Biblioteca Inteligente

**Integrantes:**

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

Índice:

[Parte #1 1](#_Toc200131100)

[Presentación de la idea y del grupo 1](#_Toc200131101)

[Objetivo 1](#_Toc200131102)

[Entregable 2](#_Toc200131103)

[Parte #2 2](#_Toc200131104)

[Diseño 2](#_Toc200131105)

[Objetivo 2](#_Toc200131106)

[Entregable 3](#_Toc200131107)

[Endpoints de la API REST 4](#_Toc200131108)

[Parte #3 5](#_Toc200131109)

[Backend 5](#_Toc200131110)

[1. Modelos y Relaciones 5](#_Toc200131111)

[2. Endpoints y Funcionalidades 5](#_Toc200131112)

[3. Seguridad y Validación 5](#_Toc200131113)

[4. Manejo de Errores 6](#_Toc200131114)

[5. Extensibilidad 6](#_Toc200131115)

[6. Resumen de Uso 6](#_Toc200131116)

# 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.

## Endpoints de la API REST

|  |  |  |  |
| --- | --- | --- | --- |
| Entidad | Método | Endpoint | Descripción |
| Usuario | POST | /api/auth/login | Iniciar sesión con legajo y contraseña |
| Usuario | POST | /api/auth/register | Registrar usuario (*Requiere 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 |
| 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 |
| Búsqueda | GET | /api/busquedas | Obtener todas las búsquedas realizadas (*Requiere token)* |
| Búsqueda | POST | /api/busquedas | Registrar una nueva búsqueda (*Requiere token y body JSON)* |
| Búsqueda | GET | /api/busquedas?usuarioId=123 | Obtener búsquedas de un usuario |
| Asistente Virtual - Historial | GET | /api/asistente/historial | Obtener todas las preguntas y respuestas (*Requiere token)* |
| Asistente Virtual - Preguntas | POST | /api/asistente/ask | Registrar una nueva pregunta (*Requiere token y body JSON)* |
| Asistente Virtual - Prompts | POST | /api/asistente/prompts | Enviar un nuevo prompt |
| Test Backend | GET | [/test](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | *(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 y rol.

**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.

**Prompt**: Guarda las preguntas que los usuarios realizan al asistente virtual (IA).

**Respuesta**: Almacena las respuestas generadas por la IA para cada prompt.

Las relaciones entre modelos permiten, por ejemplo, asociar búsquedas, prompts y respuestas a un usuario específico.

### 2. Endpoints y Funcionalidades

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

### 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 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.

### 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 y libros de manera segura y eficiente.