Desarrollo de una aplicación móvil para la gestión de pedidos y control de inventario en el restaurante Mr. Arroz Paisa, integrada con dashboard gerencial para optimizar procesos administrativos y operativos

Alejandro Rubio

Nicolas Alarcón

Universitaria Agustiniana – Uniagustiniana

Facultad de Ingenierías

Tecnología en Desarrollo de Software

Bogotá

2026

Desarrollo de una aplicación móvil para la gestión de pedidos y control de inventario en el restaurante Mr. Arroz Paisa, integrada con dashboard gerencial para optimizar procesos administrativos y operativos

Alejandro Rubio

Nicolas Alarcón

Tutor

Mauricio Alonso Villalba

Trabajo para optar al título de Tecnología en Desarrollo de Software

Universitaria Agustiniana – Uniagustiniana

Facultad de Ingenierías

Tecnología en Desarrollo de Software

Bogotá

2026

Tabla de contenido

Introducción 11
1. Datos informativos del proyecto 13
1.1 Título del proyecto 13
1.2 Breve síntesis del proyecto 13
2. Planificación del proyecto 14
2.1 Planteamiento del problema 14
2.1.1 Descripción del problema. 14
2.2 Objetivos del proyecto 16
2.2.1 Objetivo general. 16
2.2.2 Objetivos específicos. 17
2.3 Alcance de la solución 18
A. Arquitectura y acceso. 18
B. Módulos y usuarios. 18
C. Flujos operativos y automatización. 19
D. Lo que no incluye. 19
2.4 Metodología de desarrollo de software 20
3. Especificación de requisitos de software (IEEE 830) 22
3.1 Funcionalidad del producto 22
3.2 Características de los usuarios 23
3.3 Requisitos funcionales 24
Módulo 1: Interacción y autogestión (comensal). 24
Módulo 2: Sincronización operativa KDS (cocina). 25
Módulo 3: Servicio de meseros. 25
Módulo 4: Back-Office e inteligencia de negocios (administrador). 26
3.4 Diagramas de casos de uso 28
3.5 Especificaciones de los casos de uso 30
3.6 Restricciones y atributos de calidad 35
3.6.1 Restricciones. 35
3.6.2 Atributos de calidad. 35
4. Diseño del software (ISO 12207-1) 37
4.1 Diseño detallado del software 37

4.1.1 Arquitectura del frontend (PWA - React). 37
4.1.2 Arquitectura del backend (Supabase BaaS). 38
4.1.3 Modelo de datos relacional. 39
4.1.4 Diagrama de clases 40
4.1.5 Diagrama de componentes 41
4.1.6 Diagrama de actividades 41
4.1.6 Diagrama de despliegue 42
4.2 Diseño de la interfaz 43
4.3 Diseño del modelo de datos / Persistencia 50
4.4 Diseño de la arquitectura de software (Modelo C4) 50
5. Implementación 52
5.1 Herramientas utilizadas en el desarrollo del proyecto 52
5.2 Control de versiones 53
5.3 Despliegue 54
5.4 Entorno de desarrollo 54
5.5 Requisitos del hardware 55
5.5.1 Equipo del desarrollador. 55
5.5.2 Requisitos del servidor (hosting). 55
5.5.3 Requisitos del usuario. 56
6. Pruebas de software 57
6.1 Módulo 1 — Comensal (menú, carrito y pedido) 58
6.2 Módulo 2 — Cocina KDS (Kitchen Display System) 64
7. Conclusiones y recomendaciones 68
7.1 Conclusiones 68
7.2 Recomendaciones 69
Referencias 71
Anexos 72

Lista de tablas

Tabla 1 30

Tabla 2 31

Tabla 3 32

Tabla 4 33

Tabla 5 33

Tabla 6 52

Tabla 7 53

Tabla 8 55

Tabla 9 55

Tabla 10 56

Tabla 11 56

Tabla 12 56

Tabla 13 58

Tabla 14 58

Tabla 15 59

Tabla 16 59

Tabla 17 60

Tabla 18 61

Tabla 19 61

Tabla 20 62

Tabla 21 62

Tabla 22 63

Tabla 23 64

Tabla 24 64

Tabla 25 65

Tabla 26 66

Tabla 27 66

Tabla 28 67

Tabla 29 67

Lista de figuras

Figura 1 28
Figura 2 28
Figura 3 29
Figura 4 29
Figura 5 30
Figura 6 40
Figura 7 41
Figura 8 41
Figura 9 42
Figura 10 43
Figura 11 43
Figura 12 44
Figura 13 44
Figura 14 45
Figura 15 45
Figura 16 46
Figura 17 46
Figura 18 46
Figura 19 47
Figura 20 47

Figura 21 48
Figura 22 48
Figura 23 49
Figura 24 49
Figura 25 49
Figura 26 50
Figura 27 50
Figura 28 51
Figura 29 51

Resumen

El presente proyecto documenta el diseño, desarrollo y evaluación de Comendo, una Progressive Web App (PWA) orientada a la gestión de pedidos en mesa, control de inventario en tiempo real y generación de reportes gerenciales para el restaurante Mr. Arroz Paisa, ubicado en Bogotá D.C. La solución surge como respuesta a los cuellos de botella operativos derivados del modelo de servicio manual: tiempos de espera prolongados, errores en la toma de pedidos, nula visibilidad del comensal sobre el estado de su orden y ausencia de trazabilidad del inventario. Comendo permite al comensal escanear un código QR desde su smartphone para explorar el menú, personalizar su pedido excluyendo ingredientes, confirmar la orden con cálculo automático de Impoconsumo y monitorear el estado de preparación en tiempo real. El sistema integra cuatro módulos: autogestión del comensal, Kitchen Display System para cocina con alertas sonoras, vista del mesero con auto-asignación automática por carga de trabajo, y panel administrativo con dashboard analítico y exportación a Excel. La arquitectura se fundamenta en React 19 con Vite como frontend y Supabase como Backend as a Service, utilizando PostgreSQL, autenticación JWT, Supabase Realtime para sincronización mediante WebSockets y Storage para imágenes. Se aplicó la metodología MADLC, el estándar IEEE 830 para requisitos y la ISO 12207 para diseño. La evaluación mediante 32 casos de prueba funcionales validó los flujos críticos del sistema, incluyendo la deducción automatizada de inventario respetando exclusiones de ingredientes y la sincronización en tiempo real entre módulos.

Palabra clave: PWA, gestión de pedidos, código QR, inventario en tiempo real, Supabase, React, restaurantes, Kitchen Display System

Abstract

This project documents the design, development, and evaluation of Comendo, a Progressive Web App (PWA) aimed at table order management, real-time inventory control, and generation of managerial reports for the restaurant Mr. Arroz Paisa, located in Bogotá D.C. The solution emerges as a response to the operational bottlenecks derived from the manual service model: prolonged waiting times, order-taking errors, a complete lack of visibility for diners regarding the status of their order, and an absence of inventory traceability. Comendo allows diners to scan a QR code from their smartphone to browse the menu, customize their order by excluding ingredients, confirm the order with automatic calculation of the consumption tax (Impoconsumo), and monitor preparation status in real time. The system integrates four modules: diner self-service, a Kitchen Display System for the kitchen with audible alerts, a waiter view with automatic workload-based assignment, and an administrative panel with an analytical dashboard and Excel export. The architecture is built on React 19 with Vite as the frontend and Supabase as the Backend as a Service, using PostgreSQL, JWT authentication, Supabase Realtime for synchronization via WebSockets, and Storage for images. The MADLC methodology, the IEEE 830 standard for requirements, and ISO 12207 for design were applied. Evaluation through 32 functional test cases validated the system's critical workflows, including automated inventory deduction that respects ingredient exclusions and real-time synchronization across modules.

Keywords: PWA, order management, QR code, real-time inventory, Supabase, React, restaurants, Kitchen Display System

Introducción

El sector gastronómico en Bogotá D.C. ha experimentado un crecimiento constante durante la última década, impulsado por la expansión de restaurantes de comida rápida, establecimientos de servicio a la mesa y propuestas culinarias innovadoras que atienden a una población urbana en aumento. Sin embargo, este crecimiento ha evidenciado limitaciones operativas importantes, como tiempos largos de espera, errores en la toma de pedidos, congestión en caja y escasa trazabilidad de los procesos internos. Estos factores afectan directamente la experiencia del cliente, reducen la rotación de mesas y aumentan los costos derivados de reprocesos y desperdicios alimentarios.

Aunque algunos restaurantes han incorporado herramientas digitales como menús QR, sistemas POS tradicionales o aplicaciones de domicilios, estas soluciones suelen ser parciales o insuficientes, ya que no integran de manera unificada la operación del cliente, la cocina y la administración. Además, la mayoría requiere hardware especializado, instalaciones complejas o carecen de comunicación en tiempo real y automatización del inventario.

En este contexto, los avances en tecnologías web —especialmente las Progressive Web Apps (PWA), los sistemas de comunicación en tiempo real mediante canales de suscripción y la identificación rápida a través de códigos QR— abren nuevas posibilidades para optimizar la operación interna de los restaurantes y mejorar la experiencia del comensal. Las PWA permiten desarrollar aplicaciones accesibles sin instalación, ligeras y multiplataforma, lo que las convierte en una alternativa ideal para establecimientos que requieren soluciones eficientes y de bajo costo.

El presente anteproyecto propone el diseño y desarrollo de Comendo, una PWA destinada a gestionar pedidos en mesa, ofrecer seguimiento en tiempo real del estado de preparación, automatizar el inventario y centralizar la operación del restaurante en una única

plataforma digital. La solución integrará módulos para clientes, cocina, meseros y administración, permitiendo reducir tiempos de atención, minimizar errores en la toma de pedidos, mejorar la precisión del inventario y proporcionar reportes que faciliten la toma de decisiones.

La relevancia de este proyecto radica en su potencial para transformar la dinámica tradicional del servicio en restaurantes de Bogotá, al proporcionar una herramienta accesible, moderna y funcional que responde a las demandas actuales de eficiencia, digitalización y satisfacción del cliente.

1. Datos informativos del proyecto

1.1 Título del proyecto

Desarrollo de una aplicación móvil para la gestión de pedidos y control de inventario en restaurantes, integrada con dashboard gerencial para optimizar procesos administrativos y operativos en Mr. Arroz Paisa.

1.2 Breve síntesis del proyecto

En la actualidad, el restaurante **Mr. Arroz Paisa** enfrenta cuellos de botella operativos durante sus horas de alta demanda debido a su modelo tradicional de servicio; la total dependencia del personal para la toma manual de órdenes genera prolongados tiempos de espera, nula visibilidad del comensal sobre su pedido, y una desconexión administrativa donde la falta de control de inventario en tiempo real impide el análisis de tendencias comerciales. Como respuesta a esta problemática, el presente proyecto propone el desarrollo de "Comendo", una Progressive Web App (PWA) orientada a la gestión directa del servicio en mesa y la inteligencia de negocios. Esta solución tecnológica descentraliza la operación al permitir que el cliente interactúe de forma autónoma escaneando un código QR desde su smartphone para realizar su pedido y monitorear la cola de preparación en vivo; simultáneamente, el sistema sincroniza la orden con la cocina mediante Supabase Realtime, descuenta los insumos del inventario en tiempo real respetando las exclusiones de ingredientes del comensal, asigna automáticamente un mesero al pedido listo, y alimenta un dashboard gerencial con métricas de consumo, automatizando el flujo de trabajo, eliminando filas y optimizando la toma de decisiones estratégicas.

2. Planificación del proyecto

2.1 Planteamiento del problema

2.1.1 Descripción del problema.

En la mayoría de restaurantes de Bogotá D.C., el proceso tradicional de atención al cliente sigue siendo manual y lineal: el comensal debe esperar para recibir el menú, solicitar la presencia del mesero, dictar verbalmente su pedido, aguardar a que la cocina procese la orden y finalmente acercarse a caja para realizar el pago. Este flujo genera diversas dificultades que afectan tanto la experiencia del cliente como la productividad del establecimiento.

Entre los principales problemas se encuentran los elevados tiempos de espera, especialmente en horas de alta afluencia, lo que retrasa la rotación de mesas y disminuye la satisfacción del usuario. De hecho, un estudio de la Asociación Colombiana de la Industria Gastronómica (Acodres, 2023) reporta que la pérdida de comensales en ciudades como Bogotá, Cali y Bucaramanga asciende al 11%, una situación que puede estar relacionada con la insatisfacción por los tiempos de espera y la calidad del servicio. Además, son frecuentes los errores en la toma de pedidos, derivados de anotaciones manuales, malentendidos o fallas en la comunicación verbal entre cliente, mesero y cocina, ocasionando reprocesos, desperdicio de alimentos y pérdidas económicas. Según Tabblify (2023), los errores en la entrega de pedidos — como productos incorrectos o pedidos que no llegan— se encuentran entre las tres principales causas de insatisfacción de los comensales en Colombia.

Por otro lado, los clientes carecen de visibilidad sobre el estado de su pedido, lo que dificulta la gestión de expectativas y genera incertidumbre. La administración enfrenta limitaciones en la trazabilidad del inventario, provocando agotamientos inesperados, sobrestock

o falta de control sobre el consumo real de insumos. Finalmente, los procesos de cobro manual generan congestión en caja y demoras innecesarias, afectando la calidad del servicio. Esta problemática se agrava por el hecho de que solo el 45,6% de las empresas en Colombia mide los tiempos de atención y espera (Infobae, 2024), lo que dificulta la identificación y solución de estos inconvenientes. Todo esto evidencia una falta de integración tecnológica que permita un flujo de atención más ágil, controlado y eficiente.

Formulación del problema.

¿Cómo puede una PWA accesible mediante códigos QR optimizar los tiempos de atención, reducir errores operativos y mejorar el control del inventario en restaurantes de Bogotá, con el fin de incrementar la eficiencia del servicio y la satisfacción del cliente?

Justificación.

Eficiencia operativa: Automatizar procesos manuales (toma de pedidos, paso a cocina, disminución de inventario), reduciendo tiempos muertos y optimizando la asignación de personal, integrando al mesero de manera más eficiente mediante auto-asignación automática de pedidos listos.

Reducción de errores: Las órdenes ingresadas por el propio cliente minimizan malentendidos y devoluciones, facilitando la labor del mesero y permitiendo un mejor control sobre los pedidos. Además, el comensal puede excluir ingredientes directamente, evitando errores de comunicación.

Mejor experiencia de cliente: Transparencia en tiempos y estado de pedido, posibilidad de solicitudes discretas y control sobre el pago y servicio, disminuyendo la incertidumbre y aumentando la satisfacción.

Impacto económico: Mayor rotación de mesas y menor desperdicio por errores de pedido mejoran la rentabilidad; los reportes exportables a Excel permiten decisiones de compra y ofertas más acertadas.

Adopción fácil: Al ser PWA + QR no se exige instalación ni hardware complejo; basta con smartphones y un dispositivo simple en cocina/caja, lo que reduce costos y facilita la implementación.

La implementación de una solución tecnológica como Comendo responde a necesidades operativas y comerciales concretas, respaldadas por datos recientes del sector. La insatisfacción de los comensales debido a errores en los pedidos y tiempos de espera prolongados es una realidad documentada en Colombia (Tabblify, 2023; Acodres, 2023). La falta de monitoreo y control sobre estos aspectos, evidenciada por el bajo porcentaje de empresas que mide los tiempos de atención (Infobae, 2024), refuerza la urgencia de adoptar herramientas digitales que permitan optimizar la gestión de pedidos y mejorar la experiencia del cliente.

2.2 Objetivos del proyecto

2.2.1 Objetivo general.

Desarrollar una aplicación móvil (PWA) para la gestión de pedidos y monitoreo del comensal, integrada con un dashboard gerencial e inventario en tiempo real, con el fin de optimizar la operación y apoyar la toma de decisiones administrativas en el restaurante Mr. Arroz Paisa.

2.2.2 Objetivos específicos.

Diseñar la arquitectura funcional y la interfaz de la aplicación móvil para optimizar la gestión de pedidos y el monitoreo del comensal, con el fin de mejorar la interacción y eficiencia del servicio.

Implementar un sistema de gestión que integre el registro de pedidos, control de inventario con exclusiones de ingredientes, auto-asignación de meseros y visualización de información administrativa, para facilitar la operación y toma de decisiones del restaurante.

Evaluar el funcionamiento de la aplicación mediante pruebas operativas y análisis de resultados, con el propósito de verificar la eficiencia del sistema en la gestión y análisis de pedidos en tiempo real.

2.3 Alcance de la solución

A. Arquitectura y acceso.

Tecnología: La solución es una PWA (Progressive Web App) construida con React 19 y Vite, utilizando Supabase como Backend as a Service (BaaS) para base de datos PostgreSQL, autenticación, almacenamiento de archivos y comunicación en tiempo real.

Mecanismo de acceso: El acceso se realiza mediante el escaneo de códigos QR físicos ubicados en cada mesa del establecimiento.

Cero instalación: La aplicación es accesible sin necesidad de instalación previa por parte del usuario.

B. Módulos y usuarios.

La solución integra cuatro módulos funcionales diseñados para cubrir las necesidades de cada tipo de usuario en el restaurante:

Módulo para clientes (comensales): Interfaz con la que interactúa el cliente tras escanear el QR en la mesa. Permite la visualización del menú digital, la personalización de pedidos con exclusión de ingredientes y la toma de pedidos directamente desde su dispositivo móvil.

Módulo para cocina (KDS): Interfaz operativa para el personal de cocina. Incluye la visualización de los pedidos entrantes, la gestión de su estado de preparación y alertas sonoras para pedidos nuevos o urgentes (más de 15 minutos).

Módulo para meseros: Vista dedicada para los meseros del restaurante. Muestra los pedidos con estado “Listo” asignados automáticamente al mesero con menor carga de trabajo, facilitando la entrega oportuna.

Módulo de administración: Módulo para los gerentes o dueños del restaurante. Permite la gestión CRUD de productos, categorías, insumos y recetas; la generación de códigos QR por mesa; el monitoreo de mesas en tiempo real; la gestión de meseros; el control de movimientos de inventario; y la visualización de un dashboard analítico con exportación a Excel.

C. Flujos operativos y automatización.

Automatización de la toma de pedidos: El sistema automatiza el proceso desde la selección de platos, personalización de exclusiones de ingredientes, hasta la confirmación del pedido con cálculo de Impoconsumo (8%).

Automatización del paso a cocina: El pedido confirmado se visualiza instantáneamente en el KDS de cocina mediante Supabase Realtime (canales PostgreSQL).

Automatización de la disminución de inventario: Al confirmarse un pedido, el sistema descuenta automáticamente los insumos según la receta de cada plato, respetando las exclusiones del comensal. Cada movimiento queda registrado en la tabla movimientos_inventario.

Auto-asignación de mesero: Cuando un pedido cambia a estado “Listo”, el sistema asigna automáticamente el mesero con menor carga de pedidos pendientes.

Generación de reportes: El módulo de administración consolida datos de pedidos e inventario para generar informes exportables a formato Excel (.xlsx).

D. Lo que no incluye.

Pagos digitales: no se integran pasarelas de pago (pagos móviles, datáfonos integrados). El sistema registra el método de pago (efectivo o tarjeta) como información, pero el cobro se realiza externamente. Tampoco incluye: reservas de mesas, pedidos a domicilio (delivery), facturación electrónica con la DIAN, ni CRM o programas de fidelización.

2.4 Metodología de desarrollo de software

Para el desarrollo del proyecto “Comendo”, se emplea la metodología MADLC (Mobile Application Development Lifecycle). El ciclo de vida de MADLC garantiza un enfoque centrado en la experiencia del usuario (UX) y se ejecuta a través de las siguientes fases iterativas:

1. Fase de identificación. En esta etapa inicial se recolectan y analizan los requerimientos del sistema. Se definen los diferentes roles de usuario (Cliente, Cocinero, Mesero, Administrador) y sus historias de usuario. El objetivo es delimitar el alcance del proyecto, estableciendo las funcionalidades clave, como el escaneo de QR, la sincronización en tiempo real y la gestión automatizada del inventario.

2. Fase de diseño. Consiste en la conceptualización visual y técnica de la aplicación. Incluye el diseño de interfaz (wireframes y mockups) y el diseño de arquitectura (diagramas UML, modelado de base de datos PostgreSQL en Supabase y estructuración técnica del sistema).

3. Fase de desarrollo. Construcción del código fuente. Frontend: React 19 con Vite, Zustand v5 para gestión de estado, React Router v7 para enrutamiento, Lucide React para iconografía y CSS-in-JS para estilos. Backend: Supabase como BaaS, aprovechando PostgreSQL, Supabase Auth para autenticación basada en roles, Supabase Realtime para comunicación en tiempo real y Supabase Storage para almacenamiento de imágenes.

4. Fase de prototipado. Se construyen versiones funcionales tempranas de la aplicación. Se prueban los flujos críticos (realizar un pedido y verlo aparecer en el KDS de cocina) en un entorno controlado.

5. Fase de pruebas. Incluye: pruebas funcionales (transacciones de inventario, exclusiones de ingredientes y concurrencia), pruebas de red y latencia (rendimiento de Supabase

Realtime bajo diferentes condiciones de conectividad) y pruebas de usabilidad (evaluación en dispositivos físicos reales).

6. Fase de despliegue. El frontend se publica en Vercel con configuración de rewrites SPA (vercel.json). Supabase se encarga del alojamiento del backend, base de datos y almacenamiento en la nube, con certificados HTTPS incluidos.

3. Especificación de requisitos de software (IEEE 830)

3.1 Funcionalidad del producto

La aplicación móvil “Comendo” estructura su lógica de negocio en cuatro procesos macro interdependientes, diseñados bajo una arquitectura orientada a eventos y procesamiento en tiempo real mediante Supabase Realtime:

Front-Office (módulo de autogestión cliente). Servicio orientado a la interacción directa con el consumidor final. Inicia con el escaneo de un código QR que inyecta parámetros de enrutamiento (identificador de mesa). Despliega el catálogo digital, permite la gestión de un carrito de compras con persistencia de estado mediante Zustand, la exclusión de ingredientes por producto, y culmina con el despacho de la orden. Integra escucha asíncrona vía canales Realtime de Supabase para reflejar la trazabilidad del pedido en vivo.

Middle-Office (módulo KDS). Servicio de orquestación operativa protegido por rol Operario_Cocina. Centraliza el flujo de entrada de pedidos bajo un modelo de colas FIFO. Permite ejecutar transacciones de estado (Recibido → Preparando → Listo), desencadenando eventos Realtime que actualizan la interfaz del cliente. Incluye alertas sonoras mediante Web Audio API.

Service-Office (módulo mesero). Servicio orientado a la entrega de pedidos. No requiere login formal; los meseros se identifican vía localStorage. El sistema asigna automáticamente los pedidos con estado “Listo” al mesero con menor carga de trabajo pendiente.

Back-Office (módulo administrativo). Motor transaccional y analítico del sistema, protegido por rol Administrador mediante Supabase Auth. Se encarga de la conciliación del

inventario en tiempo real respetando exclusiones. Cuenta con un dashboard interactivo con KPIs, rotación de inventario, tendencias de consumo y exportación a Excel.

3.2 Características de los usuarios

El sistema interactúa con cuatro perfiles de actores principales:

Actor 1 – El comensal. Nivel de alfabetización digital heterogéneo. Entorno: dispositivo móvil propio con conexiones de red variables. Requiere interfaz intuitiva, libre de fricciones de autenticación.

Actor 2 – Personal de preparación. Nivel básico-intermedio. Entorno: tabletas en zona de cocina con alta presión temporal. Requiere botones de gran tamaño (mínimo 44x44px) y alto contraste.

Actor 3 – Gerente / Administrador. Nivel intermedio-avanzado. Entorno: computadores o tabletas. Requiere acceso seguro mediante Supabase Auth con roles, exportación de datos a Excel y gráficas de fácil interpretación.

3.3 Requisitos funcionales

Módulo 1: Interacción y autogestión (comensal).

RF-1.1 | Acceso y enrutamiento QR (Alta): El sistema debe capturar el identificador único de la mesa a través del escaneo de un código QR y habilitar el entorno de autogestión sin requerir creación de cuenta, login, ni descarga de software nativo.

RF-1.2 | Gestión de sesión efímera (Alta): Al ingresar vía QR, el sistema genera un token de sesión temporal (almacenado en localStorage) que vincula el dispositivo del comensal con la mesa específica. El hook useSesionMesa gestiona este token UUID y retorna el número visible de la mesa.

RF-1.3 | Navegación de catálogo y caché PWA (Alta): El sistema debe desplegar el menú digital agrupado por categorías. Mediante Service Workers, el catálogo base se almacena en caché para garantizar carga casi instantánea.

RF-1.4 | Motor de carrito y órdenes (Alta): El sistema permite al comensal añadir productos, modificar cantidades, excluir ingredientes específicos, visualizar el subtotal dinámico con Impoconsumo del 8%, y emitir la orden final. El estado del carrito se gestiona con Zustand (useCartStore).

RF-1.5 | Trazabilidad en tiempo real (Media): La interfaz OrderTrackingPage refleja los cambios de estado del pedido (“Recibido”, “Preparando”, “Listo”, “Entregado”) notificados mediante Supabase Realtime.

RF-1.6 | Solicitud de cuenta (Alta): Cuando el comensal solicita la cuenta, el sistema cierra todos los pedidos activos de la mesa (no solo el del URL actual), registra el método de pago (efectivo o tarjeta) y libera la mesa (estado: Libre, token_sesion_actual: null).

RF-1.7 | Exclusión de ingredientes (Media): El comensal puede excluir ingredientes específicos de un producto al momento de agregarlo al carrito. Los ingredientes excluidos no se descuentan del inventario y quedan registrados en la tabla exclusiones_pedido.

Módulo 2: Sincronización operativa KDS (cocina).

RF-2.1 | Recepción de pedidos en tiempo real (Alta): El KDS inyecta en la cola de la interfaz de cocina todo nuevo pedido confirmado instantáneamente vía Supabase Realtime (canales PostgreSQL), sin necesidad de refrescar el navegador.

RF-2.2 | Gestión transaccional de estados (Alta): El operario puede modificar el estado operativo de la orden (Recibido → Preparando → Listo), garantizando persistencia en base de datos y emisión al dispositivo del comensal en tiempo real.

RF-2.3 | Monitor de mesas activas (Alta): El sistema provee una vista en cuadrícula visualizando qué mesas están libres, cuáles tienen clientes activos y cuáles están a la espera de sus pedidos.

RF-2.4 | Alertas sensoriales (Media): El sistema emite una alerta sonora en el KDS (mediante Web Audio API, hook useKDSSound) cuando ingresa una orden nueva o un pedido excede el tiempo límite de 15 minutos.

Módulo 3: Servicio de meseros.

RF-3.1 | Auto-asignación de mesero (Alta): Cuando un pedido cambia a estado “Listo”, el servicio meseroService.js asigna automáticamente el mesero con menor carga de pedidos pendientes.

RF-3.2 | Vista del mesero (Alta): La página MeseroPage muestra los pedidos asignados con estado “Listo”, la mesa de destino y los productos a entregar, actualizándose en tiempo real mediante Supabase Realtime.

Módulo 4: Back-Office e inteligencia de negocios (administrador).

RF-4.1 | Deducción automatizada de inventario (Alta): El sistema aplica un descuento transaccional en la tabla de inventarios según la receta de cada producto, respetando exclusiones del comensal. Cada movimiento se registra en movimientos_inventario con cantidad anterior, nueva y motivo.

RF-4.2 | Gestión de catálogo CRUD (Alta): Interfaces para Crear, Leer, Actualizar y Eliminar productos (ProductosCRUD), insumos (InsumosCRUD), recetas (RecetasCRUD) y meseros (MeserosCRUD).

RF-4.3 | Generación de códigos QR por mesa (Alta): El componente MesasQR permite al administrador generar y visualizar los códigos QR únicos de cada mesa, utilizando la librería qrcode.react.

RF-4.4 | Cierre y liberación de mesa (Alta): El sistema permite emitir la cuenta final de una mesa. Una vez registrado el pago, libera la mesa (estado: Libre, token_sesion_actual: null).

RF-4.5 | Autenticación segura y roles (Alta): El acceso al panel administrativo y al KDS se restringe mediante Supabase Auth con roles almacenados en la tabla usuarios (Administrador, Operario_Cocina, Mesero). Las rutas protegidas se implementan con ProtectedRoute y ProtectedKDS.

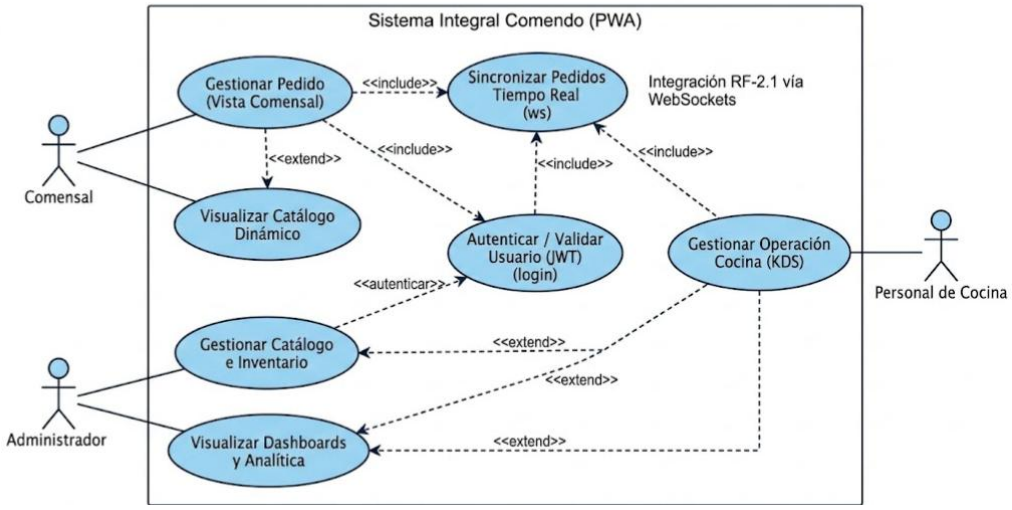
RF-4.6 | Dashboard analítico (Media): El sistema compila datos transaccionales para renderizar gráficas sobre ventas, productos de mayor/menor rotación y reportes de inventario crítico, con exportación a Excel mediante xlsx y file-saver.

RF-4.7 | Movimientos de inventario (Media): El componente MovimientosInventario permite visualizar el historial completo de movimientos de stock para trazabilidad y auditoría.

3.4 Diagramas de casos de uso

Figura 1

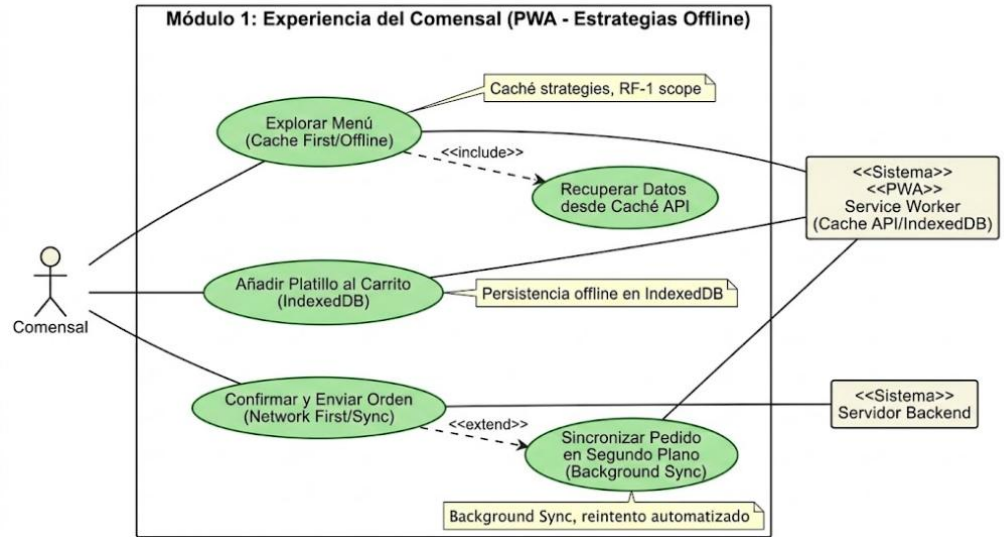
Diagramas de Casos de uso general



Nota. Fuente: Elaboración propia (2026)

Figura 2

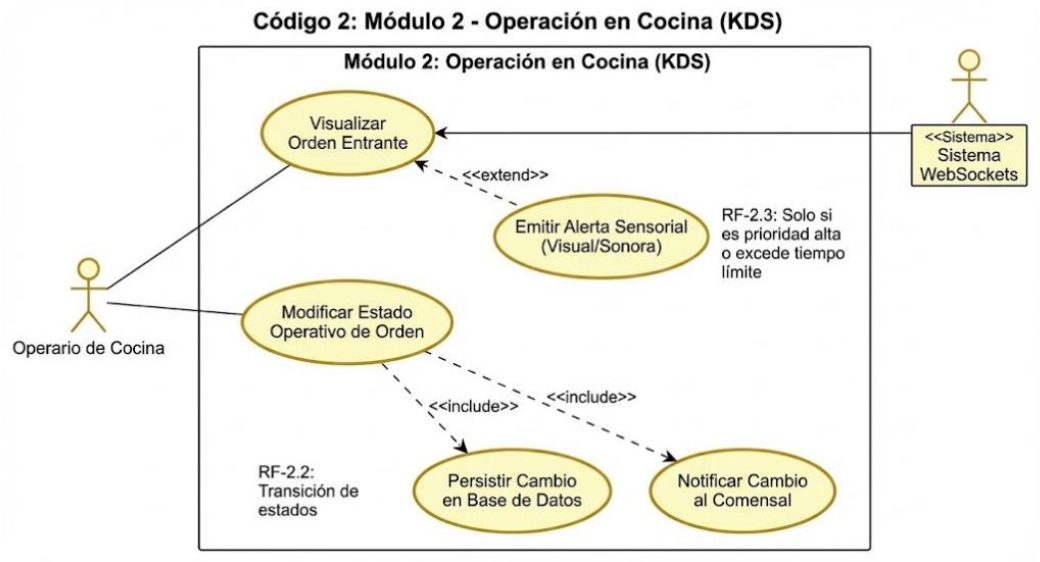
Diagramas de Casos de uso Modulo 1.



Nota. Fuente: Elaboración propia (2026)

Figura 3

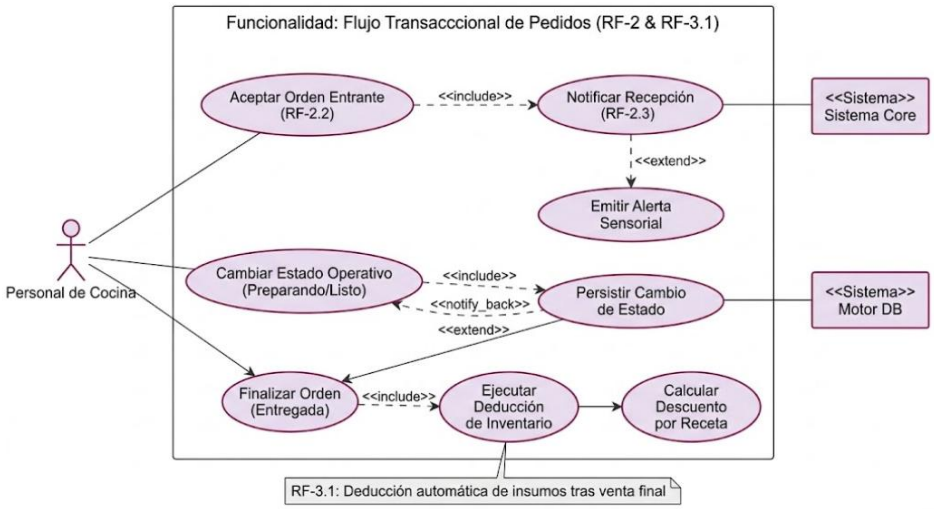
Diagramas de Casos de uso Modulo 2.



Nota. Fuente: Elaboración propia (2026)

Figura 4

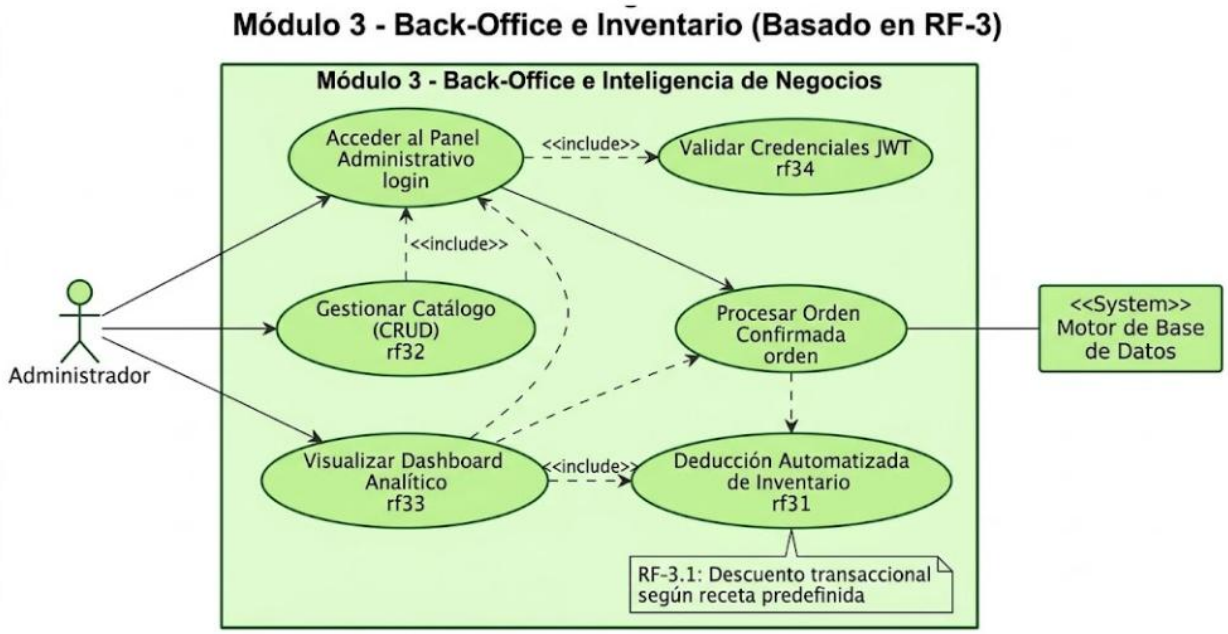
Diagramas de Casos de uso flujo transaccional de pedido.



Nota. Fuente: Elaboración propia (2026)

Figura 5

Diagramas de Casos de uso Modulo 3.



Nota. Fuente: Elaboración propia (2026)

3.5 Especificaciones de los casos de uso

Tabla 1*:*

Módulo de Acceso y Catálogo

ID CU	Nombre del Caso de Uso	Actores	Descripción	Flujo Principal	Requisito PWA / Técnico
CU-CAT-01	Escanear QR y Acceder	Cliente	El cliente escanea el QR de la mesa para abrir la PWA.	1. Escanea QR. 2. App lee ID de la mesa desde la URL. 3. Genera ID de sesión anónima en el navegador.	Lectura de parámetros de URL (React Router: useParams).

CU-CAT-02	Visualizar Catálogo/Menú	Cliente	El cliente navega por los productos disponibles organizados por categorías.	1. Abre la app. 2. Selecciona categoría. 3. Visualiza lista de productos y precios.	Cache First: El menú se guarda en caché del Service Worker para cargar al instante.
CU-CAT-03	Filtrar/Buscar Productos	Cliente	Permite buscar un producto específico por nombre.	1. Toca la barra de búsqueda. 2. Escribe el nombre. 3. Sistema filtra resultados en tiempo real.	Optimización de renderizado en React (Evitar re-renders innecesarios).

