**DESARROLLO DE API REST**

para el levantamiento de los servicios se seleccionaron las siguientes tecnologías:

* **NodeJS/Typescript**: como lenguaje de lógica del servicio.
* **AWS**: como infraestructura en nube, nuestro proveedor de servicios, específicamente:
  + **AWS Lambda**: para almacenar y ejecutar la lógica del servicio.
  + **AWS APIGateway**: para exponer el servicio.
  + **AWS IAM**: para configurar permisos.
  + **AWS Dynamo**: para configurar la base de datos.
* **Jest**: como generador de pruebas unitarias y cobertura.
* **GitHub**: como controlador de versiones y repositorio.
* **Lint/Prettier**: como formateador de código.

**HISTORIAS DE USUARIO**

A continuación, haremos un levantamiento de historias épicas:

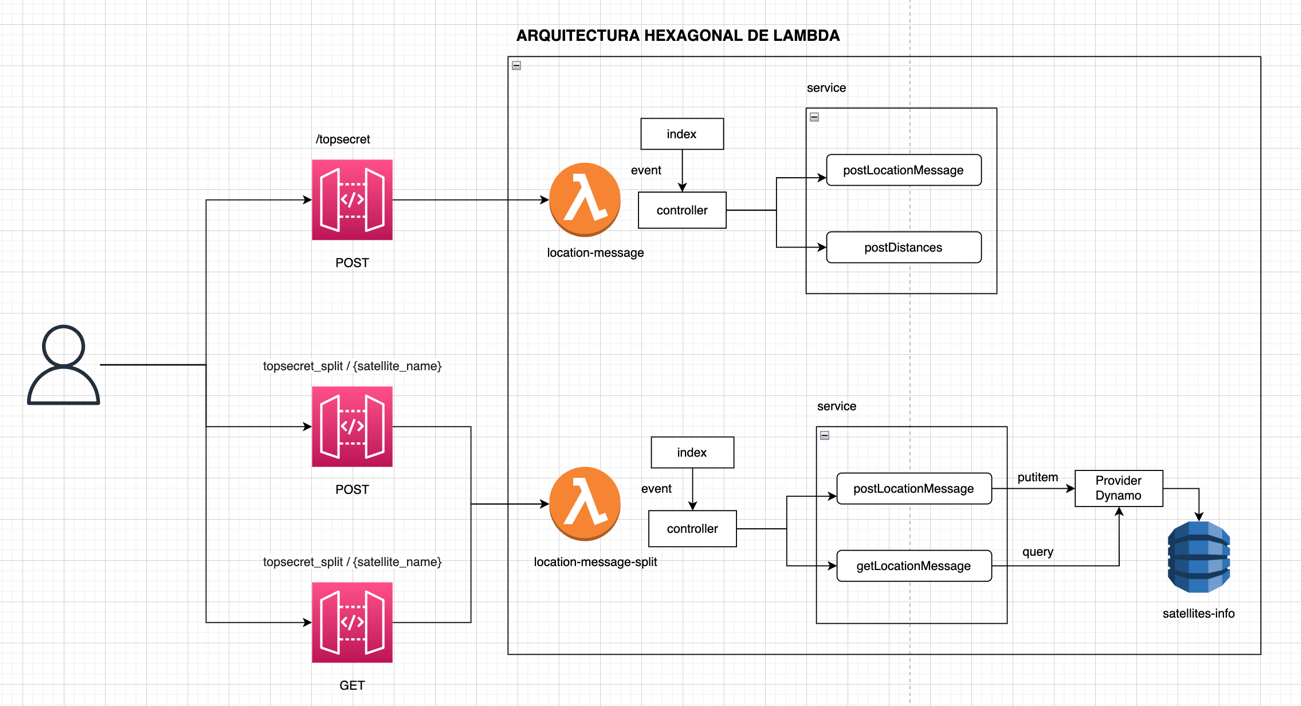
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API REST PARA PROCESAR ADN** | | | | | |
| **NUMERO HU** | **ROL** | **CARACTERISTICA** | **RAZON** | **#** | **CRITERIO DE ACEPTACION** |
| ML-1 | Yo como Cliente | Necesito consumir un servicio POST que me permita enviar un arreglo de satélites | Para detectar si la posición y el mensaje enviado | 1 | El arreglo debe ser un arreglo de tamaño 3 |
| 2 | El arreglo debe contener los satellites de nombre: \* kenobi, skywalker y sato |
| 3 | Si se logra armar el mensaje y encontrar la posición, se retorna la información retornando un código 200 |
| 4 | Si no se logra determinar la posición se debe responder con código 404 |
| 5 | Cobertura mayor a 80% |
| 6 | Código fuente en repositorio GitHub |
| 7 | Instrucciones de uso en archivo README |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API REST PARA VER ESTADISTICAS DE ADN** | | | | | |
| **NUMERO HU** | **ROL** | **CARACTERISTICA** | **RAZON** | **#** | **CRITERIO DE ACEPTACION** |
| ML-2 | Yo como Cliente | Necesito consumir 1 servicio POST que me permitan enviar por la url el nombre del satelite y además almacenar la información enviada de manera independiente a cada satélite | Almacenar la información de manera independiente | 1 | Almacenar correctamente el dato enviado |
| 2 | Cobertura mayor a 80% |
| 3 | Código fuente en repositorio GitHub |
| 4 | Instrucciones de uso en archivo README |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API REST PARA VER ESTADISTICAS DE ADN** | | | | | |
| **NUMERO HU** | **ROL** | **CARACTERISTICA** | **RAZON** | **#** | **CRITERIO DE ACEPTACION** |
| ML-3 | Yo como Cliente | Necesito consumir 1 servicio GET que me permita en la. url enviar el nombre del satelite que deseo consumir | Obtener la información de la. Posición y el mensaje si es posible | 1 | Calcular la posición y obtener el mensaje |
| 2 | Cobertura mayor a 80% |
| 3 | Código fuente en repositorio GitHub |
| 4 | Instrucciones de uso en archivo README |

