**Documentación del CRUD de Alumnos**

**Grupo 8 - Estudiantes:**

* Meneses Solorzano, Brandon Luis - U21300784
* Velez de Villa Arista Jorge Brian - U20207294
* Mendoza Tinco, Tatiana - U0831381
* Barrantes Burga, César Jeanpieer - U21218945
* López Ordinola, Miguel - U19312642

**Introducción**

Este proyecto es una aplicación que permite gestionar un registro de alumnos mediante operaciones CRUD (Crear, Leer, Actualizar y Eliminar). La aplicación está compuesta por un frontend desarrollado en Angular y un backend en Spring Boot, con la persistencia de datos en una base de datos PostgreSQL.

**Funcionalidades Principales**

1. **Crear Alumnos:** Permite registrar un nuevo alumno en el sistema con los siguientes datos: nombre, apellido, edad y email.
2. **Listar Alumnos:** Permite visualizar todos los alumnos registrados en una tabla.
3. **Editar Alumnos:** Permite modificar los datos de un alumno existente.
4. **Eliminar Alumnos:** Permite eliminar un alumno del sistema.

**Arquitectura del Proyecto**

**El proyecto sigue una arquitectura de cliente-servidor:**

* **Frontend:** Angular para la interfaz de usuario.
* **Backend:** Spring Boot para la lógica de negocio.
* **Base de Datos:** PostgreSQL para almacenar los datos de los alumnos.
* **Comunicación:** Las operaciones se realizan mediante peticiones HTTP (RESTful APIs) entre el frontend y el backend.

**Estructura del Proyecto**

**Frontend (Angular)**

1. **Componentes:**
   * **AlumnosComponent:** Componente principal que gestiona el CRUD de alumnos. Contiene la lógica de manejo de datos y la interacción con el servicio de backend.
2. **Servicios:**
   * **AlumnosService:** Servicio encargado de realizar las peticiones HTTP al backend para las operaciones CRUD.
3. **Rutas:**
   * **El frontend tiene dos rutas principales:** una para la funcionalidad de gestión de alumnos y otra para otras operaciones.

**Backend (Spring Boot)**

1. **Controlador:**
   * **AlumnoController:** Controlador que expone las APIs REST para realizar las operaciones CRUD con los alumnos.
2. **Servicio:**
   * **AlumnoService:** Servicio que contiene la lógica de negocio para manejar las operaciones CRUD.
3. **Repositorio:**
   * **AlumnoRepository:** Interfaz que extiende JpaRepository, utilizada para interactuar con la base de datos PostgreSQL y manejar la persistencia de los datos.
4. **Configuración de Base de Datos:**
   * El archivo application.properties contiene la configuración para conectarse a la base de datos PostgreSQL.

**Endpoints de la API**

**Operaciones CRUD de Alumnos**

1. **Obtener todos los alumnos:**
   * Método: GET
   * Endpoint: /api/alumnos
   * Descripción: Devuelve una lista de todos los alumnos registrados.
2. **Crear un alumno:**
   * Método: POST
   * Endpoint: /api/alumnos
   * Descripción: Permite registrar un nuevo alumno.
3. **Actualizar un alumno:**
   * Método: PUT
   * Endpoint: /api/alumnos/{id}
   * Descripción: Permite actualizar los datos de un alumno existente identificado por su ID.
4. **Eliminar un alumno:**
   * Método: DELETE
   * Endpoint: /api/alumnos/{id}
   * Descripción: Permite eliminar un alumno del sistema identificado por su ID.

**Descripción de Componentes Importantes**

**Frontend (Angular)**

1. **AlumnosComponent:**
   * Este componente maneja la lógica del CRUD de alumnos. Contiene métodos para agregar, actualizar, listar y eliminar alumnos. También se encarga de gestionar la interacción con el servicio de alumnos.
2. **AlumnosService:**
   * Este servicio se encarga de interactuar con el backend utilizando HttpClient. Se encarga de las solicitudes HTTP para las operaciones CRUD y devuelve los resultados al componente.

**Backend (Spring Boot)**

1. **AlumnoController:**
   * Controlador que expone los endpoints para las operaciones CRUD de alumnos. Recibe las solicitudes desde el frontend y devuelve la respuesta correspondiente.
2. **AlumnoService:**
   * Servicio que implementa la lógica de negocio relacionada con las operaciones CRUD. Utiliza el repositorio para interactuar con la base de datos.
3. **AlumnoRepository:**
   * Repositorio que extiende de JpaRepository para gestionar la persistencia de los datos de los alumnos en PostgreSQL.

**Pruebas con Postman**

**Puedes probar los diferentes endpoints usando Postman para verificar las operaciones CRUD:**

1. **Obtener todos los alumnos:**
   * Método: GET
   * URL: http://localhost:8080/api/alumnos
2. **Crear un alumno:**
   * Método: POST
   * URL: http://localhost:8080/api/alumnos
   * Body (JSON):

{

"nombre": "Juan",

"apellido": "Pérez",

"edad": 22,

"email": "juan.perez@example.com"

}

1. **Actualizar un alumno:**
   * Método: PUT
   * URL: http://localhost:8080/api/alumnos/{id}
   * Body (JSON):

{

"nombre": "Juan",

"apellido": "Pérez",

"edad": 23,

"email": "juan.perez@example.com"

}

1. **Eliminar un alumno:**
   * Método: DELETE
   * URL: http://localhost:8080/api/alumnos/{id}

**Valor Agregado**

Además de cumplir con las funcionalidades básicas de CRUD, este proyecto ha añadido valor con las siguientes mejoras:

1. **Interfaz Visual con DaisyUI y Angular:** Se implementó una interfaz atractiva y moderna utilizando DaisyUI junto a Angular. Esto facilita la experiencia del usuario, con un diseño responsivo y soporte para modo claro y oscuro.
2. **Estructura Optimizada con Spring Boot:** Gracias a Spring Boot, la estructura del proyecto está organizada de manera que permite una clara separación de la lógica de negocio, lo que facilita la escalabilidad y el mantenimiento del código.
3. **Pruebas con Postman:** La API ha sido probada a fondo utilizando Postman, lo que garantiza su correcto funcionamiento en todas las operaciones CRUD.
4. **Uso de PostgreSQL:** La integración con PostgreSQL asegura que los datos estén almacenados de manera segura y eficiente, y el uso de JPA facilita la interacción con la base de datos.

**Interfaces**