Nota. Fuente: Elaboración propia (2026)

Tabla 2:

Módulo de Pedidos de Mesa (Cliente)

ID CU	Nombre del Caso de Uso	Actores	Descripción	Flujo Principal	Requisito PWA / Técnico
CU-PED-01	Agregar al Carrito	Cliente	El cliente selecciona productos, cantidades y los añade a su orden.	1. Toca producto. 2. Define cantidad. 3. Agrega al carrito temporal.	Uso de Context API o Zustand para manejar el estado global del carrito.
CU-PED-02	Enviar Pedido de la Mesa	Cliente	Confirma el carrito y envía la comanda a la barra/cocina.	1. Abre carrito. 2. Revisa total. 3. Presiona "Confirmar Pedido". 4. Sistema asocia pedido a la mesa.	Background Sync: Si el WiFi del local falla un segundo, el pedido se encola y se envía al reconectar.
CU-PED-03	Consultar Estado de Orden	Cliente	El cliente puede ver si su pedido está "Recibido", "En Preparación" o "Listo".	1. Entra a "Mi Pedido". 2. Sistema muestra el estado actual de la orden de esa mesa.	Conexión en tiempo real (Supabase realtime).

Nota. Fuente: Elaboración propia (2026)

Tabla 3:

Módulo de Gestión y Recepción (Personal del Local)

ID CU	Nombre del Caso de Uso	Actores	Descripción	Flujo Principal	Requisito PWA / Técnico
CU-GES-01	Recibir Nuevo Pedido	Personal	El sistema alerta al personal que una mesa ha realizado un pedido.	1. Cliente envía pedido. 2. La pantalla del personal se actualiza automáticamente. 3. Suena alerta visual/sonora.	API de Notificaciones Push + Supabase Realtime.
CU-GES-02	Actualizar Estado del Pedido	Personal	El trabajador cambia el estado de la comanda según el progreso.	1. Ve la lista de pedidos. 2. Cambia de estado a "En Preparación" o "Entregado".	Botones grandes y accesibles (Diseño UI pensado para trabajo rápido).
CU-GES-03	Gestionar Mesas Activas	Personal	Visualiza qué mesas están ocupadas, cuáles han pedido y cuáles están libres.	1. Entra a vista de mesas. 2. Ve el mapa/cuadrícula con colores según estado.	Almacenamiento del estado en la Base de Datos.

Nota. Fuente: Elaboración propia (2026)

Tabla 4:

Módulo de Cierre y Facturación

ID CU	Nombre del Caso de Uso	Actores	Descripción	Flujo Principal	Requisito PWA / Técnico
CU-FAC-01	Solicitar la Cuenta	Cliente	El cliente pide la cuenta desde su teléfono indicando método de pago.	1. Presiona "Pedir Cuenta". 2. Selecciona Efectivo/Tarjeta. 3. Notifica al personal.	Flujo de UI sencillo e intuitivo.
CU-FAC-02	Generar Comprobante/Factura	Personal	El sistema calcula el total y permite generar el soporte de pago.	1. Selecciona la mesa a cobrar. 2. Confirma el pago. 3. Genera registro de venta.	Endpoint seguro en Supabase BaaS para transacciones.
CU-FAC-03	Liberar Mesa	Personal / Sistema	Una vez pagado, la mesa vuelve a estar disponible para un nuevo cliente.	1. Se registra el pago. 2. Se cierra la sesión de esa mesa. 3. El QR vuelve a admitir nuevos escaneos limpios.	Limpieza de tokens/sesiones asociadas a ese ID de mesa.

Nota. Fuente: Elaboración propia (2026)

Tabla 5:

Funcionalidades Nativas PWA (Opcionales, pero recomendadas)

ID CU	Nombre del Caso de Uso	Actores	Descripción	Flujo Principal	Requisito PWA / Técnico
CU-PWA-01	Instalar App en Terminales	Personal	Instalar la PWA en las tablets o teléfonos de los trabajadores.	1. Abre navegador. 2. Acepta prompt de instalación. 3. Usa la app en pantalla completa.	Manifest.json configurado con display: standalone.
CU-PWA-02	Caché de Imágenes de Productos	Sistema	Las fotos del menú se guardan en el dispositivo para no consumir datos del cliente.	1. Cliente navega. 2. Service Worker intercepta las imágenes y las guarda.	Estrategia de caché: Stale-While-Revalidate.

Nota. Fuente: Elaboración propia (2026)

3.6 Restricciones y atributos de calidad

Esta sección define las limitaciones tecnológicas y del entorno bajo las cuales debe operar el sistema, así como los atributos no funcionales que garantizan una experiencia de usuario óptima, segura y mantenible, alineándose con los estándares de calidad de software.

3.6.1 Restricciones.

Protocolo de seguridad obligatorio (HTTPS): El sistema debe estar desplegado bajo un dominio con certificado SSL válido. Tanto Vercel como Supabase proveen HTTPS por defecto.

Ecosistema web estándar: La aplicación del comensal no requiere descarga de aplicación nativa. Debe ejecutarse en navegadores móviles modernos (Chrome, Safari, Edge, Firefox).

Arquitectura de despliegue: Vercel para el frontend (capa gratuita con rewrites SPA) y Supabase para el backend, base de datos y almacenamiento (capa gratuita). No se requiere servidor backend separado.

3.6.2 Atributos de calidad.

Rendimiento: El Largest Contentful Paint debe ocurrir en menos de 2.5 segundos en conexiones 3G/4G, utilizando Lazy Loading para imágenes. El menú base se sirve desde la caché del Service Worker.

Usabilidad: Interfaz diseñada para pantallas táctiles con botones de mínimo 44x44px. El flujo desde el escaneo hasta la confirmación del pedido no supera los 4 pasos. Compatibilidad iOS con WebkitOverflowScrolling: touch y env(safe-area-inset-bottom).

Seguridad: Las URLs de los QR usan UUID v4 difíciles de predecir. El acceso al panel administrativo y al KDS se valida mediante Supabase Auth con Row Level Security (RLS).

Mantenibilidad: Código basado en Clean Code, con componentes React reutilizables y lógica de negocio centralizada en Custom Hooks (useMenu, useKDS, useActivePedido, useSesionMesa, useKDSSound) y servicios (orderService, meseroService, exportService).

4. Diseño del software (ISO 12207-1)

4.1 Diseño detallado del software

El diseño detallado se estructura utilizando una arquitectura basada en componentes para el frontend (PWA) y Supabase como Backend as a Service (BaaS) para la capa de datos y servicios.

4.1.1 Arquitectura del frontend (PWA - React).

El cliente web está dividido lógicamente según los cuatro actores principales del sistema, compartiendo la misma base de código pero renderizando vistas según la ruta y el token de sesión.

A. Aplicación del comensal. useSesionMesa gestiona el token UUID en localStorage para vincular el dispositivo con la mesa. useMenu carga categorías y productos desde Supabase. useCartStore (Zustand) gestiona el carrito con campo cartAnimationTrigger para animaciones. CartDrawer calcula el Impoconsumo (8%). ProductDetailModal permite excluir ingredientes. OrderTrackingPage muestra el estado del pedido en tiempo real vía Supabase Realtime.

B. Kitchen Display System (KDS). useKDS mantiene la conexión Realtime con Supabase, escuchando cambios en la tabla pedidos. KDSPage muestra el tablero de pedidos activos con estados: Recibido, Preparando y Listo. useKDSSound emite alertas sonoras mediante Web Audio API. Acceso protegido por ProtectedKDS (rol Operario_Cocina).

C. Back-Office (panel administrativo). AdminPage contiene sub-módulos: Dashboard, ProductosCRUD, InsumosCRUD, RecetasCRUD, MesasQR, MonitorMesas, MovimientosInventario y MeserosCRUD. Acceso protegido por ProtectedRoute (rol Administrador). exportService.js genera archivos Excel.

4.1.2 Arquitectura del backend (Supabase BaaS).

Supabase actúa como el orquestador central del sistema, proporcionando:

API REST autogenerada: Operaciones CRUD sobre todas las tablas del esquema PostgreSQL mediante peticiones HTTPS estándar.

Supabase Realtime: Servicio de WebSockets que permite la suscripción a cambios en las tablas de pedidos y mesas. Cuando un pedido cambia de estado, el KDS, la vista del comensal y la vista del mesero se actualizan instantáneamente.

Supabase Auth: Sistema de autenticación con JWT para restringir el acceso al panel administrativo y al KDS. Los roles (Administrador, Operario_Cocina, Mesero) se almacenan en la tabla usuarios.

Supabase Storage: Almacenamiento de imágenes de productos con URLs públicas optimizadas para carga rápida en dispositivos móviles.

Lógica de negocio en el frontend: orderService.js gestiona la creación de pedidos y el descuento de inventario respetando exclusiones. meseroService.js gestiona la auto-asignación. Toda la lógica de negocio se ejecuta en el frontend a través de la API REST de Supabase.

4.1.3 Modelo de datos relacional.

La persistencia de datos se gestiona a través de PostgreSQL en Supabase, garantizando la integridad referencial. El esquema normalizado comprende 10 tablas:

Tabla usuarios: id_usuario (UUID, PK), email (VARCHAR, UNIQUE), nombre (VARCHAR), rol (ENUM: Administrador / Operario_Cocina / Mesero).

Tabla mesas: id_mesa (UUID, PK), numero (INT, UNIQUE), estado (ENUM: Libre / Ocupada / Por_Pagar), token_sesion_actual (VARCHAR, NULL).

Tabla categorias: id_categoria (UUID, PK), nombre (VARCHAR).

Tabla productos: id_producto (UUID, PK), id_categoria (UUID, FK), nombre (VARCHAR), descripcion (TEXT), precio (DECIMAL), disponible (BOOLEAN), imagen_url (VARCHAR).

Tabla insumos: id_insumo (UUID, PK), nombre (VARCHAR), cantidad_stock (DECIMAL), unidad_medida (VARCHAR).

Tabla recetas: id_receta (UUID, PK), id_producto (UUID, FK), id_insumo (UUID, FK), cantidad_requerida (DECIMAL).

Tabla pedidos: id_pedido (UUID, PK), id_mesa (UUID, FK), id_mesero (UUID, FK, NULL), estado_actual (ENUM: Recibido / Preparando / Listo / Entregado / Pagado), total (DECIMAL), metodo_pago (ENUM: efectivo / tarjeta), fecha_creacion (TIMESTAMP), fecha_listo (TIMESTAMP).

Tabla detalle_pedidos: id_detalle (UUID, PK), id_pedido (UUID, FK), id_producto (UUID, FK), cantidad (INT), precio_unitario (DECIMAL), subtotal (DECIMAL), notas (TEXT).

Tabla exclusiones_pedido: id_exclusion (UUID, PK), id_detalle (UUID, FK), id_insumo (UUID, FK), nombre_insumo (VARCHAR), cantidad_no_descontada (DECIMAL), unidad_medida (VARCHAR). Registra los ingredientes que el comensal excluyó de un producto.

Tabla movimientos_inventario: id_movimiento (UUID, PK), id_insumo (UUID, FK), tipo (VARCHAR), cantidad (DECIMAL), cantidad_anterior (DECIMAL), cantidad_nueva (DECIMAL), motivo (TEXT), id_pedido (UUID, FK, NULL), fecha (TIMESTAMP). Registra cada movimiento de stock para trazabilidad.

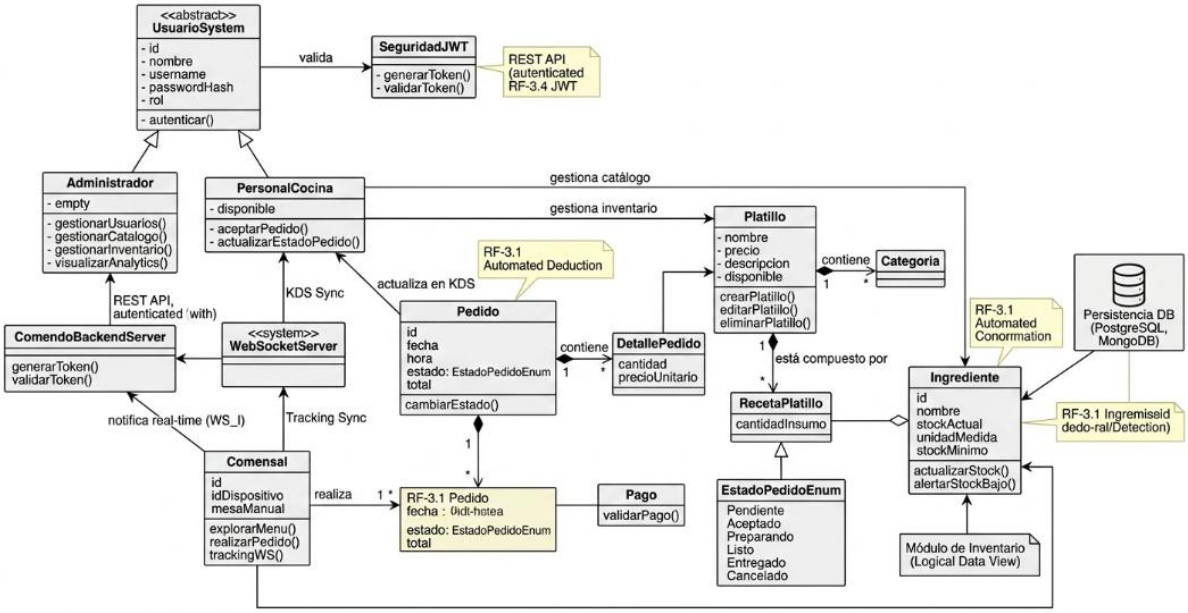
Estados del pedido: Recibido → Preparando → Listo → Entregado → Pagado.

