Mutant Detector

💡 Contexto

Este es el proyecto solicitado en un examen (detalle en el link), donde se pide crear unos servicios REST para detectar mutantes según DNA entregado y también poder ver estadísticas de los datos procesados.

✔ Entrega

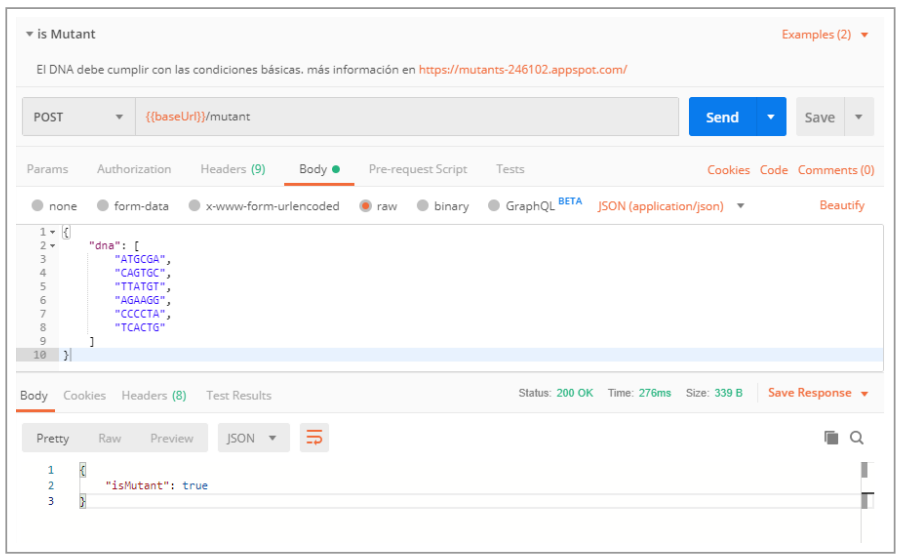
La aplicación entregada es una **App Engine** de Google Cloud Plataform (GCP) desarrollada en Nodejs, conectada también con **Cloud Firestore** de GCP como base de datos. El repositorio se encuentra en GitHub.

🖌 Probar la aplicación

La aplicación expone documentación además de los servicios (ver en: https://mutants-246102.appspot.com/). Existe un archivo de definición de la API en OpenAPI v3.0.2 con el cual se puede iniciar una colección o proyecto de API en algún cliente de APIs como por ejemplo Postman o Insomnia:

📄 **mutant.openAPI.json**

Ejemplo de la ejecución del post:



El resultado estadístico se puede probar directamente en browser: https://mutants-246102.appspot.com/stats

💻 Instalar

Pre-requisitos

Instalar previamente:

- nodejs con npm
- git
- *(Opcional)* SDK Google Cloud para comando `gcloud`

Instalción de la aplicación

Estando en una consola o terminal en el directorio donde deseas instalar la aplicación, ejecutas:

```
git clone https://github.com/glamatta/mutants.git
```

Una vez descargado el proyecto entras al directorio donde se instaló la aplicación:

```
cd mutants
```

Instalar las dependencias con:

```
npm install
```

Luego de la instalación ya puedes ejecutar el servidor:

```
npm run start
```

⚠ *debes asegurarte no tener otro servicio utilizando el puerto 80 que es el que usará la aplicación por defecto*.

Si quieres ejecutar los test debes utilizar:

```
npm run test
```

Y para ejecutar los test y además las estadísticas de *Code Coverage* (lo que actualizará también los HTMLs), ejecutar:

```
npm run coverage
```

Despliegue de la aplicación en GCP. Teniendo las configuraciones y accesos previamente configurados, se debe ejecutar:

```
gcloud app deploy
```

Análisis de DNA

Requerimiento inicial

La funcionalidad se implementa basada en la documentación entregada donde, en resumen, se debe analizar una matriz [`n` x `n`] de *strings* que sólo contienen las letras `A`, `C`, `T` y `G`. Se debe construir un servicio REST que reciba esta información y responda si esa matriz de DNA corresponde a un mutante o no, respondiendo al POST con un **http status = 200** si es mutante y **http status = 403** si no es. Es un mutante si se encuentran al menos 2 secuencias de 4 de las mismas letras consecutivas, secuencias que pueden estar horizontal, vertical u oblicuas, como lo muestran los ejemplos entregados:

No Mutante

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**	**G**

Mutante

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

Supuestos y Consideraciones

Para realizar la programación fue necesario considerar algunas definiciones que no se encontraban en la declaración inicial, la aplicación requiere:

- El objeto de entrada siempre será un `json` que contiene sólo una propiedad llamada `dna` (en minúsculas) y esta propiedad tiene como valor un arreglo de *strings*
- 4 ≤ `n` ≤ 1000
- Las *strings* NO contienen otros caracteres más que `A`, `C`, `T` y `G` en mayúsculas.
- todas las *strings* tendrán la misma cantidad de caracteres que la cantidad de *strings* para que los datos siempre sean una matriz cuadrada de [`n` x `n`]

