INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

PRACT 3.
MODELO DE DISEÑO

Profesor: Hurtado Avilés Gabriel
Grupo: 6CV3

Integrantes:
Fuentes Reyes Jayri Arath
Gathe Esquivel Arleth Elizabeth
Oviedo López Joshua
Reyes Nuñez Sebastián

Status In progress

Product requirements

Product manager, Jayri Fuentes
Engineer, Arleth Gathe
UX designer, Sebastián Reyes
UX researcher, Joshua Lopez

Introduction

Purpose

The Earthquake Alert and Visualization System with Role-Based Authentication is designed to provide real-time monitoring, analysis, and secure dissemination of seismic data. Its purpose is to:

- Deliver immediate earthquake alerts to mitigate risks to human life and infrastructure.
- Enable controlled access to critical seismic data through role-based authentication (JWT/BCrypt).

- Facilitate data-driven decision-making for disaster response via interactive visualization tools (Leaflet.js).

Problem statement

Mexico's high seismic activity demands reliable systems for real-time data processing and secure access. Current challenges include:

- Delayed or fragmented information: Existing systems lack integration with authoritative sources like the National Seismological Service (SSN).
- Security gaps: Unauthorized access to sensitive seismic data poses risks of misuse.
- Limited usability: Non-intuitive interfaces hinder effective use by non-technical stakeholders.
- This system addresses these gaps by combining real-time data trilateration, role-based security, and user-centric design.

Target audience

User Group	Demographics	Pain Points	Needs	User Group
Civil Protection	Government agencies in Mexico	Delayed alerts, incomplete data	Real-time data for emergency planning	**Civil Protection**
General Public	Residents in seismic zones	Lack of accessible, actionable alerts	Simple interface to view risks and history	**General Public**
System Admins	Technical staff at SSN/ESCOM	Manual data entry, security vulnerabilities	Secure CRUD operations and user management	**System Admins**

Objectives

Goals

1. Real-Time Visualization: Display seismic events on an interactive map using SSN data and trilateration.
2. Secure Access: Implement role-based authentication (JWT) to restrict data modification to admins.
3. Scalability: Design a microservices architecture (Spring Boot) to handle 10,000+ concurrent users.

Non-goals

1. Earthquake Prediction: Machine learning for predictions is excluded (planned for future iterations).
2. Hardware Deployment: The system focuses on software, not sensor hardware.
3. Global Coverage: Initially targets Mexico; expansion to other regions is optional.

Metrics

Quantitative Metrics

- Latency: <500 ms response time for data queries (measured via JMeter).
- Uptime: 99.9% availability (monitored with Spring Boot Actuator).
- Security: 0.5% max error rate in authentication (audit logs).
- Coverage: 95% test coverage for critical modules (JUnit + JaCoCo).