Métodos de pago: efectivo | tarjeta. El cobro se realiza externamente.

4.1.4 Diagrama de clases

Figura 6

Diagrama de clases

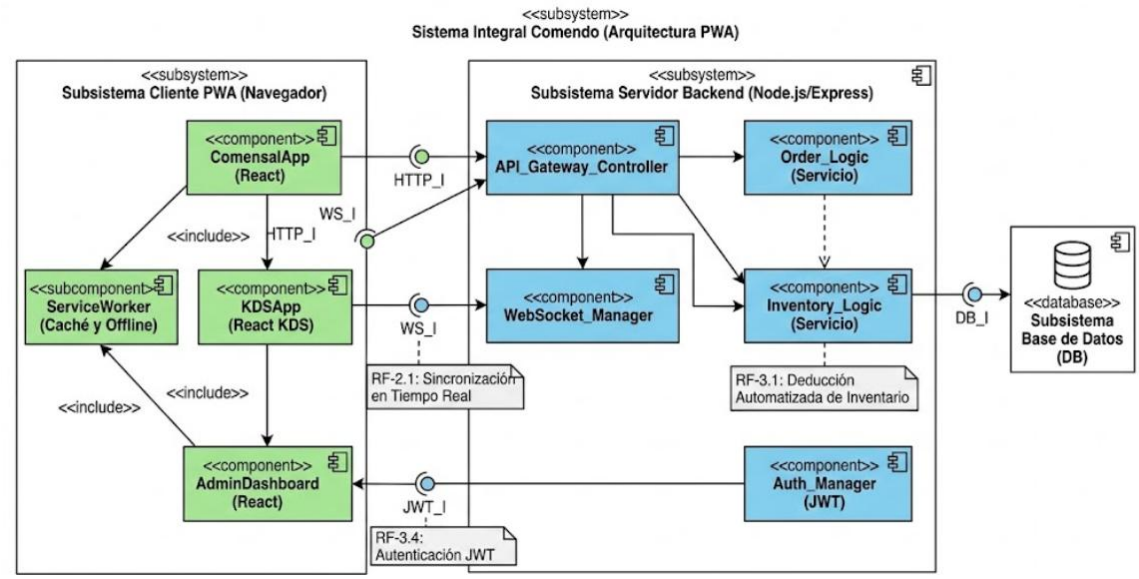


Nota. Fuente: Elaboración propia (2026)

4.1.5 Diagrama de componentes

Figura 7

Diagrama de componentes

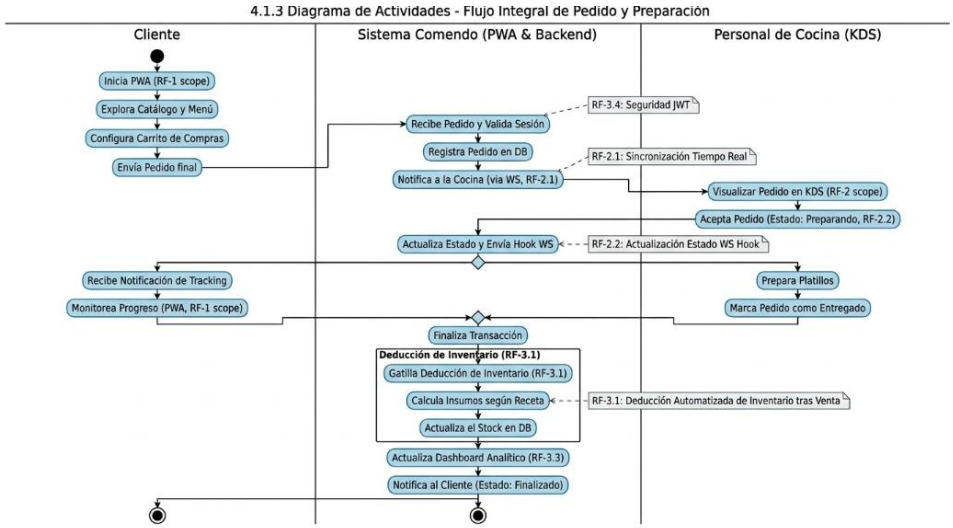


Nota. Fuente: Elaboración propia (2026)

4.1.6 Diagrama de actividades

Figura 8

Diagrama de actividades

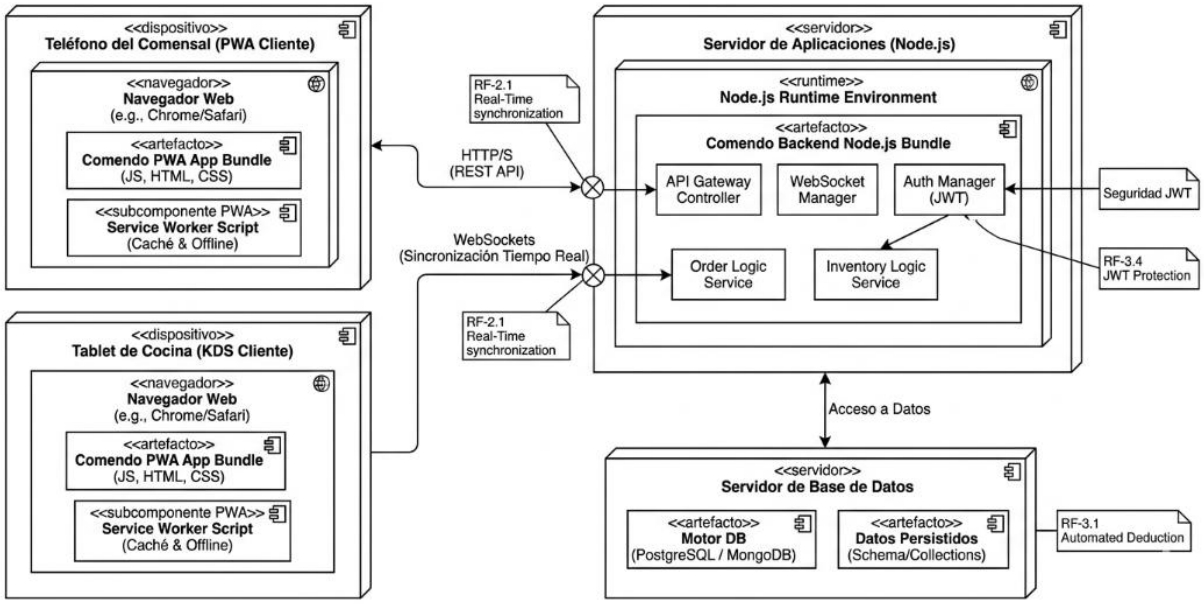


Nota. Fuente: Elaboración propia (2026)

4.1.6 Diagrama de despliegue

Figura 9

Diagrama de despliegue

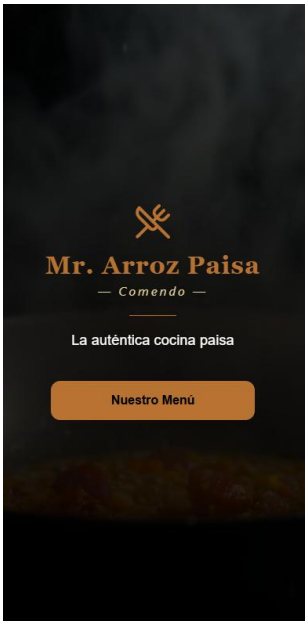


Nota. Fuente: Elaboración propia (2026)

4.2 Diseño de la interfaz

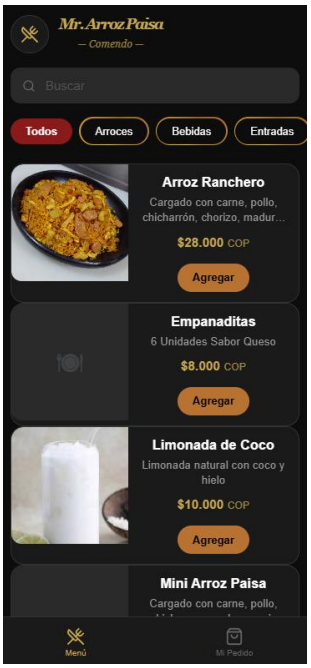
Usuario (Comensal)

Figura 10 *Home*



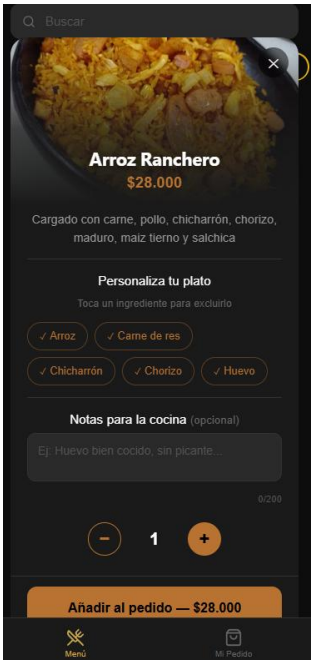
Nota. Fuente: Elaboración propia (2026)

Figura 11 *Menú*



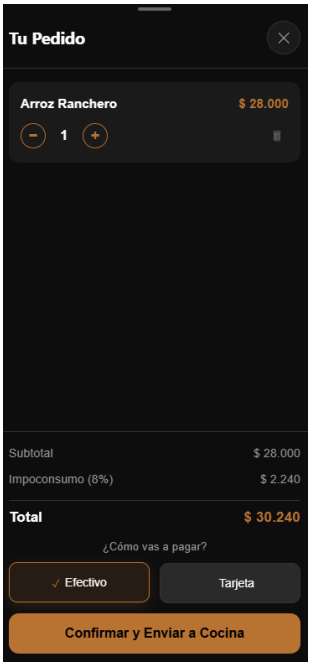
Nota. Fuente: Elaboración propia (2026)

Figura 12 *Pedido*



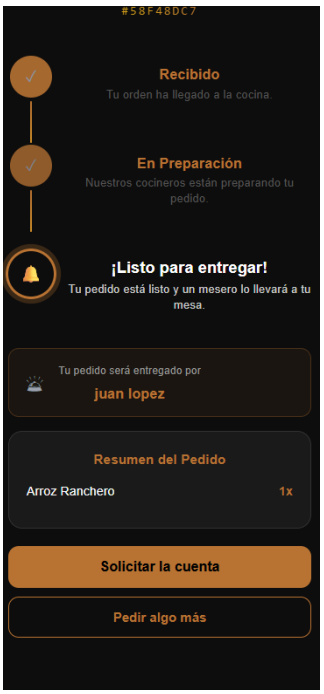
Nota. Fuente: Elaboración propia (2026)

Figura 13 *Confirmación Pedido*



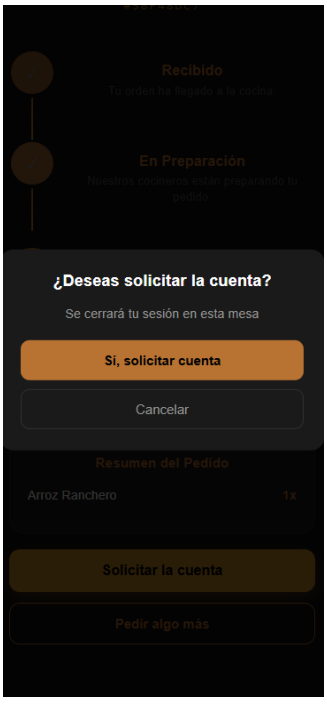
Nota. Fuente: Elaboración propia (2026)

Figura 14 *Confirmación entrega*



Nota. Fuente: Elaboración propia (2026)

Figura 15 *Solicitud cuenta*



Nota. Fuente: Elaboración propia (2026)

Figura 16 *Login*



Nota. Fuente: Elaboración propia (2026)

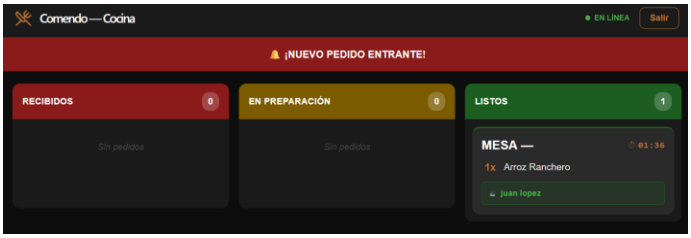
Cocina

Figura 17 *KDS Cocina*



Nota: Elaboración propia (2026)

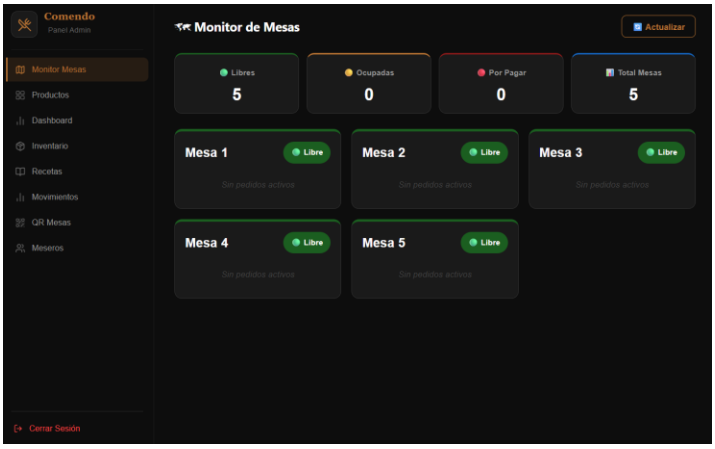
Figura 18 *Cocina cierre pedido*



Nota. Fuente: Elaboración propia (2026)

