Web Dinámico

10/12/2025

Sistema de Registro de Estudiantes y Cursos

UNIVERSIDAD AUTÓNOMA METROPOLITANA CUAJIMALPA

MARLON
MALERVA AVELAR

El Sistema de Registro de Estudiantes y Cursos es una aplicación web completa que permite gestionar estudiantes y cursos académicos. Implementa operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y establece una relación N:M entre estudiantes y cursos.

Objetivos:

- Gestionar información de estudiantes de forma eficiente
- Administrar el catálogo de cursos disponibles
- Establecer relaciones entre estudiantes y cursos
- Proporcionar una interfaz intuitiva y moderna
- Implementar validaciones en frontend y backend

Características

Gestión de Alumnos

CRUD completo con validaciones

Gestión de Cursos

Administración de catálogo

Búsqueda y Filtros

Búsqueda en tiempo real

Validaciones

Frontend y Backend

Requerimientos del sistema:

- **CRUD de Estudiantes:** Crear, leer, actualizar y eliminar estudiantes
- **CRUD de Cursos:** Gestión completa del catálogo de cursos
- **Relación N:M:** Un estudiante puede estar inscrito en múltiples cursos
- **Vista de inscritos:** Ver estudiantes inscritos por curso
- **Filtros:** Búsqueda por curso y estado del estudiante

Requerimientos técnicos

Frontend (Angular 17+)

- Angular 17+ con componentes standalone
- Angular Material para UI
- Routing para navegación
- Servicios HTTP para comunicación
- Formularios reactivos
- Tablas con paginación, ordenamiento y filtros
- Componentes organizados por módulos

Backend (Spring Boot)

- Spring Web
- Spring Data JPA
- Spring Validation
- MySQL Database
- Swagger (OpenAPI) para documentación
- Arquitectura recomendada: Controller, Service, Repository, Entity, DTO

Arquitetura General

- Capa de presentación (Frontend)
- Capa Logica (Backend)
- Capa de datos (Mysql)

Modelo de Datos

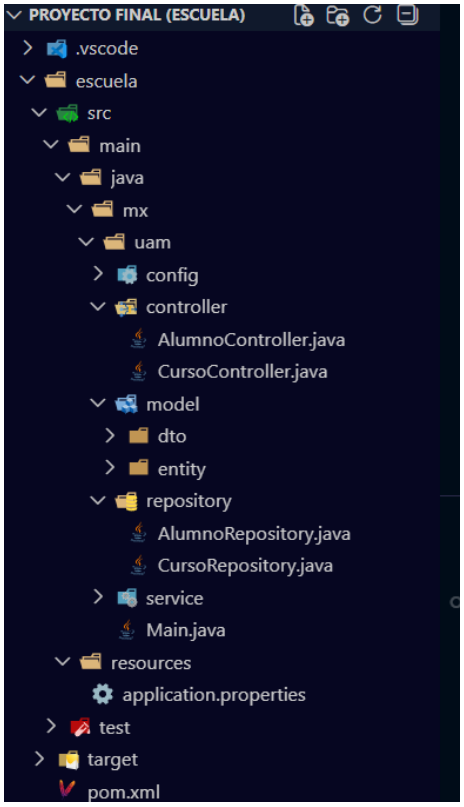
Entidad Alumno:

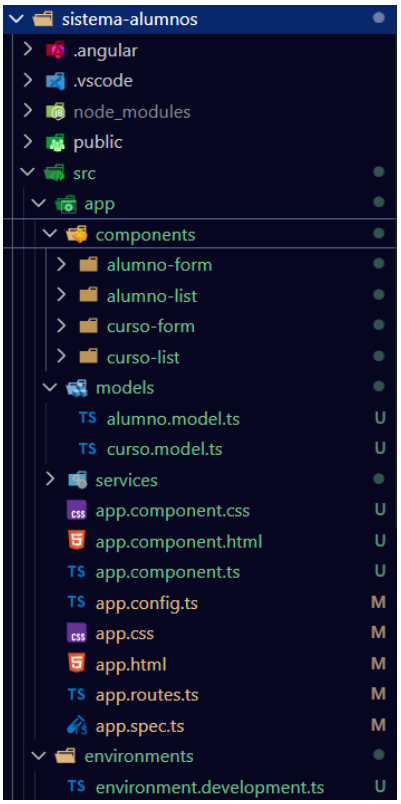
```
mysql> describe alumno;
+------------------+--------------+------+-----+---------+----------------+
| Field            | Type         | Null | Key | Default | Extra          |
+------------------+--------------+------+-----+---------+----------------+
| id               | int          | NO   | PRI | NULL    | auto_increment |
| nombre           | varchar(255) | YES  |     | NULL    |                |
| apellido_paterno | varchar(255) | YES  |     | NULL    |                |
| apellido_materno | varchar(255) | YES  |     | NULL    |                |
| edad             | int          | YES  |     | NULL    |                |
| matricula        | varchar(255) | YES  |     | NULL    |                |
| curso_id         | int          | YES  | MUL | NULL    |                |
+------------------+--------------+------+-----+---------+----------------+
```

Entidad curso:

```
mysql> describe curso;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int          | NO   | PRI | NULL    | auto_increment |
| nombre      | varchar(255) | YES  |     | NULL    |                |
| descripcion | varchar(255) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

Estructura del Proyecto (Backend y Frontend)





Backend

Configuración (application.properties)

escuela > src > main > resources > application.properties

```
1  spring.datasource.url=jdbc:mysql://localhost:3306/escuela?useSSL=false&serverTimezone=UTC
2  spring.datasource.username=Marlon
3  spring.datasource.password=12345
4  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5
6  spring.jpa.hibernate.ddl-auto=update
7  spring.jpa.show-sql=true
8  spring.jpa.properties.hibernate.format_sql=true
9
10 spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
11
```

Modelo entity Alumno y Curso

cuela > src > main > java > mx > uam > model > entity > Alumno.java > Alumno

```java
package mx.uam.model.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;

@Entity
public class Alumno {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    private String nombre;
    private String apellidoPaterno;
    private String apellidoMaterno;
    private Integer edad;
    private String matricula;

    @ManyToOne
    @JoinColumn(name = "curso_id")
    private Curso curso;

    public Integer getId() {
        return id;
    }
```

escuela > src > main > java > mx > uam > model > entity > Curso.java > Curso

```java
package mx.uam.model.entity;

import java.util.ArrayList;
import java.util.List;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.FetchType;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;

@Entity
public class Curso {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;
    private String nombre;
    private String descripcion;

    @OneToMany(mappedBy="curso",cascade=CascadeType.ALL, fetch=FetchType.LAZY)
    private List<Alumno> alumnos = new ArrayList<>();

    public Integer getId() {
        return id;
    }
```

Controller Alumno y Curso:

```java
package mx.uam.controller;

import mx.uam.model.dto.AlumnoDTO;
import mx.uam.service.AlumnoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import io.swagger.v3.oas.annotations.tags.Tag;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;

import java.util.List;

@RestController
@RequestMapping("/alumnos")
@Tag(name = "Alumno", description = "Operaciones CRUD para alumnos")
public class AlumnoController {
    @Autowired
    private AlumnoService alumnoService;

    @GetMapping
    @Operation(summary = "Obtener todos los alumnos", description = "Retorna una lista de todos los alumnos")
    public List<AlumnoDTO> getAll() {
        return alumnoService.findAll();
    }
```

```java
package mx.uam.controller;

import mx.uam.model.dto.CursoDTO;
import mx.uam.service.CursoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import io.swagger.v3.oas.annotations.tags.Tag;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;

import java.util.List;

@RestController
@RequestMapping("/cursos")
@Tag(name = "Curso", description = "Operaciones CRUD para cursos")
public class CursoController {
    @Autowired
    private CursoService cursoService;

    @GetMapping
    @Operation(summary = "Obtener todos los cursos", description = "Retorna una lista de todos los cursos")
    public List<CursoDTO> getAll() {
        return cursoService.findAll();
    }
```

Dependencias:

```xml
<dependencies>
    <!-- https://mvnrepository.com/artifact/com.mysql/mysql-connector-j -->
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <version>9.4.0</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- Swagger/OpenAPI Springdoc -->
    <dependency>
        <groupId>org.springdoc</groupId>
        <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
        <version>2.2.0</version>
    </dependency>
</dependencies>
```

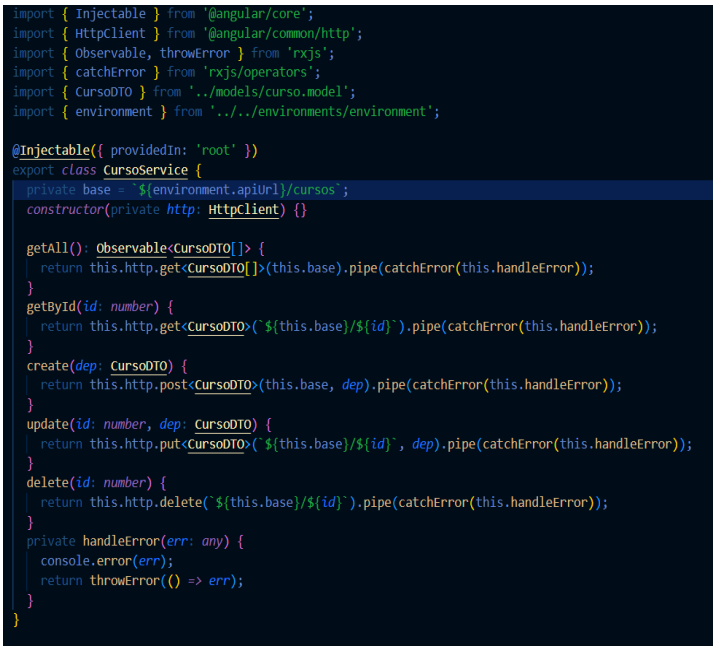
Frontend:

Main:

```typescript
import { bootstrapApplication } from '@angular/platform-browser';
import { provideRouter, Routes } from '@angular/router';
import { provideHttpClient } from '@angular/common/http';
import { AppComponent } from './app/app.component';
import { AlumnoListComponent } from './app/components/alumno-list/alumno-list.component';
import { AlumnoFormComponent } from './app/components/alumno-form/alumno-form.component';
import { CursoListComponent } from './app/components/curso-list/curso-list.component';
import { CursoFormComponent } from './app/components/curso-form/curso-form.component';

const routes: Routes = [
  { path: '', redirectTo: '/alumnos', pathMatch: 'full' as const },
  { path: 'alumnos', component: AlumnoListComponent },
  { path: 'alumnos/nuevo', component: AlumnoFormComponent },
  { path: 'alumnos/editar/:id', component: AlumnoFormComponent },
  { path: 'cursos', component: CursoListComponent },
  { path: 'cursos/nuevo', component: CursoFormComponent },
  { path: 'cursos/editar/:id', component: CursoFormComponent },
  { path: '**', redirectTo: '/alumnos' }
];

