Taller de programación avanzada: Examen Mercadolibre

Integrantes:

- Andrés Lorenzo Gil
- Diego Bustos
- Facundo Erbin
- Jorge Condori
- Juan Troncoso

Solución mediante expresiones regulares

- ¿Qué es una expresión regular?

Una expresión regular es una secuencia de caracteres que conforma un patrón de búsqueda. Se utilizan principalmente para la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones.

Las expresiones regulares son patrones utilizados para encontrar una determinada combinación de caracteres dentro de una cadena de texto.

Implementación de expresión regular

```
public void search(String sequence){
        if(Pattern.compile("(C|T|G|A)\\1{3,3}",Pattern.CASE_INSENSITIVE).matcher(sequence).find()) {
            mutantCount++;
            System.out.println("Encontrado " + sequence);
         }
}
```

Pagina para crear y aprender a usar expresiones regulares:

https://regexr.com

Búsqueda Horizontal

```
public boolean searchHorizontal(String[] sequences){
        for(String e : sequences){
            search(e);
            if(mutantCount >= 2) return true;
        }
        return false;
    }
```

Búsqueda Vertical

```
public boolean searchVertical(String[] sequences){
        String sequence = "";
            for(int i = 0 ; i < sequences.length ; i ++){
                for(int j = 0 ; j < sequences.length; j ++){
                    sequence += sequences[j].charAt(i);
                }
                search(sequence);
                if(mutantCount >= 2) return true;
            sequence = "";
        }
        return false;
    }
```

Búsqueda Oblicua / Contra oblicua

```
public boolean searchOblique (String[] sequences){
      String sequence = "";
      int height = sequences.length;
      int width = sequences[0].length();
      for (int diagonal = 1 - width; diagonal <= height - 1; diagonal += 1) {
          for (int vertical = Math.max(0, diagonal), horizontal = -Math.min(0, diagonal);
               vertical < height && horizontal < width; vertical += 1, horizontal += 1) {
              sequence += sequences[vertical].charAt(horizontal) ;
          }
          search(sequence);
          if(mutantCount >= 2) return true;
          sequence = "";
      }
      return false;
}
```

Validación utilizada

```
public static boolean isValidArray(String[] dna) {
    int size = dna.length;
    if (size < 4) {
        // La matriz como mínimo debe ser de 4X4
        return false;
    }

    for (String line : dna) {
        if (line.length() != size) {
            return false;
        }
    }

    return true;
}
```

Implementación en Swagger

- **¿Qué es Swagger?**

Swagger es un conjunto de herramientas de software de código abierto para diseñar, construir, documentar, y utilizar servicios web RESTful.

- **¿Por qué usar Swagger?**

Por que nos permite enganchar el sistema de documentación con el código del servidor para que esté sincronizado a cada cambio. Lo que resulta en una documentación muy fácil de entender.

https://desafio-meli-tpa.herokuapp.com/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config

Proyecto en swagger

Desafio Mercado Libre - Mutantes 1.0 OAS3

/v3/api-docs

Proyecto realizado para la cátedra Taller de Programación Avanzada - UTN FRM

Servers

https://desafio-meli-tpa.herokuapp.com - Generated server url

mutant-controller

GET	/api/v1/mutant	Endpoint que retorna un array con todos los ADN analizados por el sistema.
POST	/api/v1/mutant	Endpoint que recibe ADN y responde si es ADN mutante. En caso de ser ADN mutante retorna estado 200 y si no lo es retorna 403
GET	/api/v1/mutant/{size}	Endpoint para generar matrices de ADN aleatorias. Requiere que se le pase el tamaño de la matriz por parámetro
GET	/api/v1/mutant/stats	Endpoint para consultar las estadísticas del sistema.

Comparativa código y swagger GET /api/v1/mutant

```
 @Operation(
            summary = "Endpoint que retorna un array con todos los
ADN analizados por el sistema."
    )
    @ApiResponses(
            value = {
                    @ApiResponse(
                            responseCode = "200",
                            description = "Retorna todos los ADN
analizados por el sistema, mutantes como no mutantes."
                    )
            }
    )
```

Comparativa código y swagger GET /api/v1/mutant

GET `/api/v1/mutant` Endpoint que retorna un array con todos los ADN analizados por el sistema.

