

Sistema de Pedidos General

Santiago Melo
Didier Cardenas
Jean Paul García

3/11/2025

Análisis y Desarrollo de Software

Paola Gutierrez Mendieta

“PREX COL existe para conquistar la logística: optimizar, digitalizar y dominar cada ruta que conecta el progreso.”

Índice

- 1. Introducción**
- 2. Justificación**
- 3. Objetivos**
 - 3.1 General
 - 3.2 Específicos
- 4. Identificación del Problema y Desarrollo Metodológico**
 - 4.1 Solución
 - 4.2 Ingresos
 - 4.3 Valor Diferencial
 - 4.4 Levantamiento de Datos
 - 4.5 Metodología del Software
 - 4.6 Arquitectura del Software
 - 4.7 Ciclo de Vida
 - 4.8 Fases del Ciclo de Vida
- 5. Requerimientos del Software**
- 6. Anexos**

1. Introducción

El presente documento describe el desarrollo metodológico del Sistema de Pedidos Generales, una aplicación orientada a la gestión integral de pedidos entre clientes y administradores. El sistema permite el registro, autenticación, visualización de productos, creación y consulta de pedidos, y gestión de pagos dentro de un entorno seguro, escalable y centralizado.

La solución está construida sobre:

- **Backend:** Django + Django REST Framework
- **Frontend móvil:** React Native
- **Base de datos:** SQLite (desarrollo) / MySQL (producción)

Estos componentes proporcionan una arquitectura modular que facilita la escalabilidad y la incorporación de futuros módulos como informes avanzados, pasarelas de pago o seguimiento logístico.

2. Justificación

La digitalización de pedidos reduce los errores manuales, mejora la trazabilidad y aumenta la satisfacción del cliente. Este sistema sirve como base tecnológica reutilizable para negocios que requieren gestionar productos, usuarios y pedidos desde una aplicación móvil. Asimismo, PREXCOL establece un enfoque de crecimiento progresivo que contempla la expansión hacia logística avanzada y servicios integrados.

3. Objetivos

3.1 Objetivo General

Desarrollar un sistema de pedidos que permita la gestión segura de usuarios, productos y solicitudes mediante una aplicación móvil conectada a un backend RESTful.

N.º Objetivo	Indicador de Éxito
1 Implementar el backend en Django con modelos y endpoints REST.	Base de datos operativa + Endpoints funcionando + Migraciones registradas.
2 Desarrollar la app móvil en React Native con autenticación y acceso a catálogo.	Usuario puede registrarse, iniciar sesión y visualizar productos.
3 Implementar creación y consulta de pedidos con estados.	Pedido pasa por estados (pendiente → procesado → entregado).
4 Garantizar seguridad en acceso y manejo de credenciales.	Contraseñas cifradas + JWT habilitado + rutas protegidas.
5 Realizar pruebas técnicas, documentar y preparar el proyecto para ampliaciones futuras.	Informe de pruebas y documentación final aprobada.

4.1 Solución

Proveer un sistema centralizado que permita a los usuarios registrarse, autenticarse y acceder a una interfaz inicial (home) desde la app móvil, garantizando integridad y confidencialidad de credenciales y datos.

4.2 Ingresos

Fases futuras de monetización: suscripciones administrativas, integración de pasarelas de pago, módulos premium (reportes avanzados) y marketplaces.

4.3 Valor diferencial del proyecto

Arquitectura modular que permite reutilización y escalamiento: separar claramente backend (Django REST) y frontend móvil (React Native) y exponer endpoints bien documentados para futuras integraciones.

4.4 Levantamiento de datos

1. Módulo de Usuarios

- Permite registrar nuevos usuarios, iniciar sesión y cerrar sesión. Controla la información básica de cada persona, como nombre, correo y contraseña, y asegura el acceso al sistema de manera protegida.

2. Módulo de Productos

- Administra los productos disponibles para pedidos. Permite agregar, editar o eliminar productos con su respectiva información: nombre, descripción, precio e imagen.

3. Módulo de Pedidos

- Gestiona los pedidos realizados por los usuarios. Permite crear un pedido seleccionando productos, ver el resumen, el total a pagar y el estado del pedido (pendiente, entregado, etc.).

4. Módulo de Pagos

- Registra los métodos de pago utilizados y asocia cada pago a su pedido correspondiente. Los pagos pueden hacerse de forma simulada o manual.

5. Módulo de Notificaciones

- Envía mensajes o alertas dentro de la aplicación para informar al usuario sobre registros, pedidos o actualizaciones de estado.

6. Interfaz Móvil

- Es la parte visual del sistema. Muestra las pantallas de login, registro, productos, pedidos y perfil. Está desarrollada con React Native y se conecta al backend hecho en Django.

7. Módulo de Seguridad

- Protege los datos de usuarios y pedidos mediante contraseñas cifradas y validación de acceso. Controla las sesiones y evita el uso no autorizado del sistema.

Metodología Ágil: Marco SCRUM

Roles del Equipo

Rol	Responsabilidad
Product Owner	Define los requerimientos, prioriza funcionalidades y valida los entregables.
Scrum Master	Facilita el proceso, remueve impedimentos y asegura la aplicación de la metodología ágil.
Equipo de Desarrollo	Implementa backend, frontend, realiza pruebas, documentación y despliegues.

Estructura del Sprint

- Duración por sprint: 1 semana
- Objetivo por sprint: entregar un incremento funcional verificable
- Ceremonias aplicadas:
 - Daily Meeting (Diario): Seguimiento del avance y bloqueo.
 - Sprint Review (Reseña): Presentación de las funcionalidades completadas.
 - Sprint Retrospective (Retrospectiva): Identificación de oportunidades de mejora para el siguiente sprint.

Cronograma de Sprints

Semana 1: Implementación de la Arquitectura Base

Se estableció la estructura inicial de todo el proyecto:

- Configuración del entorno virtual y proyecto base en Django.
- Estructuración de la aplicación móvil en React Native.
- Definición de carpetas, dependencias y versiones del sistema.

Módulos creados:

- Modelo de Super Admin
- Modelo de Admin
- Módulo de Usuarios

Se implementaron funciones iniciales y rutas básicas para verificar la comunicación API REST → Base de Datos.

Semana 2: Creación de la Base de Datos y Endpoints

Se diseñó la estructura de datos y se configuró la base de datos (tablas, relaciones y restricciones).

- Implementación de modelos en Django
- Ejecución de migraciones
- Configuración de conexión entre Backend y la BD
- Creación de endpoints REST para intercambio de datos con la App móvil

Módulos finalizados:

- Módulo de Productos
- Módulo de Pedidos

Semana 3: Integración Frontend - Backend y Seguridad

Se conectó el frontend con el backend mediante peticiones HTTP a la API REST.

- Implementación de registro e inicio de sesión desde la App
- Visualización de productos en tiempo real
- Encriptación de contraseñas
- Autenticación mediante JWT

- Fortalecimiento del Módulo de Seguridad para garantizar protección de datos y accesos.

Resultado:

El sistema ya permite que un usuario real se registre, inicie sesión y consulte productos desde la aplicación móvil, con comunicación segura hacia el servidor.

REQUISITOS FUNCIONALES

RF1 – Registro de Usuarios

Descripción:

El sistema debe permitir que nuevos usuarios se registren proporcionando su nombre, correo electrónico y contraseña.

RPS: Al completar el formulario, los datos se guardarán en la base de datos y el usuario recibirá confirmación de registro exitoso.

RF2 – Inicio de Sesión

Descripción:

El sistema debe permitir a los usuarios iniciar sesión ingresando su correo y contraseña registrados.

RPS: El sistema validará las credenciales y, si son correctas, redirigirá al usuario a la pantalla principal (Home).

RF3 – Cierre de Sesión

Descripción:

El usuario podrá cerrar sesión desde la aplicación móvil para finalizar su acceso.

RPS: El token de autenticación será eliminado del almacenamiento local y el usuario volverá a la pantalla de inicio de sesión.

RF4 – Gestión de Productos

Descripción:

El administrador podrá agregar, modificar y eliminar productos disponibles para pedidos.

RPS: Los cambios se reflejarán automáticamente en el catálogo de productos que visualizan los usuarios.

RF5 – Visualización de Productos

Descripción:

Los usuarios podrán consultar los productos disponibles, incluyendo nombre, precio, descripción e imagen.

RPS: El sistema mostrará los productos activos en la interfaz móvil a través de peticiones al servidor.

RF6 – Creación de Pedidos

Descripción:

Los usuarios podrán crear pedidos seleccionando productos del catálogo y definiendo cantidad y método de pago.

RPS: Cada pedido se registrará en la base de datos con un estado inicial de “pendiente”.

RF7 – Consulta de Pedidos

Descripción:

El sistema debe permitir a los usuarios y administradores consultar el historial de pedidos realizados.

RPS: Se mostrará una lista con los pedidos y su estado actual (pendiente, en proceso, entregado).

RF8 – Gestión de Pagos

Descripción:

El sistema registrará los pagos asociados a cada pedido.

RPS: Se almacenará la información del método de pago y monto en la base de datos.

RF9 – Envío de Notificaciones

Descripción:

El sistema notificará al usuario sobre el registro exitoso, creación de pedidos y cambios de estado.

RPS: Las notificaciones se mostrarán en la interfaz móvil de manera automática o al actualizar la vista.

RF10 – Seguridad de Acceso

Descripción:

El sistema debe validar que solo usuarios autenticados accedan a las funciones protegidas.

RPS: Se utilizará un token JWT para identificar cada sesión activa y restringir el acceso no autorizado.

Requerimientos No Funcionales (RNF)

RNF1 – Rendimiento

Descripción:

El sistema debe responder a las solicitudes de los usuarios en un tiempo máximo de 3 segundos.

