**UNIVERSIDAD MAYOR DE SAN ANDRÉS**

**FACULTAD DE CIENCIAS PURAS Y NATURALES**

**CARRERA DE INFORMÁTICA**

****

**“PRÁCTICA 2”**

**MATERIA:** BACKEND - TAW - 251

**PARALELO:** A

**ESTUDIANTE:**

GIRONDA MINA, SAMANTHA ROUSSE

**FECHA**: 19 DE MAYO DEL 2025

LA PAZ - BOLIVIA

# Enlace al código:

<https://github.com/Crissancio/RegistroUniversidad>

## Estructura:

| D:.  │ .gitattributes  │ .gitignore  │ mvnw  │ mvnw.cmd  │ pom.xml  │ README.md  │  ├───.mvn  │ └───wrapper  │ maven-wrapper.properties  │  ├───.vscode  │ settings.json  │  ├───src  │ ├───main  │ │ ├───java  │ │ │ └───com  │ │ │ └───universidad  │ │ │ │ UniversidadApplication.java  │ │ │ │  │ │ │ ├───config  │ │ │ │ AsyncConfig.java  │ │ │ │ CacheConfig.java  │ │ │ │  │ │ │ ├───controller  │ │ │ │ DocenteController.java  │ │ │ │ EstudianteController.java  │ │ │ │ EvaluacionDocenteController.java  │ │ │ │ InscripcionController.java  │ │ │ │ MateriaController.java  │ │ │ │ WebController.java  │ │ │ │  │ │ │ ├───dto  │ │ │ │ DocenteDTO.java  │ │ │ │ EstudianteDTO.java  │ │ │ │ InscripcionDTO.java  │ │ │ │ MateriaDTO.java  │ │ │ │  │ │ │ ├───model  │ │ │ │ Docente.java  │ │ │ │ Estudiante.java  │ │ │ │ EvaluacionDocente.java  │ │ │ │ Inscripcion.java  │ │ │ │ Materia.java  │ │ │ │ Persona.java  │ │ │ │  │ │ │ ├───registro  │ │ │ │ ├───config  │ │ │ │ │ DatabaseInitializer.java  │ │ │ │ │ SecurityConfig.java  │ │ │ │ │  │ │ │ │ ├───controller  │ │ │ │ │ AuthController.java  │ │ │ │ │ UsuarioController.java  │ │ │ │ │  │ │ │ │ ├───dto  │ │ │ │ │ AuthDTO.java  │ │ │ │ │  │ │ │ │ ├───model  │ │ │ │ │ Rol.java  │ │ │ │ │ Usuario.java  │ │ │ │ │  │ │ │ │ ├───repository  │ │ │ │ │ RolRepository.java  │ │ │ │ │ UsuarioRepository.java  │ │ │ │ │  │ │ │ │ ├───security  │ │ │ │ │ JwtAuthenticationEntryPoint.java  │ │ │ │ │ JwtAuthenticationFilter.java  │ │ │ │ │ JwtUtils.java  │ │ │ │ │  │ │ │ │ └───service  │ │ │ │ UserDetailsServiceImpl.java  │ │ │ │  │ │ │ ├───repository  │ │ │ │ DocenteRepository.java  │ │ │ │ EstudianteRepository.java  │ │ │ │ EvaluacionDocenteRepository.java  │ │ │ │ InscripcionRepository.java  │ │ │ │ MateriaRepository.java  │ │ │ │  │ │ │ ├───service  │ │ │ │ │ IDocenteService.java  │ │ │ │ │ IEstudianteService.java  │ │ │ │ │ IEvaluacionDocenteService.java  │ │ │ │ │ IInscripcionService.java  │ │ │ │ │ IMateriaService.java  │ │ │ │ │  │ │ │ │ └───impl  │ │ │ │ DocenteServiceImpl.java  │ │ │ │ EstudianteServiceImpl.java  │ │ │ │ EvaluacionDocenteServiceImpl.java  │ │ │ │ InscripcionServiceImpl.java  │ │ │ │ MateriaServiceImpl.java  │ │ │ │  │ │ │ └───validation  │ │ │ ApiError.java  │ │ │ EstudianteValidator.java  │ │ │ GlobalExceptionHandler.java  │ │ │  │ │ └───resources  │ │ │ application.properties  │ │ │  │ │ ├───static  │ │ │ ├───css  │ │ │ │ style.css  │ │ │ │  │ │ │ └───js  │ │ │ auth.js  │ │ │ docentes.js  │ │ │ inscripciones.js  │ │ │ materias.js  │ │ │  │ │ └───templates  │ │ │ error.html  │ │ │ index.html  │ │ │  │ │ ├───auth  │ │ │ login.html  │ │ │ register.html  │ │ │  │ │ ├───docentes  │ │ │ form.html  │ │ │ list.html  │ │ │  │ │ ├───fragments  │ │ │ footer.html  │ │ │ header.html  │ │ │  │ │ ├───inscripciones  │ │ │ form.html  │ │ │ list.html  │ │ │  │ │ └───materias  │ │ form.html  │ │ list.html  │ │  │ └───test  │ └───java  └───target  │ mi-proyecto-spring-boot-0.0.1-SNAPSHOT.jar  │ mi-proyecto-spring-boot-0.0.1-SNAPSHOT.jar.original  │  ├───classes  │ │ application.properties  │ │  │ ├───com  │ │ └───universidad  │ │ │ UniversidadApplication.class  │ │ │  │ │ ├───config  │ │ │ AsyncConfig.class  │ │ │ CacheConfig.class  │ │ │  │ │ ├───controller  │ │ │ DocenteController.class  │ │ │ EstudianteController.class  │ │ │ EvaluacionDocenteController.class  │ │ │ InscripcionController.class  │ │ │ MateriaController.class  │ │ │ WebController.class  │ │ │  │ │ ├───dto  │ │ │ DocenteDTO$DocenteDTOBuilder.class  │ │ │ DocenteDTO.class  │ │ │ EstudianteDTO$EstudianteDTOBuilder.class  │ │ │ EstudianteDTO.class  │ │ │ InscripcionDTO$InscripcionDTOBuilder.class  │ │ │ InscripcionDTO.class  │ │ │ MateriaDTO$MateriaDTOBuilder.class  │ │ │ MateriaDTO.class  │ │ │  │ │ ├───model  │ │ │ Docente$DocenteBuilder.class  │ │ │ Docente$DocenteBuilderImpl.class  │ │ │ Docente.class  │ │ │ Estudiante$EstudianteBuilder.class  │ │ │ Estudiante$EstudianteBuilderImpl.class  │ │ │ Estudiante.class  │ │ │ EvaluacionDocente.class  │ │ │ Inscripcion$EstadoInscripcion.class  │ │ │ Inscripcion$InscripcionBuilder.class  │ │ │ Inscripcion.class  │ │ │ Materia$MateriaBuilder.class  │ │ │ Materia.class  │ │ │ Persona$PersonaBuilder.class  │ │ │ Persona.class  │ │ │  │ │ ├───registro  │ │ │ ├───config  │ │ │ │ DatabaseInitializer.class  │ │ │ │ SecurityConfig.class  │ │ │ │  │ │ │ ├───controller  │ │ │ │ AuthController.class  │ │ │ │ UsuarioController.class  │ │ │ │  │ │ │ ├───dto  │ │ │ │ AuthDTO$JwtResponse.class  │ │ │ │ AuthDTO$LoginRequest.class  │ │ │ │ AuthDTO$MessageResponse.class  │ │ │ │ AuthDTO$SignupRequest.class  │ │ │ │ AuthDTO.class  │ │ │ │  │ │ │ ├───model  │ │ │ │ Rol$NombreRol.class  │ │ │ │ Rol.class  │ │ │ │ Usuario.class  │ │ │ │  │ │ │ ├───repository  │ │ │ │ RolRepository.class  │ │ │ │ UsuarioRepository.class  │ │ │ │  │ │ │ ├───security  │ │ │ │ JwtAuthenticationEntryPoint.class  │ │ │ │ JwtAuthenticationFilter.class  │ │ │ │ JwtUtils.class  │ │ │ │  │ │ │ └───service  │ │ │ UserDetailsServiceImpl.class  │ │ │  │ │ ├───repository  │ │ │ DocenteRepository.class  │ │ │ EstudianteRepository.class  │ │ │ EvaluacionDocenteRepository.class  │ │ │ InscripcionRepository.class  │ │ │ MateriaRepository.class  │ │ │  │ │ ├───service  │ │ │ │ IDocenteService.class  │ │ │ │ IEstudianteService.class  │ │ │ │ IEvaluacionDocenteService.class  │ │ │ │ IInscripcionService.class  │ │ │ │ IMateriaService.class  │ │ │ │  │ │ │ └───impl  │ │ │ DocenteServiceImpl.class  │ │ │ EstudianteServiceImpl.class  │ │ │ EvaluacionDocenteServiceImpl.class  │ │ │ InscripcionServiceImpl.class  │ │ │ MateriaServiceImpl.class  │ │ │  │ │ └───validation  │ │ ApiError$ApiErrorBuilder.class  │ │ ApiError.class  │ │ EstudianteValidator$BusinessException.class  │ │ EstudianteValidator.class  │ │ GlobalExceptionHandler$RecursoNoDisponibleException.class  │ │ GlobalExceptionHandler.class  │ │  │ ├───static  │ │ ├───css  │ │ │ style.css  │ │ │  │ │ └───js  │ │ auth.js  │ │ docentes.js  │ │ inscripciones.js  │ │ materias.js  │ │  │ └───templates  │ │ error.html  │ │ index.html  │ │  │ ├───auth  │ │ login.html  │ │ register.html  │ │  │ ├───docentes  │ │ form.html  │ │ list.html  │ │  │ ├───fragments  │ │ footer.html  │ │ header.html  │ │  │ ├───inscripciones  │ │ form.html  │ │ list.html  │ │  │ └───materias  │ form.html  │ list.html  │  ├───generated-sources  │ └───annotations  ├───generated-test-sources  │ └───test-annotations  ├───maven-archiver  │ pom.properties  │  ├───maven-status  │ └───maven-compiler-plugin  │ ├───compile  │ │ └───default-compile  │ │ createdFiles.lst  │ │ inputFiles.lst  │ │  │ └───testCompile  │ └───default-testCompile  │ createdFiles.lst  │ inputFiles.lst  │  └───test-classes |
| --- |