Parameters | Try it out

No parameters

Responses

Code	Description	Links
200	Retorna todos los ADN analizados por el sistema, mutantes como no mutantes. Media type */* Controls Accept header. Example Value | Schema {}	*No links*

Controlador GET /api/v1/mutant

```
    @GetMapping("")
    public ResponseEntity<?> getAll () {
        try {
            List<Mutant> allMutants = mutantService.getAll();
            return
ResponseEntity.status(HttpStatus.OK).body(allMutants);
        } catch (Exception e) {
            return
ResponseEntity.status(HttpStatus.FORBIDDEN).body(("{\"error\":
\"" + e.getMessage() + "\"}"));
        }
    }
```

Comparativa código y swagger POST /api/v1/mutant

```
@Operation(
            summary = "Endpoint que recibe ADN y responde si es ADN mutante. En caso de ser ADN mutante retorna estado 200 y si no lo es retorna 403",
            description = "Se requiere una matriz de NxN con las letras (A, C, G, T)"
    )
    @ApiResponses(
            value = {
                @ApiResponse(
                        responseCode = "200",
                        description = "Retorna 200 en caso de que el ADN enviado sea mutante. Es decir tiene mínimo 2 cadenas de 4 letras iguales sucesivas en cualquier dirección. No devuelve contenido"
                ), @ApiResponse(
                        responseCode = "403",
                        description = "Retorna 403 en caso de que el ADN enviado NO sea mutante. No devuelve contenido"
                )
            }
    )
```

Comparativa código y swagger POST /api/v1/mutant

POST `/api/v1/mutant` Endpoint que recibe ADN y responde si es ADN mutante. En caso de ser ADN mutante retorna estado 200 y si no lo es retorna 403

Se requiere una matriz de NxN con las letras (A, C, G, T)

Parameters | Try it out

No parameters

Request body required | application/json

Example Value | Schema

```
{
  "id": 0,
  "dna": [
    "string"
  ],
  "mutant": true
}
```

