Documentación Técnica de la Aplicación Marketplace de Ropa

Introducción

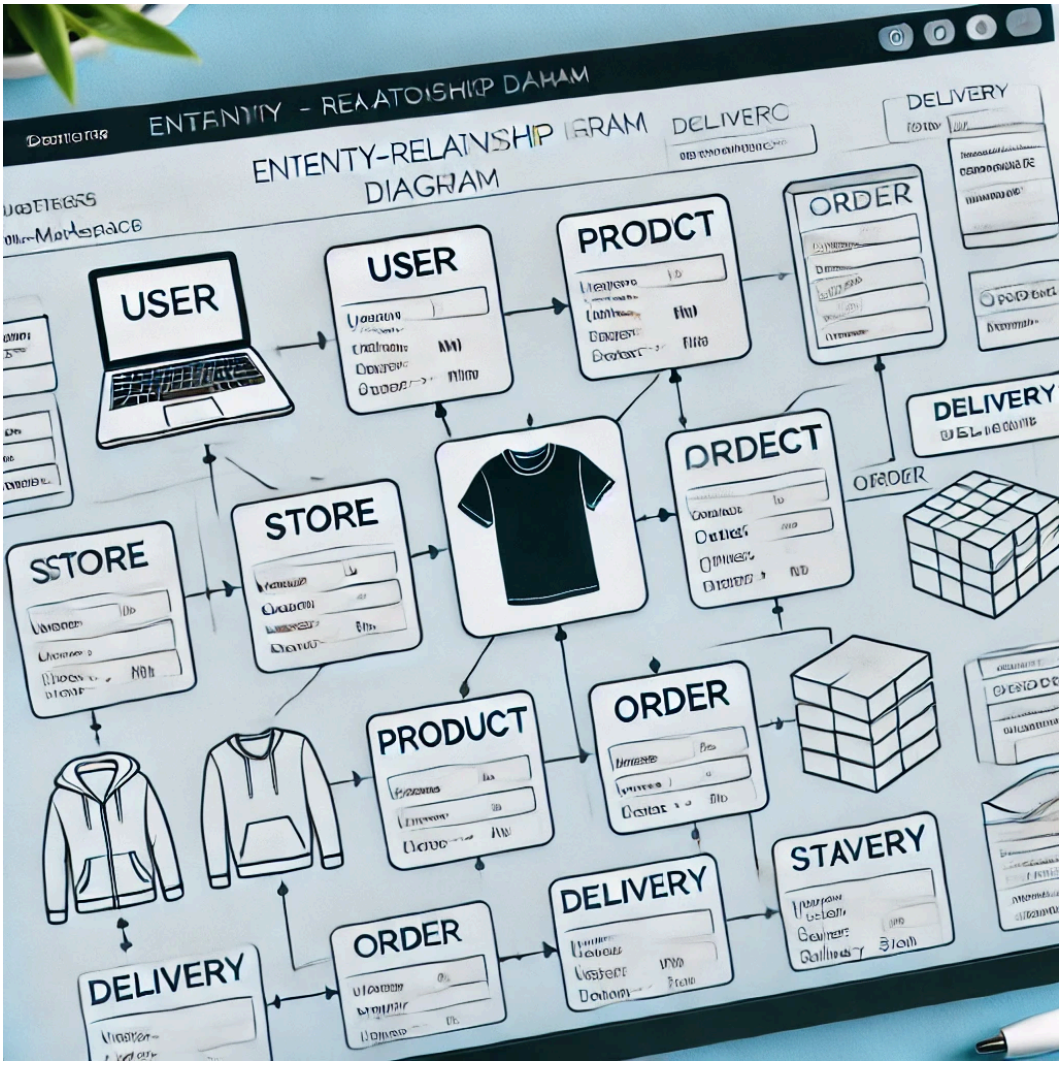
Este documento describe la arquitectura técnica y los detalles funcionales de una aplicación tipo marketplace de ropa,
orientada a globalizar tiendas locales del centro de Cali, Colombia. La aplicación permite a los usuarios navegar entre
diferentes tiendas, explorar productos y realizar compras en línea con opciones de entrega a domicilio.
Este documento abarca los modelos de datos, historias de usuario, casos de uso, modelo de clases y documentación técnica
para guiar el desarrollo y mantenimiento de la aplicación.