RPS: Las peticiones API deben procesarse de forma rápida y sin interrupciones.

RNF2 – Seguridad

Descripción:

Toda la información sensible debe transmitirse y almacenarse de forma segura.

RPS: Las contraseñas se cifrarán y los accesos se validarán mediante autenticación JWT.

RNF3 – Usabilidad

Descripción:

La interfaz móvil debe ser intuitiva y fácil de utilizar por cualquier tipo de usuario.

RPS: Los botones, menús y formularios deben estar claramente identificados y accesibles.

RNF4 – Escalabilidad

Descripción:

El sistema debe permitir la integración de nuevos módulos sin afectar las funciones existentes.

RPS: La arquitectura se diseñará en módulos independientes que faciliten el crecimiento del proyecto.

RNF5 – Compatibilidad

Descripción:

La aplicación móvil debe ser compatible con dispositivos Android.

RPS: El sistema será probado en diferentes versiones del sistema operativo Android para garantizar su correcto funcionamiento.

RNF6 – Mantenibilidad

Descripción:

El código del sistema debe ser claro y documentado para facilitar futuras modificaciones.

RPS: Se seguirán buenas prácticas de programación y comentarios en el código fuente.

RNF7 – Disponibilidad

Descripción:

El sistema debe estar disponible para su uso en cualquier momento.

RPS: El servidor deberá tener una tasa de disponibilidad mínima del 95%.

RNF8 – Portabilidad**Descripción:**

El sistema debe poder trasladarse fácilmente a otros entornos o servidores.

RPS: Se emplearán contenedores o configuraciones que permitan la migración sin pérdida de información.

ARQUITECTURA GENERAL

Prex – Col utiliza una arquitectura MTV + Cliente – Servidor REST

Django = API REST + React + Privada + Web

esto permite la separación de frontend y backend

CAPAS

1. Estructura General

El sistema se compone de tres capas principales:

- **Capa de Presentación (Frontend):**
Desarrollada en React Native, esta capa se encarga de la interacción con el usuario. Contiene las pantallas de inicio, login, registro, productos, pedidos y perfil. El frontend consume los servicios del backend mediante peticiones HTTP (API REST) y muestra la información en tiempo real.
- **Capa Lógica o de Negocio (Backend):**
Implementada en Django, gestiona la lógica del sistema, validaciones, control de usuarios, pedidos y productos.
Aquí se definen los endpoints, los modelos de datos y los controladores que procesan la información proveniente del frontend.
- **Capa de Datos (Base de Datos):**
Utiliza SQLite como gestor de base de datos, donde se almacenan los registros de usuarios, productos, pedidos y pagos.
La conexión se realiza mediante el ORM (Object Relational Mapper) de Django, lo que permite manipular los datos de manera sencilla y segura.

2. Componentes Principales

- **API REST:** Permite la comunicación entre el backend y la aplicación móvil. Todas las peticiones (registro, login, consulta de productos, creación de pedidos) se realizan mediante endpoints definidos en Django Rest Framework.
- **Autenticación JWT:** Se utiliza para proteger las rutas y validar que solo los usuarios autenticados accedan a las funciones privadas.
- **Modelo de Datos:** Define las tablas principales (Usuarios, Productos, Pedidos, Pagos) y sus relaciones.
- **Interfaz Gráfica:** Implementada en React Native con un diseño limpio e intuitivo, orientado a dispositivos móviles Android.
- **Servidor de Aplicaciones:** Django ejecuta la lógica del sistema, procesando solicitudes y respuestas del cliente.

3. Flujo de Comunicación

Usuario



Aplicación Móvil (React Native)

↓ 1. Solicitud HTTP (login, ver productos, hacer pedido)

API REST (Django / DRF)

↓ 2. Validación y lógica

Modelos Django ↔ Base de Datos

↑ 3. Respuesta procesada (JSON)

API REST

↑ 4. Respuesta HTTP

Aplicación Móvil

↑ 5. Interfaz se actualiza (pantallas, mensajes, estados)

Esta arquitectura modular permite futuras ampliaciones, como integrar notificaciones push, geolocalización o pasarelas de pago reales sin alterar el núcleo de este sistema.

Ciclo de Vida del Software

El Ciclo de Vida del Software (CVS) describe las fases por las que pasó el proyecto desde su inicio hasta su finalización.

Para el desarrollo del Sistema de Pedidos General, se aplicó un enfoque ágil basado en Scrum, con iteraciones semanales que permitieron la entrega continua de avances y mejoras.

1. Fase de Planificación

En esta fase se definieron los objetivos del proyecto, el alcance (login, registro, home y gestión básica de pedidos) y las herramientas a utilizar: Django, React Native y SQLite. Se identificaron los roles del equipo (desarrollador, diseñador, tester) y se estableció un cronograma de cinco semanas con entregas semanales.

2. Fase de Análisis

Se recopilaron los requerimientos funcionales y no funcionales del sistema.

