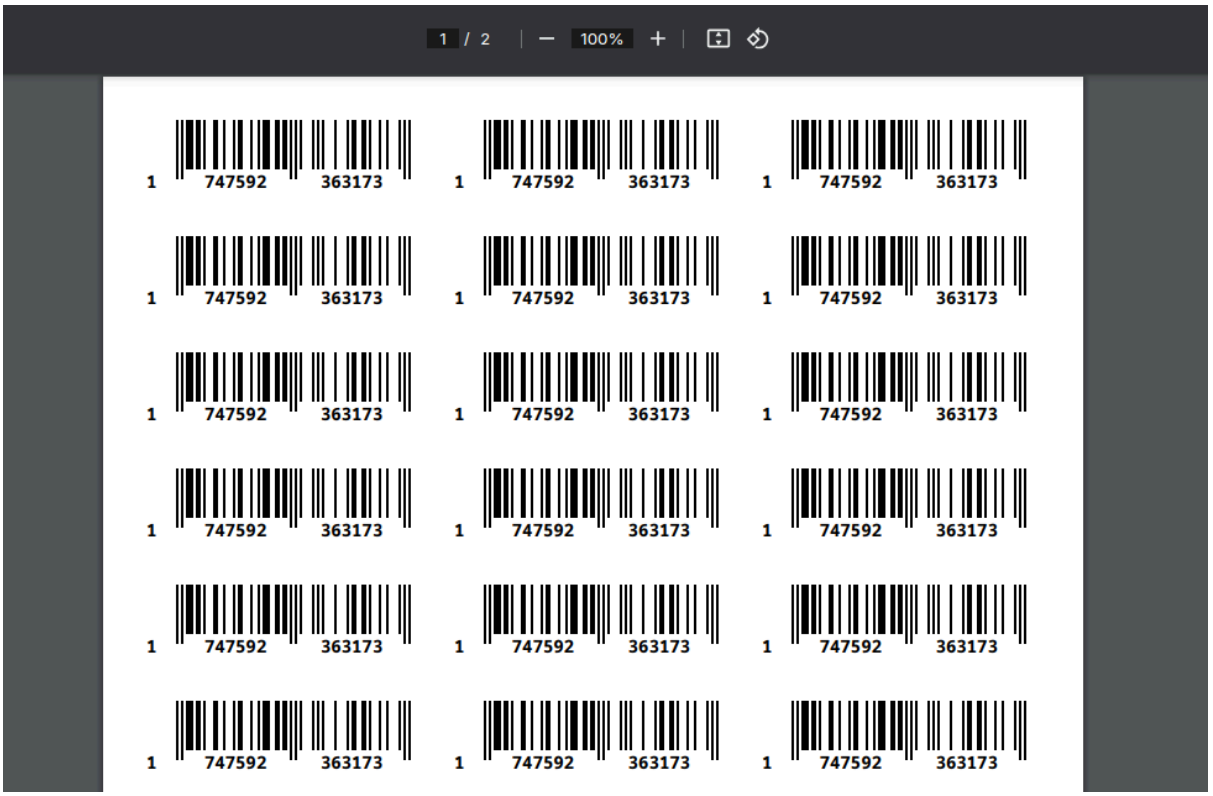
La pantalla de stock fue diseñada para concentrar la administración del inventario en una única vista operativa. Desde este módulo se facilita la consulta de existencias, la identificación de productos, la gestión de altas y modificaciones, y el acceso a información detallada de cada artículo. La incorporación de filtros responde a la

necesidad de localizar rápidamente productos dentro de un catálogo amplio, reduciendo tiempos de búsqueda y mejorando el control interno.

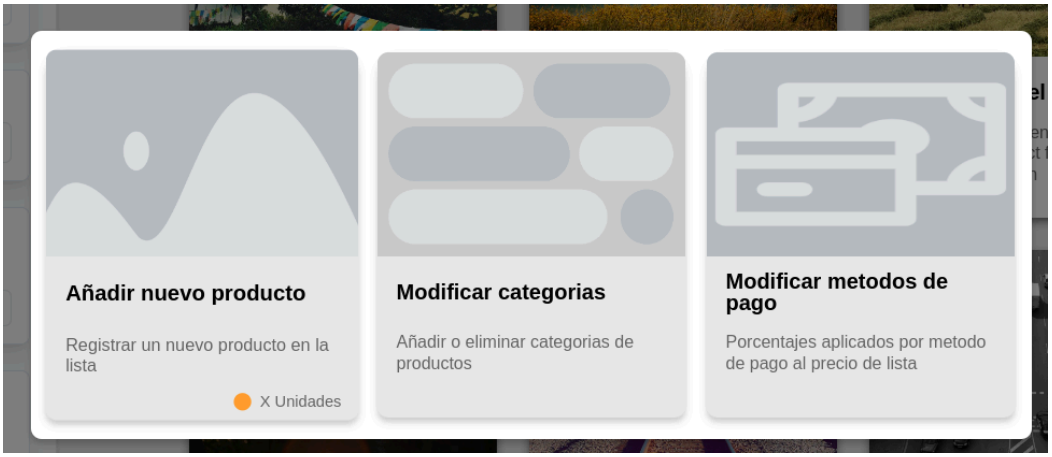
Pantalla de stock > modal de producto:



Pantalla de stock > generador de códigos de barras:

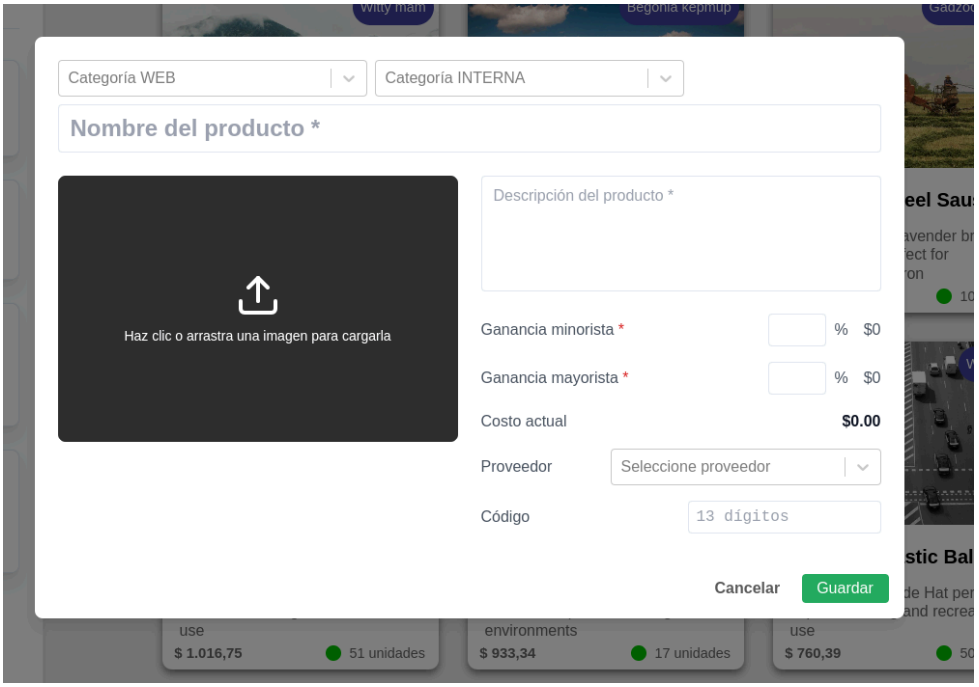


Pantalla de stock > modal de acciones:

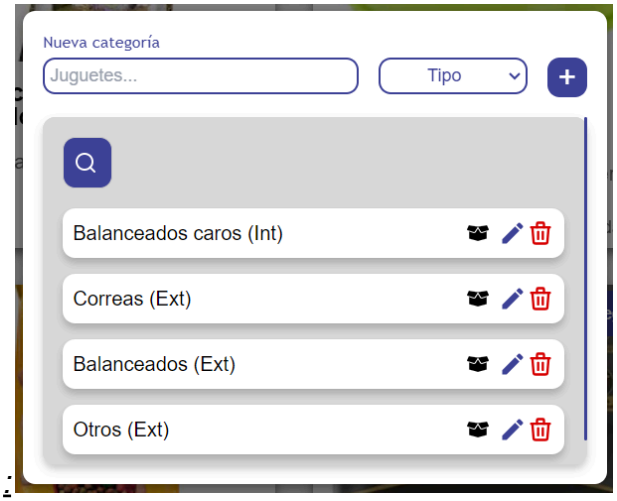


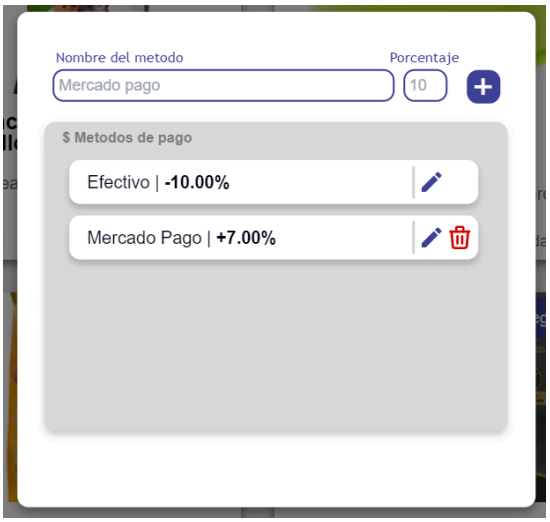
La pantalla de creación de productos fue diseñada para centralizar la carga de información necesaria para la gestión del stock, incluyendo datos identificatorios, categoría, precio, cantidades disponibles e imagen asociada. La incorporación de una imagen del producto responde a la necesidad de facilitar su reconocimiento visual tanto en la operatoria interna como en los canales digitales de venta, reduciendo errores de identificación y mejorando la experiencia de consulta.

Pantalla de stock > modal de producto (creación):

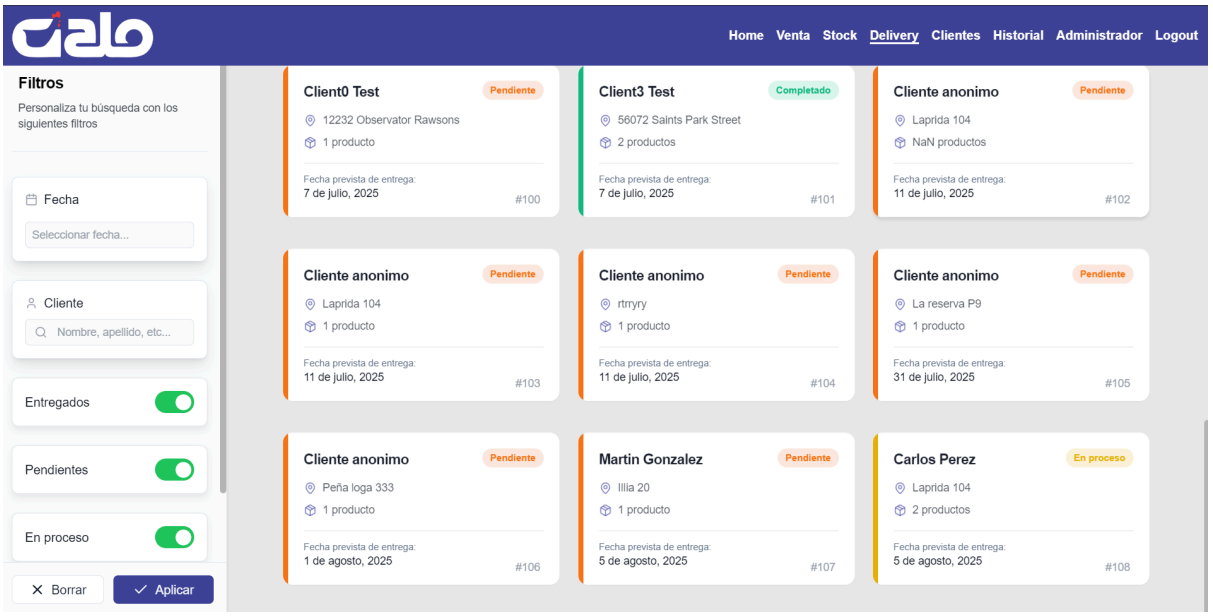


Pantalla de stock > modal de categorías y modal de métodos de pago:

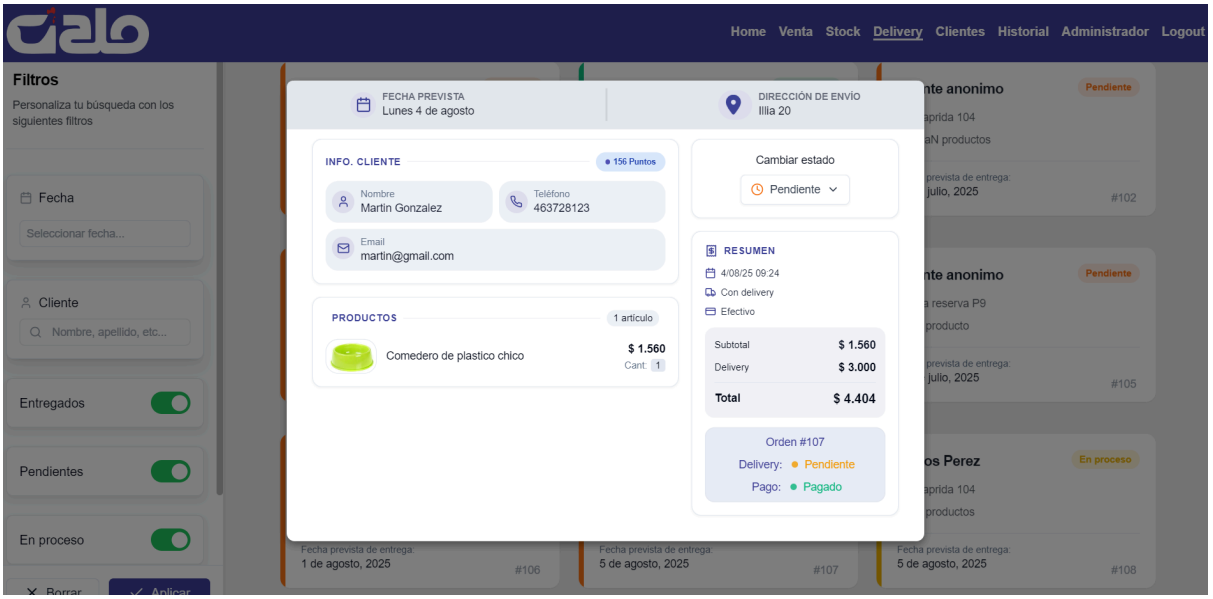




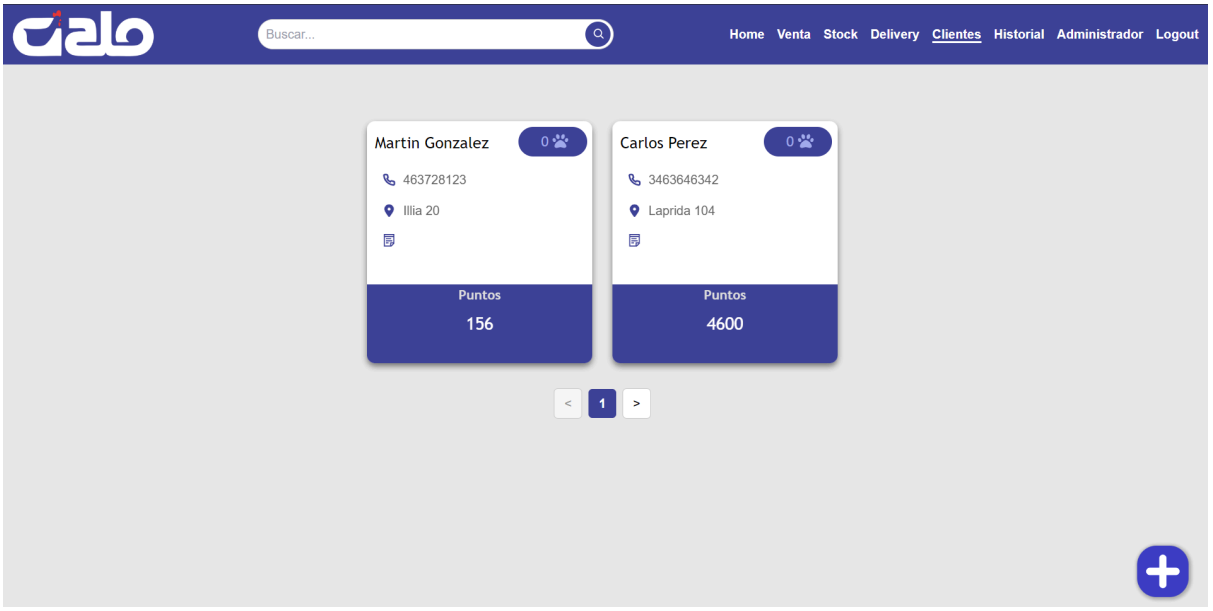
Pantalla deliveries:



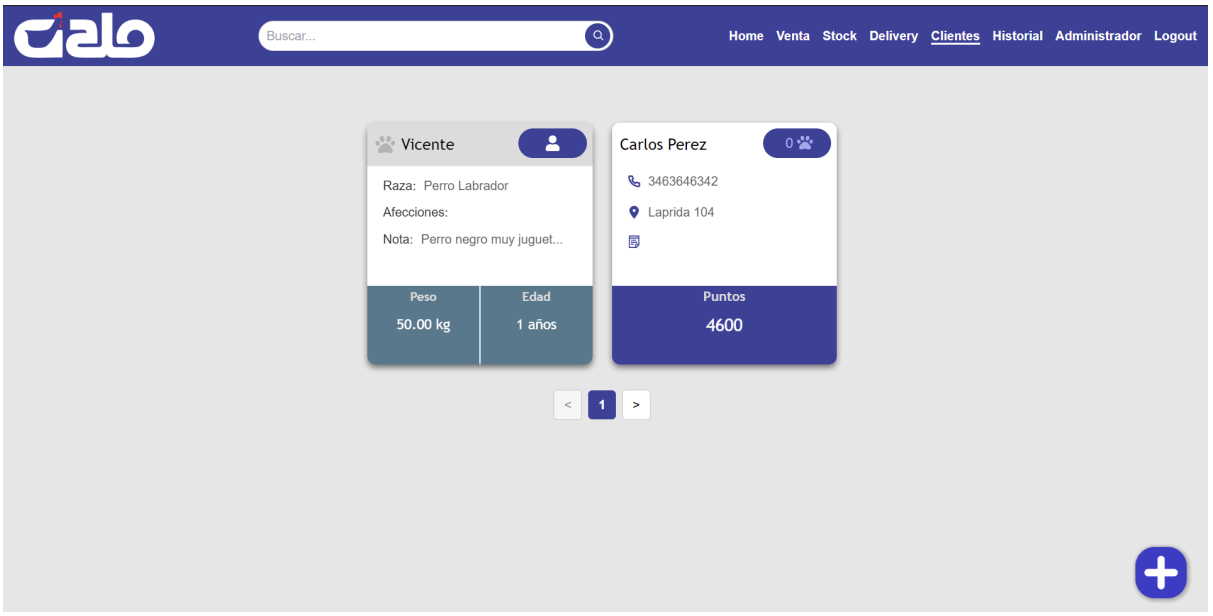
Pantalla deliveries > delivery modal:



