

BiblioTech

**Integrantes:**

* **Diaz Aimar Felipe**
* **Micheloud Ivan Jeremias**

Índice:

[Parte #1 4](#_Toc202131764)

[Presentación de la idea y del grupo 4](#_Toc202131765)

[Objetivo 4](#_Toc202131766)

[Entregable 4](#_Toc202131767)

[Parte #2 5](#_Toc202131768)

[Diseño 5](#_Toc202131769)

[Objetivo 5](#_Toc202131770)

[Entregable 5](#_Toc202131771)

[Endpoints de la API REST 7](#_Toc202131772)

[Parte #3 9](#_Toc202131773)

[Backend 9](#_Toc202131774)

[1. Modelos y Relaciones 9](#_Toc202131775)

[10](#_Toc202131776)

[2. Endpoints y Funcionalidades 10](#_Toc202131777)

[10](#_Toc202131778)

[3. Seguridad y Validación 10](#_Toc202131779)

[10](#_Toc202131780)

[4. Manejo de Errores 10](#_Toc202131781)

[5. Extensibilidad 10](#_Toc202131782)

[10](#_Toc202131783)

[6. Resumen de Uso 10](#_Toc202131784)

[Parte #4 11](#_Toc202131785)

[Frontend 11](#_Toc202131786)

[1. Estructura y Navegación 11](#_Toc202131787)

[2. Estado global y autenticación 11](#_Toc202131788)

[3. Comunicación con la API 12](#_Toc202131789)

[4. Diseño y experiencia de usuario 12](#_Toc202131790)

[Parte #5 13](#_Toc202131791)

[Presentación 13](#_Toc202131792)

[Frontend 13](#_Toc202131793)

[Backend 13](#_Toc202131794)

# Parte #1

## Presentación de la idea y del grupo

### Objetivo

* Discutir y seleccionar la idea entre varias candidatas
* Presentar y delimitar de manera general el alcance de la aplicación (qué hace y qué no hace)

### Entregable

Presentar un documento PDF con:

1. Nombre tentativo de la aplicación

Biblioteca Inteligente

2. Apellidos y nombres de los integrantes del grupo

* Felipe Diaz Aimar
* Jeremías Iván Micheloud

3. Breve descripción de la idea, para qué sirve la aplicación, qué hace y qué no hace, cómo se espera que el usuario interactúe en términos generales con ella. Debería ser un texto conciso y breve, de no más de 2 o 3 párrafos.

La aplicación consiste en una página web destinada a estudiantes de una institución educativa, donde podrán acceder a una **biblioteca virtual interactiva**. El ingreso se realiza mediante un usuario previamente registrado en la base de datos, garantizando así el acceso exclusivo a miembros autorizados. Una vez dentro, el usuario podrá **consultar el catálogo completo de libros disponibles**, junto con sus datos descriptivos, como autor, año de publicación, materia relacionada, entre otros.

La plataforma incluye un **buscador inteligente** que utiliza una API pública para mejorar la experiencia de búsqueda, y cuenta además con un **asistente virtual potenciado por IA** (utilizando una API gratuita como OpenRouter con acceso a internet), que puede ayudar al usuario a encontrar libros, realizar recomendaciones personalizadas, o responder preguntas relacionadas con el contenido disponible. La aplicación no permite editar ni subir nuevos libros por parte del usuario, ya que su objetivo principal es brindar acceso y asistencia en la exploración del material existente de forma práctica, ordenada e intuitiva.

# Parte #2

## Diseño

### Objetivo

Plantear en términos generales el diseño y la estructura de las distintas partes de la aplicación.

### Entregable

Presentar un documento PDF con:

1. Diagrama

   El contenido generado por IA puede ser incorrecto.Diagrama de entidad relación (o similar) de los modelos, donde se especifiquen los nombres de los modelos que forman parte de la aplicación, los datos que los componen y sus tipos (número, texto, etc.) y las relaciones entre ellos.

**IMPORTANTE**: esta es una primera aproximación a los modelos que utilizará la aplicación, puede que las entidades sufran modificaciones a lo largo del desarrollo (se agreguen nuevas entidades, se modifiquen campos, etc). 2.