Comparativa código y swagger POST /api/v1/mutant

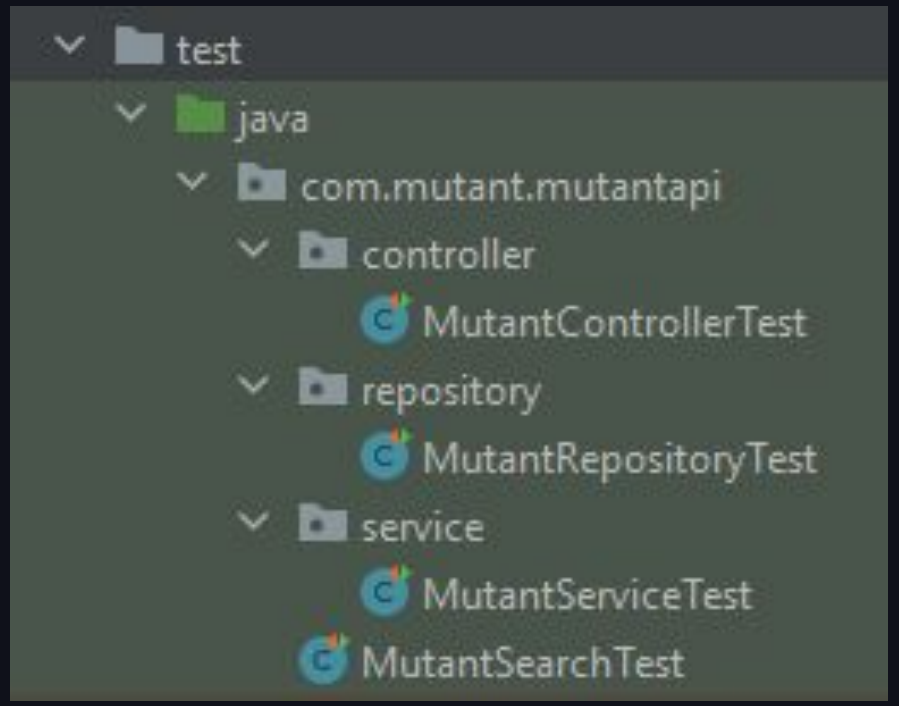
Responses

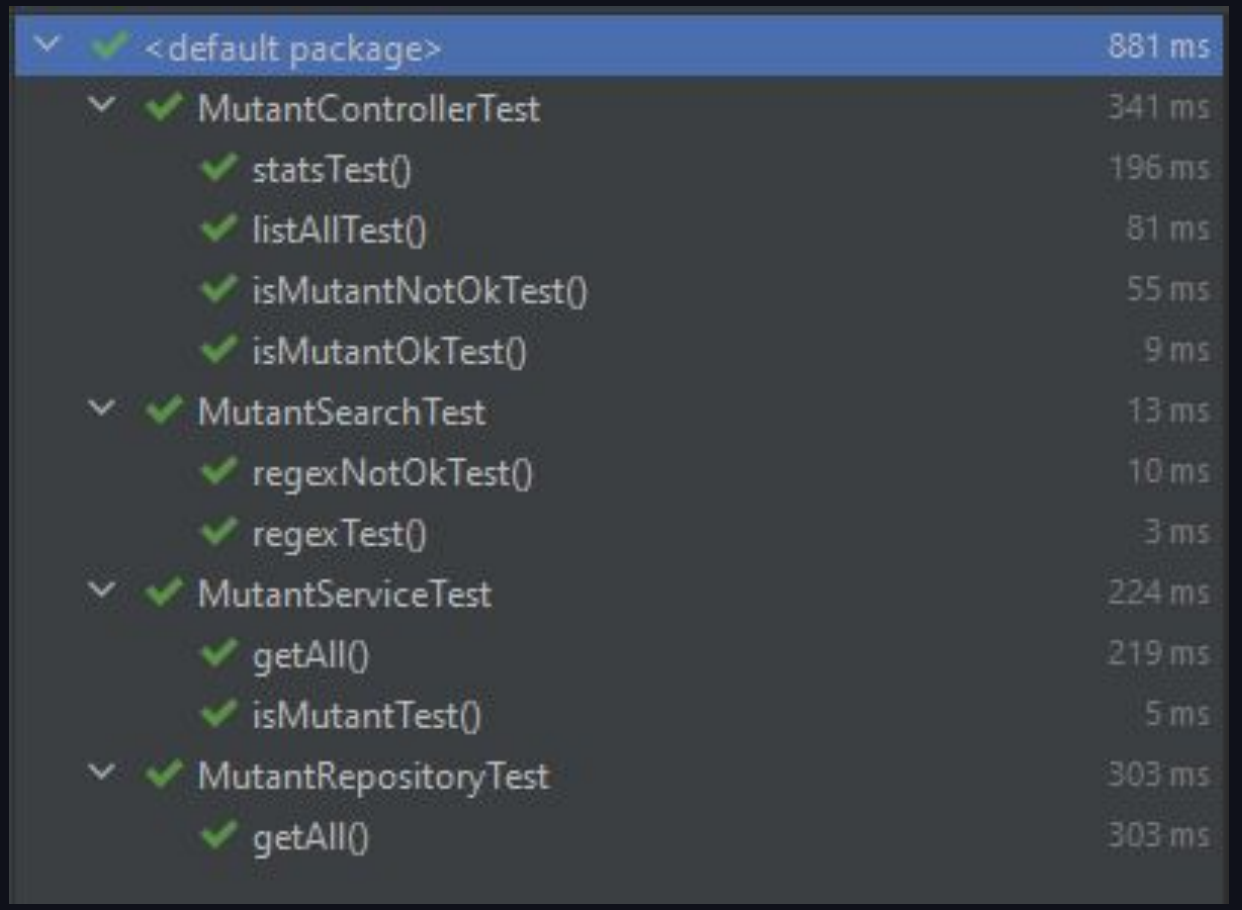
Code	Description	Links
200	Retorna 200 en caso de que el ADN enviado sea mutante. Es decir tiene mínimo 2 cadenas de 4 letras iguales sucesivas en cualquier dirección. No devuelve contenido Media type */* Controls `Accept` header. **Example Value** | Schema `string`	*No links*
403	Retorna 403 en caso de que el ADN enviado NO sea mutante. No devuelve contenido Media type */* **Example Value** | Schema `string`	*No links*

Controlador POST /api/v1/mutant

```
    @PostMapping
    public ResponseEntity isMutant(@RequestBody Mutant
mutant){
        return mutantService.isMutant(mutant.getDna()) ?
ResponseEntity.ok().build() :
ResponseEntity.status(HttpStatus.FORBIDDEN).build();
    }
```

Tests del proyecto





Cobertura de los tests

Coverage: com.mutant.mutantapi in MutantAPI

88% classes, 72% lines covered in package 'com.mutant.mutantapi'

Element	Class, %	Method, %	Line, %
controller	100% (1/1)	75% (3/4)	53% (14/26)
dto	100% (2/2)	85% (6/7)	90% (10/11)
model	100% (2/2)	100% (8/8)	100% (8/8)
mutantUtils	50% (1/2)	77% (7/9)	75% (46/61)
repository	100% (0/0)	100% (0/0)	100% (0/0)
services	100% (1/1)	50% (2/4)	71% (10/14)
MutantApiApplication	100% (1/1)	0% (0/1)	50% (1/2)