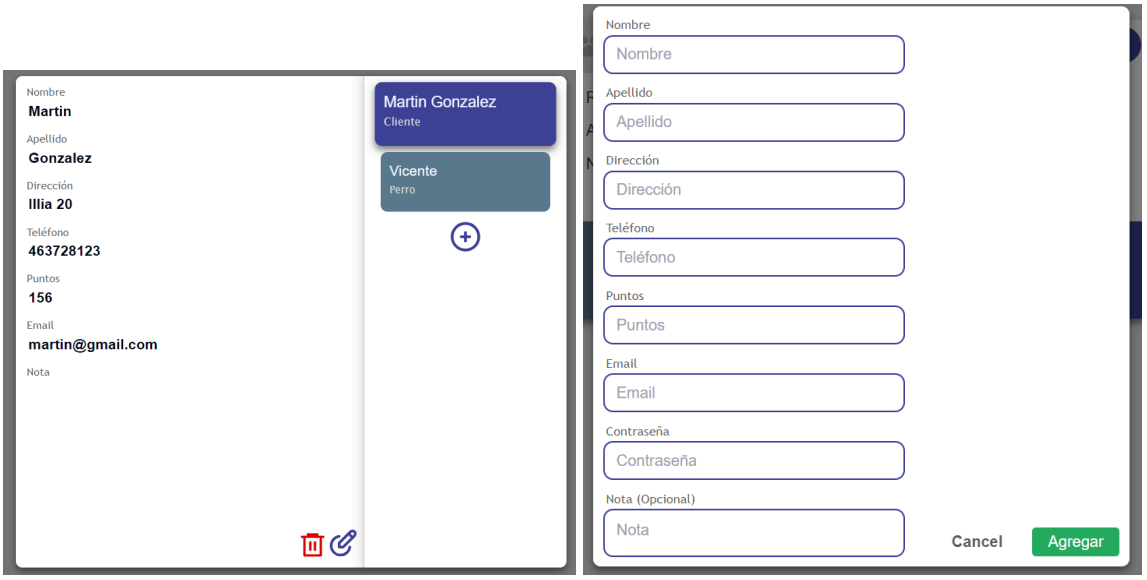
Pantalla de clientes:



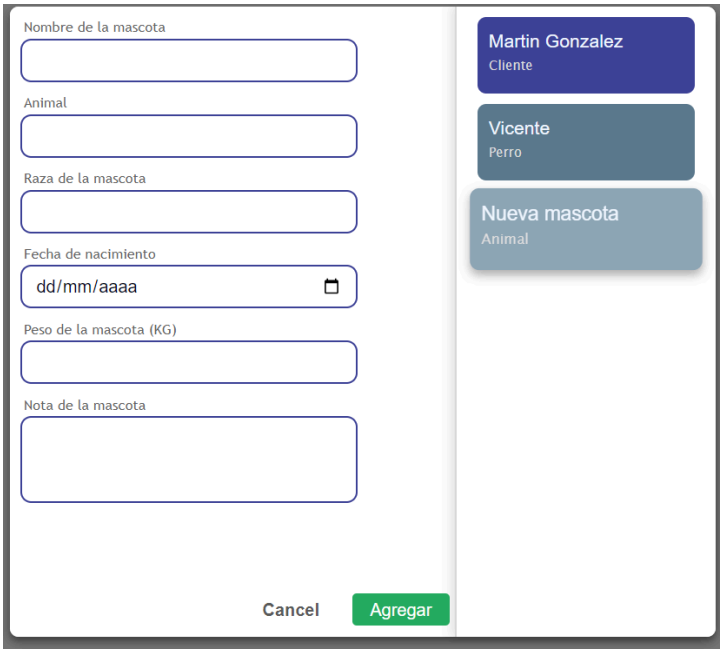
La pantalla de clientes permite centralizar la información de las personas registradas en el sistema y vincularla con los datos de sus mascotas. Esta decisión de diseño responde a la necesidad de contar con una base de información útil para la gestión comercial, el historial de compras, la personalización de promociones y la generación de notificaciones asociadas a preferencias o eventos relevantes.



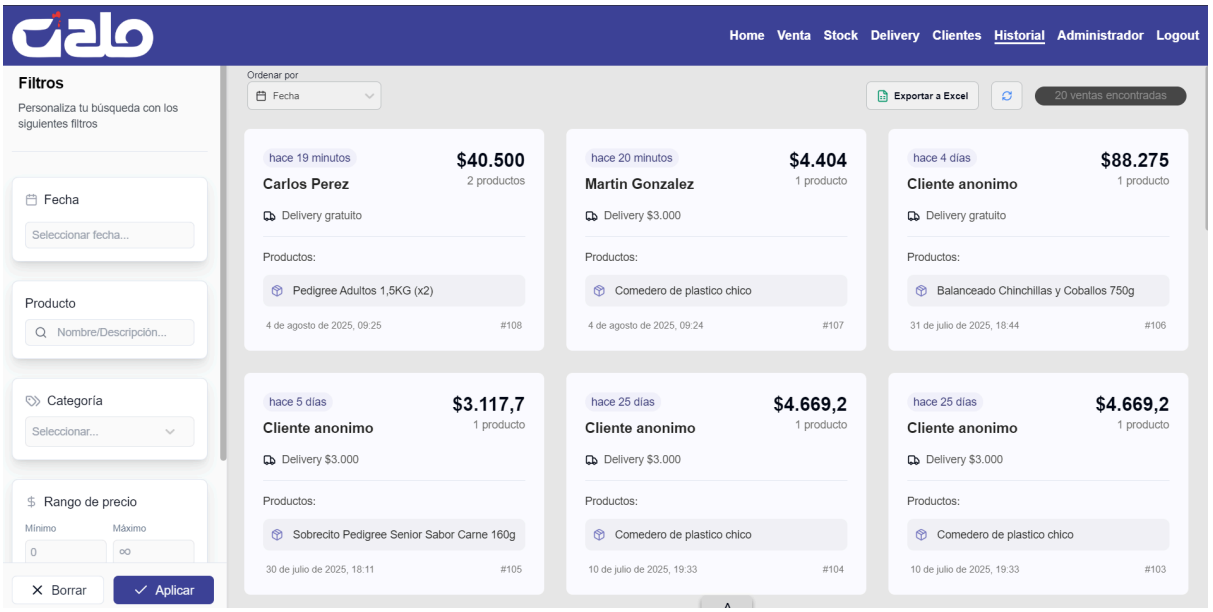
Pantalla de clientes > modal de clientes y modal de creación de clientes:



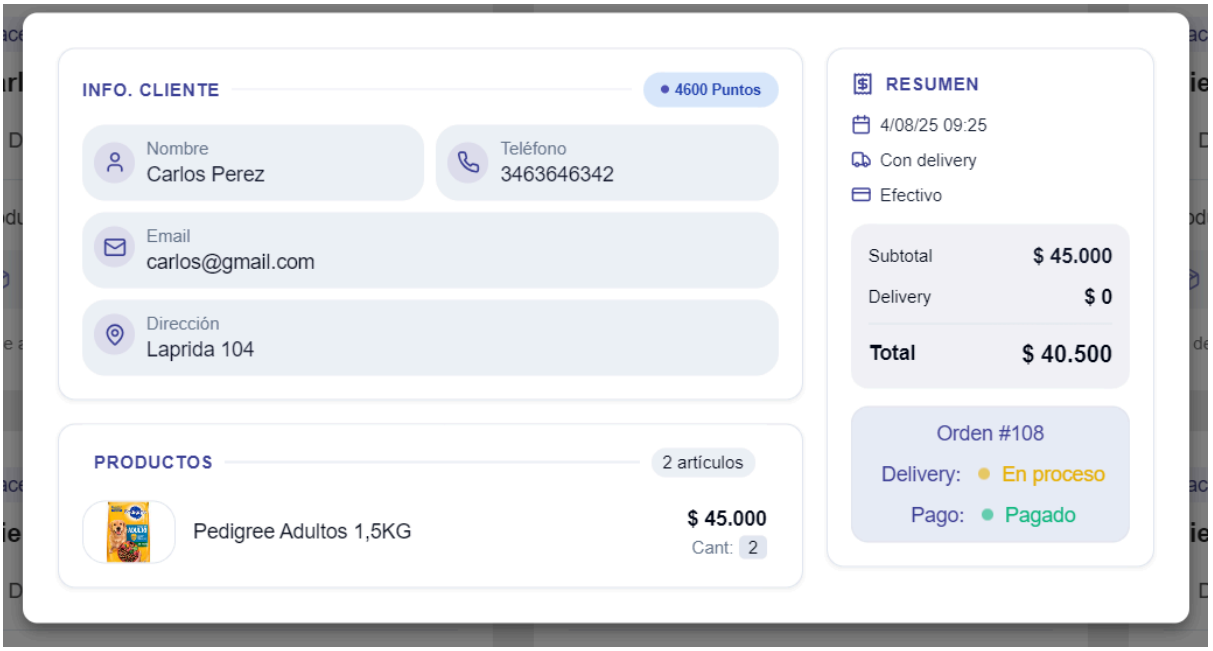
Pantalla de clientes > modal de creación de mascotas:



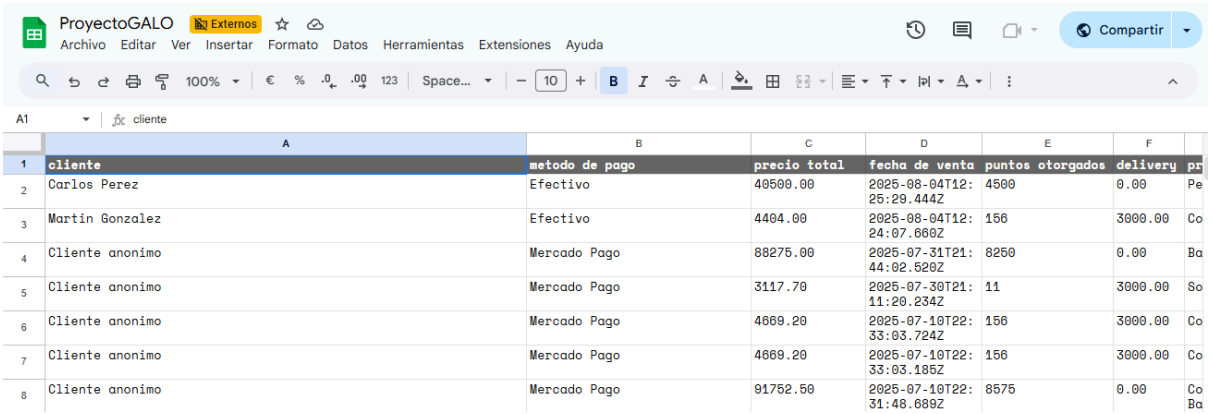
Pantalla historial:



Pantalla historial > modal de venta:



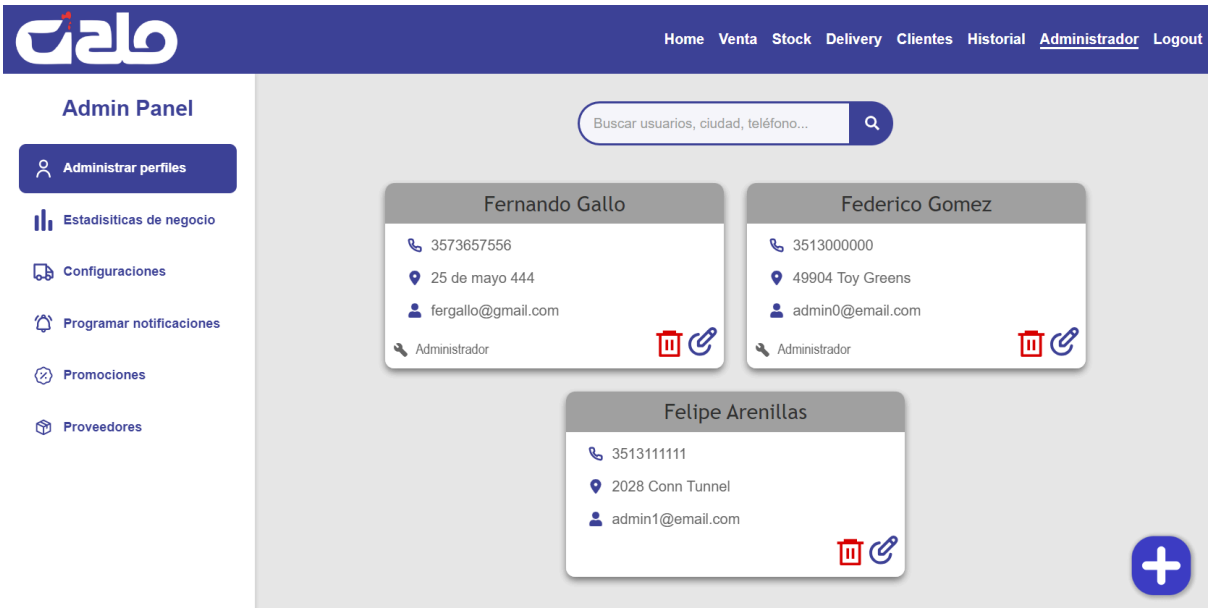
Export de historial de ventas a excel (Google Sheets):



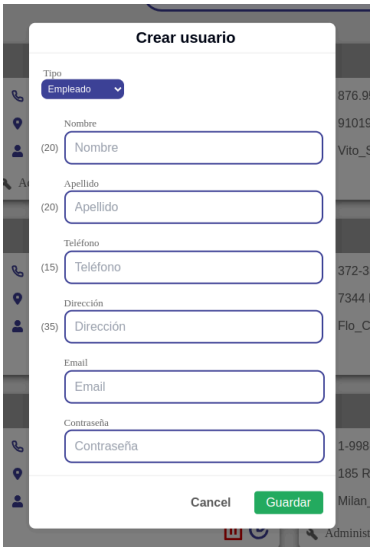
	A	B	C	D	E	F	
1	cliente	metodo de pago	precio total	fecha de venta	puntos otorgados	delivery	pr
2	Carlos Perez	Efectivo	40500.00	2025-08-04T12:25:29.444Z	4500	0.00	Pe
3	Martin Gonzalez	Efectivo	4404.00	2025-08-04T12:24:07.660Z	156	3000.00	Co
4	Cliente anonimo	Mercado Pago	88275.00	2025-07-31T21:44:02.520Z	8250	0.00	Ba
5	Cliente anonimo	Mercado Pago	3117.70	2025-07-30T21:11:20.234Z	11	3000.00	So
6	Cliente anonimo	Mercado Pago	4669.20	2025-07-10T22:33:03.724Z	156	3000.00	Co
7	Cliente anonimo	Mercado Pago	4669.20	2025-07-10T22:33:03.185Z	156	3000.00	Co
8	Cliente anonimo	Mercado Pago	91752.50	2025-07-10T22:31:48.689Z	8575	0.00	Co Ba

La funcionalidad de exportación del historial de ventas a Google Sheets fue incorporada para facilitar la consulta externa y colaborativa de reportes operativos. Al generar los archivos en la cuenta de Google Drive configurada por el cliente, el sistema permite conservar respaldos en la nube, compartir información con usuarios autorizados y revisar los datos desde distintos dispositivos sin depender de herramientas locales específicas. Esta decisión reduce la necesidad de implementar un módulo propio de reportes tabulares y aprovecha una plataforma familiar para el usuario final.

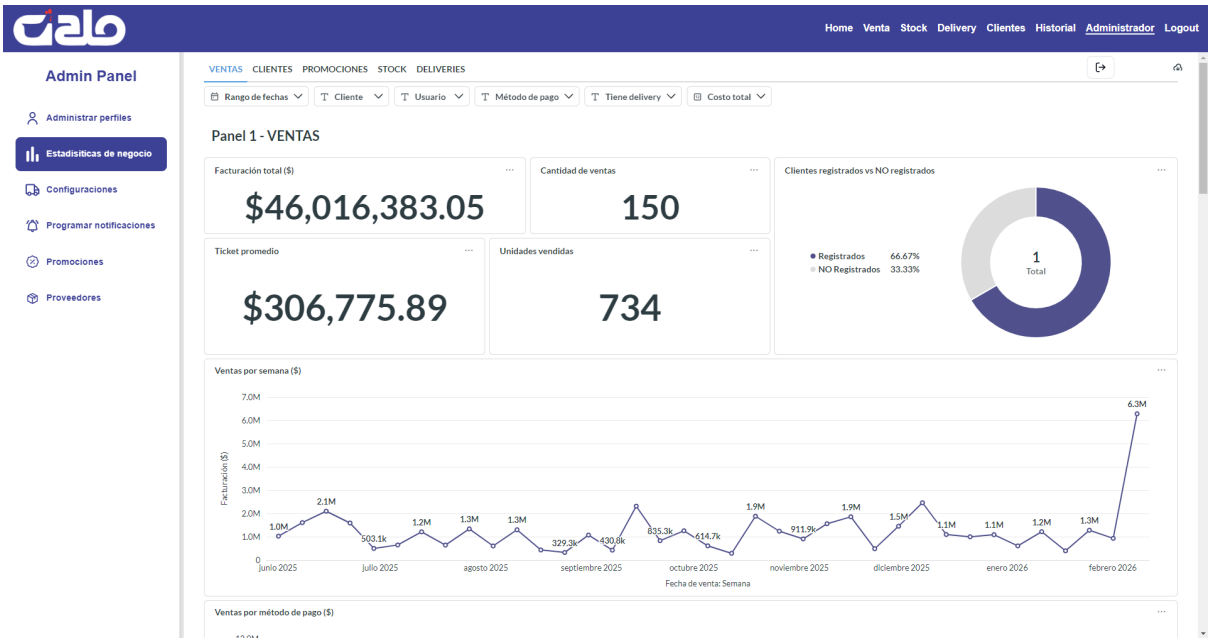
Pantalla de administrador > sección administrador de perfiles:



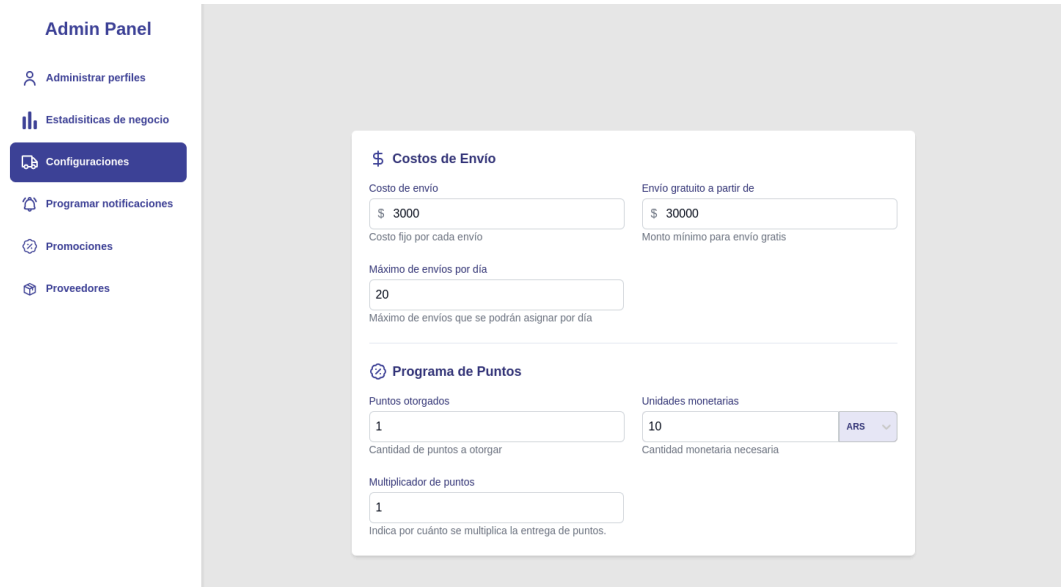
Pantalla de administrador > modal de empleados (creación):



