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

El presente documento describe el desarrollo metodológico del **Sistema de Pedidos General**, una aplicación diseñada para facilitar la gestión de pedidos entre clientes y administradores.
El sistema busca optimizar el proceso de registro, autenticación y visualización de información mediante un entorno seguro, dinámico y escalable.
El desarrollo se realizó empleando el framework Django para el backend y React Native para la interfaz móvil, garantizando conectividad, seguridad y una experiencia de usuario moderna.

2. Justificación

En la actualidad, los negocios enfrentan la necesidad de digitalizar sus procesos para optimizar tiempo y recursos. Un sistema de pedidos permite reducir errores manuales, mejorar la trazabilidad de las solicitudes y aumentar la satisfacción del cliente.
Este proyecto se justifica en la necesidad de ofrecer una **base tecnológica sólida y reusable**, donde cualquier empresa pueda gestionar sus pedidos, usuarios y operaciones de manera sencilla. Además, sirve como punto de partida para futuros módulos como pagos, reportes o seguimiento de entregas.

Django: Lógica, base de datos, API.

React: Interfaz móvil/web que consume esa API.

Valores Corporativos de PREX COL

1. **Innovación constante**
   Buscamos siempre nuevas formas de optimizar la logística mediante tecnología, automatización y digitalización avanzada.
2. **Eficiencia y precisión**
   Cada proceso, cada envío y cada dato cuenta. En **PREX COL** todo se mueve con exactitud y excelencia operativa.
3. **Compromiso con el progreso**
   Promovemos el desarrollo logístico y económico del país, impulsando la evolución del transporte y los abastos hacia el futuro digital.
4. **Transparencia y confianza**
   Operamos con claridad en cada transacción y relación, garantizando seguridad y credibilidad para nuestros aliados y clientes.
5. **Solución de raíz**
   No parchamos problemas: los entendemos, los analizamos y los resolvemos desde su origen para evitar que vuelvan a aparecer.
6. **Visión sostenible**
   Apostamos por un crecimiento responsable, buscando eficiencia energética, reducción de desperdicios y respeto por el entorno.
7. **Ambición sin límites**
   Pensamos en grande. **PREX COL** no es solo una empresa: es el inicio de un imperio logístico que transformará la forma en que se mueve el progr

3. Objetivos

3.1 Objetivo General

Desarrollar un **Sistema de Pedidos General** que implemente el registro, login y acceso al entorno principal de la aplicación, para gestionar usuarios, pedidos y operaciones utilizando Django como backend y React Native como frontend.

3.2 Objetivos Específicos

1. Diseñar y estructurar la base de datos y el backend en Django para administrar usuarios, productos, pedidos, pagos y notificaciones, garantizando control de roles y seguridad mediante autenticación y cifrado.

2. Desarrollar la interfaz móvil en React Native que permita a los usuarios registrarse, iniciar sesión, visualizar productos, realizar pedidos, gestionar pagos y recibir notificaciones de estado de manera clara y accesible.

3. Realizar pruebas funcionales, de seguridad y usabilidad en todos los módulos, y documentar los procesos técnicos y operativos para asegurar el correcto funcionamiento, mantenimiento y escalabilidad del sistema PREXCOL.

Identificación del problema y Desarrollo metodológico

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.

METODOLOGIA AGIL: Tipo SCRUM

Semana 1: Implementación de los Módulos Principales

Durante esta semana se desarrollaron los módulos básicos del sistema, estableciendo la estructura inicial del proyecto tanto en **Django** como en **React Native**.
Se crearon las carpetas del backend, el entorno virtual y el proyecto base de Django. En el frontend, se configuró la aplicación móvil con React Native y sus dependencias iniciales.
Los módulos desarrollados en esta etapa fueron:

- Módulo de Usuarios
- Módulo de Productos
- Módulo de Pedidos

Cada módulo se implementó con funciones básicas y rutas iniciales para garantizar la comunicación entre las partes del sistema.

Semana 2: Creación de Bases de Datos y Conexiones

En esta etapa se diseñó y configuró la base de datos, creando las tablas correspondientes a los módulos definidos.
Se implementaron los modelos en Django y se realizaron las migraciones para almacenar usuarios, productos y pedidos.
También se configuraron las conexiones entre el backend y la base de datos, asegurando el correcto flujo de información y validación de datos.
Por último, se crearon endpoints API REST para permitir el intercambio de información con la aplicación móvil.