**ARQUITECTURA**

Para la construcción de la solución se planteo la siguiente arquitectura:

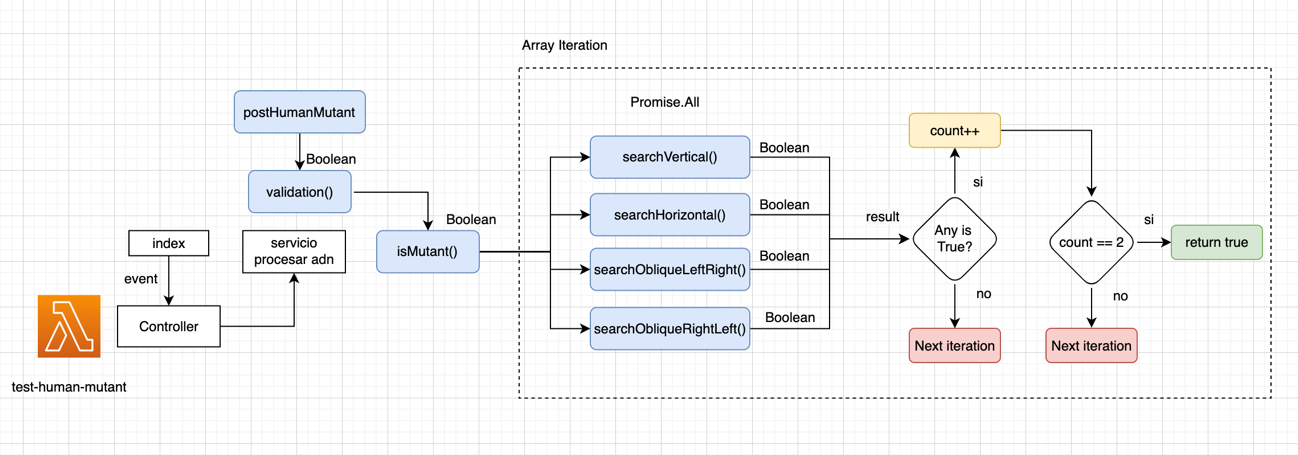
****

Como podemos ver se planteo el esquema de micro servicios en el cual cada lambda se encarga únicamente de un solo proceso, de esta manera se decidió romper un sistema monolítico y separar los servicios en lambdas independientes.

El código de cada lambda esta estructurado bajo el esquema de una arquitectura hexagonal, en la cual se separan los métodos de su implementación (patrón de diseño Facade), de esta manera podemos asegurar un código escalable y flexible a cambios de implementación. También teniendo en cuenta los principios de Clean Code, por lo cual el código se hace altamente legible y fácil de comprender.

**ALGORITMO**

Para el algoritmo que pudiera detectar si una matriz de ADN era un Humano o Mutante se planteo la siguiente solución:



* El servicio **postHumanMutant** es llamado por el controlador y este realiza las siguientes operaciones:

1. Verificamos en la base de datos que el dna ingresado no haya sido procesado anteriormente.
2. Transforma el dna ingresado en una matriz.
3. Realiza una validación de dicha matriz para que cumpla con ser de tamaño NxN, que dicho tamaño sea igual o superior al numero de letras consecutivas a encontrar, que las letras a evaluar sean las permitidas [A,T,C,G].
4. Invoca la función **isMutant** encargada del algoritmo.
5. Guardamos el dna ejecutado y el resultado en nuestra Base de datos.

