**Gestor Académico**

**Descripción del Proyecto**

**Gestor Académico** es un sistema de gestión de notas y asignaturas, orientado a una plataforma educativa, en el que profesores y estudiantes pueden interactuar con el sistema para gestionar la información académica. El sistema está desarrollado con **Spring Boot** en el backend y **React** en el frontend.

**Objetivo del Sistema**

El objetivo del sistema es gestionar:

* Los cursos o materias que un profesor puede impartir.
* Las calificaciones que los estudiantes reciben en esas materias.
* Proporcionar una interfaz fácil de usar para profesores y estudiantes, donde puedan consultar las notas de forma rápida y eficaz.

**Tecnologías Utilizadas**

**Backend:**

* **Framework**: Spring Boot (v.3.x) utilizando Jakarta en lugar de Java EE.
* **Base de datos**: H2 para el desarrollo (soporte para bases de datos como MySQL o PostgreSQL en producción).
* **ORM**: JPA (Jakarta Persistence API) para el mapeo objeto-relacional.
* **Lenguaje**: Java
* **Herramienta de construcción**: Maven

**Frontend:**

* **Framework**: React (v.18.x) para la interfaz de usuario.
* **Estilos**: CSS3 / Bootstrap
* **Lenguaje**: JavaScript (ES6+)

**Estructura del Proyecto**

**Backend (Spring Boot)**

**Directorios Principales:**

* src/main/java/com/gestor\_academico: Contiene todo el código de negocio y las clases principales del sistema.
  + **Controllers**: Contienen la lógica relacionada con las rutas y endpoints de la API. Los controladores gestionan las solicitudes HTTP y definen qué acciones tomar para cada ruta.
    - AsignaturaController.java: Controlador que gestiona los endpoints para la gestión de las asignaturas.
    - NotaController.java: Controlador que gestiona los endpoints relacionados con las notas de los estudiantes.
  + **Entities**: Contiene las clases de entidad que representan los datos en la base de datos.
    - Asignatura.java: Clase que representa una asignatura.
    - Estudiante.java: Clase que representa a un estudiante.
    - Nota.java: Clase que representa la relación entre asignaturas y estudiantes a través de las notas.
  + **Repositories**: Interactúan con la base de datos utilizando JPA.
    - AsignaturaRepository.java: Repositorio para gestionar las operaciones CRUD de la entidad Asignatura.
    - NotaRepository.java: Repositorio para gestionar las operaciones CRUD de la entidad Nota.
    - EstudianteRepository.java: Repositorio para las operaciones CRUD de Estudiante.
  + **Services**: Contiene la lógica de negocio del proyecto.
    - AsignaturaService.java: Servicio que gestiona la lógica de asignaturas.
    - EstudianteService.java: Servicio que gestiona la lógica de los estudiantes y sus relaciones con las asignaturas.
    - NotaService.java: Servicio que gestiona la lógica de las notas.
* src/main/resources: Contiene recursos como el archivo application.properties para la configuración de la base de datos H2, entre otros.

**Frontend (React)**

**Estructura del Frontend:**

* src/components: Contiene los componentes reutilizables de la interfaz de usuario.
  + AsignaturaList.js: Componente que muestra la lista de asignaturas.
  + NotaList.js: Componente que muestra las notas de los estudiantes en una asignatura.
  + EstudianteList.js: Componente que muestra los estudiantes y permite seleccionarlos para ver sus notas.
* src/pages: Contiene las páginas principales de la aplicación.
  + Home.js: Página principal de la aplicación.
  + Asignaturas.js: Página donde los usuarios pueden ver y gestionar las asignaturas.
  + Estudiantes.js: Página que permite ver y gestionar a los estudiantes.
* src/services: Contiene los servicios que interactúan con la API del backend.
  + api.js: Archivo donde se configuran las solicitudes HTTP a la API utilizando fetch o axios.
* src/App.js: Componente principal que conecta y enrutará todas las páginas.

**EndPoints del Backend**

**Asignaturas:**

* **GET** /api/asignaturas: Obtiene una lista de todas las asignaturas.
* **POST** /api/asignaturas: Crea una nueva asignatura.
* **GET** /api/asignaturas/{id}: Obtiene los detalles de una asignatura específica por su ID.
* **PUT** /api/asignaturas/{id}: Actualiza los datos de una asignatura existente.
* **DELETE** /api/asignaturas/{id}: Elimina una asignatura por su ID.

**Estudiantes:**

* **GET** /api/estudiantes: Obtiene la lista de estudiantes.
* **POST** /api/estudiantes: Crea un nuevo estudiante.
* **GET** /api/estudiantes/{id}: Obtiene los detalles de un estudiante específico por su ID.
* **PUT** /api/estudiantes/{id}: Actualiza los datos de un estudiante.
* **DELETE** /api/estudiantes/{id}: Elimina un estudiante por su ID.

**Notas:**

* **GET** /api/notas: Lista todas las notas.
* **POST** /api/notas: Añade una nueva nota para un estudiante en una asignatura.
* **GET** /api/notas/{id}: Obtiene los detalles de una nota por su ID.
* **PUT** /api/notas/{id}: Actualiza los detalles de una nota.
* **DELETE** /api/notas/{id}: Elimina una nota por su ID.

**Base de Datos**

**Tablas Principales:**

* **Asignaturas**: Representa cada curso o materia que se imparte.
  + id: Identificador único de la asignatura.
  + nombre: Nombre de la asignatura.
* **Estudiantes**: Representa a los estudiantes matriculados.
  + id: Identificador único del estudiante.
  + nombre: Nombre del estudiante.
  + correo: Correo electrónico del estudiante.
* **Notas**: Relaciona asignaturas con estudiantes y las notas que obtuvieron en ellas.
  + id: Identificador de la nota.
  + asignatura\_id: Referencia a la asignatura.
  + estudiante\_id: Referencia al estudiante.
  + calificacion: La nota o calificación obtenida.

**Ejecución del Proyecto**

1. **Backend**:
   * Ejecuta el proyecto Spring Boot con el comando mvn spring-boot:run.
   * Accede a la base de datos H2 en el navegador mediante la URL: http://localhost:8080/h2-console.
2. **Frontend**:
   * Navega al directorio del frontend.
   * Ejecuta npm start para iniciar la aplicación React en el puerto 3000.