Informe de Arquitectura del Sistema KairosMix1. Descripción General

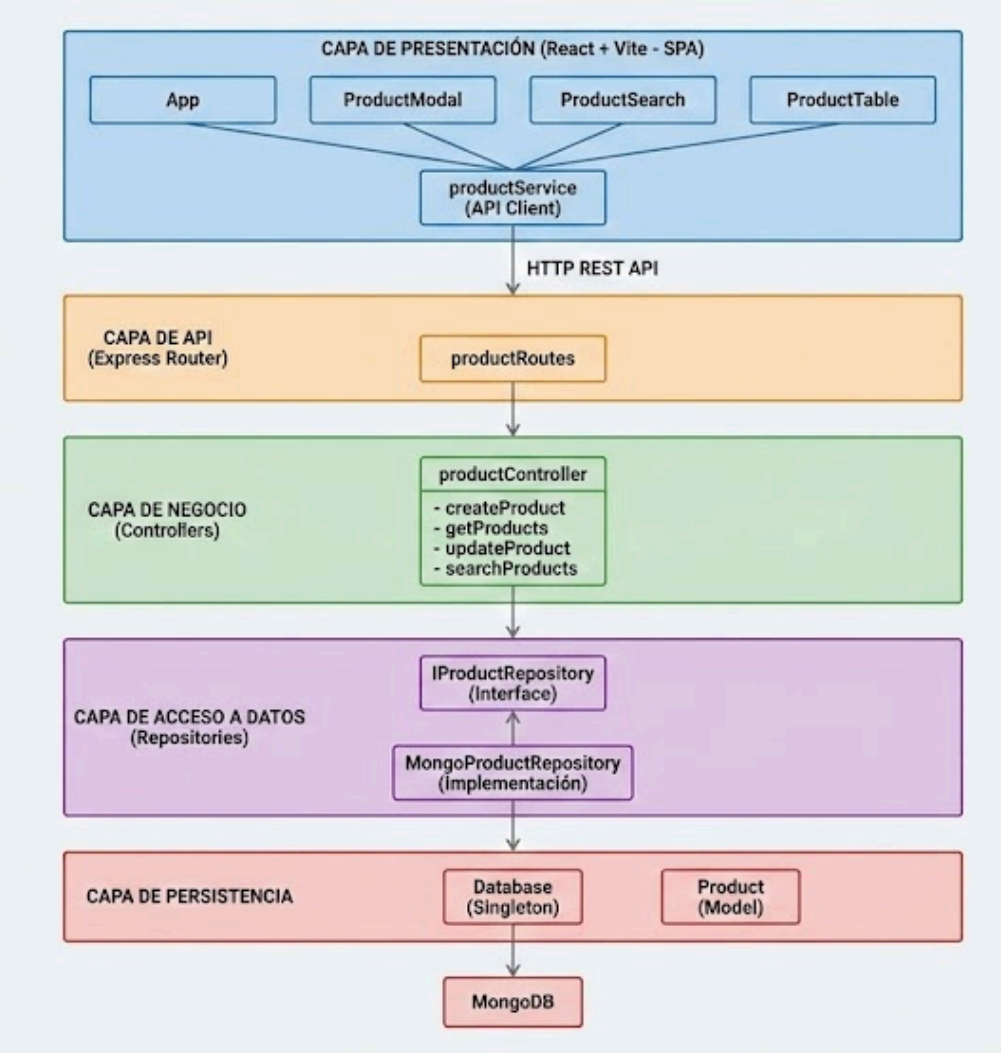
KairosMix es un sistema de gestión de productos diseñado para un negocio de café. Implementa una arquitectura **cliente-servidor** con una separación clara y definida entre el frontend y el backend.