bootstrapApplication(AppComponent, {
  providers: [
    provideRouter(routes),
    provideHttpClient()
  ]
}).catch(err => console.error(err));
```

Service Alumno y Curso

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable, throwError, catchError } from 'rxjs';
import { AlumnoDTO } from '../models/alumno.model';
import { environment } from '../../environments/environment';

@Injectable({ providedIn: 'root' })
export class AlumnoService {
  private base = `${environment.apiUrl}/alumnos`;
  constructor(private http: HttpClient) {}

  getAll(): Observable<AlumnoDTO[]> {
    return this.http.get<AlumnoDTO[]>(this.base).pipe(catchError(this.handleError));
  }
  getById(id: number) {
    return this.http.get<AlumnoDTO>(`${this.base}/${id}`).pipe(catchError(this.handleError));
  }
  create(emp: AlumnoDTO) {
    return this.http.post<AlumnoDTO>(this.base, emp).pipe(catchError(this.handleError));
  }
  update(id: number, emp: AlumnoDTO) {
    return this.http.put<AlumnoDTO>(`${this.base}/${id}`, emp).pipe(catchError(this.handleError));
  }
  delete(id: number) {
    return this.http.delete(`${this.base}/${id}`).pipe(catchError(this.handleError));
  }
  private handleError(err: any) {
    console.error(err);
    return throwError(() => err);
  }
}
```



```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError } from 'rxjs/operators';
import { CursoDTO } from '../models/curso.model';
import { environment } from '../../environments/environment';

@Injectable({ providedIn: 'root' })
export class CursoService {
  private base = `${environment.apiUrl}/cursos`;
  constructor(private http: HttpClient) {}

  getAll(): Observable<CursoDTO[]> {
    return this.http.get<CursoDTO[]>(this.base).pipe(catchError(this.handleError));
  }
  getById(id: number) {
    return this.http.get<CursoDTO>(`${this.base}/${id}`).pipe(catchError(this.handleError));
  }
  create(dep: CursoDTO) {
    return this.http.post<CursoDTO>(this.base, dep).pipe(catchError(this.handleError));
  }
  update(id: number, dep: CursoDTO) {
    return this.http.put<CursoDTO>(`${this.base}/${id}`, dep).pipe(catchError(this.handleError));
  }
  delete(id: number) {
    return this.http.delete(`${this.base}/${id}`).pipe(catchError(this.handleError));
  }
  private handleError(err: any) {
    console.error(err);
    return throwError(() => err);
  }
}
```

Funcionalidades

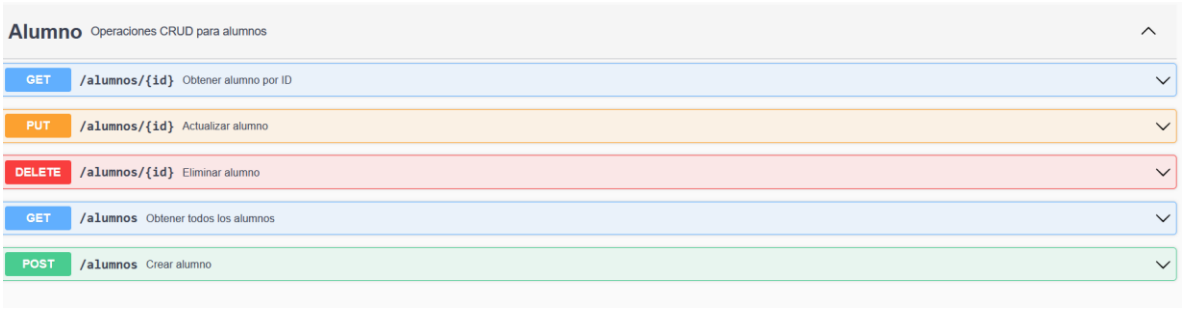
Gestión de Alumnos

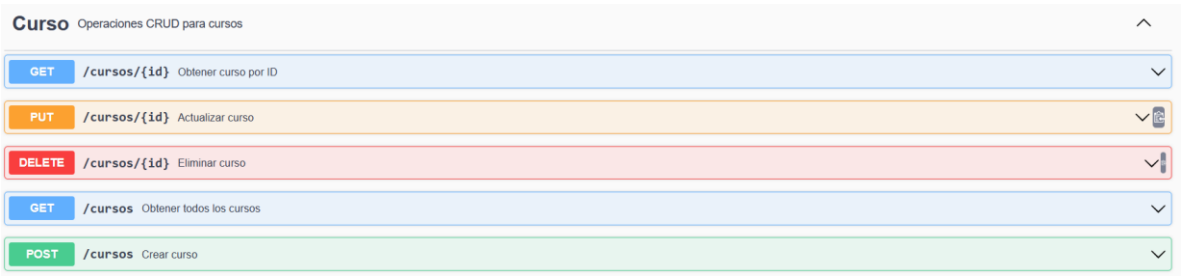
- Listar Alumnos: Vista con tabla de todos los alumnos registrados
- Búsqueda en Tiempo Real: Filtrar por nombre, apellidos o matrícula
- Crear Alumno: Formulario con validaciones completas
- Editar Alumno: Modificar datos de alumnos existentes
- Eliminar Alumno: Confirmación antes de eliminar
- Validaciones:
    - Nombre: 2-100 caracteres
    - Apellidos: 2-100 caracteres cada uno
    - Edad: 1-120 años
    - Matrícula: 4-20 caracteres, única
    - Curso: Obligatorio

