

Operación fuego Quasar

Documentación General

El objetivo es un programa que **retorne la fuente y contenido del mensaje de auxilio**. Para esto, se contó con tres satélites que permiten triangular la posición, prevee que el mensaje puede no llegar completo a cada satélite debido al campo de asteroides.

Posición de los satélites actualmente en servicio

- Kenobi: [-500, -200]
- Skywalker: [100, -100]
- Sato: [500, 100]

El programa recibe la información de distancias al emisor y los mensajes en cada satélite retornando el mensaje traducido y completo con las coordenadas de la nave. El programa permite el envío de la información de forma completa como parcial por cada satélite, además esta información es guardada y puede ser consultada a el programa en todo momento.

Información del programa

Desarrollado en C# (.NET 6) con Visual Studio Community, implementado en el cloud de Azure, se entrega información específica de la API de forma separada a este documento. El programa incluye documentación a nivel de código y pruebas unitarias de los servicios. Se contempla también una versión desarrollada en Javascript (NodeJs) no implementada pero funcional e implementable.

Repositorio: <https://github.com/asantiagociv/Operacion-Fuego-de-Quasar>

Matemática aplicada

Para ubicar la posición del emisor se utiliza la técnica de la trilateración, a partir de las distancias se miden desde los puntos de referencia conocidos que son los satélites.

La fórmula para la trilateración se puede expresar de la siguiente manera:

$$d1 = \sqrt{(x - x1)^2 + (y - y1)^2}$$

$$d2 = \sqrt{(x - x2)^2 + (y - y2)^2}$$

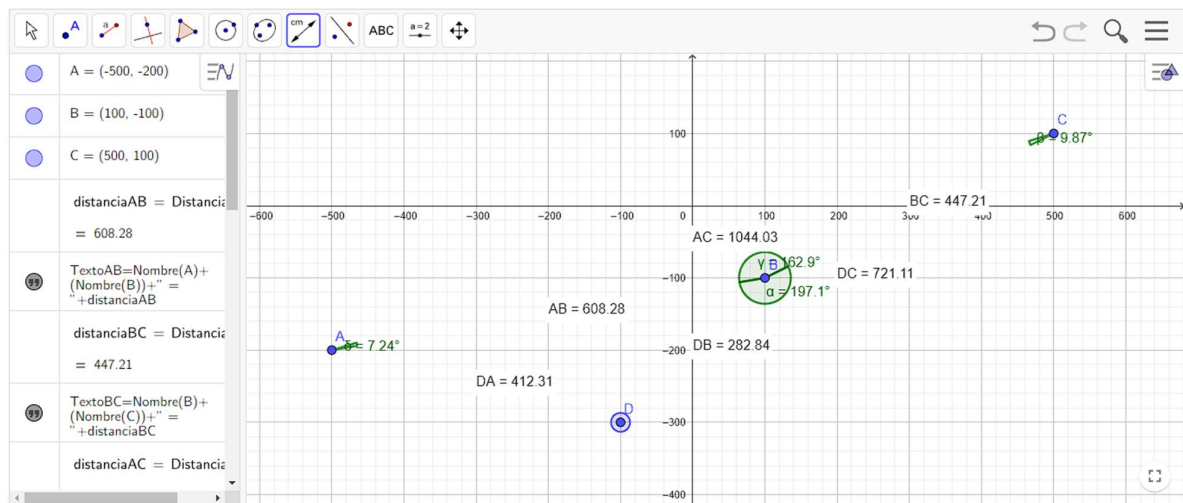
$$d3 = \sqrt{(x - x3)^2 + (y - y3)^2}$$

Donde:

- (x_1, y_1) , (x_2, y_2) , y (x_3, y_3) son las coordenadas conocidas de los puntos de referencia (en este caso, los satélites).
- d_1 , d_2 y d_3 son las distancias medidas desde el emisor del mensaje a los satélites.
- (x, y) son las coordenadas desconocidas del emisor del mensaje que se intenta determinar.

Para determinar las coordenadas del emisor del mensaje (x, y) , se pueden resolver las ecuaciones anteriores utilizando técnicas de álgebra lineal.

Ejemplo calculado en plano cartesiano:



Los puntos A, B y C son los satélites, el punto D es la nave.

Ejecutar el programa C# (Debug)

Se debe tener en consideración los siguientes requisitos:

- Se utilizará Visual Studio Community (Ejemplo 2022)
- Instalado paquete “Desarrollo ASP.NET y web” en Visual Studio Community
- Instalado SDK .NET 6.0

Paso 1

Abrir archivo “operacion_fuego_quasar.sln” en Visual Studio.

