Sistema de Alerta y Visualización de Sismos

Versión 0.0.1

Integrantes:

- Fuentes Reyes Jayri Arath
- Gathe Esquivel Arleth
- Oviedo Lopez Joshua
- Reyes Núñez Sebastián

Ingeniería de Software

Prof. Gabriel Hurtado Avilés

Grupo: 6CV3

Equipo: 4

Índice

Historial de Versiones

Fecha	Versión	Descripción	Autores
[05/03/2025]	[0.0.1]	[Sistema de alerta y visualización de sismos, Título de Empresa, Desarrollo del documento de plan de proyecto donde se establecen los objetivos alcanzar respectivamente a los módulos de Login/Signup y CRUD, restricciones, FURP y guías técnicas]	Fuentes Reyes Jayri Arath Gathe Esquivel Arleth Oviedo Lopez Joshua Reyes Nuñez Sebastián

1. Documento de Requerimientos del Proyecto

1.1 Descripción del Proyecto

El presente proyecto tiene como objetivo desarrollar un sistema de autenticación basado en roles y permisos utilizando Spring Boot. Este sistema permitirá a los usuarios registrarse e iniciar sesión de forma segura, garantizando un acceso controlado a los recursos del sistema según sus privilegios asignados. La implementación de este sistema sentará las bases para la construcción de futuras funcionalidades avanzadas, asegurando la escalabilidad y seguridad de la plataforma.

El sistema manejará distintos tipos de roles, incluyendo un Administrador con privilegios completos para realizar operaciones CRUD sobre entidades y usuarios, así como un Usuario Estándar con acceso limitado solo a la consulta de sus propios datos. Para garantizar la seguridad de la información, todas las contraseñas serán encriptadas con algoritmos seguros como BCrypt y la gestión de sesiones se llevará a cabo mediante JWT (JSON Web Token), permitiendo autenticación sin estado y facilitando la integración con otros servicios.

Además de la autenticación, el sistema incluirá mecanismos de autorización que restringirán el acceso a ciertos recursos en función de los permisos del usuario. La seguridad se implementará a nivel de controlador utilizando anotaciones como @PreAuthorize y @Secured, asegurando que solo los usuarios con los permisos adecuados puedan acceder a las distintas funcionalidades.

La información de los usuarios se almacenará en una base de datos MySQL gestionada a través de Spring Data JPA. Se proporcionará una configuración clara en application.properties para facilitar la conexión y administración de la base de datos. Adicionalmente, el desarrollo del proyecto se gestionará bajo la metodología ágil Scrum, utilizando la plataforma ClickUp para la planificación y monitoreo del progreso.

En paralelo a la implementación del sistema de autenticación, se desarrollará un **Sistema de Alerta y Visualización de Sismos**, cuyo propósito es identificar zonas de ocurrencia de eventos sísmicos en México utilizando trilateración con datos del

Servicio Sismológico Nacional (SSN). Este sistema generará grafos de conocimiento que representarán relaciones clave entre eventos sísmicos, facilitando su análisis y visualización sobre un mapa interactivo de México. Se integrarán tecnologías de bases de datos orientadas a grafos y APIs de mapas como Google Maps para proporcionar una experiencia intuitiva y enriquecida para los usuarios.

Este proyecto busca combinar la seguridad informática con el análisis de datos en tiempo real, asegurando un desarrollo sólido y bien documentado que sirva como base para futuras mejoras y expansiones del sistema.

1.2 Objetivos Principales

- Desarrollar un sistema de autenticación seguro con encriptación de contraseñas.
- Implementar roles diferenciados con permisos de acceso y restricción de acciones.
- Utilizar JWT para la gestión de sesiones.
- Almacenar información de usuarios en una base de datos MySQL con Spring Data JPA.
- Gestionar el desarrollo del proyecto con metodología ágil (Scrum).
- Desarrollar un sistema web que tenga como finalidad ser un apoyo en la gestión y seguimiento de un sismo.
- El sistema deberá poder dar seguimiento a un incidente de sismo desde el momento en el que se detecta hasta el momento que se levanta un reporte de revisión posterior a la acción y cierre del evento.
- El sistema deberá poder visualizar la información correspondiente al incidente de un sismo después de que haya finalizado.
- El sistema deberá permitirle al usuario visualizar un mapa con un área estimada del sismo.