Gestión de Cursos

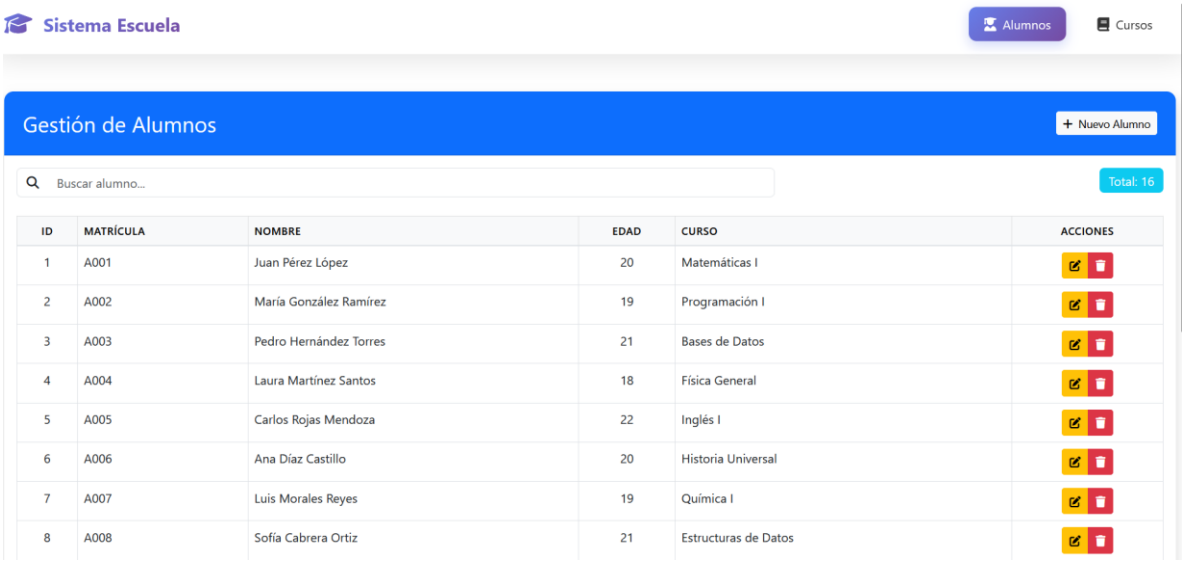
- Listar Cursos: Vista con tabla de todos los cursos
- Contador de Alumnos: Ver cuántos alumnos están inscritos
- Búsqueda en Tiempo Real: Filtrar por nombre o descripción
- Crear Curso: Formulario con validaciones
- Editar Curso: Modificar información del curso
- Eliminar Curso: Solo si no tiene alumnos inscritos
- Ver Alumnos Inscritos: Lista de alumnos por curso
- Validaciones:
    - Nombre: 3-150 caracteres
    - Descripción: 10-500 caracteres