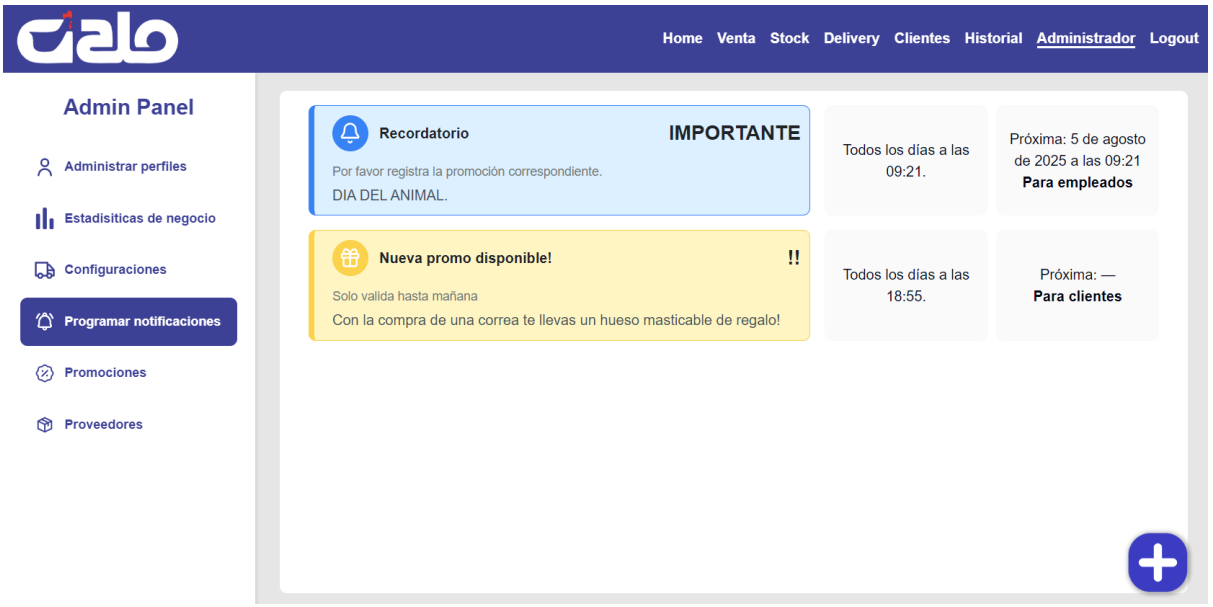
Pantalla de administrador > sección estadísticas:



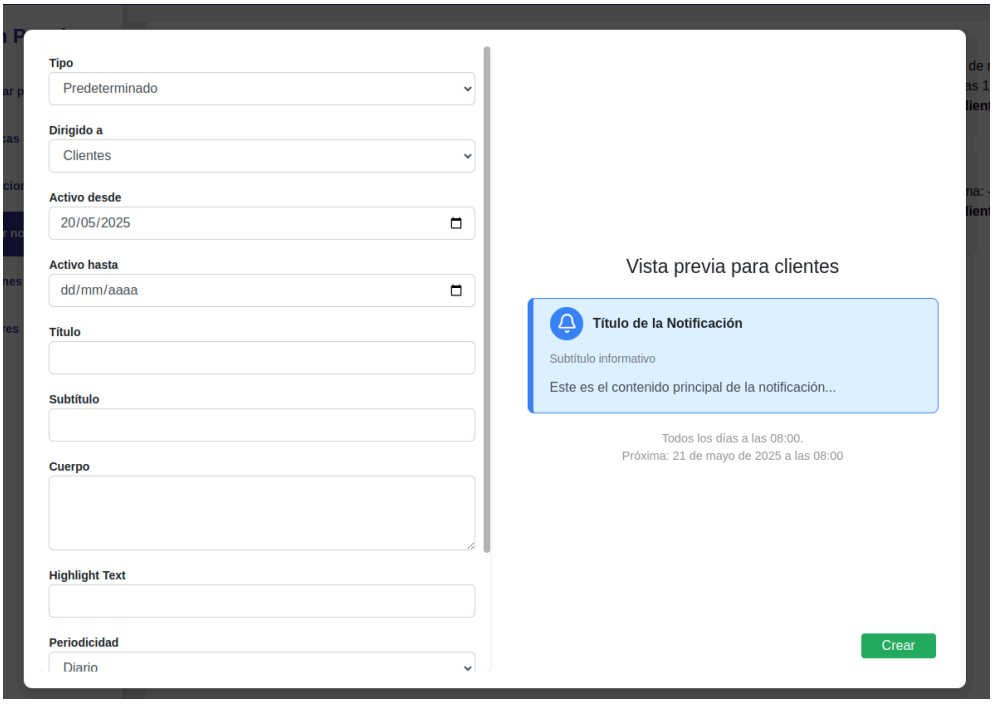
Pantalla de administrador > sección configuración:



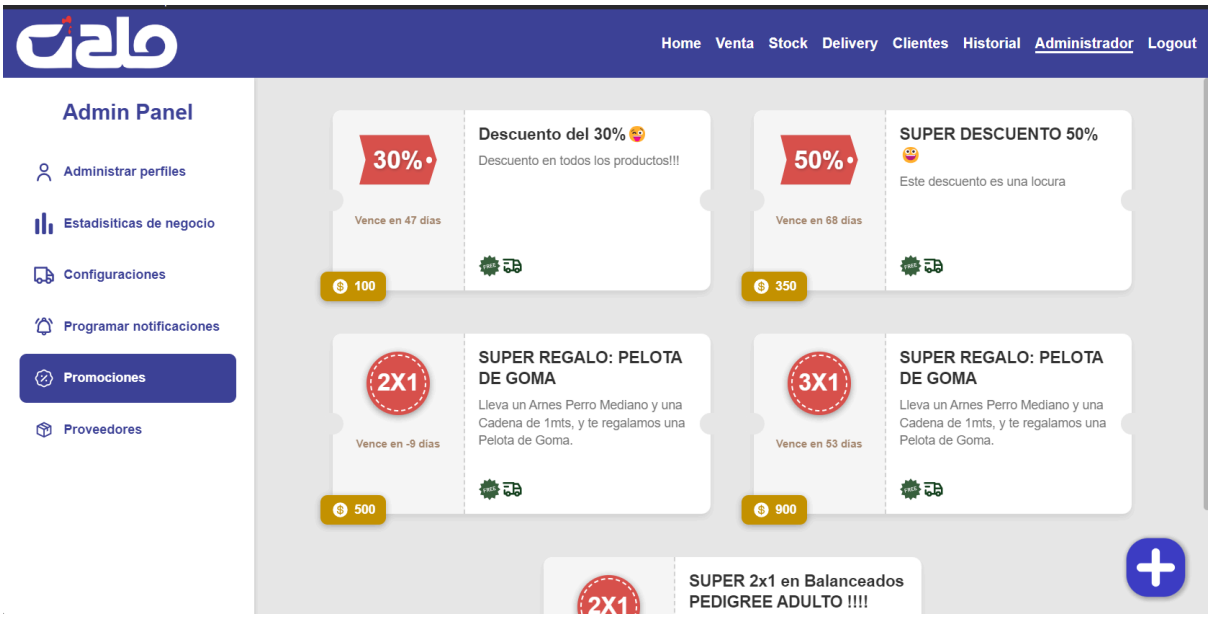
Pantalla de administrador > sección notificaciones programadas:



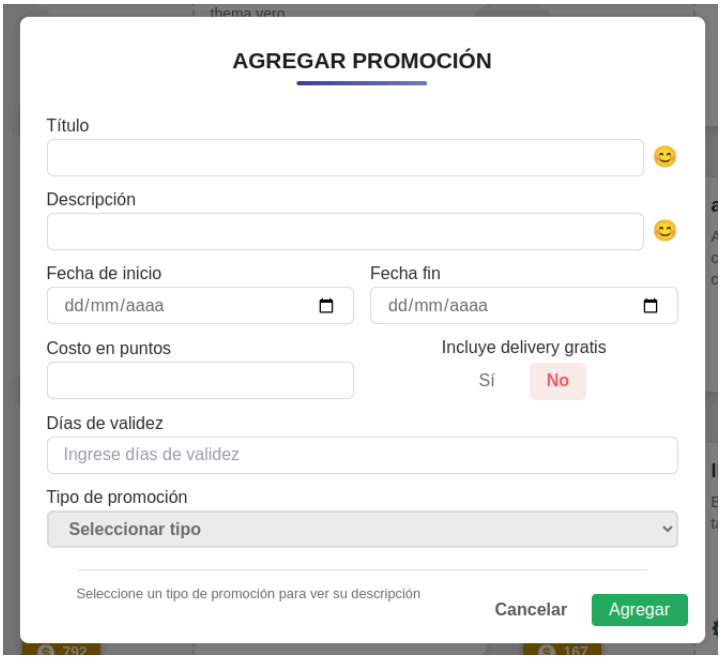
Pantalla de administrador > modal notificaciones programadas (creación):



Pantalla de administrador > sección promociones:

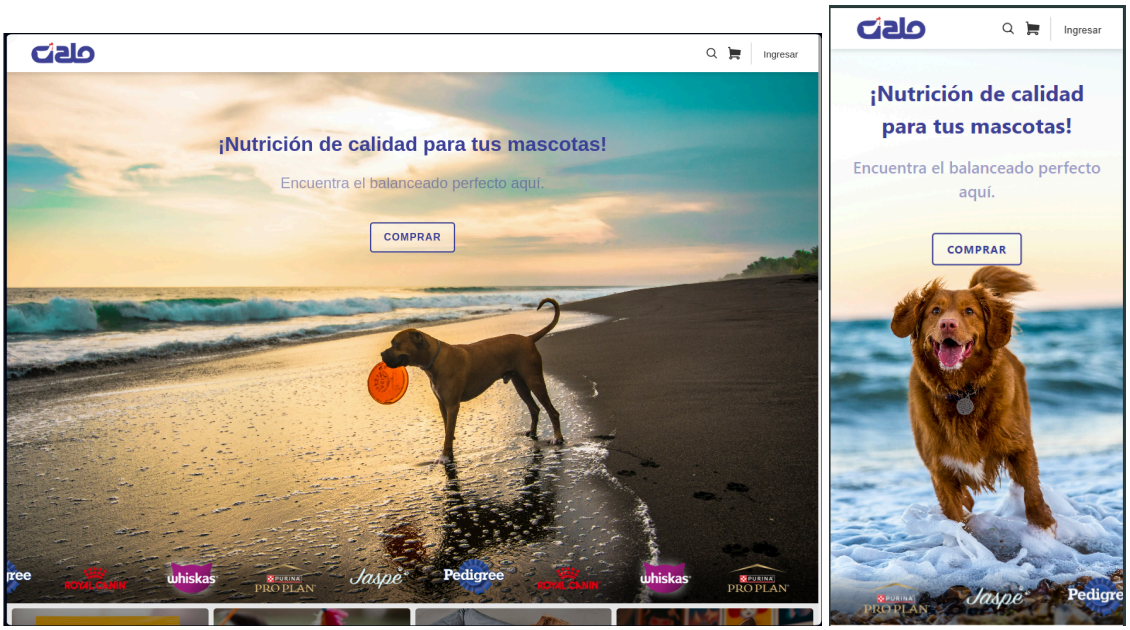


Pantalla de administrador > modal promociones (creación):

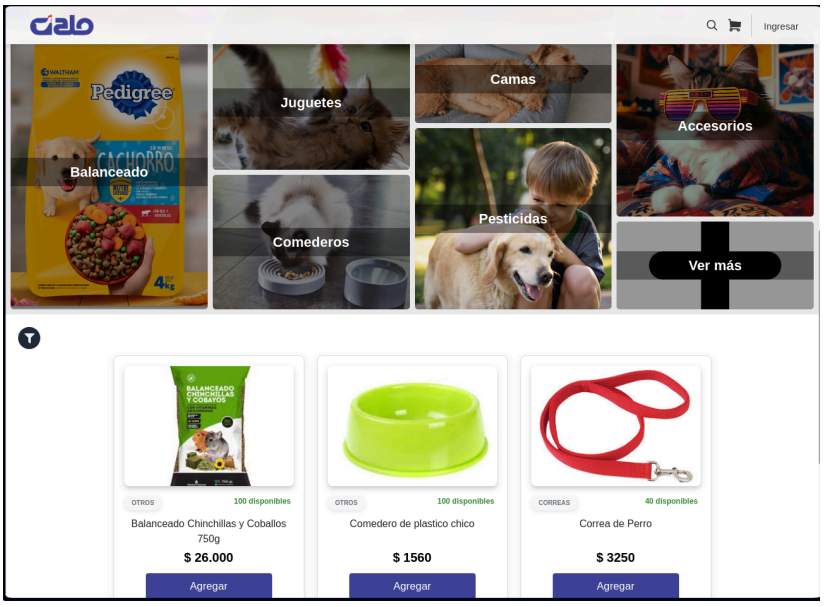


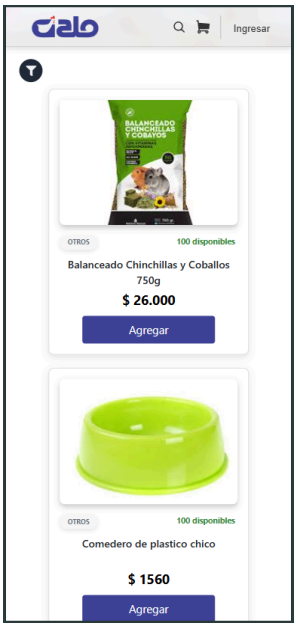
2.1.2 Pantallas Web app y Mobile App (PWA)

Home:

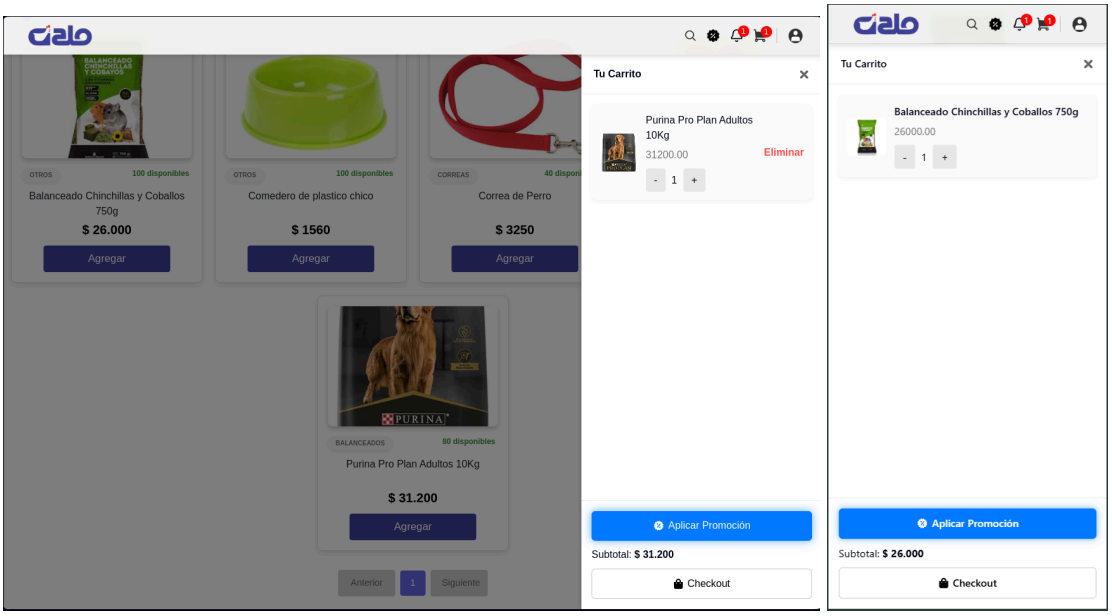


Sección de productos:

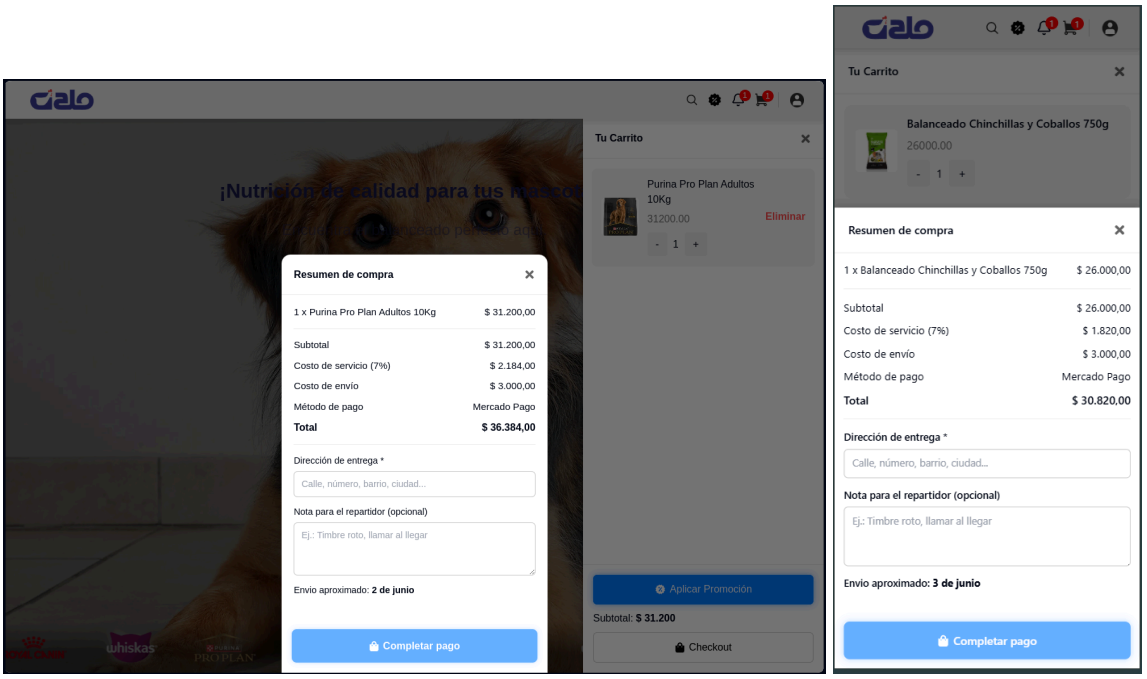




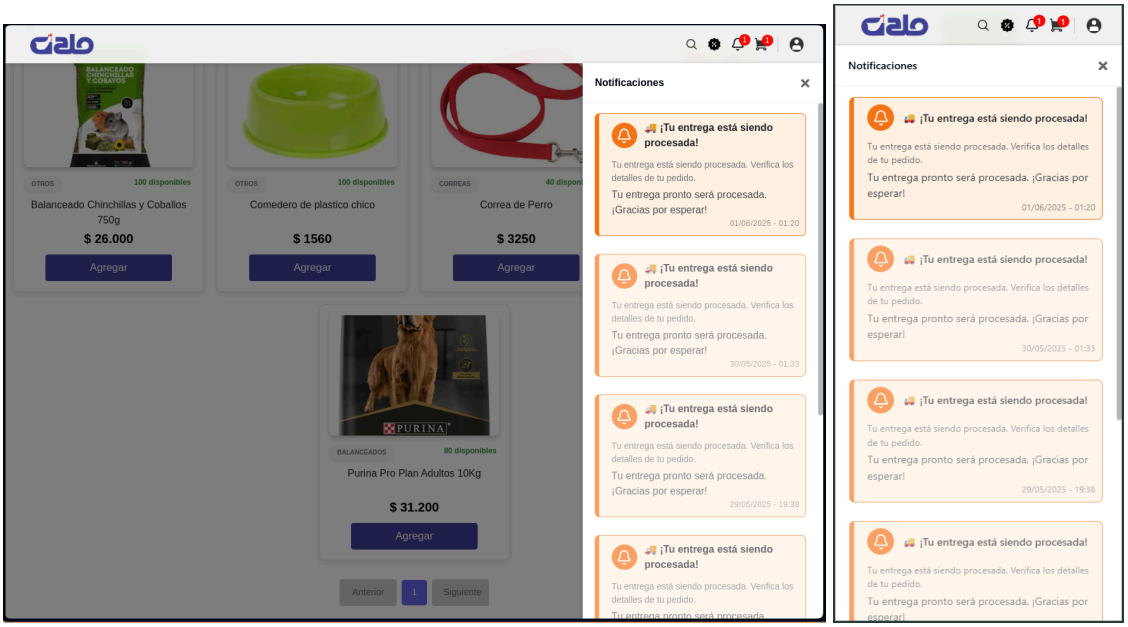
Carrito:



Checkout:

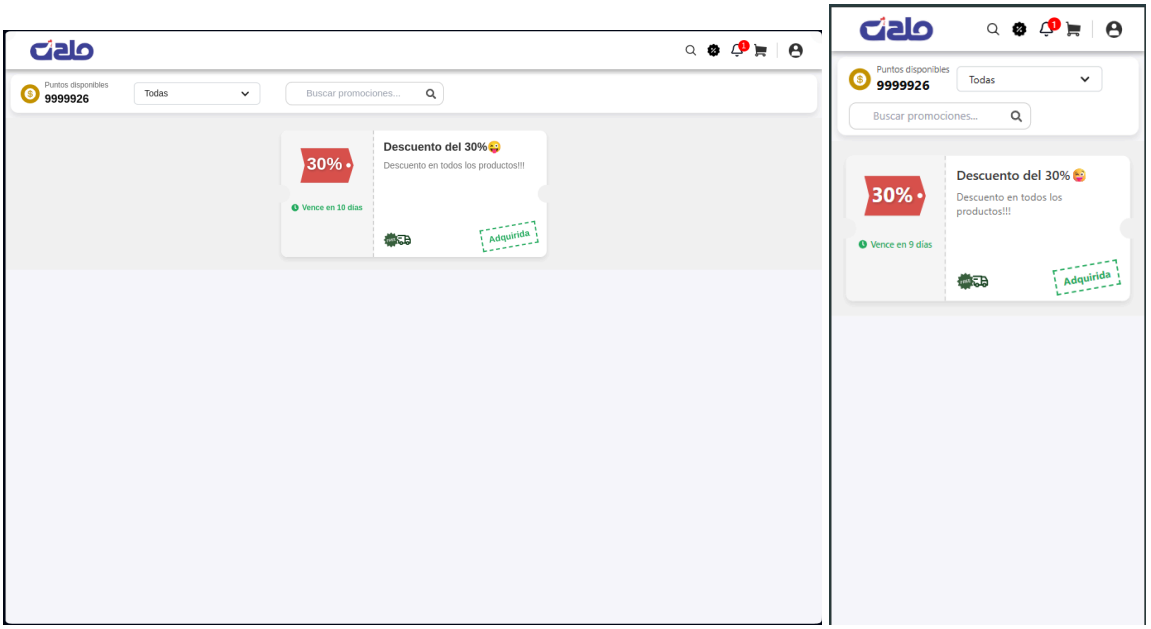


Panel de notificaciones:



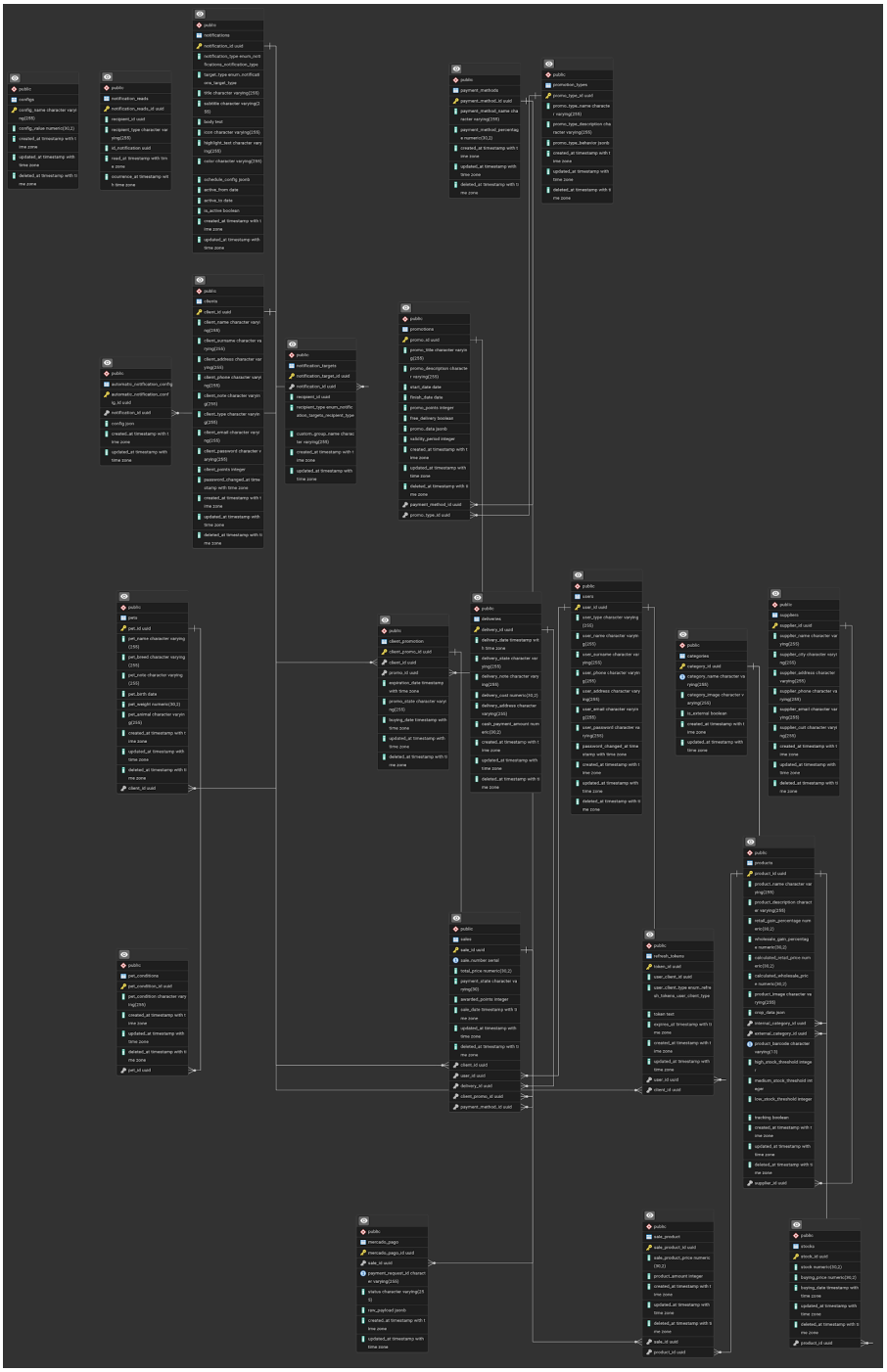
Pantalla de mi perfil:

Pantalla de promociones:



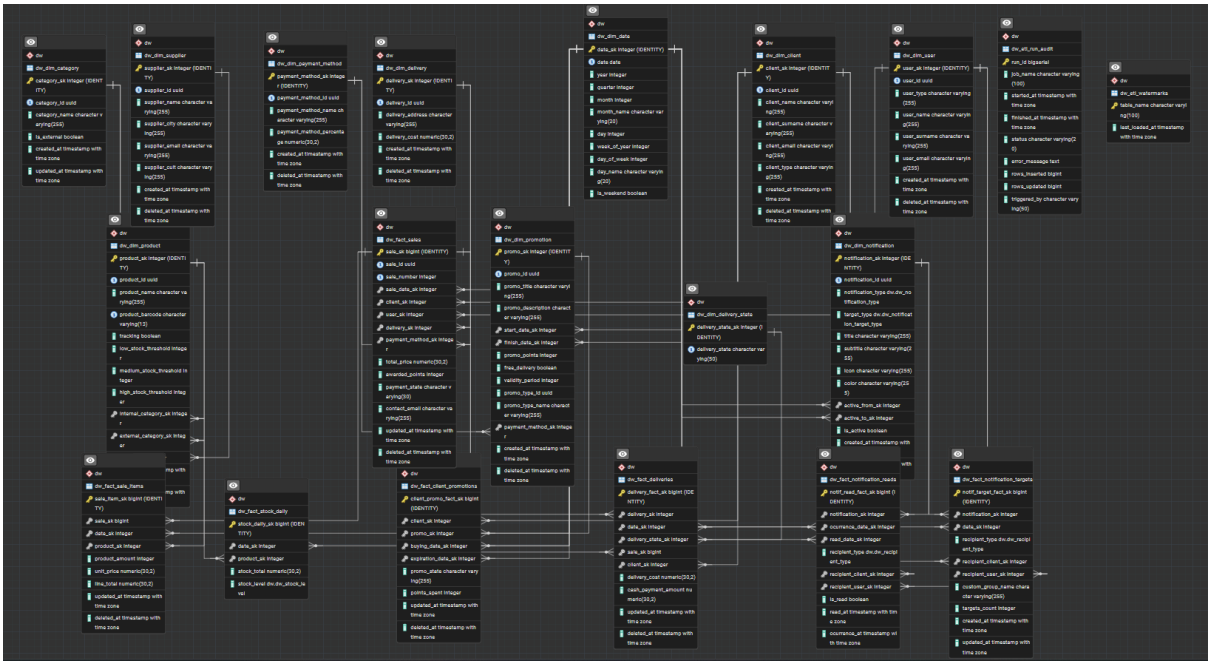
2.2 Diagramas

2.2.1 Modelo BD Principal (OLTP)



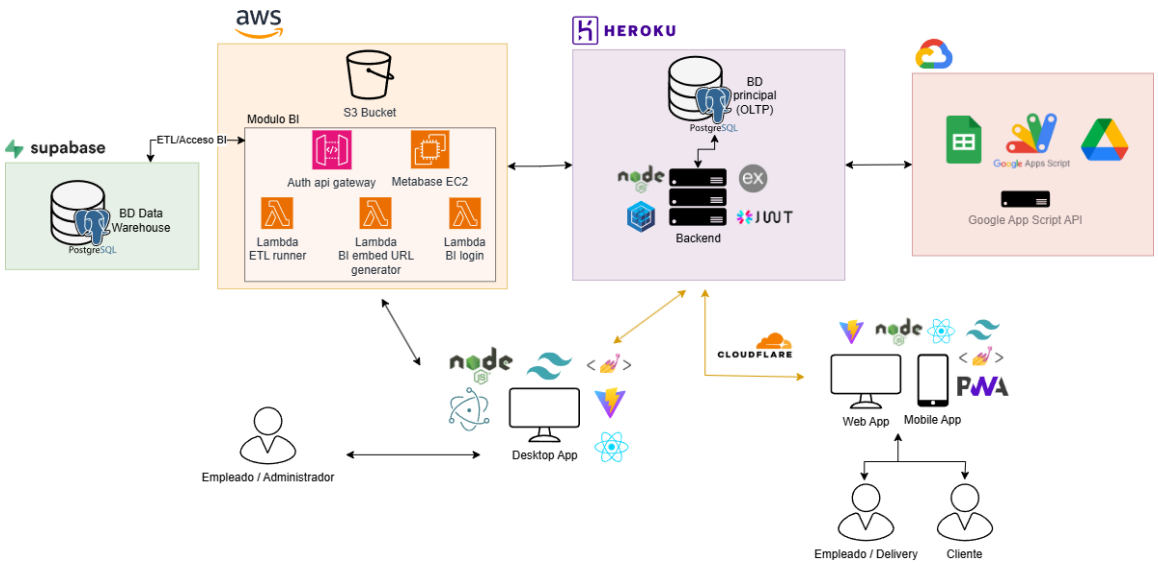
ER - DB - Galo.png

2.2.2 Modelo BD Data Warehouse



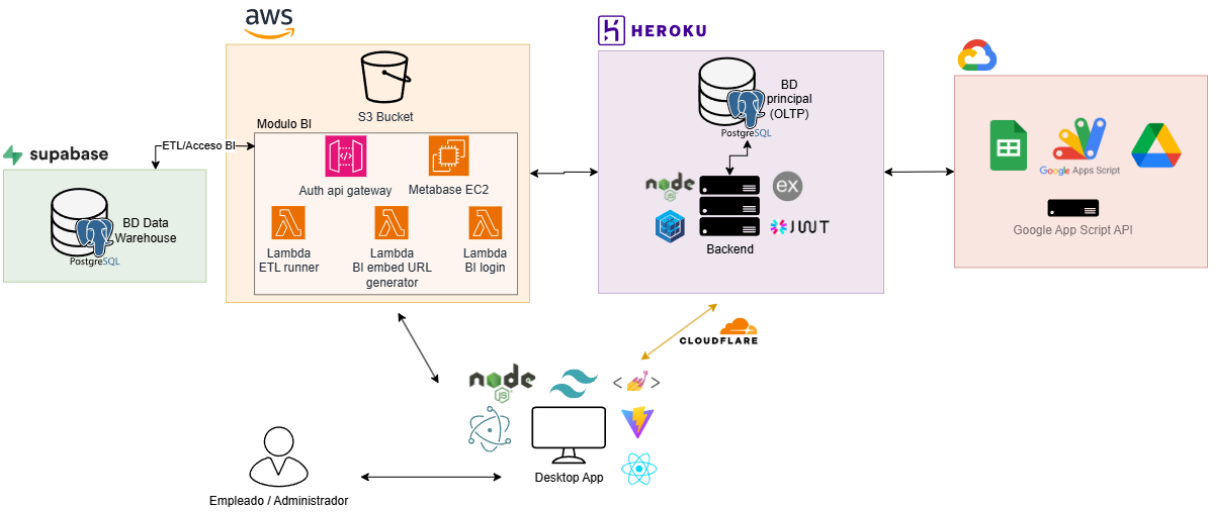
ER - DW - Galo.png

2.2.3 Diagrama del sistema completo



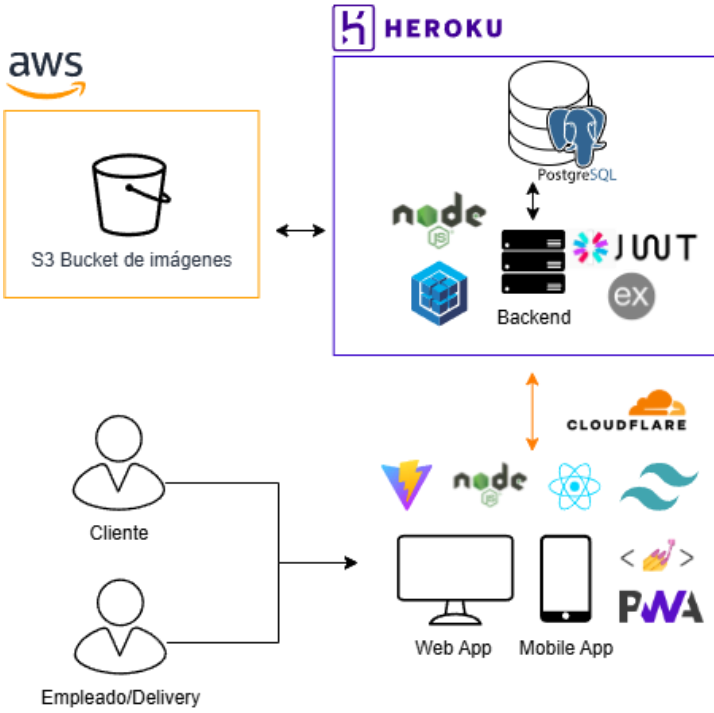
La imagen muestra un diagrama de todo el sistema y las tecnologías involucradas.

2.2.4 Diagrama de la Desktop app



La imagen muestra un diagrama de la aplicación de escritorio y las tecnologías involucradas

2.2.5 Diagrama de la Web app y Mobile app



La imagen muestra un diagrama de la aplicación web junto con la mobile y las tecnologías involucradas

2.3 Arquitectura

2.3.1 Arquitectura lógica

2.3.1.1 Introducción y actores

En este sistema intervienen dos tipos de usuarios principales. Por un lado, se encuentran los clientes y repartidores, quienes utilizan la Web App o la PWA desde sus propios dispositivos, ya sea móviles o computadoras, para consultar productos, gestionar pedidos y confirmar entregas. Por otro lado, están los empleados internos, que operan la Desktop App desarrollada con Electron desde sus máquinas locales para llevar adelante tareas administrativas y operativas del negocio.

A nivel general, tanto la Web App/PWA como la Desktop App comparten una misma capa transaccional de backend, desplegada en Heroku. Dicho backend coordina el acceso a una base de datos relacional PostgreSQL, gestiona el almacenamiento de imágenes en AWS S3 y crea hojas de cálculo en Google Sheets/Drive mediante Google Apps Scripts. Asimismo, las peticiones públicas del sistema web atraviesan primero Cloudflare, que aporta red de distribución de contenido (CDN) y protección frente a ataques DDoS.

Además de esta arquitectura operativa, la solución final incorpora una capa analítica desacoplada, orientada a la consulta de indicadores históricos y dashboards estadísticos. Esta capa se apoya en un Data Warehouse separado de la operatoria transaccional, alimentado mediante procesos ETL, y en una herramienta de visualización analítica basada en Metabase. De este modo, el acceso al panel estadístico no depende directamente del backend principal ni de la base de datos

operativa, permitiendo mantener la consulta de métricas aun ante contingencias en la infraestructura transaccional.

2.3.1.2 Capa analítica y dashboards estadísticos

Con el objetivo de brindar soporte a la toma de decisiones y mejorar la disponibilidad del panel estadístico, se incorporó una arquitectura analítica separada de la capa operativa principal. Esta arquitectura está compuesta por un proceso de extracción, transformación y carga de datos (ETL), un Data Warehouse destinado a consolidar información histórica del negocio y una herramienta de visualización basada en Metabase para la construcción y consulta de dashboards.

Los procesos ETL toman información proveniente de la base de datos operativa, la transforman y la cargan en la estructura analítica. Esta separación permite preparar los datos para consultas orientadas a indicadores, comparativas temporales y análisis gerencial, evitando cargar innecesariamente la base transaccional con consultas complejas o históricas. De esta manera, la información utilizada para fines analíticos se encuentra organizada en una capa específicamente diseñada para su explotación visual y estadística.