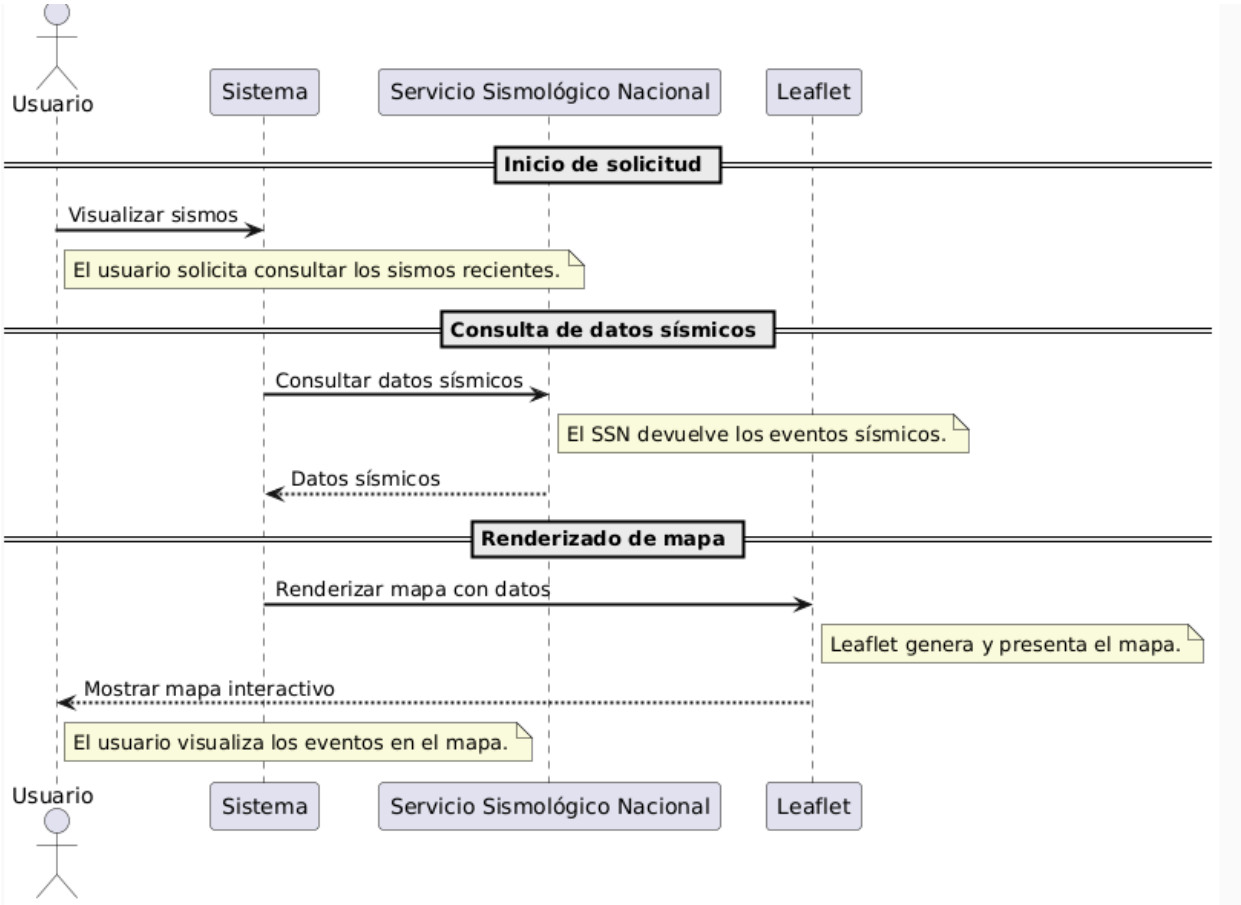
Qualitative Metrics

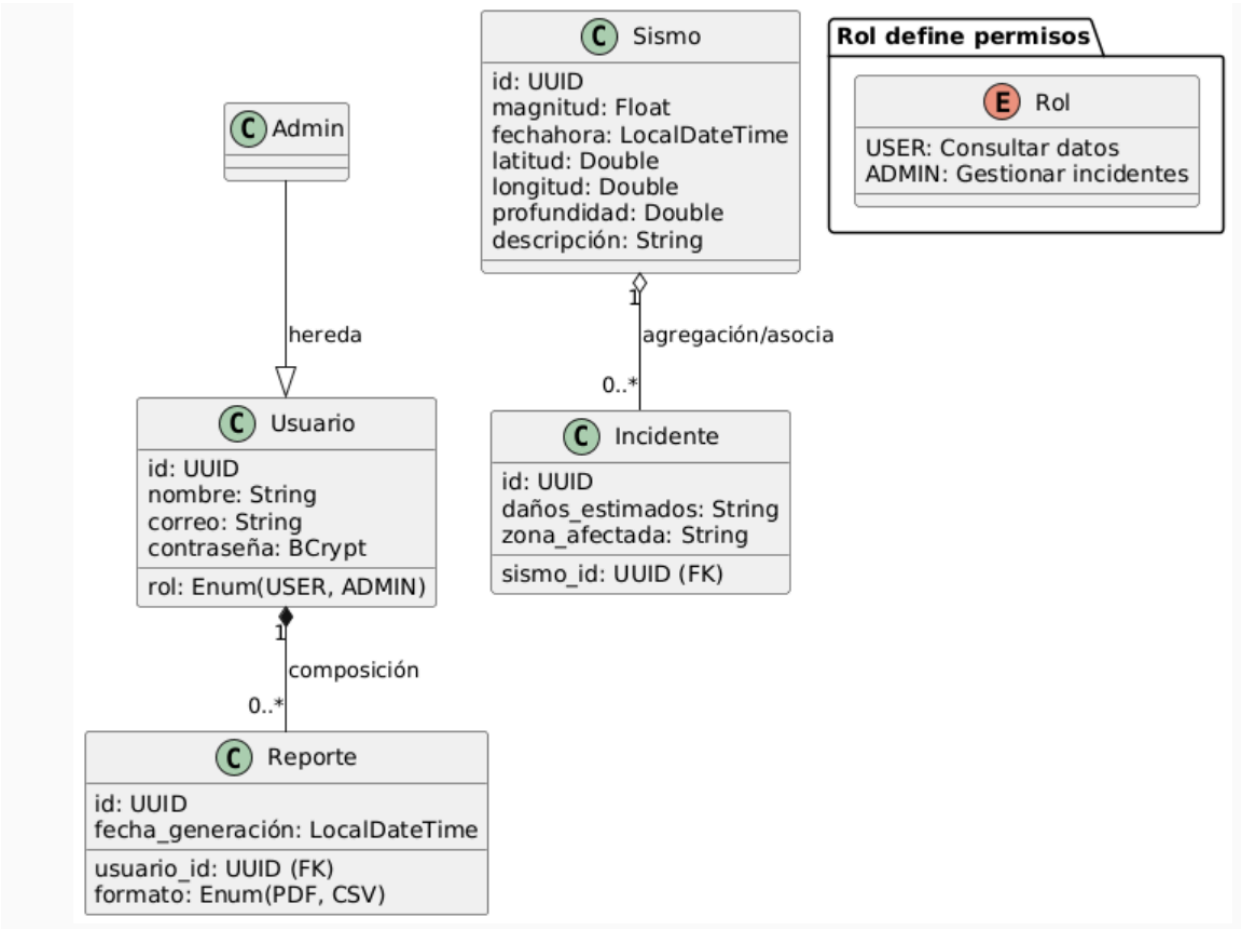
- User Satisfaction: ≥4.5/5 rating in post-implementation surveys (ease of use, clarity).
- Stakeholder Adoption: Adoption by 3+ government agencies within 6 months.
- Impact: 30% reduction in false alerts and delayed responses (measured via incident reports).