Funcionamiento Swagger

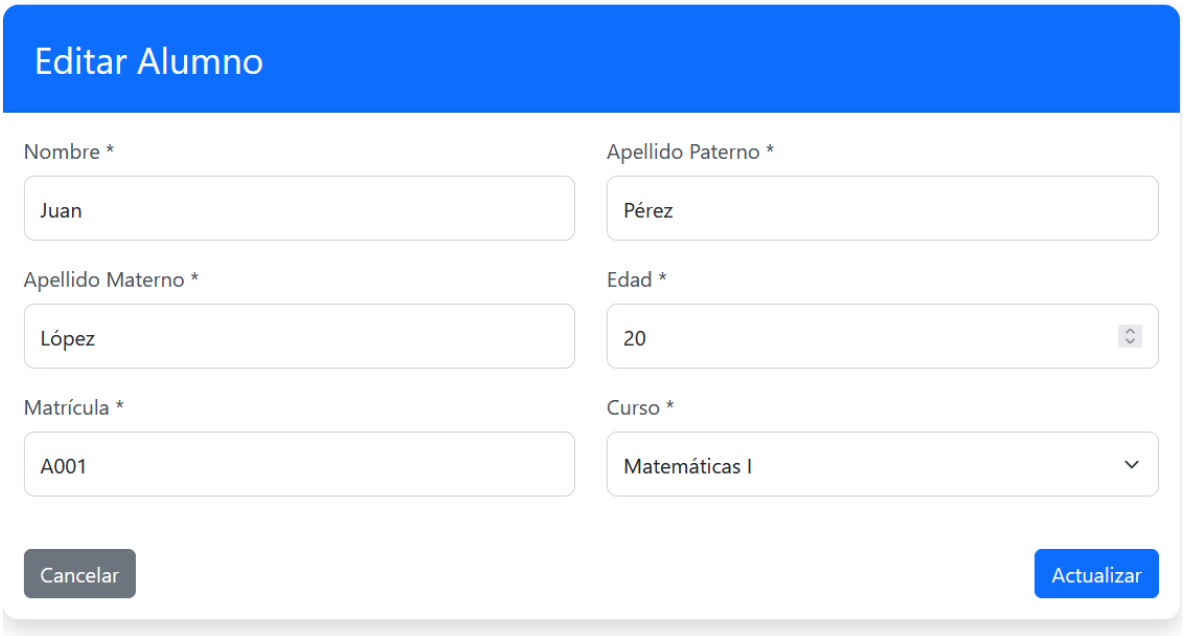


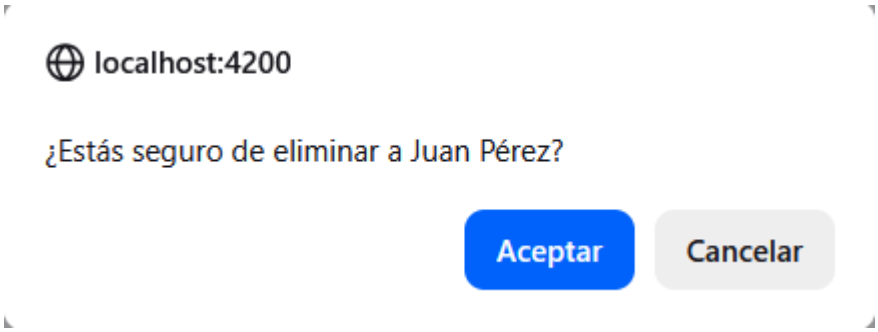


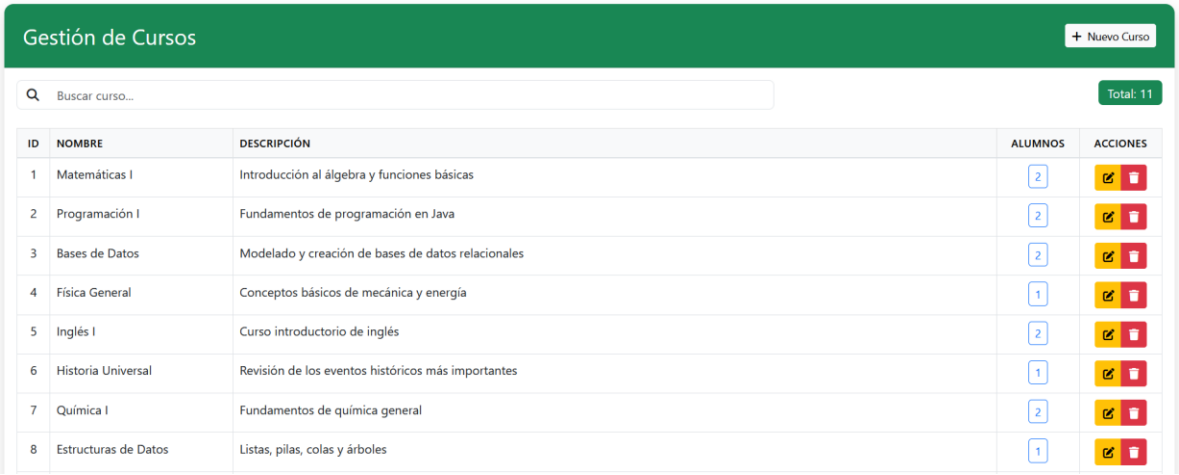
Frontend


Sistema Escuela
Alumnos
Cursos
Gestión de Alumnos
+ Nuevo Alumno
Buscar alumno...
Total: 16
ID
MATRÍCULA
NOMBRE
EDAD
CURSO
ACCIONES
1 A001 Juan Pérez López 20 Matemáticas I
2 A002 María González Ramírez 19 Programación I
3 A003 Pedro Hernández Torres 21 Bases de Datos
4 A004 Laura Martínez Santos 18 Física General
5 A005 Carlos Rojas Mendoza 22 Inglés I
6 A006 Ana Díaz Castillo 20 Historia Universal
7 A007 Luis Morales Reyes 19 Química I
8 A008 Sofía Cabrera Ortiz 21 Estructuras de Datos


Nuevo Alumno
Nombre *
Nombre
Apellido Paterno *
Apellido Paterno
Apellido Materno *
Apellido Materno
Edad *
Edad
Matrícula *
Matrícula
Curso *
Selecciona un curso
Cancelar
Guardar







ID	NOMBRE	DESCRIPCIÓN	ALUMNOS	ACCIONES
1	Matemáticas I	Introducción al álgebra y funciones básicas	2	
2	Programación I	Fundamentos de programación en Java	2	
3	Bases de Datos	Modelado y creación de bases de datos relacionales	2	
4	Física General	Conceptos básicos de mecánica y energía	1	
5	Inglés I	Curso introductorio de inglés	2	
6	Historia Universal	Revisión de los eventos históricos más importantes	1	
7	Química I	Fundamentos de química general	2	
8	Estructuras de Datos	Listas, pilas, colas y árboles	1	