1. Listado de endpoints de la API REST que provee el backend de la aplicación, el listado debe ser una tabla similar a la siguiente donde se especifica el método, la URL y una breve descripción del comportamiento del endpoint.

**IMPORTANTE**: no olvide seguir buenas prácticas para nombrar los endpoints (https://medium.com/@nadinCodeHat/rest-api-naming-conventions-and-bestpractices-1c4e781eb6a5). 3. Bocetos de las distintas vistas de la aplicación. Pueden ser diseñadas con una herramienta o fotos de bocetos dibujados a mano.

**IMPORTANTE**: tener en cuenta que las vistas de la aplicación pueden cambiar durante el desarrollo. Los bocetos son eso: bocetos. No tiene sentido dedicar mucho tiempo a que queden bonitos ya que es bastante probable que cambien en el futuro, sólo deberían servir para ilustrar de manera general el diseño y navegabilidad de la aplicación.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

## Endpoints de la API REST

|  |  |  |  |
| --- | --- | --- | --- |
| **Entidad** | **Método** | **Endpoint** | **Descripción** |
| **Usuario** | POST | /api/auth/login | Iniciar sesión con DNI y contraseña |
| **Usuario** | POST | /api/auth/register | Registrar usuario (body JSON) |
| **Usuario** | POST | /api/auth/logout | Logout (requiere token en header Authorization) |
| **Usuario** | GET | /api/usuarios | Obtener todos los usuarios |
| **Usuario** | GET | /api/usuarios/{id} | Obtener los datos de un usuario específico |
| **Usuario** | GET | /api/usuarios/dni/{dni} | Obtener usuario por DNI |
| **Usuario** | PUT | /api/usuarios/{id} | Actualizar usuario |
| **Usuario** | DELETE | /api/usuarios/{id} | Eliminar usuario |
| **Libro** | GET | /api/libros | Obtener todos los libros disponibles |
| **Libro** | GET | /api/libros/{id} | Obtener los detalles de un libro específico |
| **Libro** | GET | /api/libros/{id}/portada | Obtener la portada de un libro específico |
| **Libro** | GET | /api/libros/buscar?termino=palabra | Buscar libros por título o autor |
| **Libro** | POST | /api/libros | Agregar un libro nuevo (body JSON, puede incluir portada) |
| **Libro** | PUT | /api/libros/{id} | Modificar un libro existente |
| **Libro** | DELETE | /api/libros/{id} | Eliminar un libro (requiere token) |
| **Búsqueda** | GET | /api/busquedas | Obtener búsquedas recientes (puede filtrar por usuarioId) |
| **Búsqueda** | POST | /api/busquedas | Registrar una nueva búsqueda (body JSON) |
| **Búsqueda** | PUT | /api/busquedas/{id} | Actualizar término de búsqueda |
| **Búsqueda** | DELETE | /api/busquedas/{id} | Eliminar una búsqueda |
| **Asistente Virtual** | GET | /api/asistente/historial | Obtener todas las preguntas y respuestas del usuario (requiere token) |
| **Asistente Virtual** | POST | /api/asistente/ask | Registrar una nueva pregunta al asistente (requiere token y body JSON) |
| **Turno** | GET | /api/turnos | Obtener todos los turnos |
| **Turno** | GET | /api/turnos/full/all | Obtener todos los turnos con detalles completos |
| **Turno** | GET | /api/turnos/{id} | Obtener turnos de un usuario por su ID |
| **Turno** | GET | /api/turnos/conSala/{id} | Obtener un turno por ID con su sala relacionada |
| **Turno** | POST | /api/turnos | Crear un nuevo turno |
| **Turno** | PUT | /api/turnos/{id} | Actualizar un turno |
| **Turno** | DELETE | /api/turnos/{id} | Eliminar un turno |
| **Sala** | GET | /api/salas | Obtener todas las salas |
| **Sala** | GET | /api/salas/{id} | Obtener una sala por id |
| **Sala** | POST | /api/salas | Crear una nueva sala |
| **Sala** | DELETE | /api/salas/{id} | Eliminar una sala |
| **Invitado** | GET | /api/invitados/turno/{id\_turno} | Obtener todos los invitados de un turno |
| **Invitado** | POST | /api/invitados | Crear un nuevo invitado para un turno |
| **Invitado** | GET | /api/invitados/usuario/{id\_usuario}/pendientes | Obtener invitaciones pendientes para un usuario |
| **Invitado** | GET | /api/invitados/usuario/{id\_usuario}/aceptados | Obtener turnos aceptados como invitado para un usuario |
| **Invitado** | PUT | /api/invitados/{id} | Actualizar el estado de una invitación (aceptar/rechazar) |
| **Test Backend** | GET | /test | Verifica si el backend responde OK |