⚠ De no cumplirse alguna de esas condiciones la respuesta de la aplicación será **400: Bad Request**

En cuanto al análisis de los datos se definen los siguientes supuestos:

- un *match* es una coincidencia de 4 letras consecutivas
- los *match* pueden cruzarse si están en distintas direcciones, por ejemplo horizontal y vertical, aunque se crucen se consideran 2 *match* distintos
- Los *match* no se pueden cruzar en la misma dirección, es decir, una fila como `aaaaaaaa` tiene 2 *match* solamente y una como `aaaaaa` tiene sólo 1 *match*

💡 Estas reglas son totalmente posibles de modificar, pero dado que no había especificación detallada al respecto, la aplicación actualmente considera estas definiciones.

Test y Code Coverage

Los test automatizados constan de 14 dna que pruebas casos de mutantes, no-mutantes y dna inválidos. Se encuentran en el directorio 'test/dna'.

Resultado de logs de test y code coverage se puede visualizar en el ejemplo [testandcoverage.log]

Adicionalmente la aplicación publica también la generación del *Code Coverage* en HTML en [https://mutants-246102.appspot.com/coverage/index.html]

All files

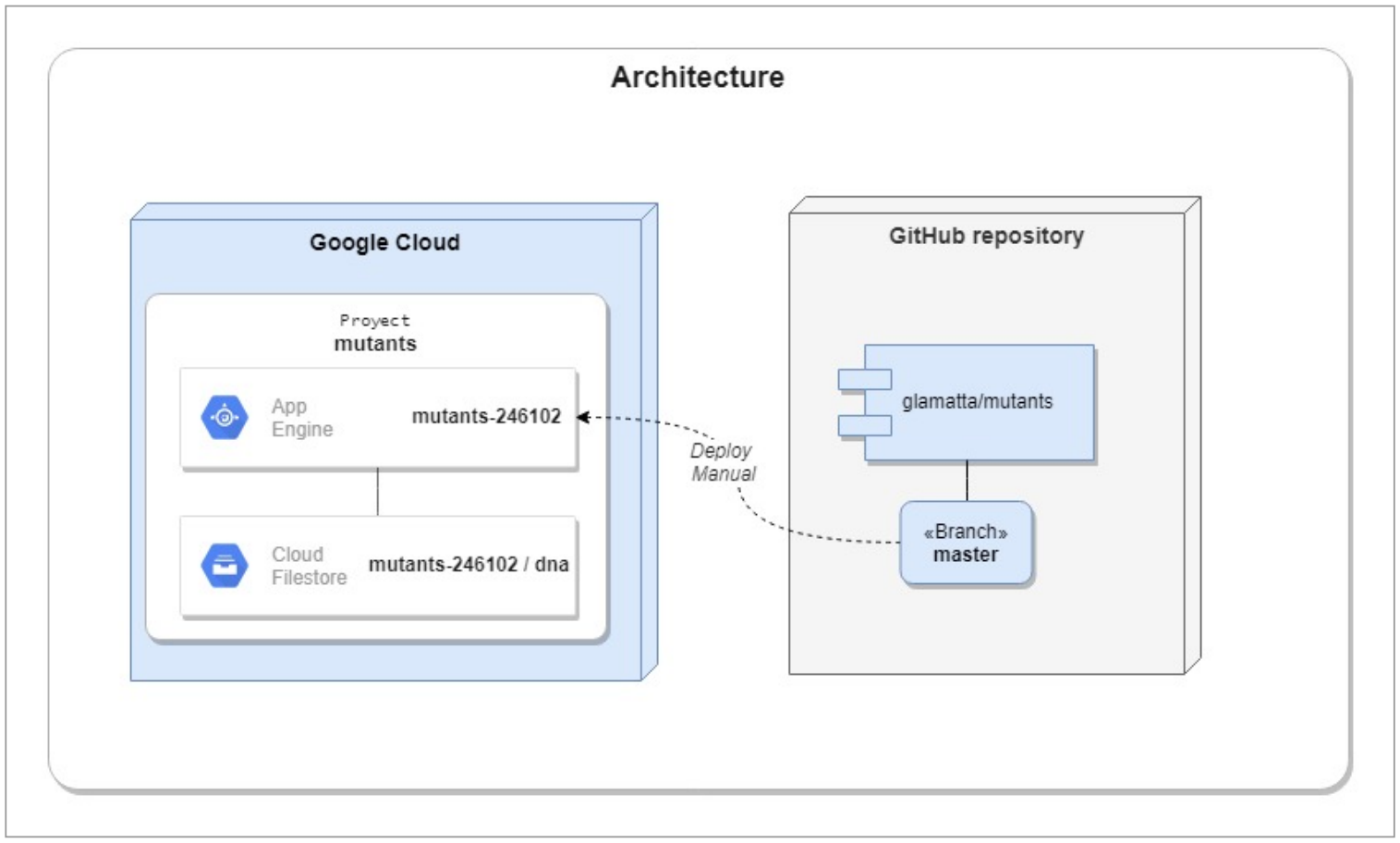
91.33% Statements 158/173 **84.51%** Branches 60/71 **70%** Functions 14/20 **93.29%** Lines 139/149