Nuevo Curso

Nombre del Curso *

Nombre del curso

Descripción *

Descripción del curso

Cancelar

Guardar

Editar Curso

Nombre del Curso *

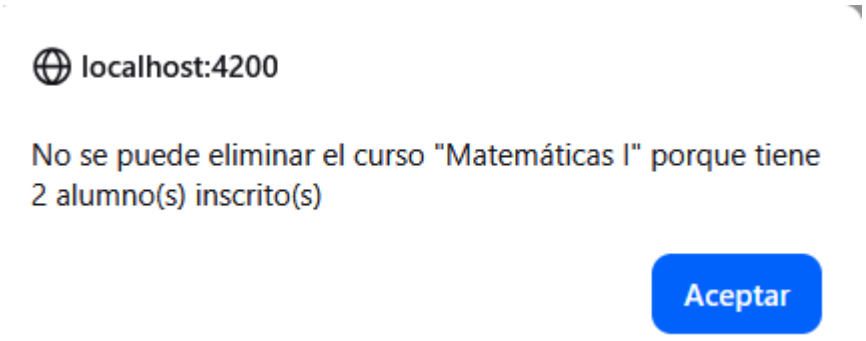
Matemáticas I

Descripción *

Introducción al álgebra y funciones básicas

Cancelar

Actualizar



Para finalizar el Sistema de Registro de Estudiantes y Cursos es una aplicación web completa que implementa las mejores prácticas de desarrollo tanto en frontend como en backend. Utiliza tecnologías modernas y escalables que permiten un mantenimiento sencillo y futuras expansiones. Teniendo una implementación del sistema completa como lo es en sus requerimientos del

CRUD completo funcional en todos los módulos, las validaciones robustas en frontend y backend, Interfaz moderna y responsive una arquitectura escalable y mantenible, código limpio y bien documentado y una experiencia de usuario fluida.

También se logro aprender y reforzar algunos datos vistos en clase y elaborados por uno mismo, como lo es:

- Desarrollo full-stack con Spring Boot y Angular
- Implementación de arquitecturas por capas
- Validaciones del lado del cliente y servidor
- Manejo de peticiones HTTP y observables
- Diseño de interfaces con Bootstrap
- Componentes standalone de Angular 17+
- Gestión de estado y formularios reactivos