Se analizaron las necesidades del cliente, los procesos de pedido, registro y autenticación de usuarios, y se elaboró el modelo de datos.

El resultado de esta fase fue la definición detallada de cómo debía comportarse el sistema.

3. Fase de Diseño

Se diseñó la arquitectura general del sistema, definiendo los módulos principales, las entidades y las relaciones entre ellas.

Se elaboraron los diagramas conceptuales, la estructura de carpetas, y las pantallas base para la aplicación móvil.

El diseño UI/UX se enfocó en la facilidad de uso y la organización clara de los elementos.

Fase de Implementación

Durante esta etapa se desarrollaron los módulos en Django y React Native.

Se construyeron los modelos de datos, vistas y controladores en el backend, junto con las pantallas de login, registro y home en el frontend.

Se integraron ambas partes mediante la API REST y se añadieron las funciones básicas de autenticación y gestión de pedidos.

4. Fase de Pruebas

En esta fase se realizaron pruebas unitarias y funcionales para verificar el correcto funcionamiento de cada módulo.

Se validaron el registro e inicio de sesión de usuarios, la creación de pedidos, la comunicación entre frontend y backend, y la persistencia de datos en la base de datos.

Se corrigieron errores menores y se optimizaron las peticiones para mejorar el rendimiento general

5. Fase de Documentación y Entre

Finalmente, se elaboró la documentación técnica del sistema, detallando los procesos, herramientas y resultados obtenidos.

Esta fase garantiza que el proyecto pueda ser mantenido y mejorado en el futuro sin pérdida de información técnica.

PLAN DE ANÁLISIS

1. Identificación del problema: Los usuarios requieren una plataforma móvil para realizar pedidos fácilmente y que el administrador pueda gestionarlos.

2. Objetivo del sistema: Crear un sistema de pedidos general que optimice la comunicación entre clientes y administradores.

3. Recolección de requisitos: A través de entrevistas, encuestas y observación del proceso actual.

4. Modelado del sistema: Se definieron casos de uso, diagramas de clases y flujos de datos.

5. Diseño de arquitectura: Se estableció un modelo cliente-servidor con Django y React Native.

6. Análisis de riesgos: Seguridad, pérdida de conexión y validación de datos.

7. Planificación: Uso de metodología Scrum, con entregas semanales y pruebas funcionales.

Requerimientos de Hardware y Software

1. Requerimientos de Hardware

a. Servidor (Backend – Django / API REST)

Tipo	Requerimiento Mínimo	Requerimiento Recomendado
Procesador	Intel Core i3 o equivalente	Intel Core i5 o superior
Memoria RAM	4 GB	8 GB o más
Almacenamiento	50 GB HDD	100 GB SSD
Conectividad	Internet 10 Mbps	Internet 20 Mbps o superior
Sistema Operativo	Windows 10 / Ubuntu 20.04	Ubuntu Server 22.04 LTS

Descripción:

El servidor ejecuta el backend en Django, gestiona la base de datos y atiende las solicitudes de la aplicación móvil.

Se recomienda el uso de Linux (Ubuntu Server) debido a su estabilidad, seguridad y rendimiento para aplicaciones web en producción.

b. Cliente (Aplicación Móvil – React Native)

Tipo	Requerimiento Mínimo	Requerimiento Recomendado
Procesador	Quad-Core 1.8 GHz	Octa-Core 2.0 GHz o superior
Memoria RAM	2 GB	4 GB o más
Almacenamiento Interno	32 GB	64 GB o más
Sistema Operativo	Android 9 (Pie)	Android 12 o superior
Conectividad	Wi-Fi / Datos móviles	Wi-Fi estable (≥ 10 Mbps)

Descripción:

El cliente es la aplicación móvil usada por el usuario final. Debe ser capaz de ejecutar React Native de forma fluida para mostrar productos, pedidos y notificaciones sin retrasos.

2. Requerimientos de Software

a. Entorno de Desarrollo

Elemento	Requerimiento
Lenguajes de Programación	Python 3.12, JavaScript (ES6)
Frameworks	Django 5.x, Django REST Framework, React Native 0.75+
Gestor de Base de Datos	SQLite3 (desarrollo) / MySQL o PostgreSQL (producción)
Servidor Web	Gunicorn + Nginx (recomendado) / Apache (alternativa)
Control de Versiones	Git y GitHub
Entorno de Pruebas	Postman, VS Code, Android Studio
Sistema Operativo de Desarrollo	Windows 10/11 o Ubuntu Desktop

Descripción:

Estas herramientas permiten desarrollar, probar y depurar el sistema.

- Django maneja la lógica del backend y la generación de API REST.
- React Native proporciona la interfaz gráfica móvil multiplataforma.

b. Librerías y Dependencias Principales

- Django REST Framework: creación de APIs REST.
- Proxy Reverse: habilita la conexión entre backend y frontend.
- Django Simple JWT: autenticación mediante tokens.
- Axios: consumo de API desde React Native.
- React Navigation: manejo de pantallas, rutas y navegación interna.
- Expo / Metro Bundler: ejecución, depuración y pruebas en entornos móviles.