Semana 3: Integración con la Interfaz Móvil y Seguridad

Durante esta semana se trabajó en la integración del backend con la aplicación móvil en React Native, utilizando peticiones HTTP para el manejo de datos.
Se implementaron funciones de registro, inicio de sesión y visualización de productos desde la aplicación.
Además, se añadió seguridad básica mediante tokens de autenticación (JWT) y encriptación de contraseñas.
También se fortaleció el Módulo de Seguridad para proteger la información de los usuarios y controlar el acceso al sistema.

Semana 4: Diseño e Implementación de la Interfaz (UI/UX)

En esta fase se mejoró el diseño visual y la experiencia de usuario.
Se aplicaron principios básicos de UI/UX en las pantallas de inicio, login, registro, productos y pedidos.
Se optimizó la navegación entre vistas, la organización de los componentes y la presentación de los datos.
Además, se ajustó el funcionamiento de los módulos visuales (Interfaz Móvil, Notificaciones y Pagos) para que la aplicación fuera más intuitiva y agradable para el usuario final.

Semana 5: Pruebas y Documentación

En la última semana se realizaron pruebas funcionales para asegurar el correcto desempeño de cada módulo.
Se verificó el registro e inicio de sesión de usuarios, la creación de pedidos y la comunicación entre la aplicación y el servidor.
Finalmente, se elaboró la documentación técnica y metodológica del proyecto, incluyendo los resultados, conclusiones y recomendaciones para futuras mejoras o ampliaciones del sistema.

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

Modelo Cliente-Servidor (servicios web RESTful) :
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 o MySQL 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

1. El usuario accede a la aplicación móvil y realiza acciones como registrarse o iniciar sesión.

2. La app envía las solicitudes al servidor Django a través de la API REST.

3. Django valida los datos, consulta la base de datos y devuelve las respuestas al cliente.

4. La aplicación muestra los resultados en la interfaz móvil.

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

4. 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.

5. 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

6. Fase de Documentación y Entre

Finalmente, se elaboró la documentación técnica y metodológica 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.

Requerimientos de Hardware

a. Servidor (Backend – Django)

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 aloja la aplicación backend desarrollada en Django, gestiona la base de datos y las peticiones provenientes del cliente.
Se recomienda el uso de un entorno Linux para mayor estabilidad y rendimiento en la ejecución de servicios web.

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 (mínimo 10 Mbps)

Descripción:
El cliente corresponde a la aplicación móvil que interactúa con el usuario final.
Debe ejecutarse de manera fluida, mostrando los datos de pedidos, productos y notificaciones sin interrupciones.

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 (producción)
Servidor Web	Gunicorn / Nginx / Apache
Control de Versiones	Git y GitHub
Entorno de Pruebas	Postman, VS Code, Android Studio
Sistema Operativo	Windows 10 / 11 o Linux (Ubuntu)

Descripción:
Estas herramientas permiten el desarrollo, depuración y prueba del sistema en un entorno controlado.
Django gestiona el backend y las API REST, mientras que React Native proporciona la interfaz móvil multiplataforma.

b. Librerías y Dependencias Principales

- Django REST Framework: para la creación de servicios API.
- Corsheaders: para permitir la conexión entre frontend y backend.
- Django JWT Simple: para autenticación por tokens.
- Axios: para manejar peticiones HTTP desde React Native.
- React Navigation: para la gestión de pantallas en la aplicación móvil.
- Expo / Metro Bundler: para la ejecución y prueba del entorno móvil.

3. Consideraciones Técnicas

- La base de datos puede migrar fácilmente de SQLite a MySQL sin pérdida de datos.
- El sistema debe contar con conexión a internet constante para sincronizar información entre cliente y servidor.
- Se recomienda el uso de entornos virtuales (venv) para mantener las dependencias del proyecto organizadas.
- Las pruebas deben realizarse tanto en emuladores Android como en dispositivos físicos para validar el rendimiento real.

