Documentación Técnica del Proyecto de Mutantes

Descripción del Proyecto

Este proyecto consiste en una API REST que permite detectar si un humano es mutante basándose en una secuencia de ADN. La API almacena los registros de ADN verificados en una base de datos H2 y proporciona estadísticas sobre los resultados de las verificaciones.

Arquitectura del Sistema

La arquitectura del sistema se basa en el patrón de diseño Modelo-Vista-Controlador (MVC). La API está construida con Spring Boot y utiliza JPA para la interacción con la base de datos H2.

- **Modelo**: Define la estructura de los datos (clase `DnaRecord`).
- **Vista**: La API REST que expone los endpoints para interactuar con los datos.
- **Controlador**: Maneja las solicitudes HTTP y realiza las operaciones necesarias utilizando los servicios.

Diagrama de Arquitectura

Incluye un diagrama que represente la arquitectura de tu sistema, mostrando cómo interactúan los componentes.

Especificaciones de la API

- **Endpoint**: `/mutant`
    - **Método**: `POST`
    - **Descripción**: Verifica si el ADN proporcionado corresponde a un mutante.
    - **Parámetros**:
        - `dna`: Array de cadenas que representa las secuencias de ADN.

Ejemplo de Solicitud:
json
Copiar código

```
{
  "dna": ["ATGCGA", "CAGTGC", "TTATGT", "AGAAGG", "CCCCTA",
"TCACTG"]
}
```

    -
    - **Respuesta**:
        - `200 OK` si es un mutante.
        - `403 Forbidden` si no es un mutante.
- **Endpoint**: `/mutant/stats`
    - **Método**: `GET`
    - **Descripción**: Devuelve estadísticas sobre las verificaciones de ADN.

Respuesta:
json
Copiar código
```
{
  "count_mutant_dna": 40,
  "count_human_dna": 100,
  "ratio": 0.4
}
```

    -

Configuración de la Base de Datos

La API utiliza una base de datos H2 en memoria para almacenar los registros de ADN verificados. La configuración en `application.properties` incluye:

properties
Copiar código
```
spring.datasource.url=jdbc:h2:mem:testdb;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=augusto
spring.datasource.password=
spring.h2.console.enabled=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Instrucciones de Uso

1. **Configuración del Proyecto**:
    - Asegúrate de tener instalado Java (JDK 17 o superior) y Maven.
    - Clona el repositorio desde GitHub.
    - Navega a la carpeta del proyecto y ejecuta `./mvnw spring-boot:run` para iniciar la aplicación.
2. **Acceder a la API**:

    - La API estará disponible en `http://localhost:8080/mutant`.
3. **Realizar Solicitudes**:
    - Puedes usar herramientas como Postman o cURL para enviar solicitudes a la API.

Plan de Pruebas

- Se han implementado pruebas automáticas que cubren las principales funcionalidades de la API.
- La cobertura de código se ha verificado y supera el 80%. Para comprobar esto, se puede utilizar herramientas como Jacoco.

Conclusiones y Aprendizajes

- **Reflexiones sobre el Proceso**: El desarrollo de esta API ha sido un proceso educativo que permitió profundizar en el uso de Spring Boot y la interacción con bases de datos.
- **Desafíos**: Uno de los principales desafíos fue manejar correctamente la lógica de detección de mutantes y garantizar que los registros se guardaran correctamente en la base de datos.
- **Mejoras Potenciales**: Considerar la implementación de autenticación para proteger los endpoints y aumentar la escalabilidad de la API.

Análisis de Rendimiento

- **Manejo de Tráfico**: La API ha sido diseñada para manejar fluctuaciones en el tráfico, con configuraciones para el servidor Tomcat que permiten un mayor número de hilos.
- **Estrategias de Escalabilidad**: Se ha considerado el uso de servicios en la nube (como Google App Engine) para implementar escalabilidad automática en función de la carga.