# Sistema de Información para la Administración Educativa (SIAE)

Camilo Andrés Rojas Perales, Johan Sebastián Salazar Jaimes

Proyecto del Taller de Panel Administrativo

Tecnología de Desarrollos de Sistemas Informáticos
📅 II Semestre 2025
👨‍🏫 Profesor: Mag. Carlos Adolfo Beltrán Castro
👨‍💻 Estudiantes: Camilo Andrés Rojas Perales - 1064709974, Johan Sebastián Salazar Jaimes - 1095304583

Imagen de Pantalla Inicial con Menú del Proyecto

## 🚀 Descripción del Proyecto

Este proyecto simula un proyecto de un sistema de información escolar en java que centralice y automatice la gestión académica y administrativa para mejorar la eficiencia y la comunicación entre estudiantes, profesores, directivos y personal administrativo con **Spring Boot – Thymeleaf**
Incluye navegación moderna entre diferentes secciones y la funcionalidad de los CRUD para los usuarios

## 📂 Estructura del Proyecto

- Lista de Menú de Opciones de navegación:
    - estudiantes.
    - docentes.
    - directivos.
    - administrativos.

| **Carpeta (Paquete)** | **Componente (Capa)** | **Descripción** |
|---|---|---|
| `modelo` | **Modelo/Entidad** | Contiene las clases de Java que mapean directamente a las tablas de la base de datos (Ej: `Estudiante`, `Usuario`). Llevan anotaciones de JPA (`@Entity`, `@Id`, `@Table`). |

| Carpeta (Paquete) | Componente (Capa) | Descripción |
|---|---|---|
| `repositorio` | **Persistencia (DAO)** | Define las interfaces que extienden de `JpaRepository`. Permiten realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sin escribir código SQL. |
| `servicio` | **Lógica de Negocio** | Contiene la lógica central de la aplicación. Se define la interfaz (`IEstudianteServicio`) y la implementación (`EstudianteServicio`). Se encarga de coordinar el repositorio y gestionar las reglas de negocio. |
| `controlador` | **Controlador (MVC)** | Contiene las clases que manejan las peticiones HTTP (URLs). Utilizan `@Controller` o `@RestController`, invocan los métodos del servicio y devuelven la vista (HTML/Thymeleaf) o los datos (JSON). |
| `recursos/templates` | **Vista (Thymeleaf)** | Almacena los archivos HTML que contienen las plantillas de Thymeleaf, responsables de renderizar la interfaz de usuario. (Ej: `estudiante/lista.html`). |
| `recursos/static` | **Recursos Estáticos** | Almacena archivos que no cambian (CSS, JavaScript, imágenes) que se cargan directamente en el navegador. |

| Carpeta (Paquete) | Componente (Capa) | Descripción |
| --- | --- | --- |
| `recursos/application.properties` | **Configuración** | Define parámetros críticos como la conexión a la base de datos (URL, usuario, clave) y la configuración de Hibernate/JPA. |

 Lista de Tecnologías Usadas

| Tecnologías | Funciónes |
| --- | --- |
| **Java Development Kit (JDK)** | Lenguaje de programación base. |
| **Spring Boot** | Framework principal para la creación de aplicaciones Java empresariales robustas y de rápida configuración. |
| **Spring Web** | Módulo que permite crear la arquitectura MVC (Controladores) y manejar peticiones HTTP/REST. |

🛠️ Instalación y ejecución

# 📂 Estructura del Proyecto (Capas)

Tu proyecto sigue la arquitectura estándar de **Modelo-Vista-Controlador (MVC)**, con una estructura de capas bien definida por Spring Boot, lo cual es fundamental para la mantenibilidad y escalabilidad.

| Carpeta (Paquete) | Componente (Capa) | Descripción |
| --- | --- | --- |
| `modelo` | **Modelo/Entidad** | Contiene las clases de Java que mapean directamente a las tablas de la base de datos (Ej: `Estudiante`, `Usuario`). Llevan anotaciones de JPA (`@Entity`, `@Id`, `@Table`). |