# Parte #3

## Backend

**Consigna**: Primera implementación del backend en Express.js utlizando Sequelize. No es necesario que sea la versión definitiva, puede sufrir cambios después. Se sugiere respetar la estructura de carpetas del ejemplo. El backend debería tener su propio repositorio.

El backend de la Biblioteca Inteligente es una API RESTful desarrollada en Node.js con Express y Sequelize (ORM para PostgreSQL). Su función principal es gestionar la información y operaciones de la biblioteca, permitiendo la administración de usuarios, libros, búsquedas, consultas a la IA y respuestas, así como la autenticación y autorización de los usuarios.

### 1. Modelos y Relaciones

El sistema cuenta con los siguientes modelos principales:

* **Usuario**: Representa a los usuarios del sistema (estudiantes, profesores, administradores). Incluye datos personales, credenciales, email, DNI y rol ([usuario](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) o admin).
  + Relación: Un usuario puede tener muchas búsquedas, prompts y turnos creados.
* **Libro**: Almacena la información de los libros disponibles en la biblioteca, incluyendo título, autor, editorial, año, idioma, disponibilidad, portada, etc.
* **Búsqueda**: Registra cada búsqueda realizada por un usuario, permitiendo mostrar el historial de búsquedas recientes.
  + Relación: Cada búsqueda pertenece a un usuario.
* **Prompt**: Guarda las preguntas que los usuarios realizan al asistente virtual (IA).
  + Relación: Cada prompt pertenece a un usuario y tiene una respuesta asociada.
* **Respuesta**: Almacena las respuestas generadas por la IA para cada prompt.
  + Relación: Cada respuesta pertenece a un prompt y a un usuario.
* **Turno**: Representa la reserva de una sala en la biblioteca para un evento o reunión. Incluye fecha, hora, temática, estado, sala y usuario creador.
  + Relación: Un turno pertenece a un usuario y a una sala, y puede tener muchos invitados.
* **Sala**: Representa las salas disponibles para reservar en la biblioteca.
  + Relación: Una sala puede tener muchos turnos.
* **InvitadosTurno**: Relaciona usuarios invitados a un turno, con estado de invitación y asistencia.

Las relaciones permiten asociar búsquedas, prompts, respuestas, turnos e invitaciones a usuarios específicos, y vincular turnos con salas e invitados.

### 

### 2. Endpoints y Funcionalidades

El backend expone endpoints RESTful organizados por entidad, cumpliendo con operaciones CRUD y validaciones. Ejemplos:

* **Usuarios**: Registro, login, consulta, actualización y eliminación de usuarios.
* **Libros**: Consulta, búsqueda, alta, modificación, eliminación y obtención de portada.
* **Búsquedas**: Registro y consulta de búsquedas recientes por usuario.
* **Prompts y Respuestas**: Envío de preguntas a la IA y consulta del historial.
* **Turnos**: Gestión de reservas de salas, incluyendo invitados y estados.
* **Salas**: Consulta, alta y eliminación de salas.
* **Invitados**: Gestión de invitaciones a turnos y su estado.

### 

### 3. Seguridad y Validación

* **Autenticación**: Se utiliza JWT para proteger los endpoints sensibles. El token debe enviarse en el header Authorization.
* **Autorización**: Algunos endpoints requieren roles específicos (por ejemplo, solo admin puede eliminar usuarios).
* **Validación de datos**: Se usa [express-validator](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") para asegurar que los datos recibidos en cada endpoint sean correctos y completos.

### 

### 4. Manejo de Errores

El backend responde con mensajes claros y códigos de estado HTTP apropiados ante errores de validación, autenticación, autorización o problemas internos. Los errores se manejan centralizadamente y se devuelven en formato JSON.

### 5. Extensibilidad

La arquitectura modular permite agregar nuevas funcionalidades, modelos o endpoints de manera sencilla, manteniendo la separación entre rutas, controladores, modelos y middlewares.