1. Modelo de Dominio y Diagrama de Clases Conceptuales

Ejercicio 1

Diagrama de Clases Conceptuales (UML)





Clases principales:

- Usuario: atributos: id, nombre, correo, contraseña (BCrypt), rol (USER/ADMIN).
- Sismo: atributos: id, magnitud, fechaHora, latitud, longitud, profundidad, descripción.
- Incidente: atributos: id, sismo_id (FK), daños_estimados, zona_afectada.
- Reporte: atributos: id, usuario_id (FK), formato (PDF/CSV), fecha_generación.

Relaciones:

- Usuario 1..* → Reporte (composición).
- Admin 1..* → Incidente (agregación).
- Sismo 1..1 → Incidente (asociación).

2. Modelo de dominio

Reglas de negocio:

- Solo los Admins pueden crear/editar/eliminar Incidentes (CU03, CU05).
- Las contraseñas se almacenan cifradas con BCrypt (invariante de seguridad).
- Cada Sismo debe tener magnitud ≥ 1.0 y ubicación geográfica válida.
- Restricciones:
- Campos obligatorios: Sismo (magnitud, fechaHora, latitud, longitud).
- Unicidad: Correo de Usuario, id de Sismo.

Restricciones:

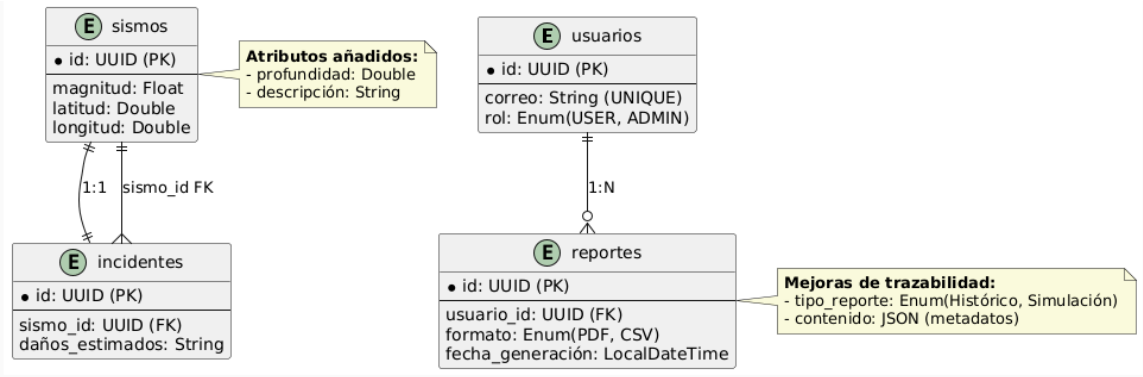
- Campos obligatorios: Sismo (magnitud, fechaHora, latitud, longitud).
- Unicidad: Correo de Usuario, id de Sismo.

3. Diccionario de datos.

Entidad	Descripción	Atributos	Restricciones
Usuario	Usuarios del sistema	id (UUID), nombre (String), correo (String, único), contraseña (String, BCrypt), rol (Enum)	Correo válido, contraseña ≥ 8 caracteres
Sismo	Evento sísmico registrado	id (UUID), magnitud (Float), fechaHora (LocalDateTime),	Magnitud ≥ 1.0

Entidad	Descripción	Atributos	Restricciones
		latitud (Double), longitud (Double)	

4. Diagrama Entidad-Relación (ER)



Tablas:

- usuarios (id PK, correo UNIQUE, rol).
- sismos (id PK, magnitud, latitud, longitud).
- incidentes (id PK, sismo_id FK, daños_estimados).

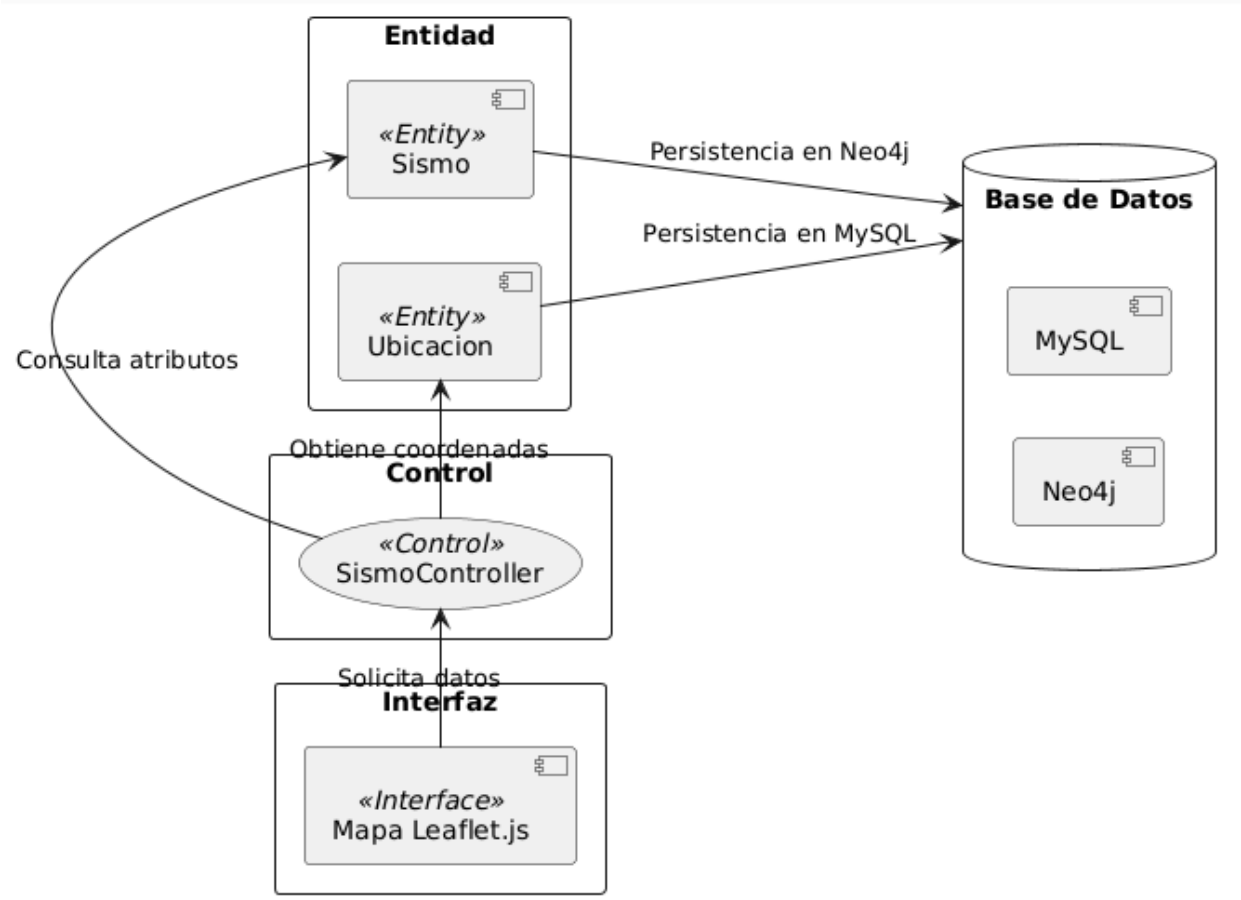
Relaciones:

- sismos → incidentes (1:1).
- usuarios → reportes (1:N).