## Manual técnico (roles, endpoints, configuración).

El sistema de Registro Universitario es una aplicación web desarrollada con Spring Boot que gestiona el registro de estudiantes, profesores, cursos y calificaciones en una institución educativa. Utiliza PostgreSQL como base de datos relacional y sigue el patrón MVC.

# Arquitectura del Sistema

* **Frontend**: Thymeleaf (plantillas HTML)
* **Backend**: Spring Boot (Java)
* **Base de Datos**: PostgreSQL
* **Autenticación**: Spring Security
* **Build Tool**: Maven

Diagrama de arquitectura:

Cliente → Spring Boot (Controller → Service → Repository) → PostgreSQL

# Requisitos Técnicos

## Software Requerido

* Java JDK 11 o superior
* Maven 3.6+
* PostgreSQL 12+
* IDE (Eclipse, IntelliJ IDEA o VS Code)

## Dependencias Principales

<dependencies>  
 <!-- Spring Boot Starter -->  
 <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>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
   
 <!-- PostgreSQL -->  
 <dependency>  
 <groupId>org.postgresql</groupId>  
 <artifactId>postgresql</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
   
 <!-- Otras dependencias -->  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
</dependencies>

# Configuración del Proyecto

## Configuración de la Base de Datos

Archivo: src/main/resources/application.properties

# Datasource  
spring.datasource.url=jdbc:postgresql://localhost:5432/registro\_universidad  
spring.datasource.username=postgres  
spring.datasource.password=tu\_contraseña  
  