Consideraciones Técnicas

- **Migración de Base de Datos:**
El sistema está diseñado para comenzar con **SQLite** durante el desarrollo por simplicidad.
Al pasar a producción, la base de datos puede migrarse a **MySQL o PostgreSQL** sin pérdida de datos utilizando herramientas como dumpdata/loaddata o migrate y mysqldump.
- **Conectividad y Sincronización:**
La aplicación móvil requiere **conexión a internet estable** para enviar y recibir datos desde el servidor Django.
Las operaciones críticas como pedidos, registro y autenticación dependen de la sincronización en tiempo real.
- **Gestión de Dependencias:**
Se recomienda el uso de **entornos virtuales (venv o virtualenv)** para aislar las dependencias del backend y evitar conflictos con otras aplicaciones del sistema operativo.
- **Pruebas y Validación en Dispositivos Reales:**
Las pruebas deben realizarse tanto en **emuladores (Android Studio)** para depuración, como en **dispositivos físicos** para evaluar:
 - rendimiento real,
 - consumo de memoria,
 - calidad de la interfaz,
 - manejo de red en escenarios reales.

Anexos

CASOS DE USO

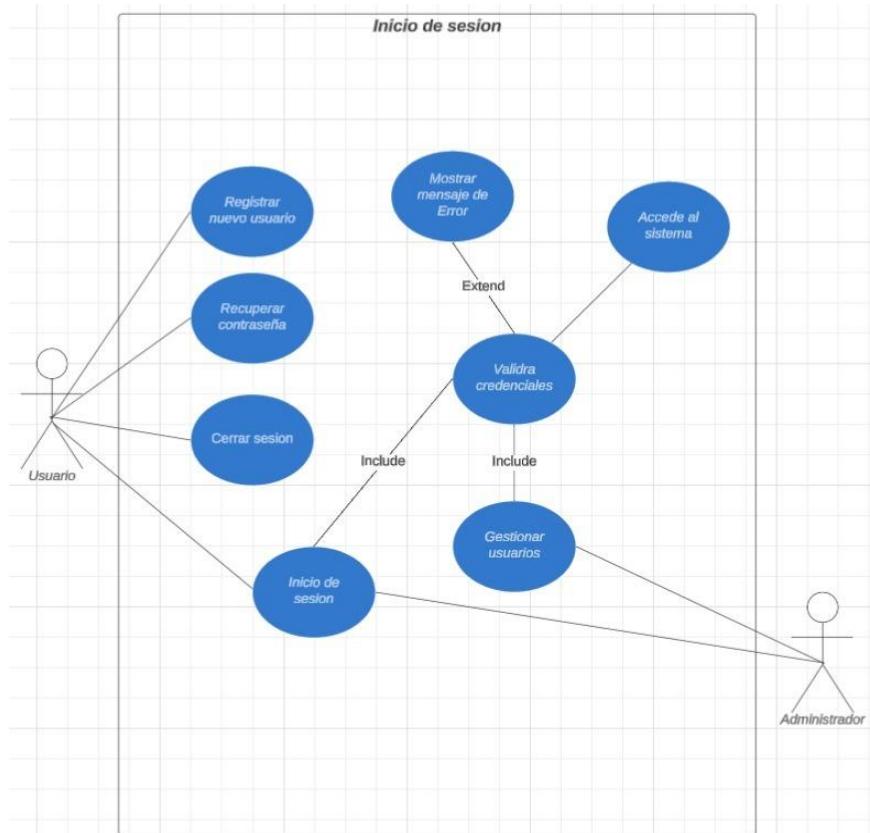


DIAGRAMA DE CLASES

Diagrama de Clases - Sistema PREXCOL

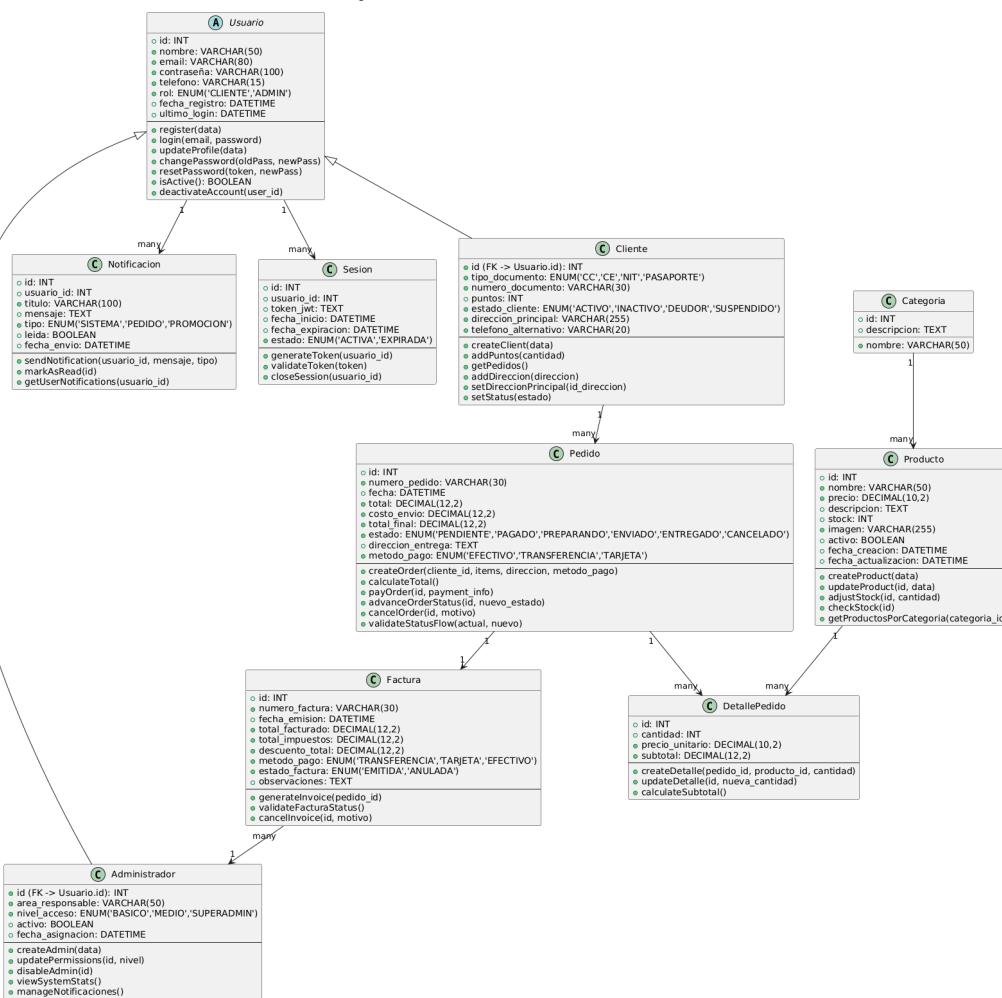


DIAGRAMA DE ACTIVIDADES (Flujo general)