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
mutants	74%	37/50	61.11%	11/18	50%	6/12	78.26%	36/46
mutants/controllers	98.29%	115/117	92.45%	49/53	100%	6/6	100%	98/98
mutants/lib	100%	6/6	100%	0/0	100%	2/2	100%	5/5

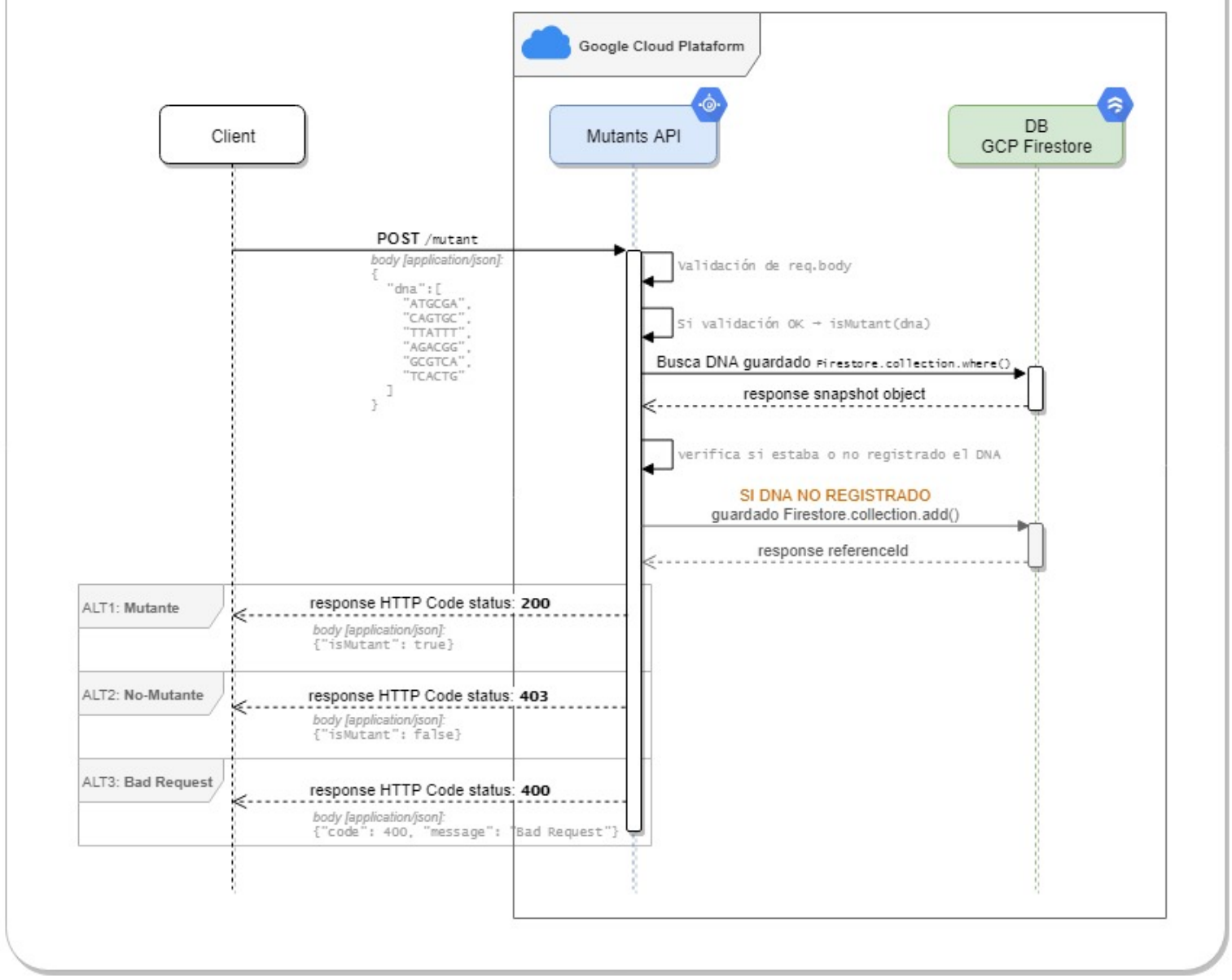
Diagramas

Arquitectura



Secuencia

Sequence *API Mutant*



Sequence *API Stats*