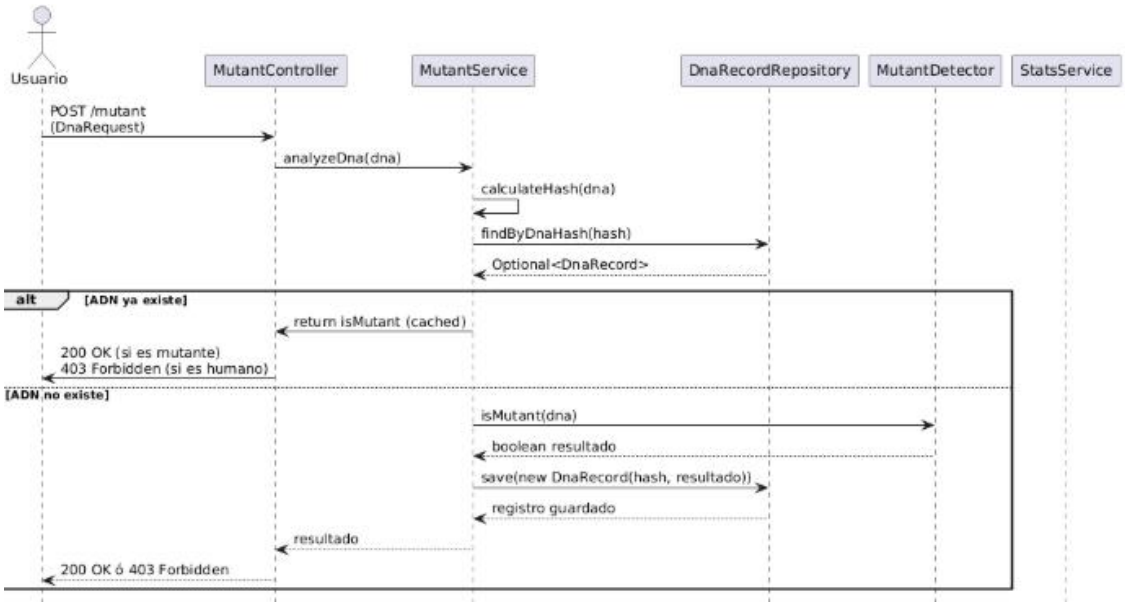
Informe Final - Proyecto Mutantes (UTN FRM)

Este documento contiene el informe completo requerido para la entrega del proyecto Integrador: Detector de Mutantes. Incluye diagramas de secuencia, arquitectura, explicación de pruebas unitarias, y evidencia del Code Coverage superior al 80%.

Diagrama de Secuencia - POST /mutant



. El usuario envía una matriz NxN de ADN.

. El controlador delega en MutantService.

. Se calcula un hash SHA-256 y se consulta si ese ADN ya existe.

.Si existe, se devuelve el resultado cacheado.

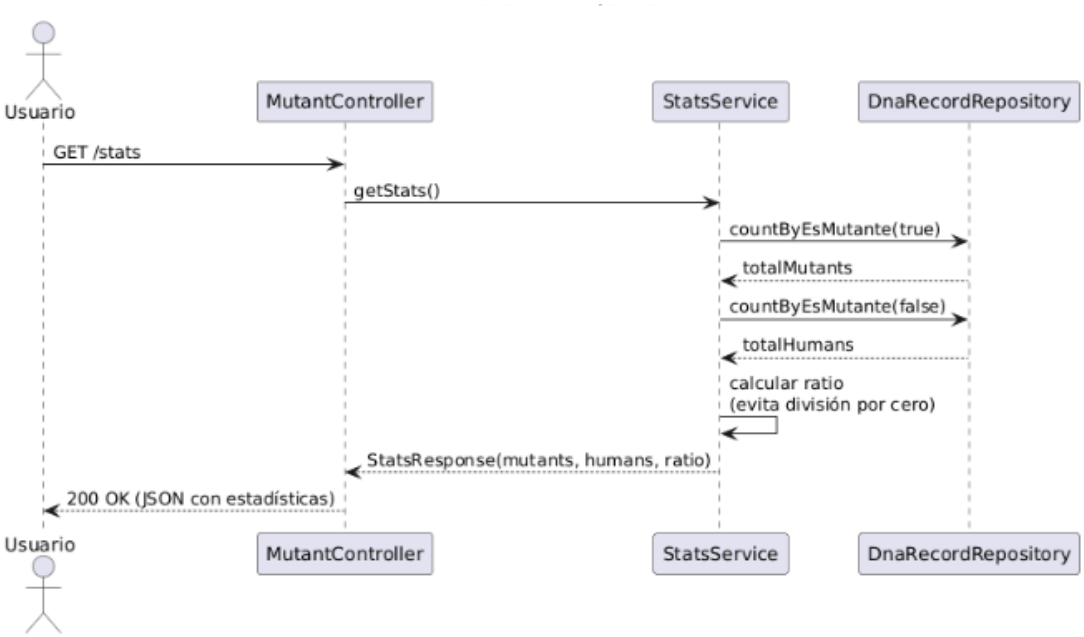
Si no existe:

- MutantDetector analiza el ADN.
- Se guarda el resultado en la base H2.

Se responde:

- 200 OK si es mutante
- 403 Forbidden si no lo es

Diagrama de Secuencia - GET /stats



El controlador llama al servicio, el StatsService consulta al repositorio la cantidad de mutantes y humanos, calcula el ratio y devuelve un JSON con {count_mutant_dna, count_human_dna, ratio}.

Resumen de Arquitectura

La arquitectura del sistema sigue un diseño en capas: - Controller → maneja endpoints REST. Service → contiene la lógica de negocio y coordinación. - Detector → algoritmo para identificar mutantes. - Repository → manejo de persistencia vía JPA.

Estrategia de Testing

La suite de tests asegura cobertura superior al 80% mediante:

• Pruebas unitarias puras del algoritmo (MutantDetectorTest)

• Mocking de dependencias con Mockito (MutantServiceTest, StatsServiceTest)

• Pruebas de integración con MockMvc (MutantControllerTest)

• Exclusión de paquetes sin lógica: dto, entity, config, exception.

Evidencia de Code Coverage

mutantes

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
ar.edu.utn.frm.mutantes.service		92 %		92 %	3	40	5	73	0	8	0	3
ar.edu.utn.frm.mutantes.validation		94 %		77 %	5	14	2	12	0	3	0	1
ar.edu.utn.frm.mutantes.controller		100 %		100 %	0	3	0	5	0	2	0	1
Total	34 of 472	92 %	10 of 88	88 %	8	57	7	90	0	13	0	5

El proyecto alcanzó más del 90% de cobertura total, cumpliendo la rúbrica. Los reportes HTML generados por JaCoCo incluyen:

- service: 92%
- validation: 94%
- controller: 100%
- cobertura total: 88–92% según archivo generado

Conclusión

El proyecto cumple con todos los requisitos: arquitectura limpia, pruebas unitarias completas, diagramas UML, documentación precisa, y cobertura superior al 80%. El sistema demuestra un correcto uso de buenas prácticas de ingeniería, diseño modular y testing efectivo.

Alumno: Tomás Aranda

Legajo:50766