5. Diagramas de Secuencia y Robustez

Ejercicio 2

Diagrama de Secuencia (CU02 - Visualizar Sismos)



- Actores: Usuario, Sistema.

Pasos:

- Usuario solicita visualizar sismos.
- Sistema valida autenticación (JWT).
- Sistema consulta SSN y BD (Neo4j).
- Sistema renderiza mapa con Leaflet.js.

- Diagrama de Robustez (CU02)

Objetos:

- Interfaz: Mapa Leaflet.js.
- Control: SismoController (Spring Boot).
- Entidad: Sismo, Ubicacion.
- Conexiones: Interfaz → Control → Entidad → BD.

- Modelo de Interfaz y Navegación

Pantallas principales:

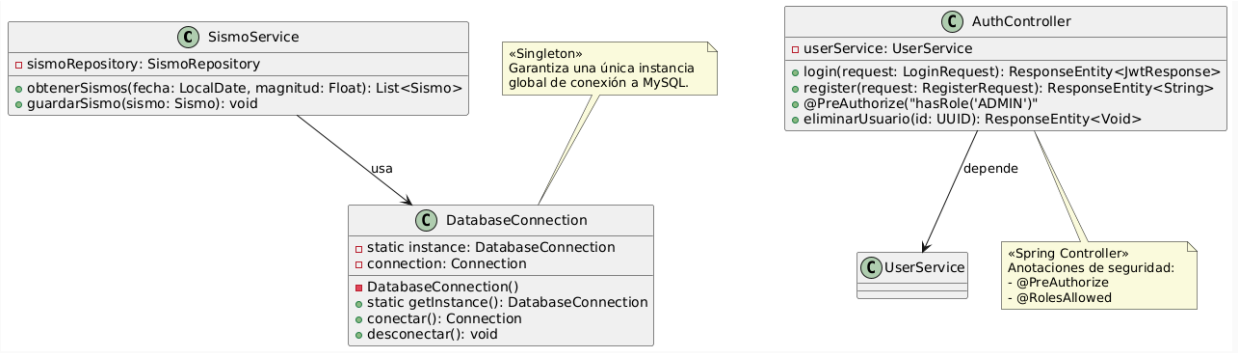
- Login (CU01).
- Mapa interactivo (CU02).
- Navegación por roles:
- Usuario: Mapa → Historial (CU04)
- Admin: Mapa → Gestionar Incidentes (CU03/CU05) → Alta Admins (CU06).

6. Implementación y Patrones

Ejercicio 3

1. Diagrama de Clases de Diseño

Diagrama de Clases de Diseño



- Clases detalladas:
    a. SismoService: Métodos: obtenerSismos(fecha, magnitud) → List<Sismo>.
    b. AuthController: Anotaciones Spring Security (@PreAuthorize("hasRole('ADMIN')")).