1.3 Reglas y Restricciones

- Solo los administradores pueden realizar operaciones CRUD sobre entidades del sistema.
- Los usuarios estándar solo pueden acceder a sus propios datos y realizar operaciones de lectura.
- Las contraseñas deben ser almacenadas de manera segura con BCrypt.
- El acceso a los endpoints debe estar restringido mediante anotaciones de seguridad (@PreAuthorize, @Secured).

1.4 Roles

- Roles de Autenticación y Acceso

1. Administrador (Admin):

   Tiene acceso completo al sistema.
   Puede realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre entidades y usuarios.
   Puede gestionar roles y permisos de otros usuarios.

2. Usuario Estándar (User):

   Acceso limitado a la consulta de sus propios datos.
   No puede realizar modificaciones en el sistema, solo operaciones de lectura.

3. Invitado (Guest):

   Acceso limitado a la visualización de información pública, como mapas de sismos o datos históricos.
   No puede realizar acciones que requieran autenticación.

- Roles Específicos para el Sistema de Sismos

4. Analista de Sismos:

   Puede acceder a datos sísmicos, realizar análisis y generar reportes.
   Puede visualizar mapas interactivos y grafos de conocimiento.

5. Supervisor de Seguridad:

Encargado de supervisar la seguridad del sistema.
Puede revisar logs de acceso y auditorías.

6. Desarrollador (Developer):

Acceso a configuraciones técnicas y guías de implementación.
Puede realizar pruebas y ajustes en el sistema.

- Roles de Soporte y Mantenimiento

7. Soporte Técnico:

Encargado de resolver problemas técnicos y brindar soporte a los usuarios.
Puede acceder a la configuración del sistema, pero no modificar datos críticos.

8. Auditor:

Puede revisar el historial de cambios y acceder a versiones anteriores del sistema.
Encargado de verificar el cumplimiento de políticas de seguridad.

- Roles de Gestión de Proyecto

9. Gestor de Proyecto (Project Manager):

Encargado de supervisar el desarrollo del proyecto.
Puede acceder a la planificación y seguimiento del progreso en herramientas como ClickUp.

10. Scrum Master:

Facilita las reuniones de Scrum y asegura que se sigan las metodologías ágiles.
Puede acceder a la documentación del proyecto y a los informes de progreso.

- Roles de Visualización y Análisis

11. Visualizador de Mapas:

Puede acceder a la visualización de mapas interactivos de sismos.
No tiene permisos para modificar datos o configuraciones.

12. Investigador:

Puede acceder a datos históricos de sismos y realizar análisis predictivos.
Puede generar reportes y gráficos, pero no modificar datos originales.

- Roles de Seguridad

13. Seguridad Avanzada:

    Puede configurar y modificar políticas de seguridad, como la gestión de JWT (JSON Web Tokens) y la encriptación de contraseñas.
    Encargado de asegurar que los endpoints estén protegidos con anotaciones como @PreAuthorize y @Secured.
    - Roles de Base de Datos

14. Administrador de Base de Datos (DBA):

    Encargado de gestionar la base de datos MySQL.
    Puede realizar operaciones de mantenimiento, como backups y restauraciones.

15. Desarrollador de Grafos:

    Encargado de gestionar la base de datos orientada a grafos y la generación de grafos de conocimiento.

2. Documento FURPS+ del Proyecto Grupal

2.1 Funcionalidad

1. Sistema de registro y gestión de usuarios

- Implementación de login y registro.
- Diferenciación de roles: Administrador y Usuario Estándar.
- Encriptación segura de contraseñas con BCrypt.
- Gestión de sesiones mediante JWT.

2.2 Usabilidad

- Interfaces de usuario intuitivas para login y registro.
- Mensajes de error y validaciones amigables para el usuario.
- Opciones intuitivas para explorar el mapa y realizar análisis.
- Permitir zoom, arrastre y selección de áreas específicas en el mapa.