* La función **isMutant** contiene un iterador para recorrer la matriz, por cada iteración hacemos lo siguiente:

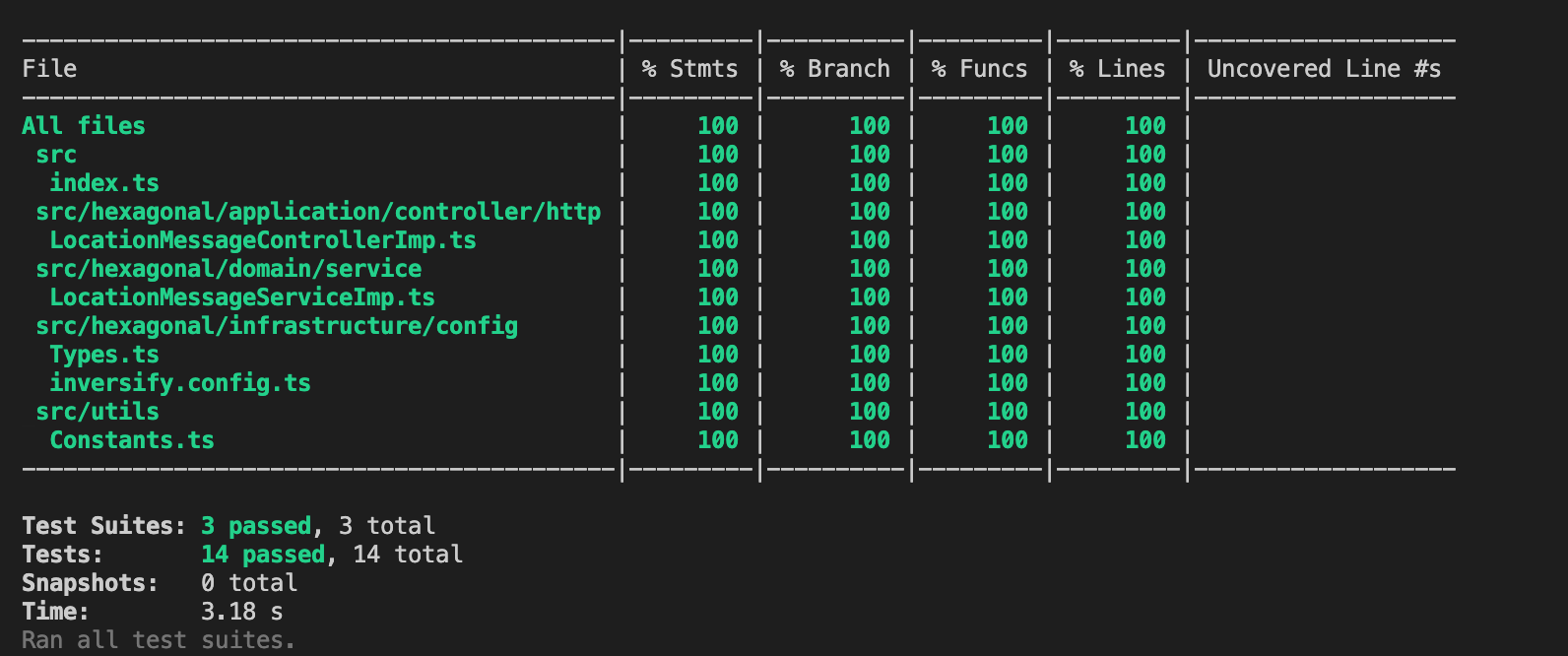
1. Validamos que la posición que se esta analizando aun no forme parte de una secuencia ya encontrada (esto se hace por medio de una variable global en el servicio, que guarda las posiciones que ya forman parte de una secuencia).
2. Si la **posición** se puede analizar, ejecutamos un **Promise.all()** en donde analizamos si la Letra que estamos evaluando (posición), tiene el resto de las N letras consecutivas restantes de manera **vertical**, **horizontal** u **oblicua.**
3. Si alguna de las evaluaciones anteriores retorna **true**, se incrementa un **contador**, y se evalúa si es igual a 2, de ser así la función finaliza retornando **true**, si no continua con la siguiente iteración.

* **NOTA:** se creo un archivo de constantes en el cual tenemos las variables:   
  \* **Letters**: de tipo arreglo: por default [A,T,C,G], aquí podemos agregar las N letras permitidas en el dna de entrada.  
  \* **sequence**: indica el numero de letras que deben estar en secuencia, por default esta en 4, pero el código se estructuro para que sea dinamico y acepte valores diferentes.

