GenoSentinel

Breaze Labs requiere el desarrollo de un sistema modular, seguro y escalable para la gestión y consulta de la información genómica y clínica de sus pacientes oncológicos. El objetivo principal es centralizar la información de **variantes genéticas somáticas** (librerías de mutaciones tumorales) y vincularla directamente con el historial clínico, sustituyendo los archivos de datos dispersos por una estructura relacional gestionada a través de APIs de microservicios.

El sistema debe implementarse siguiendo una arquitectura de **tres microservicios** desacoplados, diseñada para operar en un entorno orquestado con **Kubernetes**.

Tecnologías y Requerimientos Generales

- **Base de Datos:** Se utilizará **MySQL** para la persistencia de los datos de la clinica y mongodb para la persistencia de los datos de el microservicio de genómica.
- **Microservicio 1 (Clínica):** Nest
- **Microservicio 2 (Genómica):** Django
- **Microservicio 3 (Autenticación / Simulación de Gateway): Spring Boot**
- **Persistencia de Datos:** Es obligatorio el uso de **ORMs** (Object-Relational Mappers) en los tres microservicios para interactuar con la BD, evitando consultas SQL directas en el código de aplicación.
- **Documentación:** La documentación de la API es crucial para la integración futura. Se exige la implementación y el uso de **Swagger** para documentar todos los endpoints en los Microservicios de Clínica y Genómica.
- **Seguridad y Gateway Simulado:** Todas las peticiones de consulta o manipulación de datos deben pasar a través del Microservicio de Autenticación, que **simula un API Gateway**, validando credenciales y redirigiendo tráfico a los servicios internos.
- **Diagrama de Arquitectura:** Se requiere un diagrama que muestre el flujo de datos desde el cliente al microservicio de Autenticación (simulación de Gateway), hacia los microservicios internos y la base de datos, incluyendo comunicación interna y despliegue en Kubernetes.

Microservicios y Funcionalidad

Microservicio 1: Clínica (Nest**)**

- Gestión de Pacientes: Crear, actualizar, consultar y desactivar pacientes.
- Gestión de Tipos de Tumor: Mantener el catálogo de tumores oncológicos con sus atributos relevantes.
- Gestión de Historias Clínicas: Registrar diagnósticos, tratamientos y evolución de los pacientes.
- Transformación y exposición de datos mediante DTOs (Data Transfer Objects).
- Documentación completa en Swagger/OpenAPI.

Microservicio 2: Genómica (Django)

- Gestión de Genes: Catalogar genes de interés oncológico con sus funciones y descripciones.
- Gestión de Variantes Genéticas: Registrar mutaciones específicas, incluyendo ubicación, referencia y efecto.
- Gestión de Reportes de Pacientes: Asociar variantes genéticas a pacientes específicos, obteniendo información clínica vía el Microservicio de Clínica.
- Transformación y validación de datos mediante DTOs.
- Documentación completa en Swagger/OpenAPI.

Microservicio 3: Autenticación / Simulación de Gateway (Spring Boot)

- Autenticación de usuarios mediante credenciales y emisión de tokens JWT.
- Autorización: **El sistema manejará un único rol de usuario**, que tendrá acceso a todas las funcionalidades de los microservicios internos.
- Enrutamiento de peticiones hacia los microservicios de Clínica y Genómica, **simulando un API Gateway**.
- Monitoreo básico de estado (health check y status endpoints).

Orquestación y Despliegue en Kubernetes

- Cada microservicio debe desplegarse en **pods separados**, escalables de forma independiente.
- Los servicios internos de Clínica y Genómica deben ser accesibles únicamente a través del microservicio de Autenticación (simulación de Gateway).
- Configuración de **Secrets** y **ConfigMaps** para credenciales de BD y variables de entorno.
- Persistencia mediante **volúmenes persistentes** para la base de datos MySQL.
-

Entidades del Dominio Clínico (Microservicio de Clínica - Nest)

Entidad	Propósito	Campos Requeridos
Paciente (Patient)	Datos personales y de identificación del paciente.	id (PK, UUID), firstName, lastName, birthDate, gender, status (e.g., 'Activo', 'Seguimiento').
Tipo de Tumor (TumorType)	Catálogo de las patologías oncológicas.	id (PK), name (e.g., 'Cáncer de Mama'), systemAffected (e.g., 'Glándulas').
Historia Clínica (ClinicalRecord)	Registra los eventos de diagnóstico de cáncer. Relación con Paciente y Tipo de Tumor.	id (PK, UUID), patientId (FK), tumorTypeId (FK), diagnosisDate, stage (e.g., 'IIA', 'IV'), treatmentProtocol.

Entidades del Dominio Genómico (Microservicio de Django)

Entidad	Propósito	Campos Requeridos
Gen de Interés (Gene)	Catálogo de genes relevantes en oncología.	id (PK), symbol (e.g., 'BRCA1'), fullName, functionSummary.
Variante Genética (GeneticVariant)	Registro de una mutación específica. Relación con Gen de Interés.	id (PK, UUID), geneId (FK), chromosome (e.g., 'chr17'), position, referenceBase (e.g., 'A'), alternateBase (e.g., 'G'), impact (e.g., 'Missense', 'Frameshift').
Reporte de Variantes del Paciente (PatientVariantReport)	Librería de mutaciones encontradas en un paciente. Relación con Paciente y Variante Genética.	id (PK, UUID), patientId (FK, consultable en Microservicio Clínica), variantId (FK), detectionDate, alleleFrequency (VAF, decimal).

Consideraciones Finales

- Todos los flujos de datos entre capas de aplicación y APIs deben usar **DTOs** para garantizar validación, seguridad y desacoplamiento.
- La arquitectura debe ser completamente modular, segura y escalable, siguiendo buenas prácticas de microservicios y despliegue en Kubernetes.
- Se espera un **diagrama de arquitectura** que represente el flujo desde el cliente, pasando por el microservicio de Autenticación (simulación de Gateway), hacia los microservicios, con comunicación interna y conexión con las bases de datos.

Sustentación y Entregables

1. **Presentación en Video:**
   - Cada equipo deberá presentar su proyecto mediante **diapositivas**, explicando la arquitectura del sistema, el flujo de datos, la orquestación en Kubernetes y la interacción entre microservicios.
   - Se deberá incluir el **diagrama de arquitectura (Modelo 4×1)**, mostrando cómo se simula el Gateway y cómo se comunican los microservicios internos.
2. **Endpoints Funcionales :**
   - Todos los **endpoints de los microservicios** (Clínica y Genómica) deben estar implementados y listos para ser probados en clase a través de Swagger o alguna herramienta de testing (Postman, Insomnia, etc.).
   - Se espera que el Microservicio de Autenticación (simulación de Gateway) esté operativo, permitiendo la autenticación y autorización con el **único rol de usuario disponible**.
3. **Requisitos de Entrega:**
   - Código fuente de los tres microservicios en repositorio Git.
   - Swagger actualizado para cada microservicio.
   - Diagrama de arquitectura (PDF o imagen).
   - Frontend Angular funcionando, vinculado correctamente con los microservicios.