2.3 Confiabilidad

- Implementación de medidas de seguridad en la autenticación.
- Restricción de accesos basada en roles.
- Actualizaciones periódicas automáticas de la base de datos de sismos y grafos.
- Restauración de datos en caso de fallos.

2.4 Rendimiento

- Tiempo de respuesta óptimo para solicitudes de autenticación.
- Procesar un conjunto de 10,000 eventos sísmicos en menos de 30 segundos.
- Generar grafos de conocimiento para hasta 1,000 eventos en menos de 15 segundos.

2.5 Soporte

- Documentación clara sobre la configuración y el uso del sistema.
- Integración con ClickUp para la gestión del desarrollo.
- Exportación de datos en formatos JSON y CSV.
- Soporte para reportar errores o solicitar mejoras.

Requerimientos FURPS+ para un Sistema de Alerta y Visualización de Sismos

Propósito: Este sistema tiene como propósito identificar zonas de ocurrencia de eventos sísmicos en México mediante trilateración, utilizando datos del Servicio Sismológico Nacional (SSN) y bases de datos complementarias. El sistema generará grafos de conocimiento para modelar entidades y relaciones clave, visualizándolos sobre un mapa de México.

Funcionales (F)

1. Procesamiento de datos sísmicos:
    - Importar datos desde el SSN en tiempo real mediante API o archivo CSV.
    - Filtrar y clasificar sismos según magnitud, profundidad y ubicación geográfica.
    - Realizar cálculos de trilateración considerando datos provenientes del SSN.
2. Generación y almacenamiento de grafos de conocimiento:
    - Crear nodos para entidades clave (ubicación, magnitud, profundidad, fecha).
    - Establecer relaciones entre nodos (zona de impacto, réplicas, epicentro).
    - Utilizar bases de datos orientadas a grafos.
3. Visualización:
    - Mostrar la información en un mapa interactivo de México con ayuda de alguna API gratuita, Google Maps API o alguna tecnología similar.
    - Superponer grafos en el mapa para explorar conexiones entre eventos sísmicos.
4. Análisis histórico y predicción:

- Realizar análisis del comportamiento histórico de sismos en la costa del Pacífico.
- Implementar algoritmos de inferencia probabilística para predecir eventos futuros.

3. Guías Técnicas

Esta guía técnica proporciona un análisis detallado del desarrollo y la implementación de un Sistema de Alerta y Visualización de Sismos basado en Spring Boot, con énfasis en la seguridad, roles de usuarios y la gestión de datos sísmicos.

Objetivo: Facilitar la implementación y gestión del sistema, abarcando desde la configuración del sistema de autenticación hasta la visualización de eventos sísmicos en tiempo real.

3.1 Conexión a la Base de Datos

Configuración en application.properties:

spring.datasource.url=jdbc:mysql://localhost:3306/autenticacion_db

spring.datasource.username=root

spring.datasource.password=admin

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

Analicemos cada linea correspondiente al “application.properties” de igual forma podemos utilizar application.yaml para poder entender de mejor manera lo que sucede:

```yaml
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/autenticacion_db
    username: root
    password: admin
  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    database-platform: org.hibernate.dialect.MySQL8Dialect
```

Application.yaml

Observamos que el application.properties se segmenta en lo que es “datasource” y “jpa”, en la parte de datasource es la conexion a la base de datos:

- Url: alberga la dirección de la base, en este caso muestra una base de datos local en el puerto 3306, donde la base se llama “autenticacion_db”
- Username: La cuenta con la cual ingresara springboot a la base de datos

- Password: La contraseña que springboot usa para ingresar.

Dentro de JPA vemos como springboot manejara el schema de la base de datos, es decir, como la creara o modificara y vemos que “ddl-auto=update” es para que spring actualice de forma automatica el schema si es que se presentan cambios en nuestra JPA (Java Persistence API) entidades, de tal manera que nosotros no debemos de actualizar la base de datos de forma manual.

- “Show-sql = true”: Muestra los queries que springboot hace en la terminal cuando interactua con la base de datos.
- “database-platform=org.hibernate.dialect.MySQL8Dialect”: Le decimos Hibernate que dialecto usar para poder comprender la sintaxis de SQL que usa nuestra base de datos, en este caso al tener nuestro proyecto en MySQL 8, ponemos “MySQL8Dialect”.

Este aplication.properties nos presenta como nuestra aplicación de Spring se conecta a nuestra Base de datos local de MySQL, la configuración de Hibernate que actualiza el schema de la base de datos, muestra los queries de SQL en el log para la poder llevar un registro adecuado.

3.2 Configuración del Sistema de Seguridad (Spring Security)

La clase **SecurityConfig.java** proporcionada es una configuración de seguridad en una aplicación de Spring Boot utilizando **Spring Security** y otros mecanismos como **BCryptPasswordEncoder** para encriptación de contraseñas, manejo de sesiones y protección contra ataques comunes. Aquí te explico en detalle cada parte de la configuración:

1. Dependencias e Importaciones

```
package mx.ipn.escom.ProyectoFinal.Config;

import mx.ipn.escom.ProyectoFinal.services.UserService;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.ProviderManager;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.config.Customizer;
import org.springframework.security.web.session.HttpSessionEventPublisher;
import org.springframework.security.config.http.SessionCreationPolicy;
```

Dependencias para SecurityConfig

Estas importaciones son necesarias para configurar la seguridad del proyecto:

- **UserService**: Este servicio es el que se encargará de gestionar los detalles del usuario (como cargar los datos del usuario desde la base de datos para la autenticación).
- **AuthenticationManager**: Es la interfaz principal que se encarga de manejar el proceso de autenticación. Es responsable de recibir las credenciales de los usuarios (nombre de usuario y contraseña) y autenticar al usuario en el sistema.
- **ProviderManager**: Este es el administrador de proveedores de autenticación, que se encarga de gestionar y coordinar los distintos mecanismos de autenticación (en este caso, usando DaoAuthenticationProvider).
- **DaoAuthenticationProvider**: Es un proveedor que se utiliza para autenticar a los usuarios mediante la consulta a la base de datos.
- **BCryptPasswordEncoder**: Utilizado para encriptar y verificar contraseñas de manera segura.
- **SecurityFilterChain**: Es un filtro que permite gestionar las reglas de seguridad a nivel de las rutas HTTP de la aplicación.
- **HttpSessionEventPublisher**: Utilizado para manejar los eventos relacionados con las sesiones, como invalidarlas si se detecta que un usuario ha iniciado sesión en otro lugar.

2. Configuración de la Contraseña con BCrypt

```
@Bean
public BCryptPasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

BCryptPasswordEncoder implementación

BCrypt es un algoritmo de hashing utilizado para almacenar contraseñas de manera segura. A diferencia de los algoritmos de hash tradicionales, **BCrypt** utiliza un "salting" (un valor aleatorio agregado a la contraseña antes de hacer el hash) y un "cost factor" (que hace que el algoritmo sea más lento y, por lo tanto, más difícil de vulnerar mediante ataques de fuerza bruta).

El **BCryptPasswordEncoder** es un **PasswordEncoder** que utiliza **BCrypt** para encriptar contraseñas. En este caso, el **@Bean** está registrando el **BCryptPasswordEncoder** en el contexto de la aplicación, lo que significa que Spring Boot puede inyectarlo automáticamente en otros lugares cuando se necesite para encriptar o verificar contraseñas.

3. Configuración del AuthenticationManager

