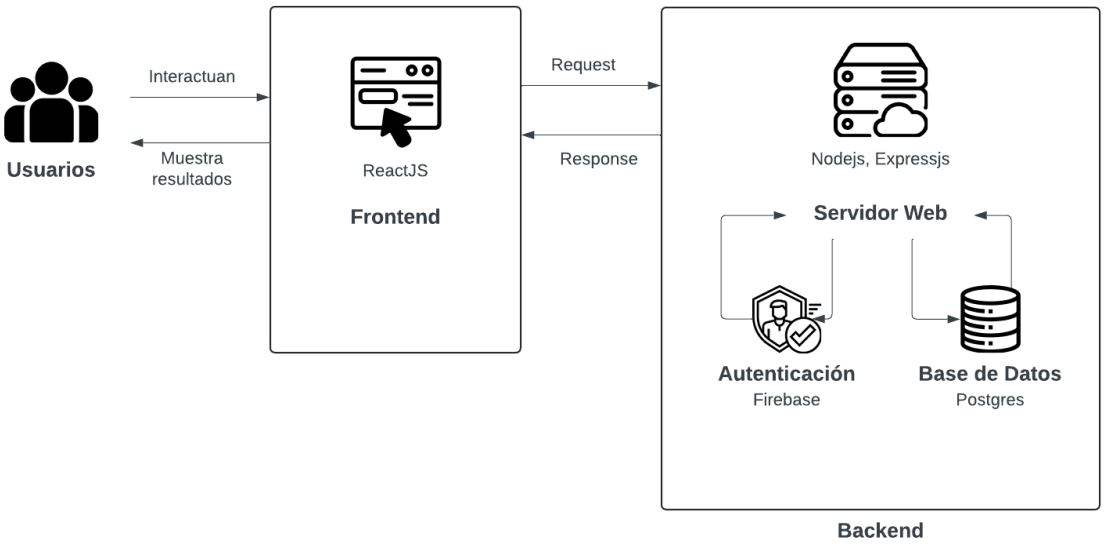
Arquitectura del sistema



2.3.1. Estructura General

El sistema está dividido en una arquitectura de tres capas:

1. **Capa de Presentación (Frontend):** Se utiliza React.js para manejar la interfaz de usuario.
2. **Capa de Aplicación (Backend):** Desarrollada con Express.js en Node.js, es responsable de la lógica de negocio, el procesamiento de las solicitudes y la comunicación con la base de datos.
3. **Capa de Datos (Base de Datos):** Alojada en un servidor de base de datos PostgreSQL, donde se almacenan y gestionan los datos.

2.3.2. Componentes Principales

Frontend (React.js)

- **React.js:** Proporciona la estructura principal del frontend, permite la creación de componentes reutilizables y maneja el estado de la aplicación.
- **Context API:** Para la gestión del estado global, que ayuda a mantener un flujo de datos consistente en componentes complejos y mejora la escalabilidad del frontend.
- **Wouter:** Para la gestión de rutas en el frontend y la navegación entre diferentes vistas de la aplicación.
- **Autenticación de Usuario:** Usa herramientas como JWT (JSON Web Token) y Firebase Auth para autenticar usuarios y asegurar sesiones.

- **Gestión de Errores:** Implementar un manejo centralizado de errores y mensajes de retroalimentación para una mejor experiencia de usuario.

Backend (Express.js en Node.js)

- **Controladores (Controllers):** Manejan las solicitudes HTTP y ejecutan la lógica de negocio. Cada controlador se conecta con los servicios correspondientes, según sea necesario.
- **Modelos (Models):** Definen la estructura de los datos y cómo interactúan con la base de datos.
- **Rutas (Routes):** Las rutas definen los puntos de entrada a la aplicación, asegurando que las solicitudes HTTP se dirijan a los controladores adecuados.
- **Servicios (Services):** Son responsables de la lógica de negocio de cada caso de uso. Esta capa es útil para desacoplar la lógica del negocio del resto del sistema y facilita las pruebas unitarias.
- **Middleware:**
    - **Autenticación y Autorización:** Middleware que valida el JWT u otro mecanismo de autenticación en cada solicitud para proteger rutas privadas.
    - **Validación de Datos:** Uso de express-validator para asegurar que los datos entrantes cumplen con los formatos requeridos antes de que lleguen a los controladores.

Capa de Datos (Base de Datos)

- **ORM:** Uso de Prisma que facilitará la interacción con la base de datos y mejorará la legibilidad del código. Facilita la gestión de datos al ofrecer una capa intermedia entre la aplicación y la base de datos, con soporte para migraciones, consultas optimizadas y un esquema intuitivo que se sincroniza automáticamente.