Sobre esta base analítica se despliega Metabase como herramienta de visualización, permitiendo construir tableros de control con métricas relevantes del negocio, tales como ventas, productos, stock, márgenes y desempeño general. A su vez, el acceso al panel analítico se encuentra desacoplado del backend principal, ya que utiliza una infraestructura independiente para la consulta del Data Warehouse. Esto permite que el módulo estadístico continúe siendo consultable incluso si la capa operativa principal o la base de datos transaccional se encuentran temporalmente fuera de servicio.

En este sentido, la arquitectura lógica final se compone de dos grandes capas complementarias: una capa transaccional, responsable de la operatoria diaria del sistema, y una capa analítica, orientada a la consolidación, visualización e interpretación de información histórica y estratégica. Esta separación no solo mejora la capacidad de análisis del sistema, sino que también aporta robustez y continuidad de acceso a los dashboards estadísticos.

2.3.1.3 Flujos principales

Flujo de la Web App / PWA (Clientes y Repartidores)

Cuando un cliente o repartidor ingresa desde su navegador, la petición HTTPS se dirige primero a Cloudflare, que actúa como red de distribución de contenido (CDN) y filtra posibles ataques (DDoS, bots, etc.). Desde Cloudflare, la solicitud llega al nodo Web App/Mobile App alojado en Heroku. El frontend sirve las páginas web (HTML/CSS/JavaScript) y maneja la navegación, autenticación y la interacción típica de e-commerce (visualización de catálogos, carrito, checkout, etc.).

En el Backend, el sistema recibe esas peticiones y ejecuta la lógica de negocio correspondiente. Si hay que leer o escribir datos transaccionales (usuarios, productos, pedidos, notificaciones, etc.), el Backend se conecta a la base de datos PostgreSQL en Heroku. Para almacenar o recuperar imágenes y otro contenido multimedia, envía/descarga objetos desde el servicio de AWS S3 (“Lectura/Escritura”). Cuando se quiere exportar datos o generar reportes, el Backend invoca Google Apps Scripts para crear o actualizar hojas de cálculo en Google Sheets/Drive (“Crea y actualiza sheets”). Una vez procesada la acción, el Backend devuelve la respuesta al frontend, que la muestra al usuario.

Flujo de la Desktop App (Empleados Internos)

El empleado inicia la Desktop App (Electron) en su máquina local. Detrás de escena, Electron arranca un navegador Chromium embebido que carga una interfaz React/Vite que conecta al Backend en Heroku. La Desktop App se comunica con el Backend mediante peticiones HTTPS o WebSocket (“HTTPS/WebSocket”) para realizar tareas internas: gestión de stock, control de entregas, administración de devoluciones, entre otros.

En el Backend, las solicitudes provenientes de la Desktop App se tratan de la misma forma que las de la Web App/PWA, tanto así como la subida/bajada de imágenes, reportes, entre otros, esto se debe a que se utiliza el mismo backend para las

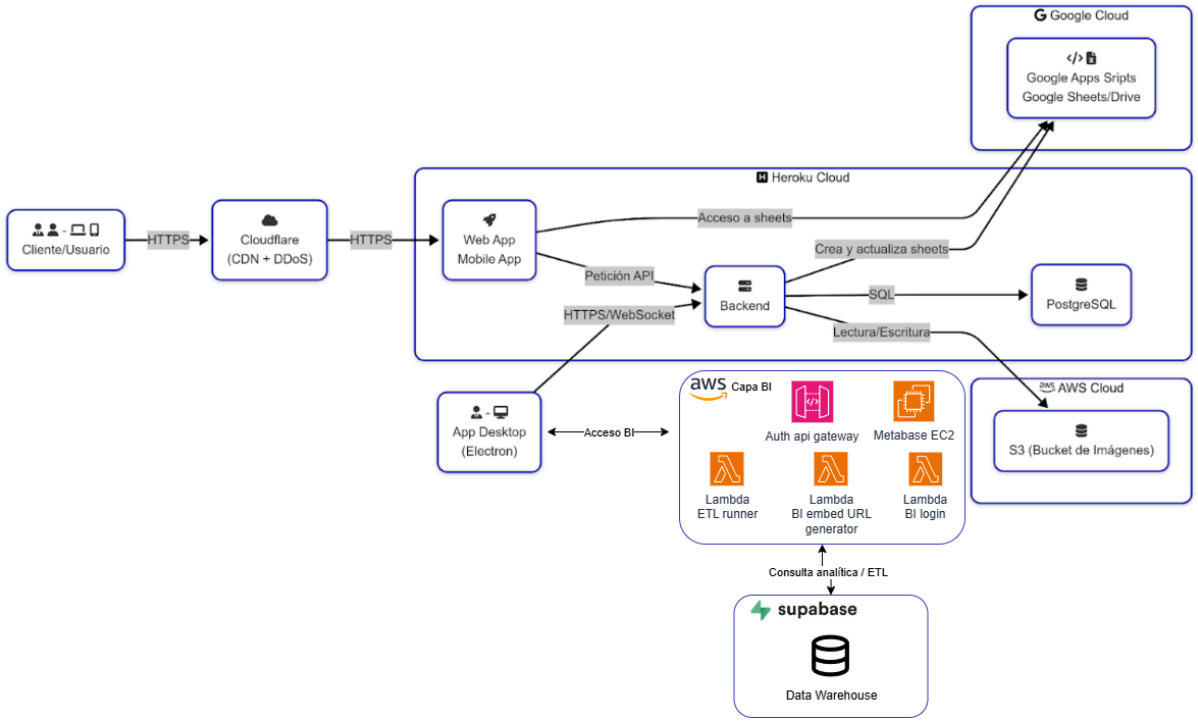
distintas aplicaciones, la única diferencia es el punto de entrada: el cliente usa un navegador público, mientras que el empleado usa la app de escritorio local.

Flujo de la capa analítica en la Desktop App

En paralelo a la operatoria diaria, los datos generados por el sistema transaccional son extraídos periódicamente mediante procesos ETL. Estos procesos transforman y consolidan la información relevante en un Data Warehouse separado, diseñado específicamente para consultas analíticas e históricas.

Una vez cargada la información en la estructura analítica, Metabase se conecta directamente al Data Warehouse para construir y mostrar dashboards estadísticos. Desde la Desktop App, el usuario con permisos adecuados puede acceder al panel analítico y consultar KPIs, métricas históricas y comparativas por período, producto, categoría o desempeño general.

Dado que esta capa se encuentra desacoplada del backend principal y de la base de datos operativa, el acceso al panel estadístico puede mantenerse disponible aun cuando la infraestructura transaccional principal no se encuentre operativa. De esta forma, la solución no solo cubre la ejecución de procesos del negocio, sino también la explotación analítica de la información en un entorno más robusto para fines de consulta gerencial.



2.3.2 Decisiones tecnológicas y justificación

PostgreSQL (Heroku Postgres)

Para la base de datos operativa del sistema se seleccionó PostgreSQL, desplegado mediante el servicio gestionado Heroku Postgres. La solución requiere administrar información fuertemente relacionada, como usuarios, productos, ventas, pedidos, promociones, entregas y movimientos de stock, por lo que resulta necesario contar con un motor relacional capaz de garantizar integridad y atomicidad en las operaciones críticas del negocio.

La elección de PostgreSQL también respondió a la necesidad de almacenar ciertos atributos con mayor flexibilidad, especialmente en componentes como la configuración de promociones y otros datos semiestructurados. Su soporte nativo para JSONB permite resolver estas necesidades dentro del mismo motor de base de datos, evitando incorporar una base documental adicional y reduciendo la

complejidad asociada a sincronización, mantenimiento y consistencia entre distintos repositorios.

Asimismo, PostgreSQL permite implementar procedimientos almacenados y triggers mediante PL/pgSQL. Esta capacidad fue utilizada para la funcionalidad de predicción de reposición, donde el registro de nuevas compras permite actualizar cálculos asociados al historial de consumo y programar notificaciones futuras dentro de una lógica centralizada y consistente.

Finalmente, el uso de Heroku Postgres permite delegar tareas operativas de infraestructura, como administración del servicio, monitoreo y mecanismos de respaldo provistos por la plataforma. De esta manera, la elección combina las capacidades técnicas de PostgreSQL con un esquema de despliegue adecuado para el alcance y las restricciones operativas del proyecto.

Node.js + Express + Sequelize

Para el desarrollo del backend se seleccionó Node.js con Express y Sequelize. Si bien durante la investigación tecnológica se analizaron alternativas como Django y NestJS, la elección de Express respondió principalmente a su adecuación al contexto del proyecto y a la experiencia previa del equipo en el ecosistema JavaScript. Esta decisión permitió mantener un stack tecnológico homogéneo con las aplicaciones frontend, reducir la curva de aprendizaje y avanzar con mayor rapidez en la implementación de los módulos funcionales.

Node.js permite atender múltiples peticiones concurrentes mediante un modelo de ejecución asíncrono, adecuado para operaciones frecuentes del sistema, como la consulta de productos, la actualización de stock, el procesamiento de pedidos y los cambios de estado de las entregas. Express proporciona una estructura liviana para exponer endpoints REST y gestionar la comunicación requerida por las aplicaciones. Por su parte, Sequelize actúa como ORM para abstraer el acceso a PostgreSQL, administrar relaciones entre entidades y simplificar las operaciones de persistencia.

En conjunto, esta combinación tecnológica permitió desarrollar sobre una misma base los servicios requeridos por la aplicación web, la PWA y la aplicación de

escritorio, manteniendo centralizada la lógica de negocio asociada a ventas, inventario, clientes, promociones, delivery y administración.

React + Vite + Tailwind CSS en styled-components

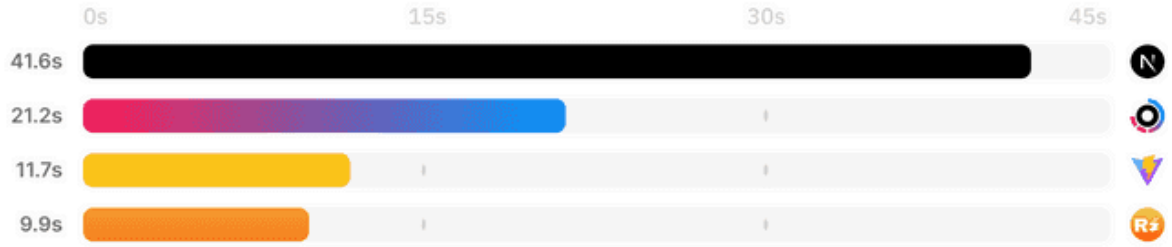
Para el desarrollo de las interfaces de usuario se seleccionó React como biblioteca principal, junto con Vite como herramienta de desarrollo y construcción de los proyectos frontend. La elección de React respondió a la experiencia previa del equipo con su modelo basado en componentes y a la posibilidad de reutilizar criterios de interfaz y lógica de presentación entre la Web App, la PWA y la Desktop App empaquetada mediante Electron.

En comparación con otras alternativas analizadas, como Angular y Vue, React permitió avanzar sobre una tecnología conocida por el equipo, reduciendo tiempos de adaptación y facilitando la implementación de componentes reutilizables para catálogos, carritos, formularios, paneles administrativos y pantallas de seguimiento de pedidos. Esta decisión resultó adecuada considerando que el proyecto contempla varias interfaces que comparten criterios visuales y funcionalidades relacionadas.

Para el entorno de desarrollo y generación de builds se seleccionó Vite, debido a su configuración liviana, su compatibilidad con React y sus tiempos de respuesta adecuados para un flujo de desarrollo iterativo. Como referencia comparativa para esta decisión, la Figura 1 presenta los tiempos reportados por KyleGill[8] para una aplicación React ejecutada bajo configuraciones con Next.js/Webpack, Next.js/Turbopack, Vite/Rollup y Vite/Rolldown. En las métricas observadas, las configuraciones basadas en Vite registran menores tiempos de arranque, compilación de página y refresco que las configuraciones basadas en Next.js. Estos resultados no constituyen una garantía universal de rendimiento, dado que dependen de la aplicación evaluada, pero respaldan la selección de una herramienta orientada a agilizar las iteraciones de desarrollo.

Cold Start

Booting up the Particl app to first render of full application with no cache.



Page Compilation

Navigating to a new page at a separate route, from click to first full render.

Hard Refresh

Browser refresh button (or an edit of a non React Fast Refresh compatible file)

Fast Refresh

React Fast Refresh time from save to changes being visible of a single component.

Figura 1. Comparación de tiempos de desarrollo entre Next.js/Webpack, Next.js/Turbopack, Vite/Rollup y Vite/Rolldown, presentados de arriba hacia abajo en cada métrica. Fuente: KyleGill[8].

En cuanto al diseño visual, se utilizaron utilidades de Tailwind CSS junto con styled-components para agilizar la construcción de interfaces y mantener estilos encapsulados por componente. Esta combinación permitió desarrollar pantallas de forma consistente, reducir la repetición de estilos y facilitar el mantenimiento visual de las distintas aplicaciones que integran la solución.

Electron (Desktop App – ERP)

Electron envuelve la base de código React/Vite, generando una aplicación de escritorio nativa con Chromium embebido. Esto acelera el desarrollo de ERP, ya que no hace falta reescribir la interfaz para Windows/macOS/Linux: se empaqueta el mismo bundle web, pero con acceso a recursos locales (sistema de archivos, impresoras). La Desktop App consume el mismo backend que la Web App, pero con rutas exclusivas (facturación interna, stock, reportes, administración) protegidas por roles.

PWA (Plataforma mobile para clientes y repartidores)

Para el acceso móvil de clientes y repartidores se seleccionó el enfoque Progressive Web App (PWA) como extensión de la aplicación web desarrollada con React y Vite. Durante el análisis se consideraron alternativas como Flutter y React Native; sin embargo, ambas implicaban incorporar un flujo adicional de generación, distribución y mantenimiento de aplicaciones destinadas a tiendas móviles. En el caso de Flutter, además, hubiese sido necesario adoptar Dart y desarrollar una interfaz separada respecto de la base construida con React.

La elección de una PWA permitió reutilizar la misma base tecnológica de la Web App y ofrecer acceso desde dispositivos móviles sin requerir la publicación inicial en Google Play o App Store, cuyos programas de desarrolladores contemplan costos y

procesos propios de distribución[9] [10]. Este enfoque resultó adecuado para el alcance del proyecto, dado que las funcionalidades requeridas por clientes y repartidores se centran en consultar productos, realizar pedidos, recibir información del estado de las entregas y operar desde un dispositivo móvil conectado a Internet.

Asimismo, la PWA permite que las actualizaciones funcionales se encuentren disponibles a partir del despliegue de la aplicación web, evitando generar y aprobar una nueva versión en una tienda para cada modificación. Su configuración contempla los componentes necesarios para habilitar características propias de este enfoque, como el manifiesto de aplicación, el service worker y la publicación mediante HTTPS, de acuerdo con las prácticas recomendadas para aplicaciones web progresivas[11].

Esta decisión permitió reducir la complejidad de desarrollo y mantenimiento, mantener coherencia tecnológica con el frontend web y disponer de una alternativa móvil suficiente para las necesidades funcionales identificadas en Club Galo.

Cloudflare (CDN + WAF + DDoS)

Cloudflare actúa como primera línea de defensa y distribución. Sirve contenido estático (JS, CSS e imágenes) desde el edge más cercano al usuario (nodo en Brasil, por ejemplo), reduciendo la latencia de 280 ms a 120 ms en Córdoba. Además, protege la API contra ataques DDoS y bots con su WAF, aplica reglas de rate-limiting en endpoints críticos (login, checkout, actualizaciones de entrega) y gestiona certificados SSL/TLS de manera automática.

AWS S3 + CloudFront

El almacenamiento de objetos (imágenes de productos, fotografías de mascotas, documentos PDF) se delega a S3 para no sobrecargar los dynos de Heroku. CloudFront distribuye esos objetos globalmente, cacheando en edge nodes, lo que garantiza alta disponibilidad y descargas rápidas, reduciendo la carga de red en Heroku.

Google Sheets / Google Apps Scripts

En lugar de mantener un microservicio de reportes, se eligió Google Sheets porque es gratuito y permite crear hojas preformateadas, compartirlas y colaborarlas en Drive. El backend invoca Apps Scripts para llenar las hojas con datos de ventas diarias, entregas pendientes o análisis de stock, evitando drásticamente el tiempo de desarrollo de un sistema de reportes propio.

MercadoPago (Procesamiento de pagos)

Para el procesamiento de pagos digitales se seleccionó MercadoPago como pasarela integrada al canal web y móvil del sistema. Durante la investigación tecnológica se comparó esta alternativa con Pagos360, considerando criterios vinculados a métodos de pago disponibles, experiencia del usuario, documentación para la integración y adecuación al contexto de compra en línea previsto para Club Galo.

La elección de MercadoPago respondió principalmente a la posibilidad de ofrecer un flujo de pago ampliamente conocido por los usuarios y compatible con compras realizadas desde dispositivos móviles. La plataforma permite procesar pagos mediante distintos medios disponibles para el cliente y facilita una experiencia de checkout integrada al canal digital desarrollado para la empresa.

Desde el punto de vista técnico y de seguridad, la integración delega el tratamiento de la información sensible de pago al proveedor especializado, evitando que los datos de tarjeta deban ser almacenados o procesados directamente por los servidores propios del sistema. El backend conserva únicamente la información necesaria para asociar el estado de la operación con el pedido correspondiente y procesar las confirmaciones recibidas mediante los mecanismos de integración provistos por la pasarela.

Esta decisión permitió incorporar cobros digitales al sistema sin desarrollar una infraestructura propia de procesamiento de pagos, reduciendo complejidad técnica y

manteniendo el foco del proyecto en la gestión comercial, el seguimiento de pedidos y la experiencia del cliente.

Heroku (Despliegue y contenedores Docker)

Para el despliegue de la capa transaccional del sistema se seleccionó Heroku como plataforma administrada para alojar la aplicación web/PWA, el backend y la base de datos operativa mediante Heroku Postgres. Esta decisión respondió a la necesidad de disponer de una infraestructura simple de operar, con un flujo de publicación accesible para el equipo y un costo inicial compatible con el alcance del proyecto.

Durante la investigación tecnológica se consideraron alternativas como Railway, Render, Vercel, AWS, Google Cloud y Microsoft Azure. Si bien las plataformas cloud de propósito general permiten configurar arquitecturas con mayor nivel de personalización, su adopción para la capa operativa principal hubiese implicado asumir tareas adicionales de configuración, monitoreo y administración de infraestructura. Para Club Galo, Heroku permitió priorizar el desarrollo funcional del sistema y reducir el esfuerzo requerido para mantener en funcionamiento los servicios transaccionales.

La utilización de contenedores Docker para los proyectos desplegados permite mantener consistencia entre los entornos de desarrollo y producción. Asimismo, el proceso de integración y despliegue continuo, implementado mediante GitHub Actions, automatiza la construcción y publicación de nuevas versiones, mientras que la plataforma centraliza la gestión de variables de entorno, registros de ejecución y releases.

En caso de incrementarse la demanda del sistema, la infraestructura desplegada en Heroku permite ampliar la capacidad contratada o incorporar más procesos de ejecución sin modificar la arquitectura funcional de las aplicaciones. De este modo, la elección ofrece un equilibrio adecuado entre simplicidad operativa, capacidad de evolución y costos iniciales para la capa transaccional de la solución.

PL/pgSQL (Inteligencia de Negocio en la BD)

El algoritmo de reposición y las notificaciones automáticas se implementan en PL/pgSQL: un trigger en la tabla de compras invoca un procedimiento almacenado que actualiza promedios y reprograma las notificaciones (por ejemplo, tres días antes del próximo promedio estimado). Esto maximiza el rendimiento al evitar viajes adicionales entre backend y BD y asegura que todo ocurra dentro de la misma transacción.

Seguridad de API y Protección contra abusos

Se definieron tres niveles de exposición: interno (ERP), semi-público (API de catálogo) y público crítico (e-commerce con guía de reparto). Para público crítico, se implementan tokens JWT rotativos, refresh tokens, cookies httpOnly, hashing de contraseñas con bcrypt, cabeceras de seguridad (HSTS, CSP, X-Frame-Options, Referrer-Policy), Cloudflare WAF y rate-limiting en endpoints sensibles (login, checkout, actualizaciones de entrega). La delegación de datos de tarjetas a MercadoPago y cifrado en reposo en PostgreSQL garantizan el cumplimiento de PCI-DSS y alineación con guías OWASP.

Integraciones externas

- **MercadoPago** para cobros y devoluciones.
- **Amazon S3 + CloudFront** para distribución de archivos estáticos.
- **Cloudflare** como CDN, WAF y mitigador de DDoS.
- **Google Cloud Platform (GCP) -** Se utiliza una API (Diseñada por nosotros) de Google Apps Scripts que conecta con Google Sheets, para reportes colaborativos vía Google Drive sin costo.
- **API de cotización de dólar** para ajustar precios dinámicos.

Metabase + Data Warehouse + procesos ETL (Dashboards estadísticos)

Para la capa analítica del sistema se incorporó una arquitectura compuesta por procesos ETL, un Data Warehouse separado de la base transaccional y Metabase como herramienta de visualización. Esta decisión permitió consolidar información histórica del negocio en una estructura orientada a consultas analíticas, evitando ejecutar reportes complejos directamente sobre la base operativa. De este modo, los dashboards estadísticos pueden construirse sobre datos preparados específicamente para el análisis de ventas, stock, productos, promociones y desempeño general, mejorando la claridad de las métricas y reduciendo el impacto sobre la operatoria diaria del sistema.

Metabase fue seleccionado porque permite construir paneles e indicadores sin desarrollar manualmente una interfaz analítica completa. Esto evitó destinar esfuerzo a implementar desde cero gráficos, filtros, consultas y lógica de visualización que una herramienta especializada ya resuelve de forma nativa. Además, su orientación a analítica de negocio resultó más adecuada para los objetivos del proyecto que otras alternativas más enfocadas en monitoreo técnico, permitiendo disponer de dashboards comprensibles, reutilizables y alineados con necesidades de gestión.

Desacople del panel analítico respecto de la capa transaccional

Una de las decisiones técnicas más importantes de esta arquitectura fue desacoplar el acceso al panel analítico del backend principal y de la base de datos operativa. Para ello, el módulo estadístico consulta directamente una fuente analítica separada y dispone de un mecanismo de acceso independiente, de modo que la visualización de indicadores pueda mantenerse disponible aun cuando la infraestructura transaccional principal se encuentre temporalmente fuera de servicio. Esta separación aporta robustez, continuidad de consulta y una mejor adecuación entre los distintos tipos de uso del sistema: por un lado, la operatoria diaria; por otro, el análisis estadístico y gerencial.

2.4 Patrones de diseño

2.4.1 Patrones de diseño utilizados

Arquitectura en capas (Layered Architecture)

El backend está organizado en cuatro capas: Controller → Service → Client → Model.

- *Controller*: recibe peticiones HTTP/WebSocket y valida parámetros.
- *Service*: orquesta la lógica de negocio, invoca a los *clients* y transforma datos en objetos de dominio.
- *Client*: encapsula llamadas a servicios externos (MercadoPago, Google Sheets) o a la base de datos mediante Sequelize.
- *Model*: define esquemas y relaciones en PostgreSQL.

  Este patrón permite que tanto las funcionalidades del e-commerce (carrito, checkout, guía de reparto en tiempo real) como las del ERP (facturación, stock, reportes) convivan sin mezclarse y posibilita el cambio de ORM con impacto mínimo.

Inversión de Dependencias (Dependency Inversion) y Principios SOLID

Cada servicio expone una interfaz y no accede directamente a Sequelize; en su lugar, utiliza un *Client* (por ejemplo, PagosClient, GoogleSheetsClient). Con ello se satisface el Principio de Inversión de Dependencias y se evita acoplar la capa de negocio a la de datos. El Principio de Responsabilidad Única se respeta creando módulos independientes para módulos de e-commerce, ERP, notificaciones y autenticación.

Composición de componentes en el frontend

Se sigue un enfoque Atomic Design:

- **Átomos**: botones, inputs, badges.
- **Moléculas**: grupos de formulario, menús desplegables sencillos.

- **Organismos**: navbar completo, tarjeta de producto con imagen, nombre y precio.

Cada componente React se escribe con styled-components y dentro del bloque de estilos se usan clases de Tailwind CSS (por ejemplo, flex, grid, px-4, text-center). Esto combina la velocidad de prototipado de Tailwind con la encapsulación y menor ruido de styled-components, manteniendo cada componente autocontenible y evitando colisiones de nombre de clases

Observer Pattern para WebSocket

Cada aplicación se suscribe a eventos importantes a través de WebSocket. Internamente se emplea un patrón Observador: cuando el backend emite un evento, los componentes suscritos reaccionan (refrescan datos o muestran alertas) sin necesidad de polling constante, optimizando uso de red y mejorando la experiencia del usuario.

Builder Pattern para consultas complejas en Sequelize

Para búsquedas avanzadas (como filtrar productos por varias categorías, rango de precios, estado de stock), se implementa un “query builder” que encadena métodos (.where(), .include(), .order()) hasta componer el objeto de opciones que se pasa a Sequelize. Esto evita condicionales anidados difíciles de leer y permite añadir filtros nuevos sin alterar la estructura base.

Template Method para generación de reportes

La creación de reportes (ventas diarias, entregas pendientes, análisis de inventario) sigue un flujo común: autenticación en Google Sheets, recolección de datos, formateo de celdas, carga a Drive y retorno de enlace. Se define un método abstracto en una clase base, y cada tipo de reporte hereda de esa clase y solo sobrescribe las secciones “obtención de datos” y “formateo específico”.

Circuit Breaker Pattern en llamadas a servicios externos

Para las integraciones con MercadoPago y la API de cotización de dólar, se implementa la lógica de “cortacircuitos”: si un servicio responde con errores o timeouts repetidos, se evita enviar nuevas peticiones durante un breve periodo, devolviendo un error controlado y evitando saturar la aplicación con llamadas que seguramente fallarán. Esto contribuye a la resiliencia y estabilidad ante fallas externas.

Fallback y política de reintentos (Retry Pattern)

Para operaciones críticas (por ejemplo, escritura en la BD al cerrar un lote de ventas en el ERP), se implementa lógica de reintento. Si una transacción falla por un timeout transitorio, se vuelve a intentar hasta un máximo de tres veces con tiempos de espera progresivos antes de devolver un error al usuario.

3 Implementación

En esta sección se presentará, de manera introductoria, la arquitectura modular del sistema y la responsabilidad de cada componente. El diseño está orientado a que cada módulo y submódulo funcione de forma completamente autónoma, encapsulando sus propias funciones, reglas de negocio y dependencias. Gracias a este enfoque, al implementar el sistema en una nueva organización se pueden seleccionar uno o varios módulos (y, en su caso, sus submódulos asociados) para conformar una solución a medida, sin necesidad de rehacer o duplicar código existente. De este modo, cada entidad podrá elegir únicamente los módulos que necesite (por ejemplo, incluir la funcionalidad de notificaciones programadas pero no la de predicción automática), y así adaptar el sistema a sus requerimientos específicos con la máxima flexibilidad.

3.1 Módulo de autenticación y gestion de usuarios

El módulo de autenticación y gestión de usuarios constituye una pieza clave del sistema, ya que define y gestiona dos tipos principales de usuarios: internos y externos. Los usuarios internos son aquellos que acceden a funciones de administración y gestión, como empleados y administradores. El sistema maneja dos roles: administrador y empleado, aunque la estructura es flexible y permitiría agregar más roles según las necesidades de la organización.

Los usuarios externos, denominados clientes, también pueden tener roles específicos, como minoristas o mayoristas, lo que permite ofrecer ventajas diferenciales, como precios ajustados al volumen de compras.

El sistema de autenticación se basa en tokens JWT, utilizando un esquema de Access Token y Refresh Token. Los Access Tokens tienen una duración corta de 2,5min para minimizar riesgos en caso de robo, mientras que los Refresh Tokens, con una duración de 7 días, permiten mantener la sesión activa sin necesidad de iniciar sesión repetidamente. Los Refresh Tokens se almacenan de manera segura y permiten que un usuario mantenga la sesión abierta en múltiples dispositivos simultáneamente. Además, en caso de que un usuario cambie su contraseña, todos los Refresh Tokens asociados se deshabilitan automáticamente, cerrando sesiones en otros dispositivos y brindando una capa adicional de seguridad.

La implementación de este sistema con JWT y Axios Interceptors asegura que, en caso de que un Access Token expire, se solicite automáticamente uno nuevo utilizando el Refresh Token, sin que el usuario perciba interrupciones. Esto se traduce en una experiencia de usuario fluida y segura, minimizando el impacto de posibles vulnerabilidades. Además, las cookies utilizadas están protegidas con flags de seguridad, dificultando su robo y asegurando una mayor integridad de la sesión.

3.2 Módulo de stock

El módulo de stock es fundamental para la gestión eficiente del inventario de la organización. Permite el control y monitoreo en tiempo real de los niveles de productos disponibles, asegurando que siempre se cuente con un abastecimiento adecuado para satisfacer la demanda. Este módulo ofrece funcionalidades para registrar entradas y salidas de productos, gestionar umbrales de stock mínimo y máximo, y generar alertas automáticas cuando los niveles de inventario alcanzan ciertos límites.

Una característica destacada del módulo de stock es la posibilidad de crear categorías internas y externas para los productos. Las categorías internas están diseñadas para el uso interno de la organización, permitiendo utilizar un lenguaje específico o técnico que pueda no ser comprensible para el cliente promedio, lo que facilita la gestión interna. Por otro lado, las categorías externas están pensadas para ser comprensibles y accesibles para los clientes, utilizando un lenguaje más natural y común, lo que mejora la experiencia del usuario y facilita la comunicación entre clientes y empleados.

Otra funcionalidad importante es la asignación de códigos de barras a los productos que no los tienen. Esto permite que, en caso de que la organización utilice escáneres de códigos de barras, el proceso de venta sea mucho más ágil y preciso, evitando errores en la introducción manual de productos. El sistema puede generar códigos de barras automáticamente y permite imprimir etiquetas para los productos, facilitando su identificación y escaneo en el punto de venta.

Además, el módulo de stock se integra con el sistema de ventas y el módulo de notificaciones, de modo que cualquier actualización en el inventario se refleje en tiempo real en todos los sistemas. Esto asegura que las promociones, ventas y alertas de reabastecimiento estén siempre actualizadas, optimizando la operatividad y reduciendo el riesgo de quiebres de stock o exceso de inventario.

3.3 Módulo de promociones

El módulo de promociones constituye la pieza de fidelización del sistema. Su propósito es premiar la lealtad de los clientes mediante un esquema de puntos acumulables y canjeables por beneficios predefinidos. Al igual que el módulo de notificaciones, la lógica se separa en capas para simplificar la administración y garantizar una experiencia fluida para el usuario final.

3.3.1 Sistema de acumulación y canje de puntos

Cada compra realizada genera una cantidad de puntos proporcional al importe abonado. Estos puntos se almacenan en el perfil del cliente y pueden ser usados para **"**comprar**"** promociones.

- **Costo en puntos:** al crear la promoción, el administrador define cuántos puntos se necesitan para acceder a ella.
- **Período de vigencia:** una vez canjeada, la promoción posee un plazo finito (en días) durante el cual puede ser aplicada.
- **Uso único:** tras aplicarse satisfactoriamente en una compra, la promoción se marca como consumida y ya no puede reutilizarse.

3.3.2 Catálogo de tipos de promoción

El sistema dispone de seis plantillas base, cada una registrada con su lógica de aplicación mediante un esquema JSON. De esta forma se cubre la mayoría de las tácticas comerciales sin necesidad de reescribir código.

1	**Descuento general**	Aplica un porcentaje uniforme de rebaja sobre todos los productos incluidos en la compra.
2	**Promoción X por Y**	Mecánica “3×2”, “2×1”, etc. El cliente añade *X* unidades de un producto y abona solo *Y*.
3	**Descuento en producto específico**	Concede un porcentaje de descuento a todas las unidades de un artículo concreto al superar una cantidad mínima.
4	**Regalo por compra de producto**	Al comprar una cantidad determinada del producto *A*, el cliente recibe uno o más artículos *B* sin costo.
5	**Descuento por paquete de productos**	Se arma un combo (por ejemplo, 2 × *A* + 1 × *B*) y se aplica un descuento global sobre el precio total del paquete.
6	**Regalo por paquete de productos**	Al adquirir un combo específico (p. ej., 2 × *A* + 1 × *B*), se entrega uno o más artículos *C* de regalo.

3.3.3 Proceso de creación de promociones (Administrador)

1. **Selección de plantilla:** el panel presenta las seis opciones descritas.

2. **Configuración de parámetros:** fechas de inicio y fin, puntos necesarios, envío gratis, descripción, etc.
3. **Publicación:** una vez guardada, la promoción queda disponible para ser canjeada por los clientes que dispongan de puntos suficientes.

3.3.4 Aplicación de promociones (Cliente)

1. **Visualización y canje:** el cliente revisa sus puntos y "compra" la promoción deseada.
2. **Aplicar:** la promoción pasa a estar disponible para aplicar en el carrito.
3. **Motor de aplicación:** cuando el cliente decide aplicarla al carrito, el sistema:
    - Añade los productos requeridos o ajusta los existentes.
    - Recalcula precios y descuentos para reflejar la oferta de forma clara y visual.
    - Verifica stock en tiempo real; si es insuficiente, muestra un mensaje y cancela la operación.
4. **Consumo definitivo:** al completar la compra, la promoción queda invalidada para futuros usos.

Con este diseño, el módulo de promociones ofrece un marco flexible y robusto que permite a la empresa desplegar campañas comerciales variadas sin intervención técnica adicional, maximizando la retención de clientes y optimizando la gestión interna.

3.4 Módulo de notificaciones

El módulo de notificaciones es una pieza clave del sistema, encargado de gestionar tanto las notificaciones programadas como las automáticas. Las notificaciones programadas se crean con antelación y pueden dirigirse tanto a clientes como a

empleados. Las notificaciones automáticas, por otro lado, se generan de forma instantánea en función de ciertas interacciones o condiciones dentro del sistema.

3.4.1 Sistema Notificaciones programadas

Estas notificaciones son versátiles y se pueden utilizar para una amplia gama de propósitos, como recordatorios de eventos, anuncios de descuentos, promociones, o avisos de actividades internas para los empleados. Estas pueden configurarse con diferentes periodicidades: diarias, semanales, mensuales, anuales o únicas. Esto permite una gran flexibilidad para adaptarse a las necesidades específicas de comunicación de la organización.

3.4.2 Sistema Notificaciones automáticas

Entre las notificaciones automáticas se encuentran:

- **Notificaciones de Delivery**: Informan al cliente sobre el estado actual de un delivery, actualizándose en tiempo real a medida que el estado del pedido cambia. Esto asegura que el comprador esté siempre al tanto del progreso del pedido, mejorando la experiencia del cliente y facilitando la gestión interna.
- **Notificaciones de Cumpleaños de Mascotas**: En el caso de clientes que registran a sus mascotas, el sistema envía una notificación automática en el cumpleaños de la mascota, incentivando la compra de regalos o productos especiales para ese día.
- **Notificaciones de Stock**: Se generan cuando el nivel de inventario de un producto alcanza ciertos umbrales críticos (alto, medio, bajo), permitiendo a la organización tomar acciones proactivas para evitar desabastecimientos.

3.4.2.1 Sub-Sistema de Predicción de necesidad

Este subsistema avanzado utiliza un algoritmo que predice cuándo un cliente podría quedarse sin ciertos productos de compra recurrente. Basándose en el historial de compras del cliente, el sistema calcula el promedio de tiempo entre compras y envía una notificación unos días antes de la fecha estimada de recompra, ayudando al cliente a reabastecerse a tiempo. Este sistema mejora su precisión con cada compra, ajustando la frecuencia de las notificaciones en función del comportamiento real del cliente, y asegurando que la experiencia de compra sea más fluida y proactiva.

3.5 Módulo de ventas

El módulo de ventas es el corazón del sistema a la hora de procesar transacciones comerciales. Sus principales características son:

- **Gestión de Carrito de Compras:**
  Permite tomar productos directamente del módulo de stock y agregarlos a un "carrito" virtual. Una vez que el usuario escanea o selecciona un producto, el sistema calcula automáticamente el precio unitario, la cantidad disponible y el importe total de la línea:
  1. **Escaneo de códigos de barras**: si la organización utiliza escáneres, basta con pasar el código de barras sobre el lector para que el producto se agregue al carrito sin necesidad de tipear nada manualmente. Esto reduce errores de registro y acelera el proceso de venta.
  2. **Búsqueda manual**: en caso de no contar con escáner, el usuario puede buscar el producto por nombre o categoría externa/interna y agregarlo al carrito.

- **Cálculo Automático de Totales y Promociones:**
  Una vez agrupados los productos en el carrito, el sistema:
  1. Aplica automáticamente descuentos o recargos según promociones activas (por ejemplo, 10 % de descuento por pago en efectivo).

2. Suma el importe total y presenta un resumen claro con subtotales y monto final.

- **Manejo de Métodos de Pago**
  Por defecto, el módulo está integrado con la pasarela de pago de MercadoPago. Desde la misma interfaz de venta, el cliente puede elegir entre distintos métodos:

  1. **MercadoPago** (tarjeta de crédito/débito, efectivo a través de puntos de cobro asociados, saldo de MercadoPago, etc.).
  2. **Pago en efectivo**: si el cliente opta por efectivo, el administrador puede configurar descuentos (por ejemplo, 10 % de descuento automático) o recargos según corresponda.
  3. **Transferencia bancaria/otros**: existe un campo libre para incluir otros métodos de pago no contemplados, con recargos o descuentos configurables desde el módulo de stock o administración.

3.5.1 Sub-Módulo de historial

El sub-módulo de historial es opcional para clientes que consideren que el costo adicional de almacenamiento de datos no es rentable. Si está habilitado, ofrece:

- **Vista Cronológica de Ventas**
  Permite listar todas las ventas realizadas, ordenadas por fecha y hora, con filtros por rango de fechas (día, semana, mes, año) o por empleado que registró la venta.

- **Agrupamiento y Filtrado Avanzado**
  1. **Por cliente**: muestra todas las compras realizadas por un mismo cliente, permitiendo analizar su comportamiento de compra a lo largo del tiempo.
  2. **Por producto**: agrupa transacciones donde aparece un producto específico, ayudando a identificar rotación y tendencias.
  3. **Por fecha**: filtra ventas según una fecha o rango de fechas.
  4. **Por rangos de importe**: permite ver cuántas ventas superaron cierta cifra y revisar los tickets correspondientes.