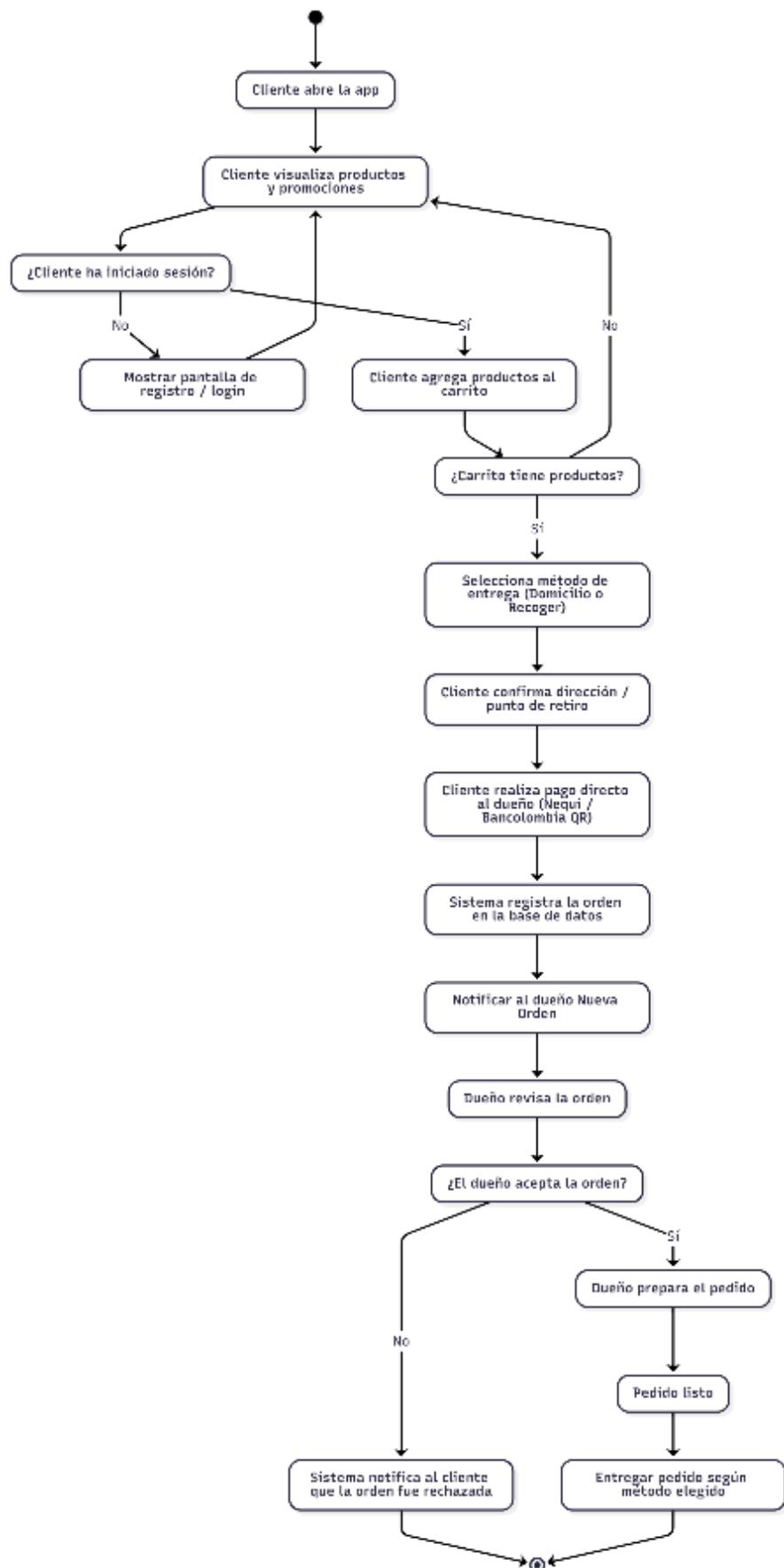
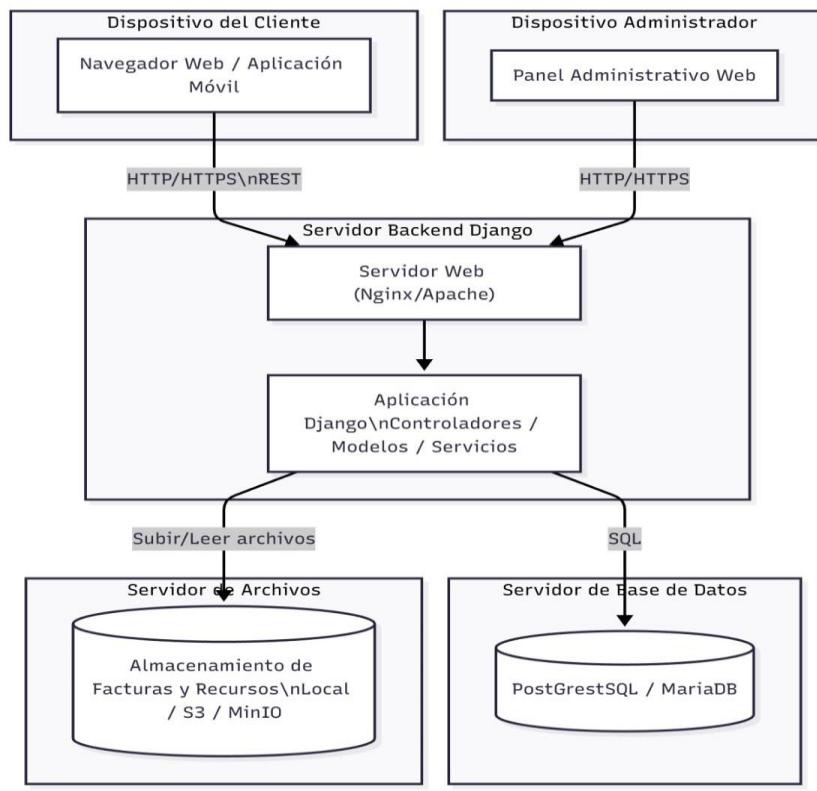


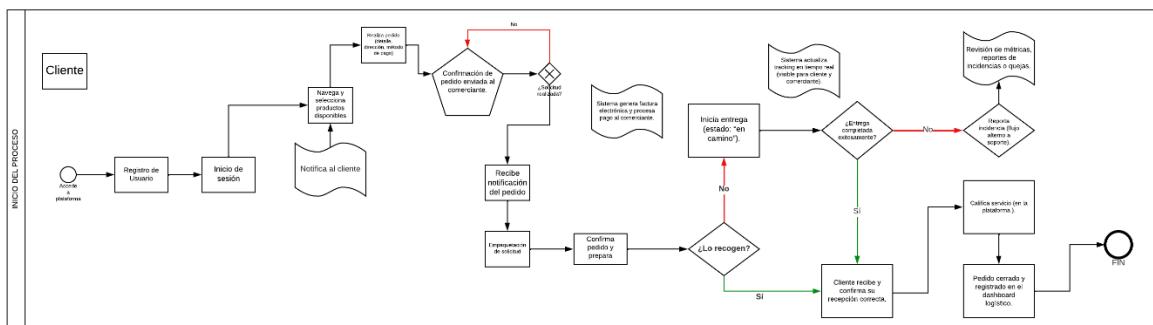
DIAGRAMA DE DESPLIEGUE



Conexiones:

- La app móvil se comunica con el servidor vía API REST.
- El servidor se conecta con la base de datos para operaciones CRUD

