**Tecnologías a implementer**

1. **Node.Js y Express:** Para crear el servidor y la lógica de la aplicación.
2. **PostgreSQL:** Para la base de datos relacional.
3. **Sequelize:** ORM para interactuar con la base de datos de una manera más cómoda.
4. **Redis:** Para el almacenamiento en cache y la gestión de sesiones.
5. **JWT:** Para la autenticación y autorización de usuarios
6. **Jest:** Para pruebas unitarias
7. **Swagger:** Para la documentación de la API

**Organización de archivos**

* La organización de archivos puede seguir una estructura modular facilitando el mantenimiento y la escalabilidad el proyecto

**Descripción de los archivos (estructura base)**

1. **Config:** Contiene archivos de configuración de la aplicación y de la base de datos
2. **Controllers:** Define los controladores que manejan las solicitudes y respuestas de la API.
3. **Middlewares:** Contiene middlewares para la autenticación, manejo de errores, etc.
4. **Models:** Define los modelos de datos que representan las tablas de la base de datos.
5. **Routes:** Define las rutas de la API y las asocia con los controladores correspondientes.
6. **Services:** Contiene la lógica del negocio y la interacción con la base de datos a través de los modelos.
7. **Utils:** Contiene utilidades generarles, por ejemplo, la función para manejar JWT y el envio de correos.
8. **App.js:** Configura la aplicación implementando Express.
9. **Server.js(index.js):** Inicia el servidor
10. **Test:** Contiene las pruebas unitarias.

**Patrones de diseño:**

1. **MVC:** Al ser una API RESTFUL solo se implementará Model y Controller para separar la lógica del negocio con la lógica de presentación.
2. **Data Access Object:** Separa la lógica que interactúa con la base de datos, permitiendo realizar cambios en la base de datos sin afectar las otras partes de la aplicación.
3. **Service Layer:** Separa la lógica del negocio de los controladores, facilitando el mantenimiento y las pruebas.

**Implementación de seguridad:**

1. **Autenticación:** Implementar JWT para manejar la autenticación de usuarios.
2. **Autorización:** Implementar roles y permisos para controlar el acceso a deferentes partes de la API