- **Métricas y Reportes**
  El sistema calcula automáticamente indicadores clave basados en las ventas almacenadas:
  1. **Volumen de ventas por día/mes/año**: para entender temporadas altas y bajas.
  2. **Producto con mayor y menor rotación**: identifica cuáles son los artículos más vendidos y los que permanecen más tiempo en stock.
  3. **Ingresos totales y promedio por venta**: aporta una visión financiera clara para tomar decisiones de precios o promociones.
  4. **Comisiones pagadas a gateways de pago**: muestra cuánto se destinó a comisiones de MercadoPago, u otros, según se requiera.

- **Privacidad y Costos de Almacenamiento**
  Dado que guardar cada ticket puede generar un volumen significativo de datos, este submódulo se activa solamente si el cliente lo solicita. Cuando está desactivado, el sistema sigue procesando ventas normalmente, pero no almacena los detalles históricos en la base de datos, reduciendo costos de almacenamiento y mejorando el rendimiento general.

3.6 Módulo de delivery

Este permite la gestión de los envíos, visualizar y administrar todos los deliveries, tanto los que están en curso como los históricos. Esto facilita el seguimiento de cada entrega y la actualización de su estado en tiempo real, brindando transparencia y control sobre el proceso de distribución.

3.7 Módulo de administración

El módulo de administración funciona como el panel central de control del sistema.

3.7.1 Sistema gestion general

Desde este panel, los administradores pueden establecer configuraciones generales, como la cantidad de puntos otorgados por cada venta, que luego pueden ser canjeados por promociones. También permite configurar el costo de los deliveries, que se establece como una tarifa fija independientemente de la distancia, así como definir un umbral de compra a partir del cual el delivery es gratuito. Además, este módulo incluye la gestión del sistema de puntos, que se puede configurar en función de la cantidad de dinero gastado por los clientes, ya sea en moneda local o en dólares, integrando así un sistema de conversión de divisas.

3.7.2 Sistema estadístico

El sistema estadístico constituye la capa analítica de la solución y tiene como objetivo brindar soporte a la toma de decisiones mediante la visualización de indicadores clave del negocio. A través de este módulo es posible consultar métricas como la cantidad de ventas, ganancias, promociones más canjeadas, productos con mayor y menor rotación, frecuencia de renovación de stock y otros indicadores relevantes para el seguimiento comercial y operativo.

Para su funcionamiento, el sistema estadístico no consulta directamente la base de datos transaccional utilizada por la operatoria diaria, sino que se apoya en una arquitectura analítica desacoplada. Los datos generados por el sistema principal son extraídos, transformados y cargados mediante procesos ETL hacia un Data Warehouse independiente, preparado específicamente para consultas históricas y analíticas. Esta separación permite organizar la información de manera más adecuada para su explotación estadística y evitar sobrecargar la infraestructura operativa con consultas complejas.

Sobre esta capa analítica se implementó un dashboard estadístico utilizando Metabase, herramienta que permite construir visualizaciones e indicadores de forma

clara e intuitiva. A través de este dashboard, los responsables del negocio pueden analizar información consolidada y comparativa, obteniendo una visión más amplia del desempeño general del sistema y del comportamiento de variables relevantes a lo largo del tiempo.

Asimismo, una de las decisiones de diseño más importantes de este módulo fue desacoplar su disponibilidad de la capa transaccional principal. De este modo, el panel estadístico puede continuar siendo consultado aun cuando el backend principal o la base de datos operativa no se encuentren disponibles temporalmente, garantizando así una mayor robustez para la consulta de información gerencial.

En este sentido, el sistema estadístico no solo amplía las capacidades funcionales de la solución, sino que incorpora una perspectiva analítica e histórica que complementa la operatoria diaria del negocio con herramientas de monitoreo, interpretación y apoyo a decisiones.

3.7.3 Sistema de gestión de usuarios

Permite crear, editar o eliminar cuentas de empleados y administradores del sistema.

3.8 Modulo de clientes

Permite registrar y gestionar la información personal de los clientes, así como crear cuentas de usuario para que puedan acceder a la aplicación web. Además, permite gestionar la información de contacto (como dirección y correo electrónico) y establecer una contraseña por defecto para el acceso a la plataforma, lo cual también puede ser útil para la recuperación de contraseñas.

3.8.1 Sub-Módulo de mascotas

Este módulo incluye un submódulo de mascotas, que permite registrar la información de las mascotas de cada cliente. Esto es especialmente útil para negocios que trabajan con animales, como forrajerías, y permite almacenar datos relevantes como nombre, fecha de nacimiento, y preferencias específicas.

3.9 Sub-módulo de reportes

El sub-módulo de reportes permite exportar información operativa del sistema a hojas de cálculo para su consulta externa, análisis puntual o almacenamiento administrativo. Puede aplicarse en distintos módulos, como stock, historial de ventas y delivery, según el tipo de información que se desee relevar.

Su implementación se apoya en servicios de Google Cloud Platform, Google Drive, Google Sheets y Google Apps Script, permitiendo generar reportes en formato de hoja de cálculo a partir de datos del sistema. De este modo, es posible exportar, por ejemplo, reportes de stock para visualizar productos disponibles, cantidades, proveedores y otros atributos relevantes, así como también reportes de ventas orientados a revisiones administrativas o estudios específicos sobre la operatoria comercial.

A diferencia del sistema estadístico, cuyo objetivo es brindar una capa analítica e histórica mediante dashboards construidos sobre un Data Warehouse, este sub-módulo se orienta principalmente a la **exportación de información operativa**. En otras palabras, mientras el sistema estadístico permite consultar indicadores consolidados y métricas visuales desde un entorno analítico, el sub-módulo de reportes permite generar archivos de apoyo en formato tabular para su revisión, uso administrativo o procesamiento externo.

De esta manera, ambas soluciones cumplen funciones complementarias dentro del sistema: por un lado, los reportes exportables facilitan la obtención de datos operativos en hojas de cálculo; por otro, la capa analítica basada en ETL, Data Warehouse y Metabase permite interpretar la información del negocio mediante dashboards estadísticos e indicadores históricos.

4 Despliegue

4.1 Infraestructura y topología

El ecosistema productivo se construyó deliberadamente sobre servicios PaaS y SaaS con el objetivo de minimizar la carga operativa y permitir que el equipo de Club Galo se concentre en la evolución funcional del negocio, y no en la administración de servidores.

Cloudflare actúa como puerta de entrada global para la capa web: gestiona DNS, certifica HTTPS, aplica protección básica frente a ataques y contribuye a reducir la latencia de acceso para los usuarios externos. A través de esta capa ingresan las peticiones dirigidas a la Web App y a la PWA utilizadas por clientes y repartidores.

La capa transaccional principal se encuentra desplegada en Heroku y concentra la operatoria diaria del sistema. Allí se alojan la aplicación web, el backend desarrollado en Node.js con Express y Sequelize, y la base de datos operativa PostgreSQL. Esta infraestructura centraliza la lógica de negocio asociada a ventas, productos, stock, pedidos, promociones, notificaciones y administración general. Asimismo, desde el backend se gestionan las integraciones con Google Apps Script para la generación de reportes operativos en Google Sheets/Drive y con AWS S3 para el almacenamiento de imágenes y otros archivos del sistema.

Además de esta capa operativa, la solución final incorpora una infraestructura analítica desacoplada para el módulo estadístico. Esta capa se compone de un

módulo BI desplegado en AWS, que incluye Metabase como herramienta de visualización, un API Gateway para el acceso al panel y funciones Lambda utilizadas para autenticación, generación de URLs embebidas y ejecución del proceso ETL. A su vez, la información consolidada se almacena en un Data Warehouse separado de la base transaccional, permitiendo consultas analíticas e históricas sin impactar directamente sobre la operatoria principal del sistema.

Esta separación entre infraestructura transaccional e infraestructura analítica constituye una de las decisiones más importantes de la arquitectura adoptada. Gracias a ella, el panel estadístico puede continuar siendo consultado aun cuando el backend principal o la base de datos operativa no se encuentren disponibles temporalmente, mejorando la robustez general de la solución para fines de consulta gerencial.

La topología final del sistema queda entonces compuesta por una capa de acceso web protegida por Cloudflare, una capa transaccional principal desplegada en Heroku, una capa de almacenamiento de archivos en AWS S3, una integración externa con Google Apps Script para reportes operativos y una capa analítica desacoplada basada en AWS y un Data Warehouse independiente. De este modo, la arquitectura separa adecuadamente las responsabilidades operativas, documentales y analíticas del sistema, facilitando la escalabilidad, el mantenimiento y la continuidad de acceso a la información relevante del negocio.

4.2 Ciclo de vida de desarrollo y CI/CD

El proceso de integración y despliegue continuo se orquesta con GitHub Actions y se basa en un modelo container-first. Cada push a la rama principal de los

repositorios del frontend (WebApp/PWA) y del backend inicia una ejecución automática cuyo objetivo es construir la imagen de contenedor, publicarla en el registro y promoverla a producción. El tiempo end-to-end es de ~2 minutos, de modo que la distancia entre “el código en Git” y “el sistema productivo” es mínimo. La base de datos corre como Heroku Postgres (servicio gestionado) y no forma parte del pipeline de construcción; se aprovisiona una vez y se consume a través del host provisto por la plataforma.

Flujo de CI/CD:

1. **Disparador del pipeline:** La acción se ejecuta automáticamente ante un push en main (o mediante ejecución manual para hotfixes). Se hace checkout del código y se restauran cachés de compilación para acelerar la construcción.
2. **Autenticación con los Registros:** El runner se autentica programáticamente contra los registros de contenedores de Docker y la API de la plataforma de Heroku, utilizando credenciales almacenadas como GitHub Secrets. Con ello obtiene permisos para publicar imágenes y operar sobre releases.
3. **Construcción y etiquetado de la imagen:** Se compila la aplicación y se construye la imagen Docker correspondiente. La imagen se etiqueta con el número de build, el commit y la rama, garantizando trazabilidad entre artefactos y código fuente.
4. **Publicación en el registro:** La imagen etiquetada se pushea al registro destino. Este paso deja disponible el artefacto inmutable que servirá como base del despliegue.
5. **Creación de release en la plataforma:** La plataforma coloca la imagen publicada en una release nueva del sistema. Esta operación versiona el despliegue y lo añade al historial, habilitando rollback inmediato a cualquier versión previa.
6. **Puesta en ejecución del proceso web:** Se escala el proceso web a un dyno y se valida que el contenedor quede en estado up. Las variables de entorno

requeridas (claves JWT, credenciales de MercadoPago, endpoints de S3, etc.) se inyectan desde la configuración de la plataforma.

Este ciclo aplica para los repositorios del frontend (WebApp/PWA) y para el backend; la base de datos por su parte fue desplegada mediante un Service de Heroku que se configura en un contenedor, el cual provee un host estable para las aplicaciones.

4.3 Gestión de datos y seguridad

La gestión de datos del sistema se organiza en dos grandes capas: una capa transaccional, utilizada por la operatoria diaria del negocio, y una capa analítica, destinada a la consulta histórica y estadística. La primera se apoya en la base de datos principal PostgreSQL desplegada junto al backend operativo, mientras que la segunda utiliza un Data Warehouse separado, alimentado mediante procesos ETL y consumido por el módulo BI. Esta separación permite proteger la operatoria principal frente a consultas analíticas intensivas y, al mismo tiempo, mejorar la disponibilidad del panel estadístico.

En materia de resguardo de información, la base de datos operativa cuenta con mecanismos automáticos de backup provistos por la plataforma de despliegue, complementados con exportaciones periódicas para disponer de copias de seguridad adicionales. Asimismo, el almacenamiento de archivos en AWS S3 se administra mediante políticas de acceso restringido y versionado de objetos, reduciendo el riesgo de pérdida accidental de recursos estáticos relevantes para la operación del sistema.

En cuanto a la seguridad de acceso, la API principal utiliza autenticación basada en JWT, con refresh token persistido y validaciones de acceso según rol. También se aplican configuraciones restrictivas de CORS y mecanismos específicos de autenticación para integraciones externas, como el procesamiento de pagos y

webhooks. De este modo, se controla el acceso a los recursos críticos del sistema y se reduce la superficie de exposición ante solicitudes no autorizadas.

Por su parte, el módulo analítico se encuentra desacoplado de la autenticación del backend principal. El acceso al panel estadístico se resuelve mediante un mecanismo independiente dentro de la capa BI, lo que permite mantener la consulta de dashboards aun cuando la infraestructura transaccional principal se encuentre fuera de servicio. Esta decisión no solo mejora la disponibilidad del módulo estadístico, sino que también separa claramente los contextos de acceso operativo y analítico.

En conjunto, la estrategia de gestión de datos y seguridad busca preservar la integridad, disponibilidad y confidencialidad de la información, diferenciando adecuadamente los componentes transaccionales, documentales y analíticos del sistema, y aplicando controles acordes a la función de cada uno.

4.4 Costos operativos y escalabilidad

La solución mantiene una estrategia de costos reducidos en su capa transaccional principal, priorizando servicios administrados y esquemas de pago por uso. Esto permite sostener la operación diaria con una inversión mensual acotada y escalar gradualmente a medida que crece la demanda del negocio.

Componente	Plan actual	Coste (USD/mes)	Escalabilidad
Heroku Dynos (x3)	3 dynos Basic	21,00	Escalable a dynos Standard o mayor cantidad de dynos sin cambios estructurales de código

Heroku Postgres	Essential-0	5,00	Escalable a planes superiores si crece el volumen transaccional
S3 + CloudFront	Uso medido	5	Escalamiento automático según almacenamiento y transferencia
Dominio	.com.ar	0,83 (10 USD/año)	Costo fijo anual
AWS capa BI	EC2 Metabase, EBS, IPv4 pública, Lambda, API Gateway y CloudWatch	30,00	Escalable aumentando la instancia, ajustando logs o separando componentes
Supabase Data Warehouse	Plan pago inicial	25,00	Escalable por almacenamiento, cómputo y transferencia

Supabase Testing DB	Ambiente no productivo / plan gratuito inicial	0,00	Puede migrarse a plan pago si el ambiente de QA lo requiere
Total mensual	—	**≈ 86,83 USD**	—

En escenarios de venta navideña o campañas comerciales, se prevé escalar temporalmente la capa transaccional mediante más dynos o dynos de mayor capacidad. Dado que este escalamiento sería puntual y limitado a períodos cortos, el impacto mensual seguiría siendo acotado en relación con el beneficio operativo esperado.

Capa analítica desacoplada

Como ampliación de la solución original, se incorporó una capa analítica independiente para el módulo estadístico, compuesta por Metabase, procesos ETL, funciones Lambda, API Gateway y un Data Warehouse separado. En el estado actual, esta infraestructura se mantiene con un costo acotado, dado que el volumen de uso del panel es reducido y la mayor parte del consumo variable resulta muy bajo.

5 Plan de pruebas

5.1 Estrategia y alcance del plan de pruebas

El plan de pruebas del sistema se definió con el objetivo de validar el correcto funcionamiento de los módulos principales de la solución, verificar la integración entre sus componentes y comprobar que los flujos implementados respondieran a los requerimientos funcionales relevados. Para ello, se adoptó una estrategia combinada que incluyó pruebas manuales funcionales, pruebas de integración entre módulos, pruebas de aceptación con el usuario y una capa inicial de pruebas automatizadas end-to-end sobre el backend principal.

Las pruebas manuales funcionales se aplicaron sobre los módulos más relevantes del sistema, incluyendo ventas, stock, promociones, clientes, notificaciones, delivery, administración y reportes. En cada caso se verificaron los flujos principales de uso, la correcta persistencia de la información, la actualización de estados y la respuesta esperada de la interfaz ante operaciones válidas o situaciones de error. Estas pruebas permitieron validar el comportamiento del sistema desde la perspectiva del usuario final y detectar ajustes funcionales durante el proceso de desarrollo.

Las pruebas de integración se orientaron a verificar la interacción entre componentes que dependen entre sí, como la relación entre ventas y stock, la aplicación de promociones en el carrito, la generación de notificaciones ante eventos del sistema, la actualización del estado de entregas y la comunicación entre frontend, backend y base de datos. También se contemplaron integraciones externas relevantes, tales como el procesamiento de pagos mediante MercadoPago, la generación de reportes exportables y el almacenamiento de recursos digitales.

De forma complementaria, se incorporó una capa básica de pruebas automatizadas end-to-end sobre el backend principal. Estas pruebas consistieron en invocar endpoints relevantes, validar las respuestas obtenidas y comprobar la correcta persistencia, modificación o recuperación de datos en la base correspondiente. Si

bien esta cobertura automatizada no abarca la totalidad del sistema, constituye una base inicial de aseguramiento técnico sobre operaciones críticas de la API y su integración con la capa de datos.

Las pruebas de aceptación de usuario se realizaron con foco en los flujos operativos más importantes para Club Galo. En estas instancias, el usuario validó funcionalidades vinculadas a ventas, inventario, delivery, promociones y administración general, aportando observaciones sobre comportamiento, usabilidad y ajustes requeridos. Las incidencias detectadas durante el proceso fueron registradas como tareas en Jira, permitiendo su seguimiento, priorización y corrección dentro de las iteraciones de trabajo.

El criterio general de aceptación estableció que una funcionalidad se consideraba validada cuando cumplía el requerimiento asociado, permitía completar el flujo principal sin errores bloqueantes, persistía correctamente la información involucrada y no generaba inconsistencias en módulos relacionados. De esta manera, el plan de pruebas permitió combinar una validación funcional desde la experiencia de uso con una verificación técnica inicial sobre la integración entre API, base de datos y servicios externos.

5.2 Matriz de pruebas manuales

Área	Caso	Datos clave	Resultado
Ventas	PM-V-01 — Checkout con promo + QR	3 ítems, 1 promo %	Factura registrada, link válido, webhook ‘paid’
Stock	PM-S-02 — Alerta de stock bajo	Restar unidades hasta umbral	Notificación interna en < 2 s

Fidelización	PM-F-03 — Puntos con multiplicador	Ticket 50.000 ARS	+100 pts con mult ×2
Notificaciones	PM-N-04 — Predicción de reposición	Compra periódica cada 30 d	Push 3 d antes
Delivery	PM-D-05 — Cambio de estado	Estado ‘En camino’ → ‘Entregado’	Cliente y user actualizados en vivo

Cada ciclo de pruebas dura ~90 min. Se documentan desvíos en Jira con nuevas tarjetas con la etiqueta “error”.

5.3 Procedimiento de ejecución

El procedimiento de ejecución de pruebas se estructuró en distintas etapas, combinando validaciones manuales y automatizadas. En primer lugar, se desplegaba la rama de desarrollo correspondiente al entorno de prueba. Luego, se refrescaba la base de datos con información dummy o de prueba, de manera de contar con un estado controlado y reproducible para la ejecución de los distintos casos.

Sobre esta base, se ejecutaba la matriz de Pruebas Manuales Exhaustivas (PME) en un orden predefinido, procurando minimizar el impacto de estados previos entre casos de prueba. Esta secuencia permitía validar de forma progresiva los distintos módulos funcionales del sistema, verificando tanto el comportamiento esperado como la interacción entre componentes.

De forma complementaria, para el backend principal se incorporó una ejecución de pruebas automatizadas end-to-end (E2E). Estas pruebas consistían en invocar los endpoints correspondientes, validar la respuesta obtenida y comprobar posteriormente la persistencia, actualización o recuperación correcta de los datos en la base de datos. De esta manera, el procedimiento no solo contempló la validación funcional manual del sistema, sino también una instancia automatizada de verificación técnica sobre la integración entre la API y la capa de datos.