2. Stack Tecnológico

Capa	Tecnología	Versión
Frontend	React + Vite	React 19.2.0, Vite 7.2.4
Backend	Express + TypeScript	Express 5.2.1, TS 5.9.3
Base de Datos	MongoDB (Mongoose)	Mongoose 9.0.1
Comunicación	REST API	JSON

3. Arquitectura General

El sistema utiliza una **arquitectura de N-Capas (N-Tier)** con los siguientes niveles:



4. Arquitectura del Backend4.1 Estructura de Directorios
backend/kairosmix-back/

```
├── src/
│   ├── index.ts            # Punto de entrada
│   ├── controller/
│   │   └── productController.ts
│   ├── database/
│   │   └── Database.ts      # Singleton
│   ├── models/
│   │   └── Product.ts       # Esquema Mongoose
│   ├── repositories/
│   │   └── ProductRepository.ts
│   └── routes/
│       └── productRoutes.ts
├── package.json
└── tsconfig.json
```

4.2 Endpoints de la API REST

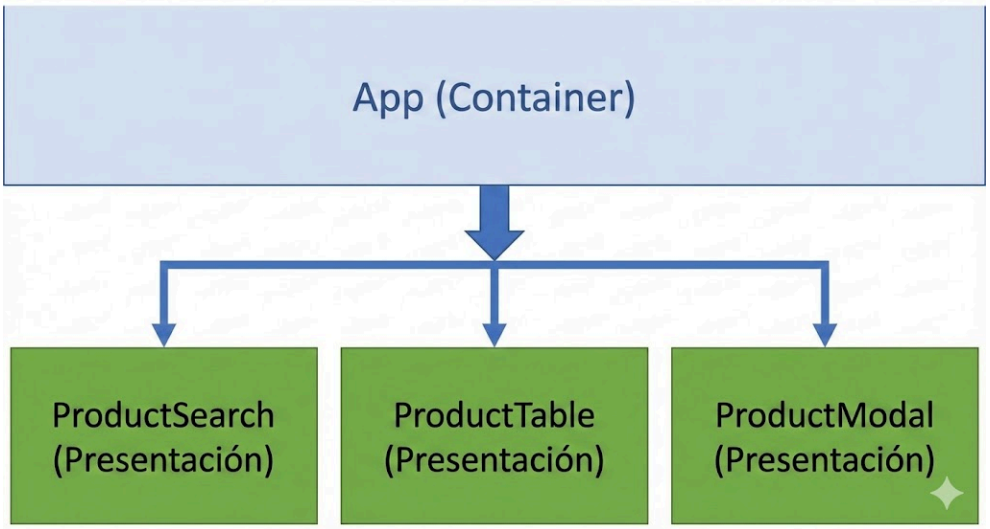
Método	Endpoint	Descripción
`POST`	`/api/products`	Crear producto
`GET`	`/api/products`	Obtener todos los productos
`GET`	`/api/products/sea…`	Buscar productos
`PUT`	`/api/products/:id`	Actualizar producto

5. Arquitectura del Frontend5.1 Estructura de Directorios
frontend/kairosmix-front/

```
├── src/
│   ├── App.jsx             # Componente principal
│   ├── App.css
│   ├── main.jsx            # Punto de entrada
│   ├── index.css
│   ├── components/
│   │   ├── ProductForm.jsx
│   │   ├── ProductModal.jsx
│   │   ├── ProductModal.css
│   │   ├── ProductSearch.jsx
│   │   ├── ProductSearch.css
│   │   ├── ProductTable.jsx
│   │   └── ProductTable.css
│   └── services/
│       └── productService.js # Cliente API
├── index.html
├── package.json
└── vite.config.js
```