2. Patrón Singleton (Conexión BD)

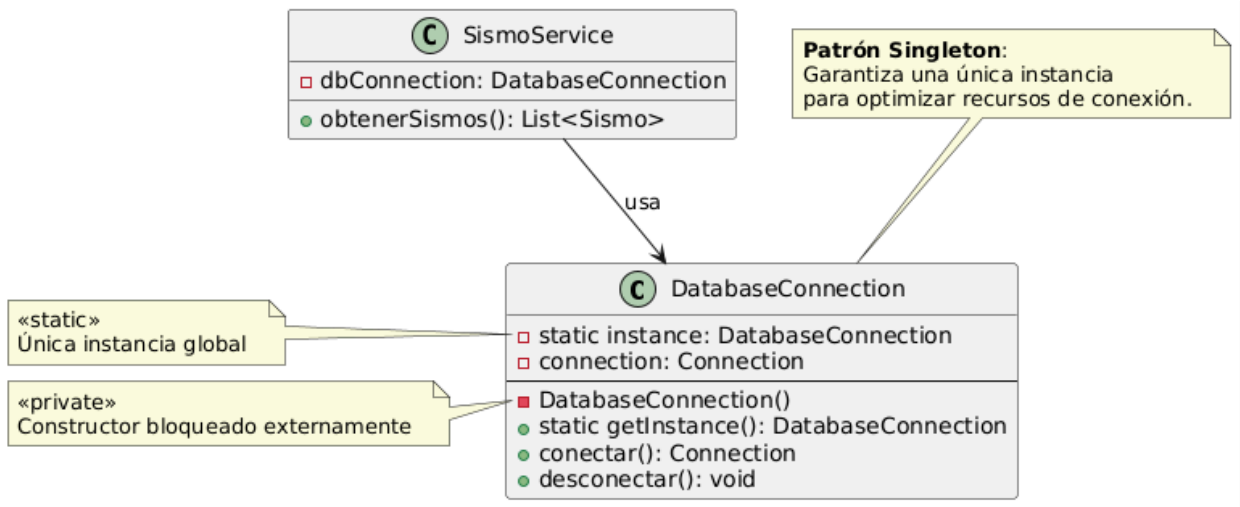
- Justificación: Garantiza una única instancia de conexión a MySQL, optimizando recursos.
- Implementación:

```java
public class DatabaseConnection {
    private static DatabaseConnection instance;
    private DatabaseConnection() {}
    public static synchronized DatabaseConnection getInstance() {
        if (instance == null) instance = new DatabaseConnection();
        return instance;
    }
}
```


public class DatabaseConnection {
	private static DatabaseConnection instance;
	private DatabaseConnection() {}
	public static synchronized DatabaseConnection getInstance() {
		if (instance == null) instance = new DatabaseConnection();
		return instance;
	}
}

Diagrama UML del patrón:

Singleton UML



3. Modelo de Implementación

Tecnologías:

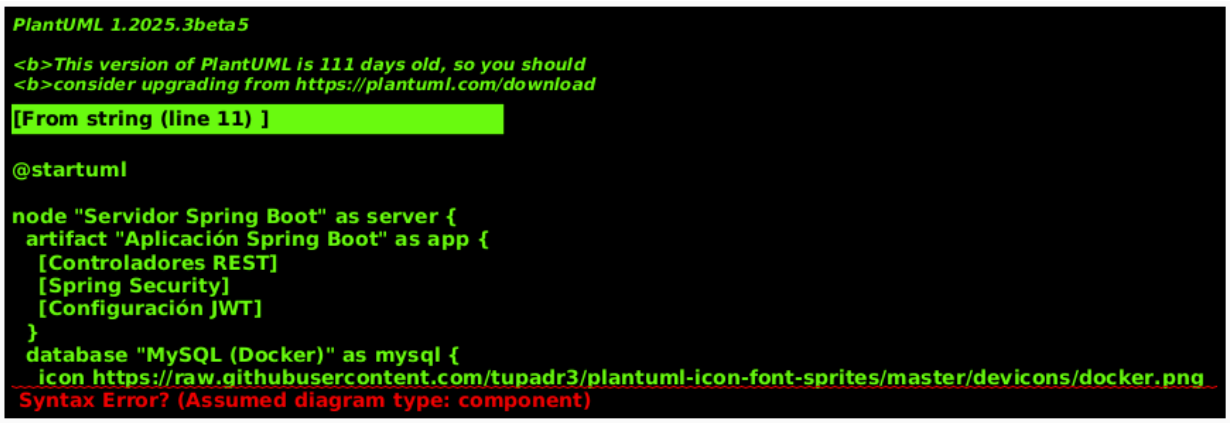
- Backend: Spring Boot 3.1, JWT, Spring Data JPA.
- Frontend: Leaflet.js, Thymeleaf.
- Bases de datos: MySQL (usuarios), Neo4j (sismos).

Arquitectura: Microservicios con API Gateway.

Seguridad: HTTPS, BCrypt, OAuth2.

4. Diagrama de Despliegue

Diagrama de Despliegue



Componentes:

- Servidor Spring Boot (4GB RAM, Intel i5).
- Clientes web (navegadores).
- Bases de datos (MySQL, Neo4j en Docker).

Protocolos:

- REST/HTTPS, WebSockets (futuro).