1. Modelo de Datos

El modelo de datos define las entidades principales y las relaciones entre ellas dentro de la aplicación de marketplace de ropa.
Estas son las entidades principales:
- Usuario: incluye campos como nombre de usuario, email, contraseña, dirección.
- Tienda: nombre de la tienda, dirección de la tienda, dueño.
- Producto: nombre del producto, precio, talla, color, id de la tienda relacionada.
- Carrito: productos añadidos al carrito, cantidad, precio total.
- Orden: estado de la orden (pendiente, enviada, entregada), fecha de compra, método de pago.
- Entrega: información de envío, estado de la entrega, fecha de entrega estimada.

Las relaciones clave son:
- Un Usuario puede tener muchas Ordenes.
- Una Tienda puede tener muchos Productos.
- Una Orden puede tener muchos Productos.
- Un Usuario puede seguir muchas Tiendas (relación muchos a muchos).

Diagrama Entidad-Relación (ERD)



2. Historias de Usuario

Las historias de usuario son esenciales para definir las necesidades y expectativas del cliente. A continuación, se describen algunas historias clave:

- Como cliente, quiero poder buscar productos por categorías para encontrar lo que necesito de manera rápida y eficiente.
- Como cliente, quiero agregar productos al carrito para poder realizar la compra de una sola vez.
- Como cliente, quiero recibir notificaciones del estado de mi pedido para estar informado en todo momento.
- Como tienda, quiero gestionar el inventario para mantener los productos actualizados.
- Como tienda, quiero ver estadísticas de mis ventas para mejorar mi negocio.
- Como repartidor, quiero ver la dirección del cliente y el estado del pedido para realizar la

entrega en el menor tiempo posible.
- Como administrador del sistema, quiero poder gestionar a los usuarios, tiendas, productos y repartos para mantener el sistema en orden.

3. Casos de Uso

Se describen los casos de uso principales de la aplicación:
- Registro y autenticación de usuarios: Permitir que tanto clientes como tiendas puedan registrarse e iniciar sesión.
- Búsqueda de productos: Los clientes pueden buscar productos por tienda o categoría, y agregarlos a su carrito.
- Proceso de pago y envío: El cliente realiza el pago en línea y selecciona la dirección de entrega.
- Gestión de inventario: Las tiendas pueden agregar, modificar o eliminar productos de su catálogo.
- Notificaciones: Las tiendas y los usuarios reciben notificaciones automáticas sobre el estado de los pedidos.
- Entrega de productos: Los repartidores acceden a la información de la entrega y actualizan el estado del pedido.

4. Modelo de Clases

El modelo de clases describe la estructura interna del software. Algunas de las clases principales son:
- Clase Usuario: Métodos para registrar, iniciar sesión, y ver el historial de pedidos.
- Clase Tienda: Métodos para gestionar inventario y recibir pedidos.
- Clase Producto: Atributos como nombre, precio, talla, disponibilidad.
- Clase Pedido: Métodos para crear y gestionar pedidos, verificar estado.
- Clase Repartidor: Métodos para acceder a los pedidos asignados y actualizar el estado de las entregas.

5. Documentación Técnica

La documentación técnica detalla la arquitectura del sistema:
- Frontend: Se utilizará Flutter o React Native para el desarrollo de la aplicación móvil, asegurando una interfaz de usuario amigable y moderna.
- Backend: Se recomienda Django junto con Django REST Framework para gestionar la API

que maneja productos, pedidos, y usuarios.
- Base de Datos: PostgreSQL o MongoDB será la base de datos principal para almacenar la información de usuarios, tiendas, productos y pedidos.
- Servicios de Pago: Se integrarán servicios de pago como Stripe o PayPal para permitir transacciones seguras dentro de la aplicación.
- Notificaciones: Las notificaciones push se implementarán usando Firebase Cloud Messaging (FCM) para informar a los usuarios y tiendas sobre el estado de sus pedidos.
- Arquitectura: El sistema utilizará una arquitectura basada en microservicios, donde cada componente interactúa a través de APIs REST, proporcionando escalabilidad y flexibilidad.