**EXAMEN MERCADO LIBRE – FELIPE ANDRES JAMIY GIRON**

Este documento es el encargado de evidenciar y explicar el uso del api rest desarrollada para cumplir con el reto de mercado libre, por lo cual se adjuntan imágenes de como se hizo la aplicación y el uso de esta.

* **Desarrollo Api Rest**

La aplicación se realizó con Spring boot y uso del lenguaje Java, se uso como gestor de dependencias Maven y se creo una bd en memoria h2 para realizar la persistencia y consulta de los candidatos, además esta expuesta públicamente por medio de amazon aws.

La siguiente es una evidencia de la estructura que se utilizo en el desarrollo del proyecto.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Se utilizaron interfaces para definir el comportamiento de la app, además de un controlador que es la puerta de entrada a la app y un service para desarrollar la lógica de negocio.

* **Lógica de negocio**

El propósito del api rest es determinar por medio del dna si el candidato es un mutante o un humano, para esto se tiene como regla que es mutante si el dna tiene mas de una secuencia con cuatro letras iguales consecutivas.

La aplicación esta en la capacidad de recibir una lista de String con las letas del dna, crear una matriz apartir de esta y validar las horizontales, verticales y diagonales aun que siempre validando que si ya cumple la condición de negocio hacer el return y no completar la iteración.

Con las horizontales no hay necesidad de recorrer una matriz ya que se busca en cada String si existe la secuencia que indica si es mutante, en el caso de hallar dos secuencias hace el retorno de la respuesta http y un true indicando que se encontró un mutante, en caso contrario sigue la ejecución del código (no es necesario recorrer toda la lista de String el verifica en cada iteración si se cumple la condición para no obligarla a recorrer completa)..

Texto

Descripción generada automáticamente con confianza media

En caso de no encontrar si es mutante la aplicación realizara la búsqueda de este en las verticales, para esto se arma la matriz, luego convirtiendo cada columna en un string y verificar si en esta se encuentra la secuencia, en caso de encontrarla se hace el return afirmando que se encontró un mutante (no es necesario recorrer toda la matrix el verifica en cada iteración si se cumple la condición para no obligarla a recorrer completa).

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

En caso de no encontrar si es un mutante se hace las diagonales, pero la aplicación en cada iteración encuentra la diagonal la cual puede tener la cantidad de letras que pueden hacer una secuencia, obviando diagonales innecesarias como la de las esquinas, además haya en la misma iteración la diagonal inversa, y las diagonales contrarias.

A continuación, una ilustración de lo que seria una iteración de las diagonales.

Imagen que contiene juego, texto, dibujo

Descripción generada automáticamente

En caso de no encontrar en la iteración las coincidencias haya las mismas diagonales, pero una posición menos hasta llegar a las diagonales principales.

Gráfico, Forma

Descripción generada automáticamente con confianza media

Por último, una ilustración de la ultima iteración de las diagonales (tener en cuenta que la aplicación tiene en cuenta si se hayo una sentencia anteriormente y apenas completa dos se retorna con los valores esperados).

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

Otro servicio el cual es requerido es la consulta de unos datos estadísticos, cuando la aplicación haya un mutante lo registra en una tabla la cual tiene un tipo, en este se almacena por medio de un enum si es mutante o humano y este nos ayuda al tema estadístico del servicio /stats.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **Pruebas Unitarias**

Además, se pide las pruebas unitarias con una cobertura del 80% por esta razón se creo una clase para hacer las pruebas unitarias y se utiliza jacoco para hacer la cobertura de estas.

Para esto se adjuntan imágenes las cuales evidencias la cobertura de las pruebas.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **Uso Api rest de manera local**

Para hacer las pruebas de manera local se pueden seguir estos pasos.

Interfaz de usuario gráfica, Aplicación, Correo electrónico

Descripción generada automáticamente

Desde postman o SoapUI se hacen las peticiones de manera local de la siguiente forma.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **Uso Api rest desde Amazon aws**

Para el uso de los servicios /mutant y /stats se debe acceder por la ip publica 3.83.31.251, el puerto 8080 y el path correspondiente.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente