# Sistema de Alerta y Visualización de Sismos - Gestión de Usuarios con Spring Boot

## Descripción General del Sistema

### Arquitectura

- **Framework**: Spring Boot 3.4.2
- **Lenguaje de Programación**: Java 21
- **Gestión de Dependencias**: Maven
- **Base de Datos**: MySQL

## Componentes Principales

### 1. Configuración de Seguridad

Archivo: `SecurityConfig.java`

- Configuración de Spring Security
- Gestión de autenticación y autorización
- Configuración de rutas protegidas y públicas

#### Características Clave:

- Desactivación de CSRF (configurable)
- Rutas públicas: `/registro`, `/login`, `/testConnection`
- Rutas protegidas por roles:
  - `/admin/**`: Acceso exclusivo para ROLE_ADMIN
  - `/user/**`: Acceso para ROLE_USER

### 2. Modelo de Datos

#### UserModel

- Entidad JPA para representar usuarios
- Atributos:
  - `id`: Identificador único
  - `username`: Nombre de usuario (único)
  - `email`: Correo electrónico (único)
  - `password`: Contraseña (encriptada)
  - `roles`: Conjunto de roles asignados

### Role

- Entidad JPA para representar roles de usuario
- Atributos:
    - `id`: Identificador único
    - `name`: Nombre del rol (ej. ROLE_USER, ROLE_ADMIN)

## 3. Repositorios

- `UserRepository`: Operaciones CRUD para usuarios
- `RoleRepository`: Operaciones CRUD para roles
- Utiliza Spring Data JPA para consultas

## 4. Servicios

### UsuarioService

Gestiona la lógica de negocio para usuarios:

- Registro de usuarios
- Actualización de perfiles
- Gestión de roles
- Encriptación de contraseñas

### CustomUserDetailsService

- Implementa `UserDetailsService`
- Carga usuarios desde la base de datos
- Convierte roles de usuario en authorities de Spring Security

## 5. Controladores

### Funcionalidades Principales

- `LoginController`: Manejo de página de login
- `RegistroController`: Registro de nuevos usuarios
- `HomeController`: Página de inicio con acceso diferenciado por roles
- `AdminController`: Gestión de usuarios (solo para administradores)
- `PerfilController`: Gestión de perfil de usuario

## 6. Configuración de Base de Datos

Archivo: `application.properties` (no mostrado en los documentos) Configuración típica:

```
properties
```

Copy

```
spring.datasource.url=jdbc:mysql://localhost:3306/tarea2
spring.datasource.username=admin
spring.datasource.password=admin
spring.jpa.hibernate.ddl-auto=update
```

### 7. Dependencias Principales

- Spring Boot Web
- Spring Security
- Spring Data JPA
- Thymeleaf (templates)
- MySQL Connector
- Bootstrap (frontend)

# Flujo de Autenticación

1. Usuario accede a `/login`
2. Ingresa credenciales
3. `CustomUserDetailsService` valida credenciales
4. Spring Security autentica y asigna roles
5. Redirige según rol (admin/user)

# Seguridad y Mejores Prácticas

### Autenticación

- Contraseñas encriptadas con BCrypt
- Manejo de sesiones
- Protección contra ataques (CSRF configurable)

### Autorización

- Control de acceso basado en roles
- Uso de `@PreAuthorize` para validaciones de seguridad
- Rutas diferenciadas según privilegios

# Consideraciones de Implementación

### Aspectos a Mejorar

- Implementar validaciones más robustas
- Añadir manejo de excepciones personalizado
- Considerar habilitar CSRF en producción
- Implementar recuperación de contraseña

### Recomendaciones

- Usar variables de entorno para credenciales
- Implementar logging
- Añadir validaciones de complejidad de contraseña
- Considerar implementación de autenticación de dos factores

## Tecnologías y Versiones

- Java 21
- Spring Boot 3.4.2
- Spring Security
- MySQL 8.0.33
- Bootstrap 5.3.2