Finalmente, al cierre de cada ciclo de trabajo se realizaban pruebas de aceptación con foco en la experiencia de uso, permitiendo contrastar el comportamiento implementado con los requerimientos esperados para cada funcionalidad.

5.4 Resultados y hallazgos

Durante la etapa beta se identificaron distintos incidentes funcionales y de comportamiento, la mayoría de los cuales fueron resueltos dentro de la misma iteración de trabajo. Entre los hallazgos más relevantes se encontraron ajustes de integración, mejoras de comportamiento en la interfaz y correcciones vinculadas a flujos específicos de uso, quedando únicamente algunos refinamientos menores planificados para versiones posteriores.

En paralelo, la incorporación de pruebas automatizadas end-to-end sobre el backend permitió validar una capa adicional de calidad técnica, verificando el correcto funcionamiento de los endpoints y su interacción con la base de datos. Estas pruebas sirvieron como una instancia inicial de aseguramiento automatizado, útil para detectar errores de integración y confirmar que las operaciones críticas de lectura, escritura y actualización se ejecutaran de forma consistente.

En términos generales, los resultados obtenidos muestran que la estrategia combinada de pruebas manuales, aceptación de usuario y validaciones automatizadas permitió fortalecer la confiabilidad de la solución, tanto desde la

perspectiva funcional como desde la perspectiva técnica. No obstante, también se reconoce que la cobertura automatizada implementada constituye una base inicial, susceptible de ampliarse en futuras iteraciones mediante más casos de prueba y una mayor profundidad de validación.

5.5 Pruebas de aceptación (UAT)

Se realizaron cuatro sesiones presenciales (enero–abril 2025) donde el cliente operó la Desktop App en días hábiles y brindó feedback oral. Entre los ajustes destacaron: cambio del sistema de aumentos de precios por porcentaje de ganancias, reordenamiento de botones en modal de delivery.

6 Metodología de trabajo

6.1 Organización y roles

Felipe Arenillas asumió la doble función de Product Owner y Scrum Master, gestionando el backlog, priorizando historias y canalizando el feedback del cliente. El Equipo de Desarrollo lo integraron Felipe Arenillas, Octavio Garcia y Julián Gergolet Albornoz, responsables del análisis, el diseño, la codificación y las pruebas manuales. El dueño de Club Galo intervino como stakeholder principal, validando funcionalidades y proponiendo ajustes. Los profesores Federico Porrini, Ignacio Carreño y Juan Vulcano actuaron como asesores académicos, revisando decisiones de arquitectura y buenas prácticas.

6.2 Ceremonias y artefactos

El proyecto se condujo con SCRUM durante veintidós sprints quincenales, desde abril de 2024 hasta mayo de 2025. Cada ciclo comenzó con una reunión de

planificación, continuó con una breve coordinación diaria de quince minutos —informal, ya que el equipo trabajaba de manera presencial— y concluyó con una revisión conjunta del incremento y una retrospectiva ligera. Al cierre de cada sprint el incremento quedaba disponible en un entorno de staging para ser evaluado por los profesores tutores y por el dueño de Club Galo.

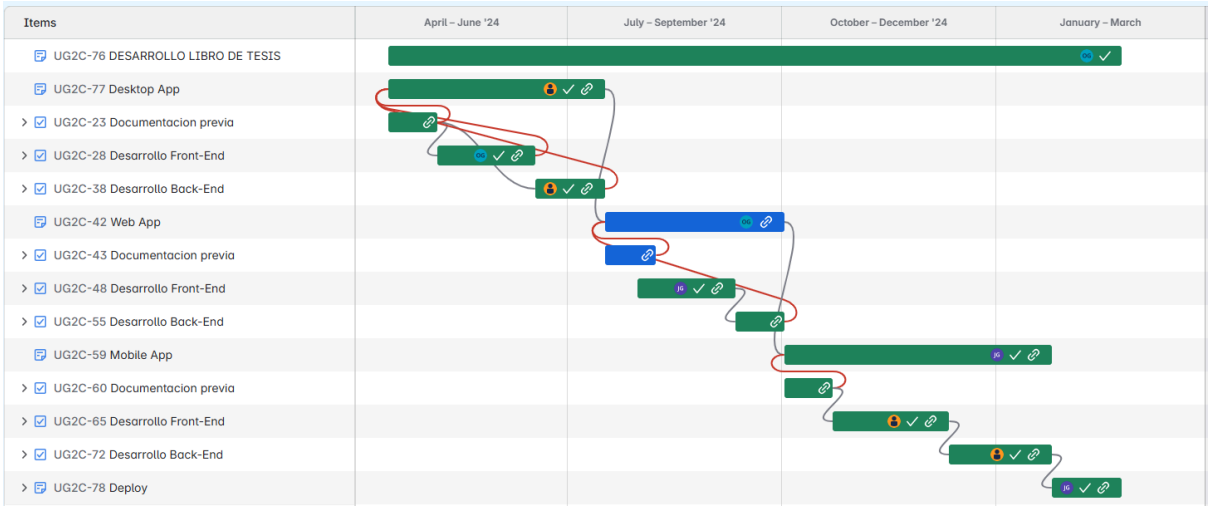
El Product Backlog se refinó cada dos sprints; de él surgía un Sprint Backlog que se volcaba a Jira al inicio del ciclo. La Definition of Done exigía código revisado por pares, checklist de QA manual y despliegue exitoso en staging.

6.3 Herramientas de apoyo

La planificación y el seguimiento se realizaron en Jira, complementado con Confluence para documentar decisiones técnicas y con GitHub Actions para la integración y el despliegue continuos en Heroku. El uso de estas herramientas permitió mantener trazabilidad sin añadir complejidad innecesaria al flujo de trabajo.

6.4 Cronograma

El diagrama de Gantt muestra la secuencia y los plazos de las distintas etapas del desarrollo del sistema, dividiendo el proyecto en tres grandes bloques: la aplicación de escritorio (Desktop App), la aplicación web (Web App) y la aplicación móvil tipo PWA (Mobile App). Cada bloque incluye una fase inicial de documentación previa, seguida del desarrollo del Front-End y del Back-End, con vínculos de dependencia que aseguran el flujo correcto entre tareas. Las barras de color se distribuyen a lo largo de los cuatrimestres de 2024, reflejando la planificación desde abril hasta marzo de 2025; garantizando la coordinación temporal de todos los componentes.



6.5 Gestión de riesgos y calidad

La matriz de riesgos se revisó al término de cada sprint. Los tres focos más críticos fueron el cambio de React Native a PWA, la migración de MySQL a PostgreSQL y la decisión de sustituir un backend separado por aplicación por un backend común que centralizará la lógica para desktop, web y PWA. Cada riesgo se mitigó con prototipos y pruebas piloto tempranas que permitieron validar compatibilidad y rendimiento antes de avanzar. La calidad se garantiza con revisiones obligatorias de pull requests y con pruebas exploratorias antes de fusionar código a la rama principal, asegurando que cada incremento cumpliera los criterios de aceptación acordados.

Mejoras a futuro

El roadmap prioriza impacto visible al usuario y retorno de inversión antes que refactorizaciones internas, manteniendo ciclos de entrega ≤ 8 semanas.

N.º	Hito	Descripción	Valor / Beneficio	Complejidad	Depende de
1	Push Notifications	Web Push (Service Worker) + FCM fallback	Comunicación proactiva, retención	Media	Establecer FCM keys
2	Acciones en notificaciones	Deep-link al producto / pedido	Conversión directa desde el aviso	Baja	1
3	Recomendador ML	K-NN sobre historial de ventas	+Ticket medio; cross-sell	Alta	Datos > 6 meses
4	Facturación ARCA	WSAA/WSFE, CAE online	Cumpl. fiscal, menos tareas manuales	Alta	OK backend
5	Desktop App Web	Portar Electron a PWA standalone	Menor mantenimiento + Familiaridad de los usuarios	Media	1,2
6	Multicaja	Registrar pagos por caja	Control contable fino	Media	—

7	Multisucursal 2.0	Stock y KPI segregados	Escalar modelo SaaS	Alta	6
8	Apps nativas	Capacitor + store listing	Mayor alcance móvil	Media	1,2
X	AWS completo	Evaluar ahorro vs Heroku	—	Baja	Costo ↑
X	Rolldown	Migrar Vite/Rollup	Build más veloz	Baja	Estable 1.0

Beneficios Post-Implementación

1 Reducción de costos operativos

- **Conteo manual de Inventario:** 192 h/año liberadas (2 empleados × 8 h × 12 meses), a 5 USD/h ⇒ ≈ 960 USD anuales. Esto equivale a tiempo que antes se destinaba a tareas manuales de gestión de inventario y que ahora puede reasignarse a ventas, atención al cliente u otras actividades de mayor valor.
- **Pérdida de ventas por quiebre de stock:** al registrar cantidades en línea se evita ~3 % de faltantes (estimado por Club Galo), con un promedio de 2000 clientes mensuales, y un promedio de ticket de ~23 USD, lo que nos da un estimado de 1380 USD en pérdida de ingreso bruto por falta de stock al mes,

con una ganancia promedio del 40% por producto nos da que el ingreso neto perdido por quiebre stock es de 552 USD/mes.

- **Gastos por sobre-compra de stock:** El equipo de Club Galo estima que por año aproximadamente están gastando de más en stock que no se vende, que se rompe o incluso se vence ~762 USD

2 Incremento de ingresos

La meta interna es capturar 15 % de las transacciones vía canal online; con ticket medio de 23 USD, cada venta adicional cubre ~5× el costo mensual de la nube.

3 Eficiencia y calidad de vida

Los empleados concentran tiempo en asesoramiento técnico, reducen tareas repetitivas y mejoran su satisfacción laboral (encuesta informal 4,6/5 tras la beta). El soporte 12×7 garantiza autonomía post-venta.

4 Impacto Económico

4.1 Modelo de ROI

En este análisis, se consideran beneficios todos los costos evitados tras la implementación: horas liberadas, ventas recuperadas por evitar quiebres de stock y reducción del sobre-stock. El período considerado es anual y los montos están expresados en USD. Como referencia cambiaria informativa, se utiliza 1 USD = 1.420 ARS.

- **Ahorro en costos operativos (inventario):** El conteo manual insumía 192 h/año (2 personas × 8 h × 12 meses). Valorizado a 5 USD/h, el costo evitado anual es de 960 USD.
- **Ingresos netos recuperados por quiebre de stock:** Con 2.000 clientes/mes y ticket promedio de 23 USD, la venta bruta mensual es 46.000 USD. La tasa de faltantes estimada por el cliente es 3 %, equivalente a 1.380 USD/mes

brutos. Aplicando margen neto del 40 %, el ingreso neto perdido era 552 USD/mes, es decir 6.624 USD/año.

- **Ahorro por sobre-stock:** El sobre-stock histórico se estima en 762 USD/año. Para ser prudentes, se computa como logro del sistema sólo una reducción del 30 %, es decir 228,60 USD/año.

Ahora, considerando la inversión necesaria para operar y mantener el sistema:

- **Infraestructura y hosting**: Incluye Heroku, AWS, Supabase, almacenamiento, dominio y componentes asociados a la capa transaccional y analítica. Se estima un costo mensual inicial de **87 USD**, equivalente a **1.044 USD/año**.
- **Licencia/mantenimiento**: 30 USD/mes → 360 USD/año.
- **Total Inversión** (I) = 1.404 USD/año.

Concepto	Anual (USD)
Ahorro en costos operativos	960.00
Perdida ingresos por quiebre de stock 552USD/mes	6624.00

Ahorro por sobre stock (~30% del sobre stock total)	228.60
Beneficio total (BT)	**7812.60**
Infraestructura y hosting, 87 USD/mes	1.044,00
Licencia de uso y mantenimiento del sistema Club Galo (30 USD/mes)	360.00
Inversión anual total (I)	**1.404,00**
ROI neto	**4,56**

En términos de payback, los beneficios mensuales aproximados —(960/12) + 552 + (228,60/12)— suman aproximadamente 651 USD/mes. Considerando una inversión anual de 1.404 USD, el período de recuperación estimado es cercano a 2,2 meses.

Las hipótesis empleadas son conservadoras: el ahorro por sobre-stock solo computa un 30 % de mejora, la pérdida por faltantes usa el 3 % reportado por el cliente, el margen neto asumido es del 40 % y la infraestructura se calcula incluyendo tanto la capa transaccional como la nueva capa analítica desacoplada.

5 Impacto Social

- **Capital humano**: capacitación formal (2 jornadas × 3 h) y soporte extendido mejoran las competencias digitales de la plantilla.
- **Experiencia del cliente**: omnicanal, pagos QR y programa de puntos fomentan lealtad y reducen barreras de compra.
- **Ecosistema PyME**: al ofrecer el sistema como SaaS, se democratizan herramientas reservadas a grandes cadenas..

6 Impacto medioambiental

- **Menor uso de papel y consumibles**: El inventario se actualiza en tiempo real y los reportes pueden exportarse a Google Sheets/Drive, donde pueden auditarse, comentarse y ajustarse desde un celular o una tablet sin impresiones. El efecto acumulado es una reducción sostenida del papel, de los insumos de impresión (tinta/tóner) y de los descartes asociados.
- **Reducción de la huella de carbono:** El módulo de delivery aporta visibilidad del estado de cada pedido y favorece la consolidación de recorridos. Al contar con avisos al cliente y actualizaciones en tiempo real, se reducen los viajes fallidos. En paralelo, las notificaciones de reposición —enviadas antes de que el producto se agote— disminuyen compras de urgencia y repartos fuera de ruta. Todo esto se traduce en menos kilómetros recorridos, menor consumo de combustible y, por tanto, menor huella de carbono asociada a la distribución local.
- **Reducción del desperdicio de alimentos y empaques:** El seguimiento de inventario con umbrales y la predicción de reposición ayudan a reducir sobre-stock en productos de baja rotación y a anticipar la compra en ítems críticos. Menos sobre-stock implica menos roturas y vencimientos; y menos

vencimientos significa menos alimento desechado y menos empaques enviados a basura.

Conclusión

El proyecto desarrollado demuestra que una PyME familiar puede incorporar herramientas propias del retail digital y de la analítica de negocio sin incurrir en inversiones desproporcionadas, integrando en una misma solución procesos de venta, stock, delivery, marketing, fidelización y análisis estadístico. A través de la combinación de aplicaciones desktop, web y PWA respaldadas por un backend común, se logró construir una plataforma capaz de acompañar tanto la operatoria diaria del negocio como sus necesidades de crecimiento y modernización.

Los objetivos planteados al inicio del trabajo, tales como centralizar la información, reducir tareas manuales, abrir un canal online y mejorar la gestión comercial, fueron cumplidos dentro del alcance previsto. Asimismo, la incorporación del módulo analítico amplió el valor de la solución al permitir la consulta de indicadores históricos y dashboards estadísticos orientados al soporte de decisiones. La implementación de una capa desacoplada basada en procesos ETL, Data Warehouse y Metabase permitió no solo fortalecer la capacidad de análisis del negocio, sino también asegurar la disponibilidad del panel estadístico aun ante contingencias en la infraestructura transaccional principal.

Desde la perspectiva de la calidad, el trabajo combinó validaciones funcionales manuales, pruebas de aceptación de usuario y una capa inicial de pruebas automatizadas end-to-end sobre el backend, contribuyendo a reforzar la confiabilidad general del sistema. Si bien esta cobertura automatizada constituye una base inicial y todavía existen oportunidades de ampliación, representa un avance concreto en el aseguramiento técnico de la solución implementada.

Desde el punto de vista académico, la tesis evidencia la aplicación integrada de conocimientos de ingeniería de software, arquitectura de sistemas, bases de datos, experiencia de usuario, despliegue en la nube y analítica de datos. Desde el punto de vista empresarial, sienta las bases de una solución escalable y adaptable a futuras necesidades del negocio, tanto en su dimensión operativa como en su dimensión gerencial.

El desarrollo del proyecto también permitió obtener aprendizajes relevantes respecto del proceso de construcción de una solución de software aplicada a un contexto real. En primer lugar, se evidenció la importancia de estimar las tareas con márgenes adecuados y de dedicar tiempo suficiente al refinamiento del diseño antes de avanzar con la implementación. Asimismo, las validaciones tempranas y periódicas con el cliente resultaron fundamentales para detectar ajustes necesarios, evitar desarrollos que no aportaran valor y orientar la solución hacia las necesidades concretas del negocio. A nivel interno, el intercambio de criterios técnicos entre los integrantes del equipo fortaleció las decisiones adoptadas, especialmente cuando estas fueron documentadas y consensuadas. Finalmente, el registro sistemático de incidencias en Jira facilitó el seguimiento de errores, su priorización y la mejora progresiva de la calidad del producto desarrollado.

Como líneas de trabajo futuro, quedan abiertas la ampliación de la cobertura automatizada de pruebas, la profundización de la analítica avanzada, la incorporación de nuevas métricas y dashboards, y la evolución de funcionalidades orientadas a una operación aún más robusta y escalable. En este sentido, el proyecto no solo resuelve problemáticas actuales de Club Galo, sino que también establece una base sólida para su evolución tecnológica en el tiempo.

Bibliografía

Mordor-Intelligence[1]: https://www.mordorintelligence.ar/industry-reports/argentina-pet-food-market

Estudio Anual CACE[2]: https://cace.org.ar/prensa/estudioanual-2024/

La Nación - El país sudamericano que mas dinero gasta en alimentos para mascotas[3]: https://www.lanacion.com.ar/lifestyle/mascotas/el-pais-sudamericano-que-mas-dinero-gasta-en-alimentos-para-mascotas-nid23082024/

Erthalion[4]: https://erthalion.info/2017/12/21/advanced-json-benchmarks/

EnterpriseDB[5]: https://www.enterprisedb.com/blog/postgresql-vs-mysql-360-degree-comparison-syntax-performance-scalability-and-features

ByteBase[6]: https://www.bytebase.com/blog/postgres-vs-mysql/

Radixweb[7]: https://radixweb.com/blog/nestjs-vs-expressjs

KyleGill[8]: https://www.kylegill.com/essays/vite-vs-turbopack/

GooglePlay[9]: https://support.google.com/googleplay/android-developer

AppleStore[10]: https://developer.apple.com/programs/

WEB PWA[11]: https://web.dev/pwa-checklist/

Mercado Pago Developers[12]: https://www.mercadopago.com.ar/developers

Pagos360[13]: https://www.pagos360.com/

OWASP[14]: https://owasp.org

Auth0[15]: https://auth0.com/blog/

NestJS[16]: https://docs.nestjs.com

Express[17]: https://expressjs.com/en/guide/routing.html

Django[18]: https://docs.djangoproject.com/en/5.2/

Metabase[19]: https://www.metabase.com/docs/latest/

Grafana[20]: https://grafana.com/docs/

Power BI[21]: https://learn.microsoft.com/es-es/power-bi/

Tableau[22]: https://help.tableau.com/current/pro/desktop/en-us/dashboards.htm

Looker Studio[23]: https://docs.cloud.google.com/looker/docs/studio

Chart.js[24]: https://www.chartjs.org/docs/latest/

Recharts[25]: https://recharts.org/

D3.js[26]: https://d3js.org/what-is-d3