Carrera de Pods

Las Carreras de Vainas era un deporte muy popular en las zonas menos desarrolladas de la galaxia, como en Tatooine y Malastare. Se las conocía como "La prueba definitiva de valentía y habilidad de Tatooine". Era, sin duda, uno de los deportes más peligrosos que se habían inventado. Estas carreras no solo eran para entretenimiento, también las apuestas que se hacían llegaron a ser tan importantes que, en algunos casos, la economía y negocios de esas zonas llegaron a girar en torno a ellas.

Tu misión es crear un programa que retorne la posición exacta que utilizara el jurado para la determinación de los ganadores y la información de salud para control de las escuderías (temperatura de motores anti gravitación, nivel de energía, temperatura de baterías, porcentaje de averías).

Para esto, cuentas con tres antenas que te permitirán triangular la posición, ¡pero cuidado! Los mensajes pueden no llegar completo a cada antena debido a las interferencias provocadas por el campo de plasma de los competidores.

Posición de las antenas en servicio

- Antena0: [-500, -200]
- Antena1: [100, -100]
- Antena2: [500, 100]

Nivel 1

Crear un programa con las siguientes firmas:

// input: distancia al emisor (pod) tal cual se recibe en cada antena

// output: las coordenadas ‘x’ e ‘y’ del emisor del mensaje

func GetLocation(distances ...float32) (x, y float32)

// input: las métricas tal cual es recibido en cada antena

// output: las métricas tal cual lo genera el emisor y el nombre del pod

func GetMessage(messages ...[]string) (msg string)

Consideraciones:

- La unidad de distancia en los parámetros de GetLocation es la misma que la que se utiliza para indicar la posición de cada antena.
- Las métricas recibidas en cada antena, se recibe en forma de arreglo de strings.
- Cuando un parámetro del mensaje no pueda ser determinado, se reemplaza por un string en blanco en el array.
    - Ejemplo: [“590C”, “1MWh”, “”, “60%”]
- Considerar que existe un desfasaje (a determinar) en las métricas que se recibe en cada antena.
    - Ejemplo:
        - Antena0: [“590C”, “1MWh”, “110C”, “”]
        - Antena1: [“590C”, “”, “110C”, “60%”]
        - Antena2: [“1MWh”,””, “60%”,””]

Nivel 2

Crear una API REST, hostear esa API en un cloud computing libre (Google App Engine, Amazon AWS, etc), crear el servicio /podhealth/ en donde se pueda obtener la ubicación de la nave y las métricas que emite.

El servicio recibirá la información de la nave(pod) a través de un HTTP POST con un payload con el siguiente formato:

POST → /podhealth/

```
{
 "antenas": [
        {
           "name": "antena0",
           "pod": "Anakin Skywalker",
           "distance": 210.0,
           "metrics": ["590C","","","60%"]
        },
        {
           "name": "antena1",
           "pod": "Anakin Skywalker",
           "distance": 225.5,
           "metrics": ["","1MWh","","60%"]
         },
         {
           "name": "antena2",
           "pod": "Anakin Skywalker",
           "distance": 252.7,
           "metrics": ["590C","","110C, ",""]
        }
] }
```

La respuesta, por otro lado, deberá tener la siguiente forma:
RESPONSE CODE: 200

```
{
        "pod": "Anakin Skywalker",
        "position": {
          "x": -210.0,
          "y": 95.5
        },
        "metrics": "590C,1MWh, 110C, 60%"
}
```

En caso que no se pueda determinar la posición o las métricas, retorna:

RESPONSE CODE: 404

Nivel 3

Considerar que el mensaje ahora se debe poder recibir en diferentes POST al nuevo servicio /podhealth_split/, respetando la misma firma que antes. Por ejemplo:

POST → /podhealth_split/{antena_name}

```
{
"pod": "Anakin Skywalker",
"distance": 100.0,
 "message": [“590C”, "", "", "60%"]
 }
```

Crear un nuevo servicio /podhealth_split/ que acepte POST y GET. En el GET la respuesta deberá indicar la posición y las métricas en caso que sea posible determinarlo y tener la misma estructura del ejemplo del Nivel 2.

Caso contrario, deberá responder un mensaje de error indicando que no hay suficiente información.

Entregables

- Código fuente en repositorio privado de GitHub
- Documentación que indique cómo ejecutar el programa
- Documentación del proyecto que considere importante
- URL en donde este hosteado el servicio
- Contemplar buenas prácticas (tip: imaginar que estas poniendo una aplicación productiva).

Nivel 4:

Fase 2 Carrera de Pods

Teniendo en cuenta que es posible que el sistema diseñado Carrera de Pods requiera mayor precisión ampliando considerablemente los muestreos por segundo y también se dispare el número de peticiones realizadas al sistema, diseñar una arquitectura que soporte los siguientes requerimientos no funcionales:

1. El sistema debe soportar 4000 RPM (Requests per minute)

2. El sistema debe estar protegido a posibles ataques de hackers.

3. Escalabilidad: la arquitectura debe soportar que se pueda aumentar o disminuir la cantidad de antenas consumidores para descifrar o determinar la ubicación.

4. Garantizar la entrega de los mensajes a 3 servicios POST intergalácticos (sin dependencia entre ellos) desde podhealth_split para notificar a jurados independientes en diferentes ubicaciones, dichos servicios tienen picos de timeout e indisponibilidad sin un patrón determinado (no se ha determinado en qué franja horaria se presentan las fallas), sin embargo, es de vital importancia que los mensajes lleguen a su destino.

5. Se debe almacenar registro de logs para realizar análisis histórico.

6. Indicadores: los servicios serán monitoreados 7x24 por su importancia, Plantear los indicadores que debe monitorear el equipo de desarrollo para anticipar o actuar ante cualquier problema que se genere.

Entregables

1. Diagrama de arquitectura de componentes en el que muestre las interacciones entre servicios/consumidores y las tecnologías a usar para solucionar el problema.

2. Diagrama de clases que explique el funcionamiento del sistema y los patrones de desarrollo a implementar.

3. Listado de lineamientos para asegurar que el equipo de desarrollo cumpla con los requerimientos no funcionales.

4. Bonus: documentación técnica adicional que pueda mejorar el entendimiento al equipo de desarrollo.

Nota sobre uso de asistencia IA

En los casos en que se haya requerido asistencia de Inteligencia Artificial para la elaboración, diseño o resolución del desafío, se debe **documentar la cronología de los prompts utilizados y el modelo de IA empleado**.
Esta información debe quedar registrada junto con los entregables del proyecto para garantizar **transparencia, trazabilidad.**