| Carpeta (Paquete) | Componente (Capa) | Descripción |
|---|---|---|
| `repositorio` | **Persistencia (DAO)** | Define las interfaces que extienden de `JpaRepository`. Permiten realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sin escribir código SQL. |
| `servicio` | **Lógica de Negocio** | Contiene la lógica central de la aplicación. Se define la interfaz (`IEstudianteServicio`) y la implementación (`EstudianteServicio`). Se encarga de coordinar el repositorio y gestionar las reglas de negocio. |
| `controlador` | **Controlador (MVC)** | Contiene las clases que manejan las peticiones HTTP (URLs). Utilizan `@Controller` o `@RestController`, invocan los métodos del servicio y devuelven la vista (HTML/Thymeleaf) o los datos (JSON). |
| `recursos/templates` | **Vista (Thymeleaf)** | Almacena los archivos HTML que contienen las plantillas de Thymeleaf, responsables de renderizar la interfaz de usuario. (Ej: `estudiante/lista.html`). |
| `recursos/static` | **Recursos Estáticos** | Almacena archivos que no cambian (CSS, JavaScript, imágenes) que se cargan directamente en el navegador. |

| Carpeta (Paquete) | Componente (Capa) | Descripción |
| --- | --- | --- |
| `recursos/application.properties` | **Configuración** | Define parámetros críticos como la conexión a la base de datos (URL, usuario, clave) y la configuración de Hibernate/JPA. |

---

# □ Lista de Tecnologías Usadas

Este proyecto utiliza el siguiente *stack* de tecnologías, basado en tu implementación de Spring Boot:

### 💻 Framework y Lenguaje

| Tecnología | Función |
| --- | --- |
| **Java Development Kit (JDK)** | Lenguaje de programación base. |
| **Spring Boot** | Framework principal para la creación de aplicaciones Java empresariales robustas y de rápida configuración. |
| **Spring Web** | Módulo que permite crear la arquitectura MVC (Controladores) y manejar peticiones HTTP/REST. |

### 💾 Persistencia y Base de Datos

| Tecnología | Función |
| --- | --- |
| **Spring Data JPA** | Abstracción que simplifica la interacción con la base de datos, permitiendo definir repositorios (CRUD) con solo interfaces. |
| **Hibernate** | Implementación por defecto de JPA, responsable del **Mapeo Objeto-Relacional (ORM)**. Convierte las entidades Java en consultas SQL. |
| **MySQL Connector** | Driver JDBC que permite a Java conectarse y comunicarse con la base de datos MySQL. |

| Tecnología | Función |
| --- | --- |
| **MySQL (DB)** | Sistema de gestión de base de datos relacional (RDBMS) utilizado para almacenar la información del sistema (estudiantes, usuarios, etc.). |

### 🌐 Frontend y Plantillas

| Tecnología | Función |
| --- | --- |
| **Thymeleaf** | Motor de plantillas que permite generar vistas HTML dinámicas, integrando datos del modelo desde el controlador. |
| **Bootstrap** | Framework de CSS/JavaScript para diseño *responsive* y estilizado de la interfaz de usuario. |

---

# 🔧 Instalación y Ejecución

Para poner en marcha el proyecto, se deben seguir estos pasos:

### 1. Requisitos Previos

- Tener instalado **Java JDK** (versión 17 o superior, según la que uses).
- Tener instalado un **Sistema de Gestión de Base de Datos (SGBD)**, como MySQL.
- Tener una herramienta de construcción (build tool): **Maven** o **Gradle**.
- Un Entorno de Desarrollo Integrado (**IDE**): IntelliJ IDEA, Eclipse, o VS Code.

### 2. Configuración de la Base de Datos

1. Crear la base de datos (esquema) en MySQL con el nombre configurado en el proyecto (ej: `siae`).
2. Ejecutar el script SQL para crear la tabla `estudiantes` y las demás tablas necesarias, asegurando que los nombres de las columnas (`codigo`, `nombre_completo`, `correo_institucional`, etc.) coincidan con el mapeo en la entidad `Estudiante.java`.

### 3. Configuración de Spring Boot

Abrir el archivo `src/main/resources/application.properties` y ajustar las credenciales:
Properties
```
spring.datasource.url=jdbc:mysql://localhost:3306/siae?serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=12345
```

## 4. Compilación y Ejecución

1. **Limpiar y Compilar:** Abrir la terminal en la raíz del proyecto y ejecutar:
   - **Con Maven:** `mvn clean install`
   - **Con Gradle:** `./gradlew clean build`
2. **Ejecutar la Aplicación:**
   - **Desde el IDE:** Ejecutar la clase principal de Spring Boot (`@SpringBootApplication`).
   - **Desde la Terminal:** Ejecutar el archivo JAR generado en la carpeta `target/`

     Bash
     ```
     java -jar target/SIAE.jar
     ```

## 5. Acceso

Una vez que la aplicación se inicia correctamente (buscando el mensaje de inicio de Tomcat), se puede acceder a la interfaz:

- **URL de Acceso:** `http://localhost:8081`