**Patrón de diseño**

Se eligió el patrón de diseño MVC en este se separa la lógica de la aplicación en tres componentes principales modelo, vista y controlador. Esta división permite gestionar los datos, la presentación y la lógica de negocio de forma independiente. Al dividir responsabilidades, el código se vuelve más organizado y fácil de mantener.

En comparación con otros patrones, como MVVM o MVP, MVC es mucho más simple y flexibilidad en aplicaciones web. MVVM puede añadir complejidad innecesaria en entornos donde no se requiere un fuerte enlace de datos, mientras que MVP tiende a ser más adecuado para aplicaciones de escritorio o móviles.

**Arquitectura**

La arquitectura de 3 capas organiza la aplicación en tres niveles la primera es la capa de presentación, lógica de negocio y capa de datos. La capa de presentación maneja la interfaz de usuario, la lógica de negocio contiene las reglas y la lógica de la aplicación, mientras que la Capa de Datos interactúa con la base de datos. Este enfoque ofrece una clara separación de responsabilidades. Cada capa puede desarrollarse, actualizarse y escalarse de manera independiente. Además, refuerza la seguridad al mantener separadas las capas de presentación y datos

Frente a otras arquitecturas, como la monolítica o de microservicios, la arquitectura de 3 capas ofrece un equilibrio ideal. Es más manejable que los microservicios, evitando la complejidad innecesaria, y supera a la monolítica en modularidad y escalabilidad.

**Lenguajes elegidos y tecnologías.**

**Vite**

Vite es una herramienta de desarrollo moderna que destaca por su rapidez y eficiencia. Diseñada para aplicaciones frontend, especialmente con React, Vite ofrece un arranque rápido y recarga en caliente (HMR), permitiendo tiempos de desarrollo más cortos.

A diferencia de herramientas como Webpack o Parcel, Vite simplifica el flujo de trabajo y es más adecuado para proyectos grandes con muchos módulos

**ORM: Prisma**

Prisma es un ORM moderno que simplifica la interacción con bases de datos relacionales como PostgreSQL. Su esquema declarativo permite modelar la base de datos fácilmente, y sus migraciones automatizadas facilitan la evolución del esquema.

Comparado con ORMs tradicionales como Sequelize o TypeORM, Prisma es ideal porque optimiza el flujo de trabajo backend y asegura la integridad de los datos.

**Base de Datos: PostgreSQL**

PostgreSQL es una base de datos relacional robusta, ideal para aplicaciones que requieren integridad de datos y consultas avanzadas. Su capacidad para manejar grandes volúmenes de datos y múltiples conexiones garantiza escalabilidad.

En comparación con MySQL, MongoDB o SQLite, PostgreSQL es mejor por su equilibrio entre flexibilidad y robustez. En este caso su integración con Prisma y su soporte para operaciones complejas lo hacen una buena opción.

**Lenguajes Elegidos: JavaScript/Node.js (Backend) y React (Frontend)**

JavaScript y Node.js permiten un desarrollo full-stack al utilizar el mismo lenguaje en el frontend y backend nos facilita el hecho de no aprender más lenguajes de programación y así el desarrollo sea mucho más lineal y no se utiliza tiempo en aprender otro lenguaje. Node.js sobresale en la gestión de solicitudes gracias a su modelo asincrónico, mientras que React ofrece una interfaz modular y rápida mediante componentes reutilizables y el uso del DOM virtual.

React, en comparación con frameworks como Angular o Vue.js, se adapta mejor a este contexto por su flexibilidad y ecosistema amplio. Node.js, por su parte, supera a Python/Django o PHP/Laravel en términos de asincronía y compatibilidad con JSON. Este stack proporciona una solución eficiente y moderna para aplicaciones web.