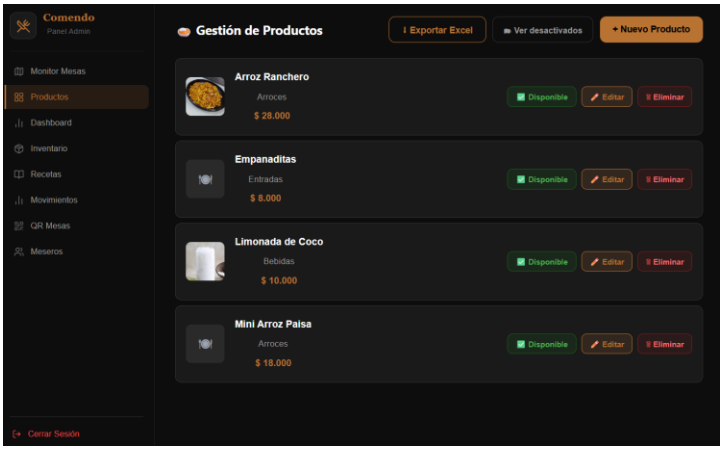
Administrador

Figura 19 *Monitoreo de mesas*



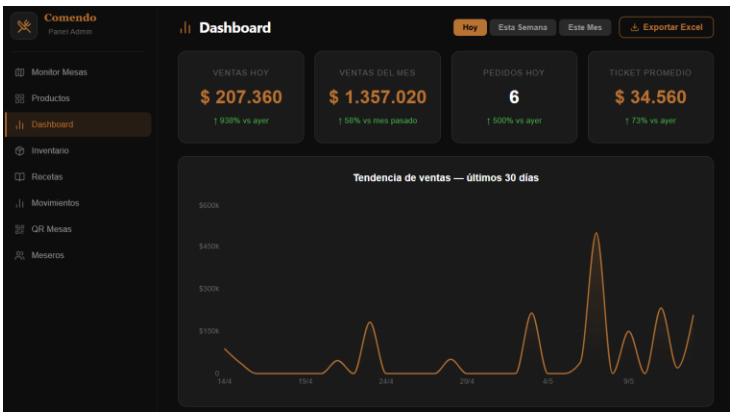
Nota. Fuente: Elaboración propia (2026)

Figura 20 *Productos*



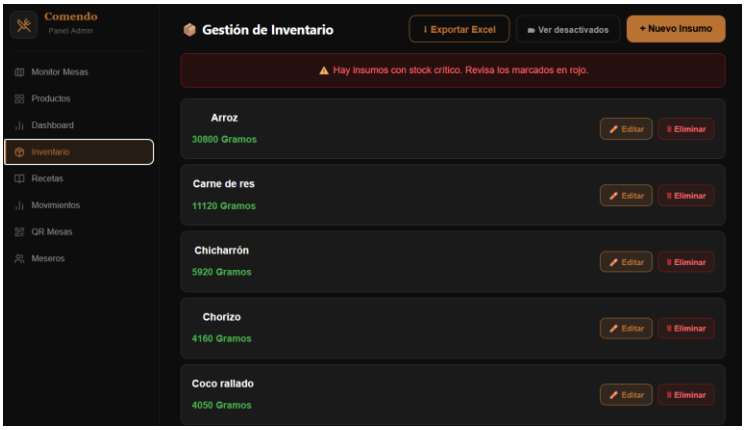
Nota. Fuente: Elaboración propia (2026)

Figura 21 *Dashboard*



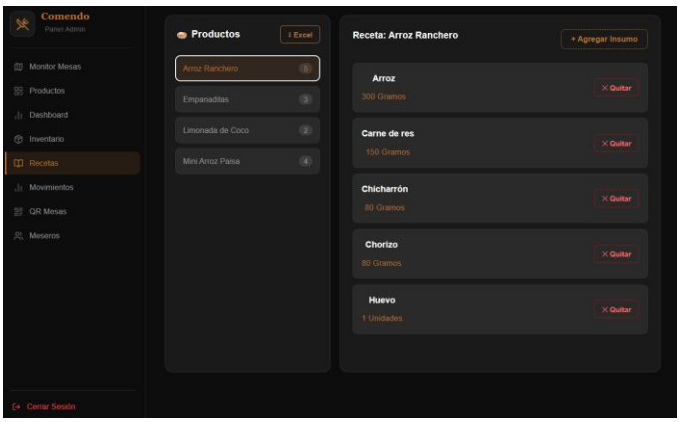
Nota. Fuente: Elaboración propia (2026)

Figura 22 *Gestión inventario*



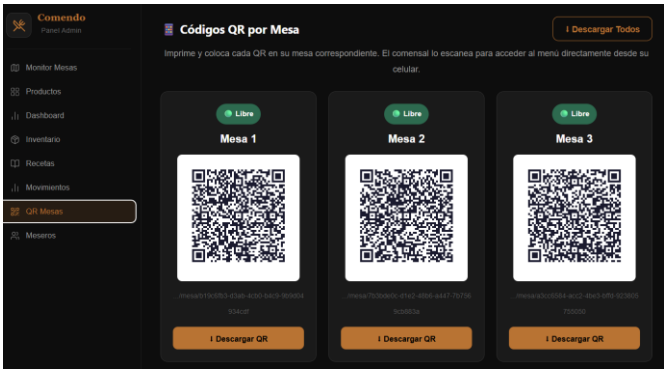
Nota. Fuente: Elaboración propia (2026)

Figura 23 *Recetas*



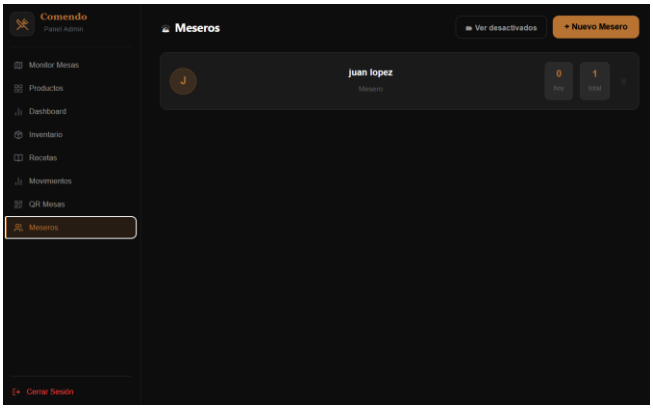
Nota. Fuente: Elaboración propia (2026)

Figura 24 *Codigos QR*



Nota. Fuente: Elaboración propia (2026)

Figura 25 *Meseros*

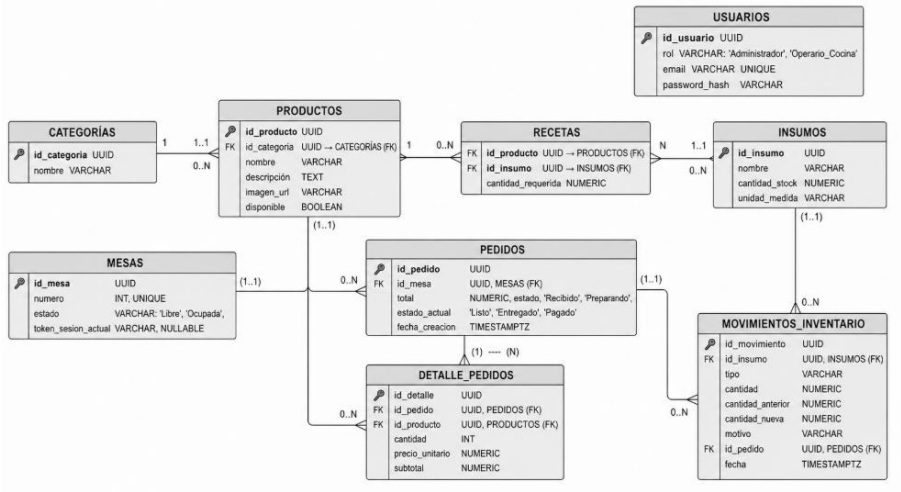


Nota. Fuente: Elaboración propia (2026)

4.3 Diseño del modelo de datos / Persistencia

Figura 26

Modelo entidad relación

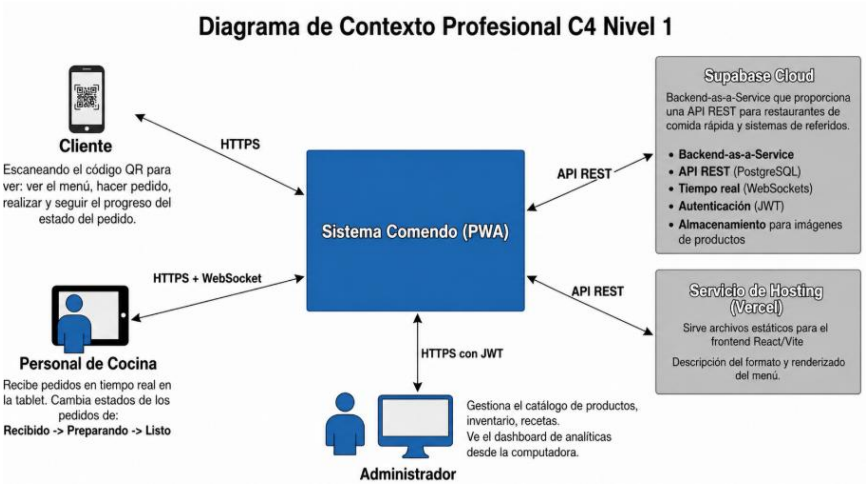


Nota. Fuente: Elaboración propia (2026)

4.4 Diseño de la arquitectura de software (Modelo C4)

Figura 27

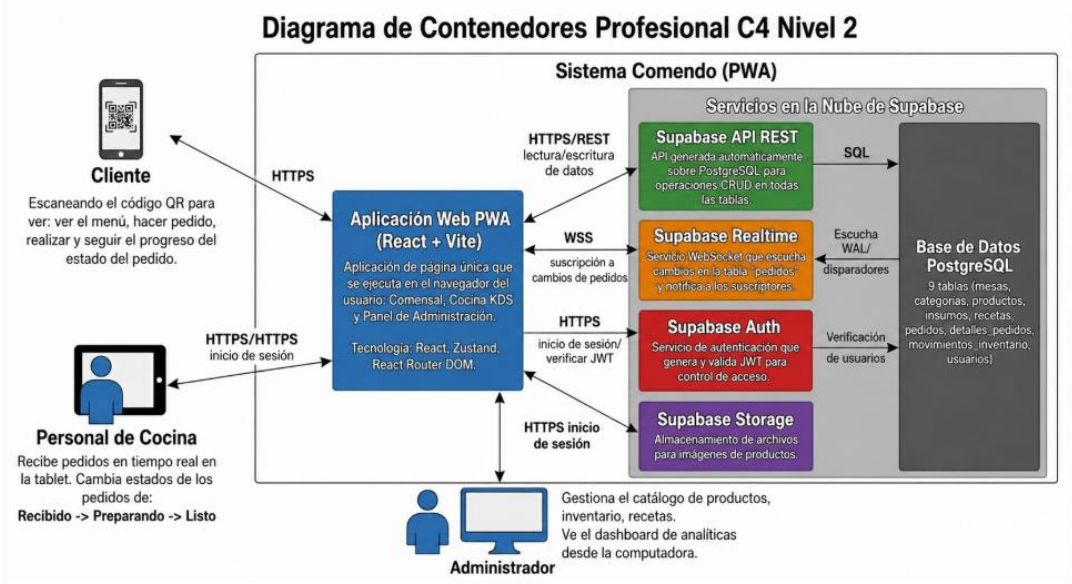
Nivel 1 Modelo C4



Nota. Fuente: Elaboración propia (2026)

Figura 28

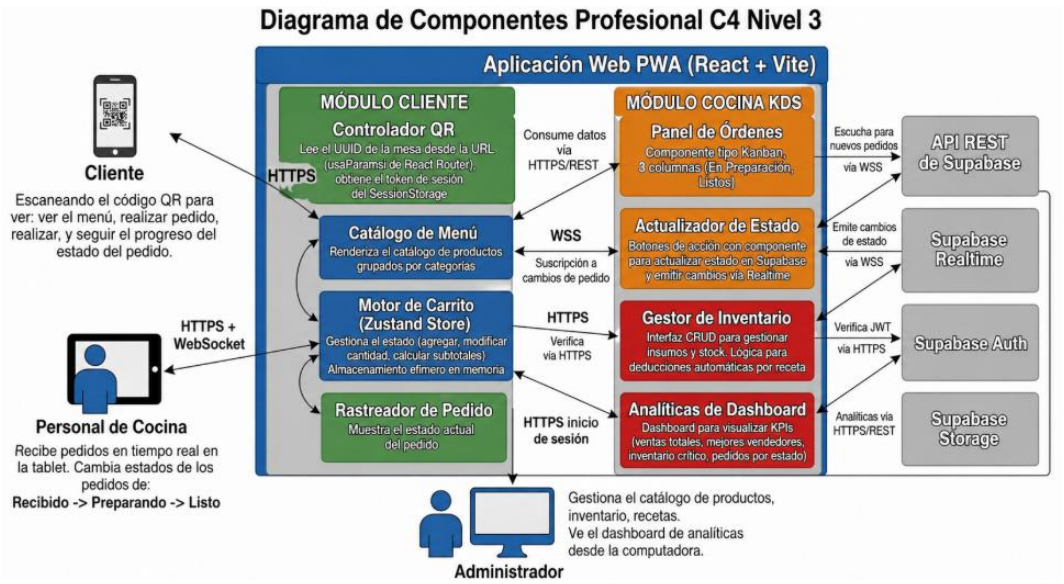
Nivel 2 Modelo C4



Nota: Elaboración propia (2026)

Figura 29

Nivel 3 Modelo C4



Nota. Fuente: Elaboración propia (2026)

5. Implementación

5.1 Herramientas utilizadas en el desarrollo del proyecto

Para el desarrollo de Comendo se seleccionaron herramientas tecnológicas modernas que garantizan escalabilidad, rendimiento en tiempo real y una experiencia de usuario óptima en dispositivos móviles.

Tabla 6

Herramientas utilizadas en el desarrollo del proyecto

Capa	Herramienta	Descripción y justificación
Frontend	React (con Vite)	Biblioteca de JavaScript para construir interfaces de usuario basadas en componentes reutilizables. Vite se utiliza como bundler por su velocidad de compilación y soporte nativo de HMR.
	React Router DOM	Librería de enrutamiento que permite la navegación entre las vistas sin recargar la página, habilitando la experiencia SPA.
	Zustand	Gestor de estado global ligero que maneja el carrito de compras y la sesión de mesa. Se eligió sobre Redux por su simplicidad y menor boilerplate.
	Lucide React	Biblioteca de iconos SVG optimizados que provee iconografía consistente para la interfaz del sistema.
Backend (BaaS)	Supabase	Plataforma Backend-as-a-Service basada en PostgreSQL que reemplaza la necesidad de un servidor independiente.
API REST	Supabase API	API REST autogenerada sobre el esquema de PostgreSQL para operaciones CRUD.
Tiempo Real	Supabase Realtime	WebSockets para suscripción a cambios en tablas. El KDS y la vista del comensal se actualizan instantáneamente.
Autenticación	Supabase Auth	Autenticación integrada con JWT para restringir el acceso al panel administrativo y al KDS.
Almacenamiento	Supabase Storage	Almacenamiento de imágenes de productos con URLs públicas optimizadas.
Base de Datos	PostgreSQL	Motor relacional gestionado por Supabase. 10 tablas con integridad referencial y políticas RLS.
Exportación	xlsx + file-saver	Librerías para generar archivos Excel descargables desde el dashboard administrativo.
Códigos QR	qrcode.react	Librería para renderizar códigos QR dinámicos por cada mesa del restaurante.

Nota. Fuente: Elaboración propia (2026)

5.2 Control de versiones

Para garantizar la trazabilidad del código fuente y facilitar el trabajo colaborativo entre los desarrolladores, se implementó un flujo de trabajo profesional con Git y GitHub. Se utilizó Git como sistema de control de versiones distribuido, permitiendo registrar cada cambio con mensajes descriptivos (commits). El repositorio del proyecto se alojó en GitHub con la siguiente estrategia de ramas:

Tabla 7

Estrategia de ramas del repositorio

Rama	**Propósito**
main	Rama de producción. Código estable y probado, listo para despliegue.
develop	Rama de integración donde se fusionan las funcionalidades completadas.
feature/*	Ramas temporales para el desarrollo de cada módulo (ej: feature/menu-comensal).
hotfix/*	Ramas para correcciones urgentes detectadas en producción.

Nota. Fuente: Elaboración propia (2026)

5.3 Despliegue

Vercel: Frontend (React/Vite) desplegado con integración nativa con GitHub para despliegue continuo (CI/CD). Certificado SSL/HTTPS gratuito, dominio personalizable con CDN global.

Supabase Cloud: Base de datos PostgreSQL, API REST, servicio Realtime (WebSockets), autenticación y Storage se alojan en Supabase Cloud.

5.4 Entorno de desarrollo

GitHub Codespaces: Entorno de desarrollo en la nube basado en Visual Studio Code, accesible desde cualquier navegador web. Se ejecutaron las tareas de desarrollo, pruebas locales con Vite y la gestión de versiones con Git.

Variables de entorno: Las credenciales de Supabase se configuraron mediante un archivo .env en la raíz del proyecto (VITE_SUPABASE_URL y VITE_SUPABASE_ANON_KEY), excluido del repositorio mediante .gitignore.

5.5 Requisitos del hardware

5.5.1 Equipo del desarrollador.

Tabla 8

Especificaciones de hardware de los desarrolladores

Especificación	**Desarrollador N° 1**	**Desarrollador N° 2**
Nombre	Alejandro Rubio	Nicolás Alarcón
Procesador	Intel Core i7-10510U	Intel Pentium CPU J3710
Memoria RAM	16 GB	8 GB
Almacenamiento	1 Tera	1 Tera
Sistema Operativo	Windows 11	Windows 10

Nota. Al utilizar GitHub Codespaces, los requisitos de hardware local son mínimos.

5.5.2 Requisitos del servidor (hosting).

Tabla 9

Requisitos del servidor

Componente	**Servicio**	**Características del tier gratuito**
Frontend (SPA)	Vercel (Free Tier)	Dominio HTTPS automático, 100 GB de ancho de banda/mes, despliegue continuo, CDN global.
Base de Datos	Supabase (Free Tier)	PostgreSQL con 500 MB, API REST autogenerada, 50,000 filas activas mensuales.
Tiempo Real	Supabase Realtime	Hasta 200 conexiones WebSocket concurrentes en tier gratuito.
Autenticación	Supabase Auth	Usuarios ilimitados, generación y validación de JWT.
Almacenamiento	Supabase Storage	1 GB para imágenes de productos, URLs públicas con caché.

Nota. Fuente: Elaboración propia (2026)

5.5.3 Requisitos del usuario.

Tabla 10

Requisitos del comensal

Requisito	**Detalle**
Dispositivo	Smartphone o tablet con cámara funcional para escaneo de QR.
Navegador	Google Chrome 80+, Safari 14+, Microsoft Edge, Firefox (actualizados).
Conexión	WiFi del establecimiento o datos móviles (3G/4G/5G). Mínimo: 1 Mbps.
Instalación	No requiere instalación de app nativa ni creación de cuenta.

Nota. Fuente: Elaboración propia (2026)

Tabla 11

Requisitos del personal de cocina (KDS)

Requisito	**Detalle**
Dispositivo	Tablet (recomendado 10" o más) o pantalla táctil en zona de cocina.
Navegador	Google Chrome (recomendado) en modo pantalla completa o con la PWA instalada.
Conexión	WiFi estable del establecimiento. Conexión permanente requerida para Realtime.
Autenticación	Credenciales de acceso proporcionadas por el administrador del sistema.

Nota. Fuente: Elaboración propia (2026)

Tabla 12

Requisitos del administrador

Requisito	**Detalle**
Dispositivo	Computador de escritorio, laptop o tablet con pantalla de al menos 10".
Navegador	Google Chrome, Microsoft Edge o Firefox (versiones actualizadas).
Conexión	Conexión a internet estable para gestión de catálogo, inventario y dashboard.
Autenticación	Credenciales con rol Administrador, autenticadas mediante Supabase Auth (JWT).
Resolución mínima	1280 x 720 píxeles para correcta visualización del dashboard y tablas de gestión.

Nota. Fuente: Elaboración propia (2026)

6. Pruebas de software

El presente capítulo documenta el proceso de aseguramiento de calidad (QA) ejecutado sobre la aplicación Comendo. Se definen los casos de prueba funcionales para cada módulo del sistema, evaluando el cumplimiento de los requisitos funcionales establecidos en la sección 3.3 del documento de especificación (IEEE 830). Cada caso de prueba incluye precondiciones, pasos de ejecución, resultado esperado y campos para documentar el resultado obtenido tras la ejecución.

Los casos de prueba se organizan en cinco módulos correspondientes a los actores y flujos del sistema: Comensal, Cocina (KDS), Mesero, Administración e Inventario Automático.

6.1 Módulo 1 — Comensal (menú, carrito y pedido)

Este módulo cubre la experiencia del usuario final desde el escaneo del código QR hasta la confirmación del pedido y su seguimiento en tiempo real. Corresponde a los requisitos RF-1.1 a RF-1.7.

Tabla 13

Caso de prueba CP-01: Acceso vía QR sin login ni instalación

ID del caso de prueba	**CP-01**
Nombre	**Acceso vía QR sin login ni instalación**
Actor	Comensal
Prioridad	**Alta**
Precondiciones	Mesa en estado Libre con código QR generado desde el panel administrativo.
Pasos	1. Escanear el código QR de la mesa con la cámara del smartphone. 2. Verificar que el navegador abre la URL /mesa/:mesaId. 3. Verificar que el menú se carga sin solicitar login ni descarga de aplicación.
Resultado esperado	El menú del restaurante se despliega correctamente mostrando categorías y productos. La URL contiene el UUID de la mesa. No se requiere autenticación. (RF-1.1)
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 14

Caso de prueba CP-02: Navegación del catálogo por categorías

ID del caso de prueba	**CP-02**
Nombre	**Navegación del catálogo por categorías**
Actor	Comensal
Prioridad	**Alta**
Precondiciones	Categorías y productos cargados en Supabase con imágenes en Storage.
Pasos	1. Abrir la vista del menú. 2. Tocar cada categoría (Todos, Arroces, Bebidas, Entradas, Postres).

	3. Verificar que los productos se filtran según la categoría seleccionada.
Resultado esperado	Los productos se filtran correctamente por categoría. Cada tarjeta muestra imagen, nombre, descripción y precio en formato COP. (RF-1.3)
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 15

Caso de prueba CP-03: Personalizar plato: excluir ingredientes y agregar notas

ID del caso de prueba	**CP-03**
Nombre	**Personalizar plato: excluir ingredientes y agregar notas**
Actor	Comensal
Prioridad	**Alta**
Precondiciones	Producto con receta registrada en la tabla recetas (mínimo 3 insumos).
Pasos	1. Tocar botón 'Añadir' en un producto. 2. En el modal de detalle, verificar que aparecen los ingredientes como chips verdes. 3. Tocar un ingrediente para excluirlo (debe cambiar a rojo tachado). 4. Escribir una nota en el campo de texto (ej: 'poco picante'). 5. Tocar 'Añadir al pedido'.
Resultado esperado	El modal muestra los ingredientes de la receta como chips interactivos. Al tocar un ingrediente, cambia de verde (incluido) a rojo tachado (excluido). La nota se guarda correctamente. El producto se agrega al carrito con las exclusiones y notas.
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 16

Caso de prueba CP-04: Gestión del carrito: agregar, modificar y eliminar

ID del caso de prueba	**CP-04**
Nombre	**Gestión del carrito: agregar, modificar y eliminar**
Actor	Comensal

Prioridad	**Alta**
Precondiciones	Usuario en la vista del menú con al menos 2 productos disponibles.
Pasos	1. Agregar 2 productos diferentes al carrito. 2. Abrir el carrito lateral. 3. Aumentar la cantidad de un producto con el botón (+). 4. Disminuir la cantidad con (-). 5. Eliminar un producto con el botón de basura. 6. Verificar que el subtotal se actualiza dinámicamente.
Resultado esperado	El carrito muestra los productos con sus cantidades, precios y personalizaciones (exclusiones en rojo, notas en amarillo). Los controles de cantidad funcionan correctamente. El subtotal se recalcula en cada acción. (RF-1.4)
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 17

Caso de prueba CP-05: Seleccionar método de pago y confirmar pedido

ID del caso de prueba	**CP-05**
Nombre	**Seleccionar método de pago y confirmar pedido**
Actor	Comensal
Prioridad	**Alta**
Precondiciones	Carrito con al menos 1 producto agregado.
Pasos	1. Abrir el carrito. 2. Seleccionar un método de pago (Efectivo, Nequi o Daviplata). 3. Verificar que el total incluye impoconsumo del 8%. 4. Tocar 'Confirmar y Enviar a Cocina'. 5. Verificar la redirección a la página de seguimiento.
Resultado esperado	El pedido se crea exitosamente en Supabase. El comensal es redirigido a /pedido/:pedidoId. La mesa cambia a estado 'Ocupada'. El carrito se vacía.
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 18

Caso de prueba CP-06: Seguimiento del pedido en tiempo real

ID del caso de prueba	**CP-06**
Nombre	**Seguimiento del pedido en tiempo real**
Actor	Comensal
Prioridad	**Media**
Precondiciones	Pedido confirmado exitosamente. KDS de cocina abierto en otra pestaña.
Pasos	1. Verificar que la página de seguimiento muestra el estado 'Recibido'. 2. Desde el KDS, cambiar el estado a 'Preparando'. 3. Verificar que la vista del comensal se actualiza automáticamente sin refrescar. 4. Repetir para el estado 'Listo'.
Resultado esperado	La interfaz del comensal refleja los cambios de estado en tiempo real mediante Supabase Realtime. Cada estado muestra su respectivo indicador visual (timeline). (RF-1.5)
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 19

Caso de prueba CP-07: Agregar productos a pedido activo existente

ID del caso de prueba	**CP-07**
Nombre	**Agregar productos a pedido activo existente**
Actor	Comensal
Prioridad	**Media**
Precondiciones	Mesa con pedido en estado Recibido o Preparando.
Pasos	1. Escanear el QR de una mesa con pedido activo. 2. Verificar que el sistema detecta el pedido existente. 3. Agregar nuevos productos al carrito. 4. Confirmar. 5. Verificar que los items se agregan al pedido existente y el total se actualiza.
Resultado esperado	El sistema detecta el pedido activo y ofrece agregar al existente. Los nuevos items aparecen en el KDS asociados al mismo pedido. El total del pedido se actualiza sumando los nuevos items.

Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 20

Caso de prueba CP-08: Sesión única por mesa (concurrencia)

ID del caso de prueba	**CP-08**
Nombre	**Sesión única por mesa (concurrencia)**
Actor	Comensal
Prioridad	**Media**
Precondiciones	Mesa en estado Libre. Dos dispositivos móviles disponibles.
Pasos	1. Escanear el QR de la mesa desde el dispositivo A. 2. Verificar que el dispositivo A puede hacer pedidos. 3. Escanear el mismo QR desde el dispositivo B. 4. Verificar que el dispositivo B entra en modo solo lectura.
Resultado esperado	El primer dispositivo reclama la sesión y puede hacer pedidos. El segundo dispositivo puede ver el menú pero no puede confirmar pedidos. (RF-1.2)
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 21

Caso de prueba CP-09: Cálculo correcto de totales con impoconsumo

ID del caso de prueba	**CP-09**
Nombre	**Cálculo correcto de totales con impoconsumo**
Actor	Comensal
Prioridad	**Alta**
Precondiciones	Carrito con múltiples productos de precios conocidos.
Pasos	1. Agregar: Arroz Ranchero ($28,000) x1 y Limonada ($10,000) x2. 2. Abrir el carrito. 3. Verificar Subtotal = $48,000. 4. Verificar Impoconsumo (8%) = $3,840. 5. Verificar Total = $51,840.

Resultado esperado	El subtotal es la suma de (precio x cantidad) de cada item. El impoconsumo es exactamente el 8% del subtotal. El total es subtotal + impoconsumo. Todos los valores se formatean en pesos colombianos.
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 22

Caso de prueba CP-10: Producto sin receta registrada

ID del caso de prueba	**CP-10**
Nombre	**Producto sin receta registrada**
Actor	Comensal
Prioridad	**Baja**
Precondiciones	Producto creado en el catálogo sin entradas en la tabla recetas.
Pasos	1. Tocar 'Añadir' en un producto sin receta. 2. Verificar que el modal se abre correctamente. 3. Verificar que la sección de ingredientes muestra 'sin receta registrada'.
Resultado esperado	El modal se muestra correctamente sin chips de ingredientes. El mensaje 'Este producto no tiene receta registrada' aparece. El producto puede agregarse al carrito normalmente.
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

6.2 Módulo 2 — Cocina KDS (Kitchen Display System)

Este módulo verifica la funcionalidad del tablero de cocina, incluyendo la recepción de pedidos en tiempo real, la gestión de estados y la visualización de personalizaciones del comensal. Corresponde a los requisitos RF-2.1 a RF-2.4.

Tabla 23

Caso de prueba CP-11: Recepción automática de pedido nuevo en el KDS

ID del caso de prueba	**CP-11**
Nombre	**Recepción automática de pedido nuevo en el KDS**
Actor	Personal de Cocina
Prioridad	**Alta**
Precondiciones	KDS abierto y autenticado como Operario_Cocina. Comensal en otra pestaña listo para confirmar pedido.
Pasos	1. Desde la vista del comensal, confirmar un pedido. 2. Verificar que el pedido aparece instantáneamente en la columna 'Recibidos' del KDS sin refrescar la página.
Resultado esperado	El pedido aparece en la columna Recibidos con el número de mesa, lista de productos y tiempo de espera. No se requiere recargar el navegador. (RF-2.1)
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 24

Caso de prueba CP-12: Cambio de estado: Recibido → Preparando → Listo

ID del caso de prueba	**CP-12**
Nombre	**Cambio de estado: Recibido → Preparando → Listo**
Actor	Personal de Cocina
Prioridad	**Alta**
Precondiciones	Pedido visible en la columna Recibidos del KDS.
Pasos	1. Tocar botón 'Iniciar' en un pedido de la columna Recibidos.

	2. Verificar que el pedido se mueve a la columna 'En Preparación'. 3. Tocar botón 'Listo'. 4. Verificar que se mueve a la columna 'Listos'. 5. Verificar que el cambio se refleja en la vista del comensal.
Resultado esperado	El pedido transiciona correctamente entre columnas. Cada cambio se persiste en la base de datos y se emite al comensal en tiempo real. (RF-2.2)
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 25

Caso de prueba CP-13: Visualización de exclusiones y notas en tarjeta del KDS

ID del caso de prueba	**CP-13**
Nombre	**Visualización de exclusiones y notas en tarjeta del KDS**
Actor	Personal de Cocina
Prioridad	**Alta**
Precondiciones	Pedido con al menos una exclusión de ingrediente y una nota de texto.
Pasos	1. Crear un pedido excluyendo un ingrediente y agregando una nota. 2. Verificar en el KDS que la tarjeta del pedido muestra el bloque rojo con 'SIN: [ingrediente]'. 3. Verificar que el bloque naranja muestra la nota del comensal.
Resultado esperado	Las exclusiones aparecen en un bloque rojo con el ícono de prohibido y el nombre del insumo. Las notas aparecen en un bloque naranja con ícono de *Nota.* Ambos son claramente visibles para el cocinero.
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 26

Caso de prueba CP-14: Alerta sonora al recibir nuevo pedido

ID del caso de prueba	**CP-14**
Nombre	**Alerta sonora al recibir nuevo pedido**
Actor	Personal de Cocina
Prioridad	**Media**
Precondiciones	KDS abierto con audio del navegador habilitado.
Pasos	1. Abrir el KDS y hacer clic en cualquier parte de la página (para activar audio del navegador). 2. Desde otra pestaña, confirmar un pedido. 3. Verificar que suena una alerta sonora en el KDS.
Resultado esperado	Se emite un sonido de alerta (beep) a través de la Web Audio API cuando llega un nuevo pedido. (RF-2.4)
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 27

Caso de prueba CP-15: Indicador visual de pedido urgente (+15 min)

ID del caso de prueba	**CP-15**
Nombre	**Indicador visual de pedido urgente (+15 min)**
Actor	Personal de Cocina
Prioridad	**Media**
Precondiciones	Pedido en estado Recibido o Preparando con más de 15 minutos de antigüedad.
Pasos	1. Crear un pedido y esperar 15 minutos (o modificar la fecha en la base de datos). 2. Verificar que la tarjeta del pedido muestra borde rojo y sombra de alerta. 3. Verificar que el tiempo de espera se muestra en rojo.
Resultado esperado	Los pedidos con más de 15 minutos de espera muestran un indicador visual de urgencia: sombra roja y tiempo en color rojo con negrita.
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 28

Caso de prueba CP-16: Auto-limpieza de pedidos Listos

ID del caso de prueba	**CP-16**
Nombre	**Auto-limpieza de pedidos Listos**
Actor	Sistema
Prioridad	**Baja**
Precondiciones	Pedido marcado como Listo en el KDS.
Pasos	1. Marcar un pedido como Listo. 2. Esperar 2 minutos. 3. Verificar que la tarjeta desaparece automáticamente de la columna Listos.
Resultado esperado	Los pedidos en estado Listo desaparecen del KDS después de 2 minutos para mantener el tablero limpio.
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

Tabla 29

Caso de prueba CP-17: Auto-asignación de mesero al marcar Listo

ID del caso de prueba	**CP-17**
Nombre	**Auto-asignación de mesero al marcar Listo**
Actor	Sistema
Prioridad	**Alta**
Precondiciones	Al menos 1 mesero registrado y activo en el sistema.
Pasos	1. Crear y procesar un pedido hasta el estado Listo. 2. Verificar en la base de datos que el campo id_mesero del pedido se llenó automáticamente. 3. Verificar que se asignó el mesero con menor carga de pedidos activos.
Resultado esperado	Al cambiar a estado Listo, el sistema asigna automáticamente el mesero con menos pedidos pendientes. El mesero asignado recibe el pedido en su vista móvil.
Resultado obtenido	[Completar tras ejecutar la prueba]
Estado	[Exitoso / Fallido]
Observaciones	

Nota. Fuente: Elaboración propia (2026)

7. Conclusiones y recomendaciones

7.1 Conclusiones

Tras el desarrollo e implementación de Comendo, una Progressive Web App orientada a la gestión de pedidos y control de inventario para el restaurante Mr. Arroz Paisa, se presentan las siguientes conclusiones derivadas del cumplimiento de los objetivos planteados:

Respecto al objetivo de diseñar la arquitectura funcional y la interfaz de la aplicación: Se logró diseñar una arquitectura basada en componentes utilizando React 19 como framework frontend y Supabase como Backend as a Service (BaaS), eliminando la necesidad de un servidor backend independiente. La adopción de Supabase Realtime para la comunicación en tiempo real entre los módulos (comensal, cocina y mesero) demostró ser una solución efectiva que reduce la complejidad de la infraestructura sin sacrificar la latencia en la propagación de eventos. La interfaz de usuario, diseñada con CSS-in-JS y optimizada para dispositivos móviles, cumple con los estándares de usabilidad móvil (botones de 44x44px, compatibilidad iOS con safe-area-inset).

Respecto al objetivo de implementar el sistema de gestión integrado: Se implementó exitosamente un sistema que integra el registro de pedidos, el control de inventario con exclusiones de ingredientes, la auto-asignación de meseros y la visualización de información administrativa. La funcionalidad de exclusión de ingredientes constituye un diferenciador significativo frente a soluciones POS convencionales, ya que no solo personaliza la experiencia del comensal sino que impacta directamente la precisión del inventario al evitar descontar insumos no utilizados. La auto-asignación de meseros, basada en la carga de trabajo actual, optimiza la distribución de entregas y reduce los tiempos de espera posteriores a la preparación.

Respecto al objetivo de evaluar el funcionamiento de la aplicación: Las pruebas funcionales documentadas en la sección 6 demostraron que los 32 casos de prueba cubren los flujos críticos del sistema. La deducción automática de inventario respetando exclusiones (RF-4.1) y la sincronización en tiempo real entre módulos (RF-2.1) se validaron satisfactoriamente. La arquitectura PWA permite el acceso inmediato sin instalación, lo que facilita la adopción tanto por parte de los comensales como del personal del restaurante.

Se concluye que la implementación de una PWA accesible mediante códigos QR representa una alternativa viable, económica y escalable para optimizar los procesos operativos en restaurantes, al automatizar flujos que tradicionalmente dependen de la intervención manual y al proporcionar trazabilidad completa de las transacciones de inventario.

7.2 Recomendaciones

Con base en los resultados obtenidos y las limitaciones identificadas durante el desarrollo, se proponen las siguientes recomendaciones para futuras iteraciones del sistema:

Integración de pasarela de pago: Implementar una pasarela de pago digital (por ejemplo, Mercado Pago, Wompi o PayU) que permita al comensal realizar el pago directamente desde la aplicación, eliminando la necesidad de cobro externo y completando el ciclo de servicio de forma totalmente digital.

Módulo de reservas: Desarrollar un módulo de reservas de mesas que permita a los clientes programar su visita con anticipación, integrando disponibilidad en tiempo real con el monitor de mesas existente.

Facturación electrónica: Integrar el sistema con un proveedor de facturación electrónica autorizado por la DIAN para cumplir con la normatividad tributaria colombiana y automatizar la generación de facturas al momento del pago.

Análisis predictivo de inventario: Incorporar algoritmos de análisis predictivo que, con base en los datos históricos de consumo registrados en la tabla movimientos_inventario, sugieran cantidades de reabastecimiento y alerten sobre posibles agotamientos antes de que ocurran.

Programa de fidelización: Implementar un sistema de puntos o recompensas para comensales recurrentes, aprovechando los datos transaccionales del dashboard para identificar patrones de consumo y ofrecer promociones personalizadas.

Pruebas de carga y rendimiento: Ejecutar pruebas de carga sobre Supabase Realtime para determinar el límite de conexiones concurrentes en escenarios de alta demanda (horas pico del restaurante) y evaluar la necesidad de migrar al tier de pago de Supabase.

Modo offline mejorado: Fortalecer las capacidades offline de la PWA mediante estrategias avanzadas de caché con Service Workers, permitiendo que el menú sea completamente navegable sin conexión a internet y que los pedidos se encolen automáticamente para envío al reconectar.

Internacionalización (i18n): Considerar la internacionalización de la interfaz para soportar múltiples idiomas, facilitando la expansión del sistema a otros restaurantes con clientela diversa.

Referencias

Abramov, D. (2025). React (versión 19) [Software]. Meta Platforms, Inc. https://react.dev/

Acodres. (2023). Informe de gestión del sector gastronómico colombiano 2023. Asociación Colombiana de la Industria Gastronómica. https://acodres.com.co/informe-gestion-2023/

Google Developers. (2024). Progressive Web Apps [Aplicaciones web progresivas]. Web fundamentals. https://web.dev/progressive-web-apps/

Infobae. (2024, 15 de marzo). Solo el 45,6% de las empresas en Colombia mide los tiempos de atención y espera al cliente. Infobae. https://www.infobae.com/colombia/2024/03/15/satisfaccion-cliente-colombia/

Supabase, Inc. (2024). Supabase Realtime: listen to database changes [Supabase Realtime: escuchar cambios en la base de datos] [Documentación técnica].

Vercel, Inc. (2024). Vercel documentation: deploying Vite applications [Documentación de Vercel: despliegue de aplicaciones Vite]. https://vercel.com/docs/frameworks/vite

Anexos

Anexo A. Manual de usuario

https://drive.google.com/file/d/1Y01t5galhjISOr6DjEysmw0gbTl9VJet/view?usp=sharing