# JPA/Hibernate  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect  
spring.jpa.properties.hibernate.format\_sql=true  
  
# Server  
server.port=8080

## Configuración de Seguridad

Archivo: SecurityConfig.java

@Configuration  
@EnableWebSecurity  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
   
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http  
 .authorizeRequests()  
 .antMatchers("/", "/home", "/register").permitAll()  
 .antMatchers("/admin/\*\*").hasRole("ADMIN")  
 .antMatchers("/profesor/\*\*").hasRole("PROFESOR")  
 .antMatchers("/estudiante/\*\*").hasRole("ESTUDIANTE")  
 .anyRequest().authenticated()  
 .and()  
 .formLogin()  
 .loginPage("/login")  
 .permitAll()  
 .and()  
 .logout()  
 .permitAll();  
 }  
   
 // Configuración de autenticación y codificación de contraseñas  
}

# Estructura de la Base de Datos

## Diagrama ER

Usuario (id, username, password, rol)  
Estudiante (id, nombre, apellido, email, usuario\_id)  
Profesor (id, nombre, apellido, email, usuario\_id)  
Curso (id, nombre, descripción, profesor\_id)  
Calificación (id, estudiante\_id, curso\_id, nota, comentario)  
Inscripción (id, estudiante\_id, curso\_id, fecha)

## Scripts SQL

CREATE TABLE usuario (  
 id SERIAL PRIMARY KEY,  
 username VARCHAR(50) UNIQUE NOT NULL,  
 password VARCHAR(100) NOT NULL,  
 rol VARCHAR(20) NOT NULL  
);  
  
CREATE TABLE estudiante (  
 id SERIAL PRIMARY KEY,  
 nombre VARCHAR(50) NOT NULL,  
 apellido VARCHAR(50) NOT NULL,  
 email VARCHAR(100) UNIQUE NOT NULL,  
 usuario\_id INTEGER REFERENCES usuario(id)  
);  
  
-- Tablas restantes con relaciones similares

# Roles y Seguridad

## Roles del Sistema

1. **ADMIN**: Acceso completo al sistema
   * Gestionar usuarios
   * Gestionar cursos
   * Asignar profesores
   * Generar reportes
2. **PROFESOR**: Acceso limitado
   * Ver sus cursos
   * Registrar calificaciones
   * Ver listado de estudiantes
3. **ESTUDIANTE**: Acceso básico
   * Ver sus cursos
   * Ver calificaciones
   * Inscribirse a cursos

## Flujo de Autenticación

1. Usuario ingresa credenciales
2. Sistema verifica en base de datos
3. Se establece sesión con roles correspondientes
4. Se redirige según rol

# Endpoints API

## Autenticación

| Método | Endpoint | Descripción | Rol Requerido |
| --- | --- | --- | --- |
| GET | /login | Página de login | Público |
| POST | /login | Procesar login | Público |
| GET | /logout | Cerrar sesión | Todos |
| POST | /register | Registrar nuevo usuario | Público |

## Administración (ADMIN)

| Método | Endpoint | Descripción |
| --- | --- | --- |
| GET | /admin/usuarios | Listar usuarios |
| POST | /admin/usuarios | Crear usuario |
| GET | /admin/cursos | Listar cursos |
| POST | /admin/cursos | Crear curso |
| GET | /admin/reportes | Generar reportes |

## Profesor

| Método | Endpoint | Descripción |
| --- | --- | --- |
| GET | /profesor/cursos | Ver cursos asignados |
| GET | /profesor/curso/{id} | Ver detalles de curso |
| POST | /profesor/calificacion | Registrar calificación |

## Estudiante

| Método | Endpoint | Descripción |
| --- | --- | --- |
| GET | /estudiante/cursos | Ver cursos inscritos |
| POST | /estudiante/inscripcion | Inscribirse a curso |
| GET | /estudiante/calificaciones | Ver calificaciones |

# Despliegue

## Local

1. Clonar repositorio
2. Configurar PostgreSQL (crear DB y usuario)
3. Actualizar application.properties
4. Ejecutar:

mvn spring-boot:run