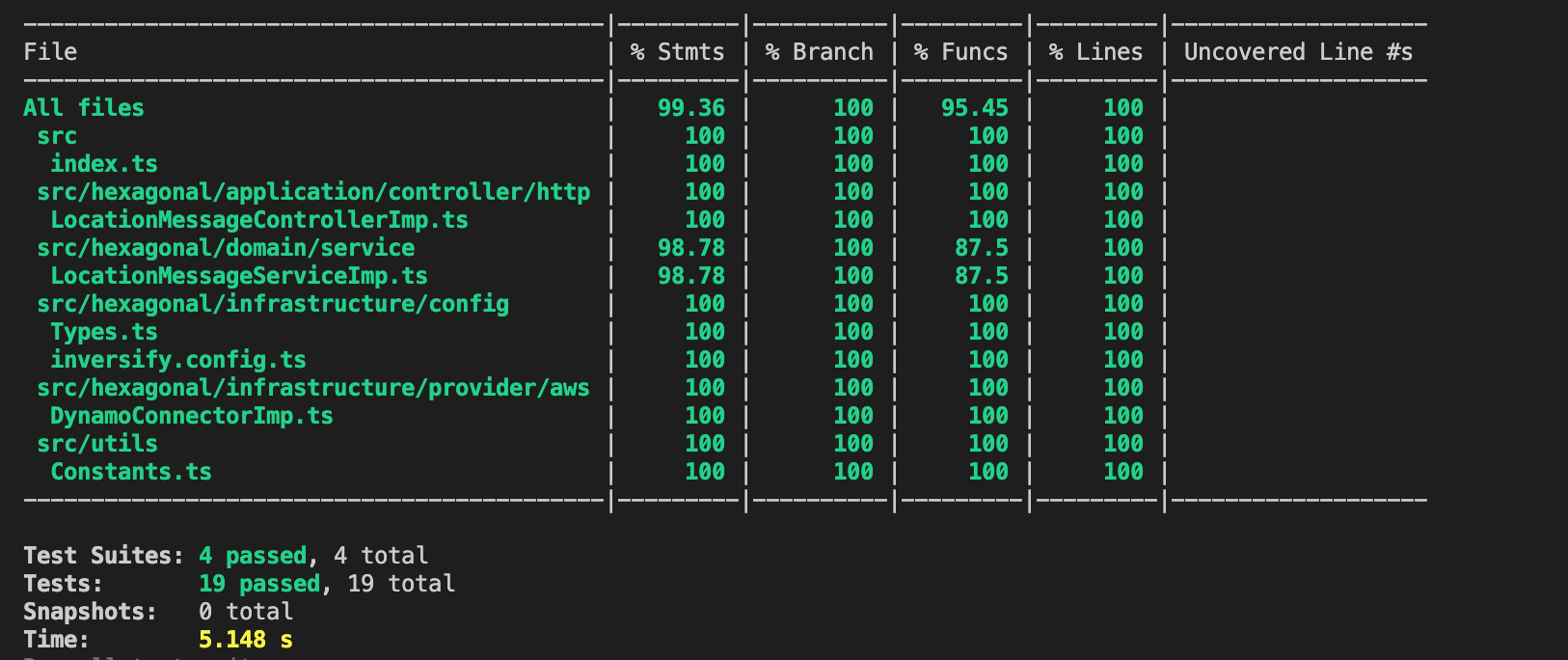
**TEST Y COBERTURA**

Se configuraron y ejecutaron las pruebas unitarias del código arrojando los siguientes resultados:

* Código de la Lambda que procesa el ADN



* Código de la lambda que obtiene las estadísticas



**RESPUESTAS DE LAS API**

|  |  |  |
| --- | --- | --- |
| **RESPUESTA DE LA API PROCESAR ADN** | | |
| **codigo** | **descripcion** | **respuesta** |
| 200 | Respuesta cuando el dna procesado da como resultado: Mutant | * Mutant |
| 403 | Respuesta cuando el dna procesado da como resultado: Human | * Human |
| 404 | Respuesta cuando existe alguna inconsistencia en el proceso | * El dna provisionado no es correcto * El dna provisionado ya fue procesado, resultado: Human/Mutant |

|  |  |  |
| --- | --- | --- |
| **RESPUESTA DE LA API STATS** | | |
| **codigo** | **descripcion** | **respuesta** |
| 200 | Respuesta exitosa | {  "count\_mutant\_dna": 0,  "count\_human\_dna": 0,  "ratio": 0  } |
| 404 | Respuesta cuando no existen datos guardados | * Actualmente la base de datos se encuentra vacia |