### 

### 6. Resumen de Uso

El backend está preparado para ser consumido por cualquier frontend moderno (por ejemplo, React), permitiendo la gestión integral de la biblioteca, la interacción con la IA y la administración de usuarios, libros, turnos y salas de manera segura y eficiente.

# Parte #4

## Frontend

Consigna: Primera implementación del frontend utilizando React. No es necesario que sea la versión definitiva, puede sufrir cambios después. Se sugiere mantener una estructura modular y componentes reutilizables. El frontend debe tener su propio repositorio.

El frontend de la Biblioteca Inteligente es una aplicación desarrollada con **React.js**, utilizando **Vite** como entorno de construcción para una experiencia de desarrollo más rápida y ligera. Su propósito es ofrecer una interfaz clara, amigable y eficiente para que los usuarios puedan interactuar con las funcionalidades del sistema.

### 1. Estructura y Navegación

La aplicación está compuesta por múltiples vistas accesibles mediante **React Router**, permitiendo la navegación entre:

* Página de inicio y autenticación (login y registro).
* Catálogo de libros.
* Asistente virtual potenciado por IA.
* Gestión de turnos y reservas.
* Visualización y gestión del perfil del usuario.

Cada vista se encuentra encapsulada en un componente principal, y a su vez, compuesta por subcomponentes reutilizables, lo que facilita la organización y el mantenimiento del código.

A su vez cuenta con un panel de usuarios con rol “administrador”, lo que permite agregar, editar y visualizar, los usuarios registrados, los libros que existen y los turnos creados.

### 2. Estado global y autenticación

Se utiliza **useContext** combinado con useReducer para el manejo del estado global, especialmente para conservar los datos del usuario autenticado y su token JWT. Esto permite que las vistas protegidas solo sean accesibles cuando el usuario tiene una sesión activa.

La autenticación se gestiona a través del backend utilizando JWT. Los tokens se almacenan en localStorage y se envían en el header de las peticiones a rutas privadas del backend.

### 3. Comunicación con la API

La aplicación consume los endpoints REST expuestos por el backend utilizando fetch, con peticiones GET, POST, PUT y DELETE según corresponda. Las respuestas se manejan con promesas y se incluyen validaciones básicas en los formularios para mejorar la experiencia de usuario.

Se implementó también un interceptor simple que adjunta el token automáticamente en las solicitudes protegidas.

### 4. Diseño y experiencia de usuario

Se utilizaron estilos personalizados con CSS nativo, junto con librerías externas para íconos y algunos elementos visuales.

Se priorizó la usabilidad, la navegación clara y el acceso rápido a las funcionalidades principales.

# Parte #5

## Presentación

### Frontend

El **frontend** de la aplicación, desarrollado con **React**, fue desplegado en **Vercel**, una plataforma especializada en el alojamiento de aplicaciones web modernas. Esta elección permite tiempos de carga optimizados, despliegues automáticos conectados al repositorio de GitHub y una experiencia ágil para el usuario final. Vercel maneja internamente el build del proyecto y genera una URL pública que permite acceder a la aplicación desde cualquier dispositivo con conexión a internet.

La aplicación está disponible en la siguiente URL:

🔗 <https://repositoriofrontendbibliotech.vercel.app/>

El repositorio en el que se encuentra el Frontend es el siguiente:

🔗 <https://github.com/JereMicheloud/Repositorio-Frontend-BibliotecaInteligente.git>

### Backend

El **backend** de la aplicación, desarrollado en **Node.js** utilizando **Express** y **Sequelize**, fue desplegado en **Render**, una plataforma que permite alojar servidores con soporte para bases de datos, procesos de background y endpoints protegidos. Render facilita la gestión del servidor, el despliegue continuo desde el repositorio y permite mantener activa la API REST consumida por el frontend. También ofrece funcionalidades como control de procesos, reinicio automático y monitorización básica del servicio.

La API se encuentra publicada en la siguiente URL:

🔗 <https://repositorio-backend-bibliotecainteligente.onrender.com>

El repositorio en el que se encuentra el Backend es el siguiente:

🔗 <https://github.com/FelipeDiazAimar/Repositorio-Backend-BibliotecaInteligente.git>