DIAGRAMA BPMN



HISTORIAS DE USUARIO HU1:

Código Actor	Descripción	Criterios de Aceptación
HU1 Usuario	Registrarse en la aplicación para crear una cuenta personal.	<ul style="list-style-type: none"> - El usuario debe ingresar nombre, correo y contraseña. - El sistema valida la información. - Se confirma el registro y se habilita el acceso. - El usuario ingresa correo y contraseña.
HU2 Usuario	Iniciar sesión para acceder a su cuenta previamente registrada.	<ul style="list-style-type: none"> - El sistema verifica credenciales. - Si son válidas, se permite el acceso. - Si son incorrectas, se muestra error.
HU3 Cliente	Visualizar el catálogo de productos disponible en la aplicación.	<ul style="list-style-type: none"> - El usuario puede ver una lista con nombre, imagen, precio y descripción de los productos. - La información debe actualizarse en tiempo real desde el servidor.
HU4 Cliente	Realizar y confirmar un pedido desde la aplicación.	<ul style="list-style-type: none"> - El usuario selecciona productos y cantidades. - Se muestra el valor total. - El sistema registra el pedido y genera un número de seguimiento.
HU5 Administrador	Gestionar el catálogo de productos (crear, editar, eliminar).	<ul style="list-style-type: none"> - El administrador puede agregar nuevos productos. - Puede actualizar información existente. - Puede desactivar productos cuando no estén disponibles. - El administrador puede ver la lista de pedidos ordenados por estado.
HU6 Administrador	Consultar y gestionar los pedidos recibidos para organizar la entrega.	<ul style="list-style-type: none"> - Puede actualizar el estado (Recibido, En preparación, Enviado, Entregado). - El sistema notifica al cliente los cambios.

MAPA DE NAVEGACION

FUNCIONALIDAD	CRUD	MODULO	REQUERIMIENTO
Cliente:			
Login	(C, R, U, D)	Usuarios	RF2, RF10
Registro	(C)	Usuarios	RF1
Home (Catálogo)	(R)	Productos	RF5
Detalle Producto	(R)	Productos	RF5
Carrito	(C, R, U, D)	Pedidos	RF6
Checkout (Pago)	(C, R, U, D)	Pagos	RF8
Pedidos (Historial)	(R)	Pedidos	RF7
Perfil de Usuario	(C, R, U)	Usuarios	RF3
Notificaciones	(R)	Notificaciones	RF9
Cerrar Sesión	(R, U)	Seguridad	RF10
Admin:			
Login Admin	(C, R, U, D)	Usuarios / Seguridad	RF2, RF10
Dashboard General	(R, U)	Interfaz Móvil / Control	
Gestión de Productos	(CRUD)	Productos	RF4, RF5
Gestión de Pedidos	(R, U, D)	Pedidos	RF7
Gestión de Usuarios	(R, U)	Usuarios	RF1, RF2
Configuración del Sistema	(C, R, U, D)	Seguridad	RF10
Cerrar Sesión	(R, U)	Seguridad	RF10

ESTRUCTURA JERARQUICA

CLIENTE

Login (C, R, U, D) – Módulo Usuarios – RF2, RF10

Registro (C) – Módulo Usuarios – RF1

Home (Catálogo) (R) – Módulo Productos – RF5

Detalle Producto (R) – Módulo Productos – RF5

Carrito (C, R, U, D) – Módulo Pedidos – RF6

Checkout (C, R, U, D) – Módulo Pagos – RF8

Confirmar Pedido – Módulo Pedidos – RF6

Pedidos (Historial) (R) – Módulo Pedidos – RF7

Perfil (C, R, U) – Módulo Usuarios – RF3

Notificaciones (R) – Módulo Notificaciones – RF9

Cerrar Sesión (R, U) – Módulo Seguridad – RF10

ADMIN

Login Admin (C, R, U, D) – Módulo Usuarios / Seguridad – RF2, RF10

Dashboard (R, U) – Módulo Interfaz Móvil / Control General

Productos (CRUD) – Módulo Productos – RF4, RF5

Pedidos (Gestión de Pedidos) (R, U, D) – Módulo Pedidos – RF7

Usuarios (R, U) – Módulo Usuarios – RF1, RF2

Configuración (C, R, U, D) – Módulo Seguridad – RF10

Cerrar Sesión (R, U) – Módulo Seguridad – RF10

CONDICIONES GENERALES DEL SISTEMA (RNF)

- **RNF1 Rendimiento:** respuesta < 3 s.
- **RNF2 Seguridad:** cifrado + JWT.
- **RNF3 Usabilidad:** interfaz intuitiva.
- **RNF4 Escalabilidad:** módulos independientes.
- **RNF5 Compatibilidad:** Android.
- **RNF6 Mantenibilidad:** código limpio y comentado.
- **RNF7 Disponibilidad:** servidor 95%.
- **RNF8 Portabilidad:** contenedores Docker / migrable.