

Examen Mutantes Mercado Libre

Documentación

García Matías Daniel

Contenido

Tecnologías involucradas.

Instrucciones para ejecutar la aplicación.

Pruebas de Stress.

Code Coverage.

Lógica aplicada para resolver si un humano es mutante o no.

Anotaciones.

Tecnologías involucradas:

- NodeJS
- Postgres
- Sequelize
- NPM
- Mocha
- Istanbul
- Heroku
- Docker
- JMeter

Instrucciones para ejecutar la aplicación.

Cómo correr la aplicación localmente:

Requisito: Tener instalado Docker.

```
$ sh ./run-app.sh
```

En caso de no tener Docker instalado o ya sea para desarrollar:

Requisito: Tener instalado npm y Node (versión 8 ó posterior).

```
$ sh ./run-db.sh
```

```
$ npm install
```

```
$ npm run local
```

El servidor estará corriendo en el puerto 8080.

Cómo correr los tests:

```
$ sh ./run-db.sh
```

```
$ npm install
```

```
$ npm test
```

Cómo usar la aplicación en la nube:

La aplicación está hosteada en <https://mutantes-meli.herokuapp.com/>

Endpoint para verificar si un humano es mutante:

<https://mutantes-meli.herokuapp.com/api/xmen/mutant>

Ejemplo:

Method: POST.

Header: {Content-Type:application/json}

*Body: {"dna":[
 "TTATTC",
 "CTGTAC",
 "TTGTGT",
 "CGATGG",
 "CCCTTA",
 "ACACTG"]}]}*

Endpoint para obtener las estadísticas: <https://mutantes-meli.herokuapp.com/api/xmen/stats>

Pruebas de Stress

En caso de hacerle pruebas de stress a la aplicación en la nube, tener en cuenta que Heroku tiene un límite de 10.000 filas en la base de datos.

A la aplicación local se le hicieron pruebas de stress con JMeter, las cuales arrojaron los siguientes resultados:

Número de hilos: 10.

Periodo de subida: 1s

Bucles: 10.000

Tiempo transcurrido: 00:02:32

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
Petición HTTP	100000	15	1	148	6,30	0,00%	655,4/sec	146,56	170,88	229,0

En el repositorio está incluido el archivo [stress-test.jmx](#)

Code Coverage


Para el Code Coverage se utilizó [Istanbul](#)

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	94.77	94.87	84.38	94.77	
mutantes	100	100	100	100	
index.js	100	100	100	100	
mutantes/config	90	100	66.67	90	
app.config.js	100	100	100	100	
database.config.js	85.71	100	50	85.71	21
mutantes/controllers	89.47	100	75	89.47	
xMen.controller.js	89.47	100	75	89.47	34,37
mutantes/core	96.59	93.94	100	96.59	
Stat.core.js	100	100	100	100	
xMen.core.js	95.95	93.1	100	95.95	113,116,117
mutantes/models	100	100	100	100	
xMen.model.js	100	100	100	100	
mutantes/routes	100	100	100	100	
index.route.js	100	100	100	100	
xMen.route.js	100	100	100	100	
mutantes/services	77.78	100	66.67	77.78	
xMen.service.js	77.78	100	66.67	77.78	10,17

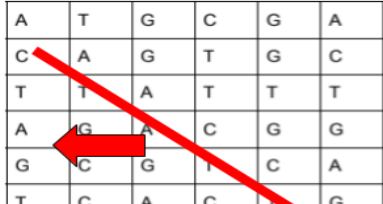
Lógica aplicada para resolver si un humano es mutante o no.

El algoritmo utilizado contiene los siguientes pasos en el siguiente orden:

- Chequea los strings de forma horizontal**, tal como vienen en el request, con una expresión regular que evalúa si en el valor existen al menos 4 letras iguales seguidas. Con un contador, si encuentra dos secuencias, retorna y corta el procedimiento posterior.
- Chequea los strings de forma vertical**. Para esto divide cada carácter de cada cadena y transforma todo a una matriz. Luego itera cada columna, y por cada columna itera la fila. Concatena los caracteres y una vez finalizada la iteración de la fila evalúa con la expresión regular si cumple o no. Con el mismo contador, si encuentra dos secuencias retorna y corta el procedimiento posterior.
- Chequea los strings de forma oblicua**. Este es el caso más complejo de entender e incluso de explicar. Utiliza también una matriz de caracteres, el contador para evaluar si existen dos dichas secuencias, y la expresión regular.
 - Itera desde la primer fila y la primer columna, hasta llegar hasta última fila y última columna, concatena esos caracteres y evalúa si cumple la condición.
Sigue recorriendo hacia la derecha evaluando lo mismo.
 - Itera desde la segunda fila y la primer columna, hasta llegar a la última fila y anteúltima columna, concatena esos caracteres y evalúa si cumple la condición.



A	T	G	C	G	A
C	A	G	T	G	C
T	T	A	T	T	T
A	G	A	C	G	G
G	C	G	T	C	A
T	C	A	C	T	A



A	T	G	C	G	A
C	A	G	T	G	C
T	T	A	T	T	T
A	G	A	C	G	G
G	C	G	T	C	A
T	C	A	C	T	A

- Sigue recorriendo hacia la izquierda evaluando lo mismo.
- c. Luego invierte la matriz y hace el procedimiento a y b.

Posterior a esto devuelve el response a la petición:

- Si es mutante
 - Status: 200 Ok.
 - Body: {"isMutant":true}
- Si no es mutante:
 - Status: 403 Forbidden.
 - Body:{"isMutant":false}

Asíncronamente persiste los datos en la base de datos.

Anotaciones

Por qué los endpoints no son simplemente /mutant y /stats como dice el ejercicio.
Simplemente por seguir una convención de Rest.