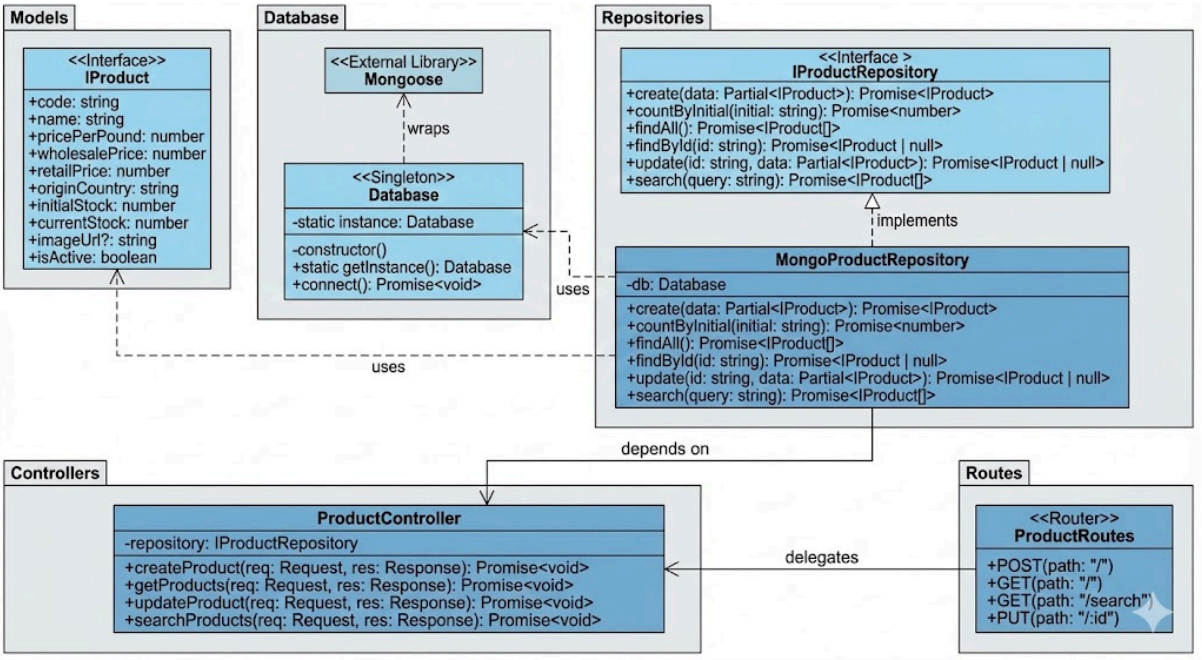
5.2 Arquitectura de Componentes

Patrón Component-Based Architecture con flujo de datos unidireccional:

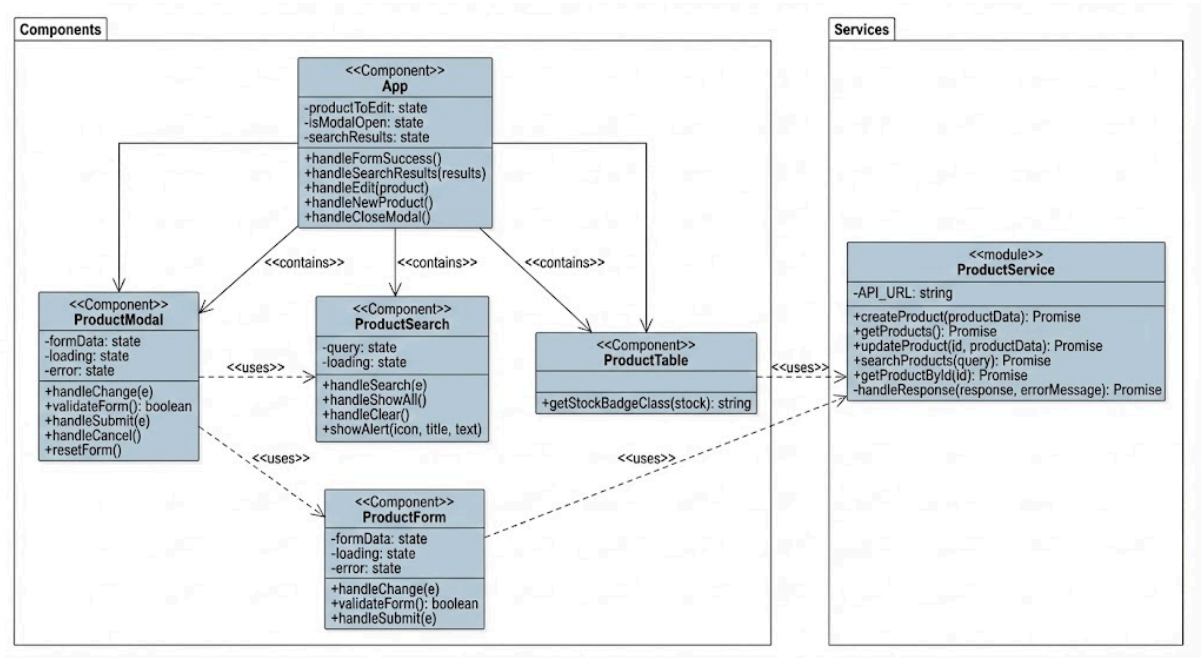


6. Diagramas de Clases

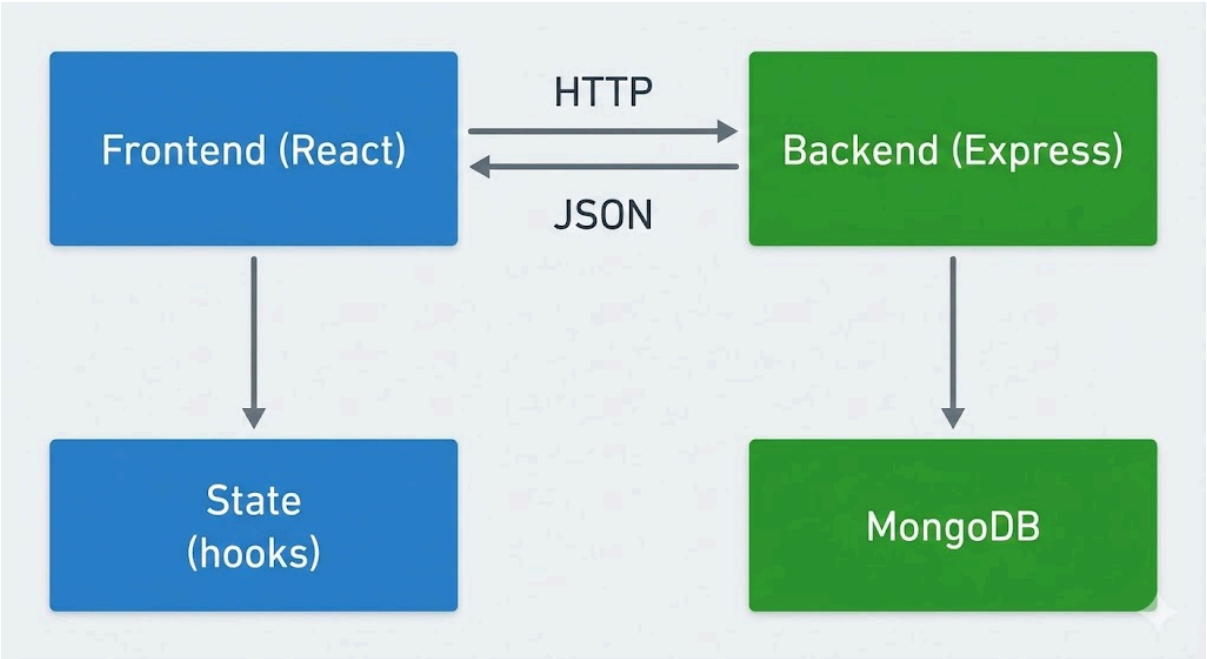
6.1 Diagrama de Clases del Backend



6.2 Diagrama de Clases del Frontend



9. Flujo de Datos



10. Principios Arquitectónicos Aplicados

Principio	Aplicación
Separación de Responsabilidades	Frontend/Backend independientes, capas bien definidas
Single Responsibility	Cada componente/clase tiene una única responsabilidad

Dependency Inversion	Controllers dependen de interfaces (IProductRepository)
Modularidad	Código organizado en módulos reutilizables
Stateless API	API REST sin estado, cada petición es independiente