```java
@Bean
public AuthenticationManager authenticationManager(UserService userService) {
    DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
    authProvider.setUserDetailsService(userService);
    authProvider.setPasswordEncoder(passwordEncoder());
    return new ProviderManager(List.of(authProvider));
}
```

4. Configuración de SecurityFilterChain
   Esta es la parte más importante del código, ya que es donde configuramos las reglas de seguridad para nuestras rutas y gestionamos las sesiones de los usuarios.

   authorizeHttpRequests(): Esta sección configura qué usuarios pueden acceder a ciertas rutas en función de sus roles. Aquí se definen las reglas para la autorización de los usuarios.

   - **.requestMatchers("/api/usuarios/**").hasAuthority("ROLE_ADMIN**"):** Las rutas bajo /api/usuarios/** solo serán accesibles por usuarios con el rol ROLE_ADMIN.
   - **.requestMatchers("/admin", "/admin/**").hasAuthority("ROLE_ADMIN"):** Lo mismo ocurre con las rutas bajo /admin que solo pueden ser accedidas por usuarios con el rol ROLE_ADMIN.
   - **.requestMatchers("/usuario/**").hasAuthority("ROLE_USER"):** Las rutas bajo /usuario solo pueden ser accedidas por usuarios con el rol ROLE_USER.
   - **.anyRequest().permitAll**(): Las demás rutas estarán accesibles para todos los usuarios (sin autenticación).
   - **httpBasic()**: Habilita la autenticación básica (básicamente, un pop-up para ingresar el nombre de usuario y la contraseña). Esto se usa generalmente en aplicaciones de servicio web, aunque se utiliza menos en aplicaciones web con formularios de login modernos.

   - **formLogin():** Aquí configuramos un login personalizado. En lugar de la página de login predeterminada de Spring Security, definimos una página personalizada en /login, que es donde los usuarios ingresarán sus credenciales.

   - **.loginPage("/login")**: Especifica la URL para la página de inicio de sesión personalizada.
   - **.loginProcessingUrl("/perform_login")**: URL para el procesamiento de la autenticación.
   - **.defaultSuccessUrl("/redirect", true)**: En caso de éxito, el usuario será redirigido a /redirect.

- **.failureUrl("/login?error=true")**: En caso de error de autenticación, el usuario será redirigido de nuevo a /login?error=true.
- **logout()**: Configura el cierre de sesión. Cuando un usuario haga logout, será redirigido a la página de login con el parámetro ?logout en la URL.
- **.logoutUrl("/logout")**: URL donde el usuario puede cerrar sesión.
- **.logoutSuccessUrl("/login?logout")**: URL a la que será redirigido el usuario después de hacer logout.
- **.sessionFixation().migrateSession()**: Protege contra un ataque de secuestro de sesión, asegurándose de que se cree una nueva sesión cuando el usuario inicie sesión.
- **.maximumSessions(1)**: Limita a un solo inicio de sesión por usuario. Si un usuario intenta iniciar sesión en otro dispositivo o ventana, se cerrará su sesión en el primer dispositivo.
- **.expiredUrl("/login?expired")**: Redirige al usuario a la página de login si la sesión ha expirado.
- **csrf()**: Deshabilita la protección CSRF (Cross-Site Request Forgery). Esto se hace porque en muchos casos, cuando la aplicación utiliza APIs RESTful o se integra con un front-end separado, esta protección no es necesaria.

5. Manejo de Eventos de Sesión

```java
// Manejo de eventos de sesión (para invalidar sesión si el usuario se autentica en otro lado)
@Bean
public HttpSessionEventPublisher httpSessionEventPublisher() {
    return new HttpSessionEventPublisher();
}
```

HttpSessionEventPublisher

HttpSessionEventPublisher: Este bean es importante para manejar eventos de sesión. Permite, por ejemplo, invalidar la sesión de un usuario si se detecta que ha iniciado sesión en otro dispositivo. De este modo, se protegen los usuarios contra el secuestro de sesiones.

3.3 Roles, Permisos y Endpoints Protegidos

Login y Manejo de Roles:

1. SecurityConfig.java - reglas de seguridad, rutas protegidas, rutas con autenticacion
2. AuthRestController.java - login desde Postman
3. LoginController.java - maneja pantalla de login
4. Login.html

Base de datos

1. application.properties - configura conexión a la BD
2. UserRepository.java/RoleRepository.java - interactúan con la BD
3. Usuario.java/Rol.java - representan las tablas de la BD

Conf Spring Boot

1. application.properties
2. pom.xml – Dependencias del Proyecto