Documentación del Sistema de Pedidos General

CASOS DE USO

DIAGRAMA DE CLASES

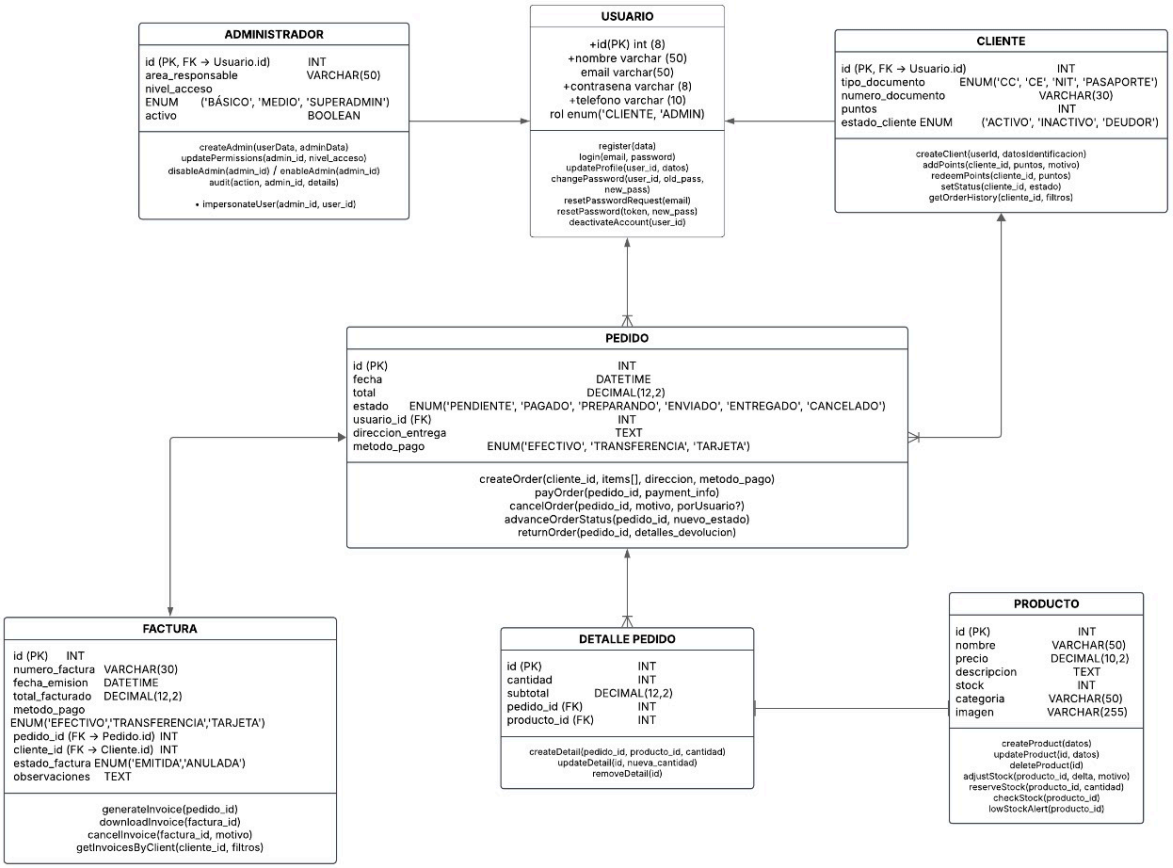


DIAGRAMA DE ACTIVIDADES Flujo principal:

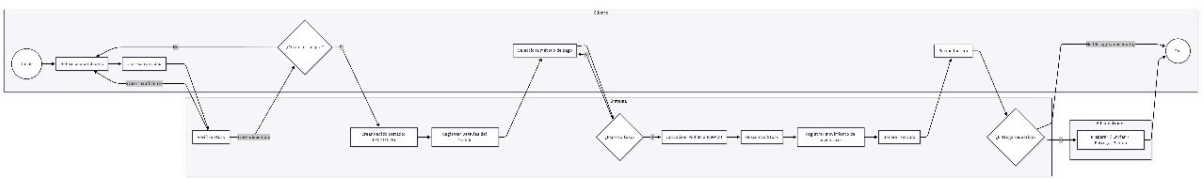
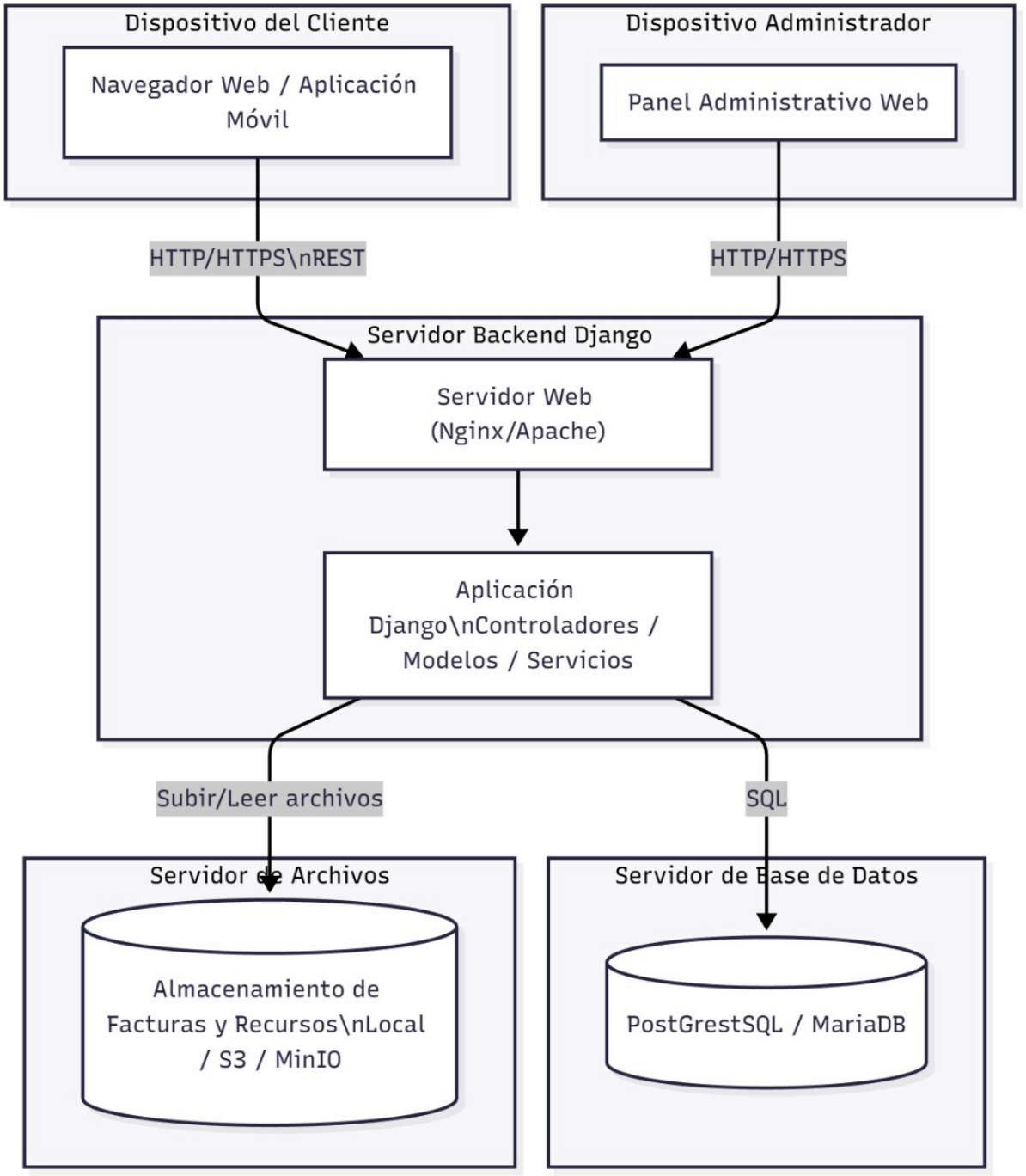


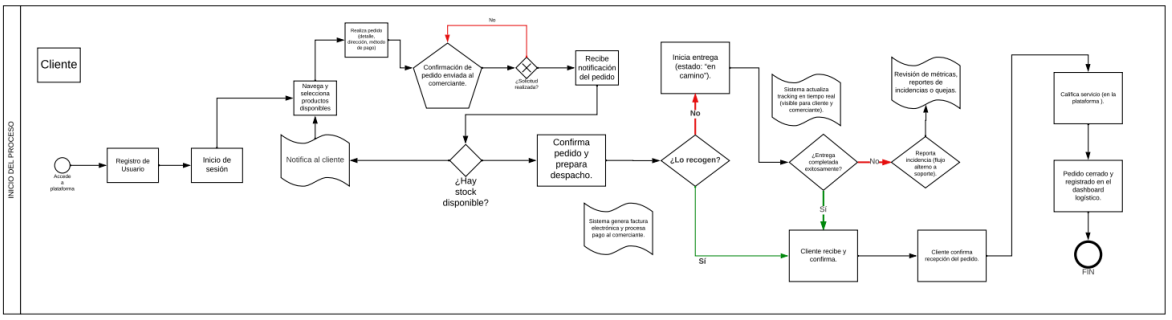
DIAGRAMA DE DESPLIEGUE



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:

HU1: Como usuario, quiero registrarme en la app para poder hacer pedidos.
HU2: Como usuario, quiero iniciar sesión para acceder a mi cuenta.
HU3: Como cliente, quiero ver productos disponibles para elegir los que deseo.
HU4: Como cliente, quiero generar un pedido para recibir mis productos.
HU5: Como administrador, quiero gestionar los productos para mantener el catálogo actualizado.
HU6: Como administrador, quiero revisar los pedidos para organizarlos eficientemente.

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.