Paso 2

Restaurar los paquetes NuGet en caso de ser necesario, luego en la parte superior la opción de “Debug” y luego start como IIS Express.

La aplicación iniciara en debug abriendo automáticamente la pagina de la API, el puerto en que inicia la API puede ser modificado en el archivo launchSetting.json dentro de Properties y en el tag iisSettings.

Publicar en Azure

Se debe tener en consideración los siguientes requisitos:

- Cuenta de Azure
- Se utilizará Visual Studio Community (Ejemplo 2022)
- Instalado SDK .NET 6.0

Paso 1

Abrir el archivo “operacion_fuego_quasar.sln” en Visual Studio.

Paso 2

Restaurar los paquetes NuGet en caso de ser necesario, inicialmente podemos compilar primero para confirmar que este todo correcto, de igual forma el proceso de publicación compila la aplicación.

Paso 3

En Azure utilizaremos “Azure App Service” que permite hostear cualquier aplicación Web y APIs Rest (.NET, NodeJs, PHP, Java, etc.). Debemos tener creado un App Service Plan, podemos buscar este apartado en el buscador, creamos uno nuevo si no tenemos, con las especificaciones de pago que deseemos, en este caso se utilizó una suscripción de tipo gratuita, sistema operativo Windows con locación en Brazil South.

Microsoft Azure

Actualización

Buscar recursos, servicios y documentos (G+/)

[Inicio](#) > [Planes de App Service](#) >

Crear plan de App Service

Datos básicos

Etiquetas

Revisar y crear

Los planes de App Service le ofrecen flexibilidad para asignar aplicaciones específicas a un conjunto de recursos concreto y para optimizar aún más el uso de los recursos de Azure. De esta forma, si quiere ahorrar gastos en el entorno de prueba, puede compartir un plan entre varias aplicaciones. [Más información](#)

Detalles del proyecto

Seleccione una suscripción para administrar los recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción *

Evaluación gratuita

Grupo de recursos *

(Nuevo) Grupo de recursos

[Crear nuevo](#)

Detalles del plan de App Service

Nombre *

Escriba el nombre de su plan de App Service

Sistema operativo *

☒ Linux ☐ Windows

Región *

East US

Plan de tarifa

El plan de tarifa de App Service determina la ubicación, las características, los costos y los recursos del proceso asociados a la aplicación. [Más información](#)

Plan de precios

Gratis F1 (Infraestructura compartida)

Redundancia de zona

Un plan de App Service se puede implementar como un servicio con redundancia de zona en las regiones que lo admiten. Esta es una decisión que solo se toma en el momento de la implementación. Después de la implementación, no se podrá hacer que un plan de App Service tenga redundancia de zona. [Más información](#)

Redundancia de zona

☐ **Habilitada:** Su plan de App Service y las aplicaciones que contiene tendrán redundancia de zona. El número mínimo de instancias del plan

Revisar y crear

< Anterior

Siguiente: Etiquetas >

Paso 4

A continuación, debemos crear un App Service, el nombre de la instancia definirá la url de nuestra aplicación, colocamos las especificaciones de pago deseadas, en este caso se utilizó una suscripción de tipo gratuita, sistema operativo Windows con locación en Brazil South y en este caso seleccionamos como “Pila de entorno de ejecución” una aplicación de .NET 6.

Crear aplicación web ...

Datos básicos Implementación Redes Supervisión Etiquetas Revisar y crear

App Service Web Apps le permite generar, implementar y escalar rápidamente aplicaciones empresariales web, móviles y de API que se ejecutan en cualquier plataforma. Satisfaga los estrictos requisitos de rendimiento, escalabilidad, seguridad y cumplimiento sin renunciar a una plataforma totalmente administrada para el mantenimiento de la infraestructura. [Más información](#)

Detalles del proyecto

Seleccione una suscripción para administrar los recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción *

Grupo de recursos *

[Crear nuevo](#)

Detalles de instancia

¿Necesita una base de datos? [Pruebe la nueva experiencia de web y base de datos.](#)

Nombre *

Publicar * ☒ Código ☐ Contenedor Docker ☐ Aplicación web estática

Pila del entorno en tiempo de ejecución *

Sistema operativo ☒ Linux ☐ Windows

Región *

[¿No encuentra su plan de App Service? Pruebe otra región o seleccione su App Service Environment.](#)

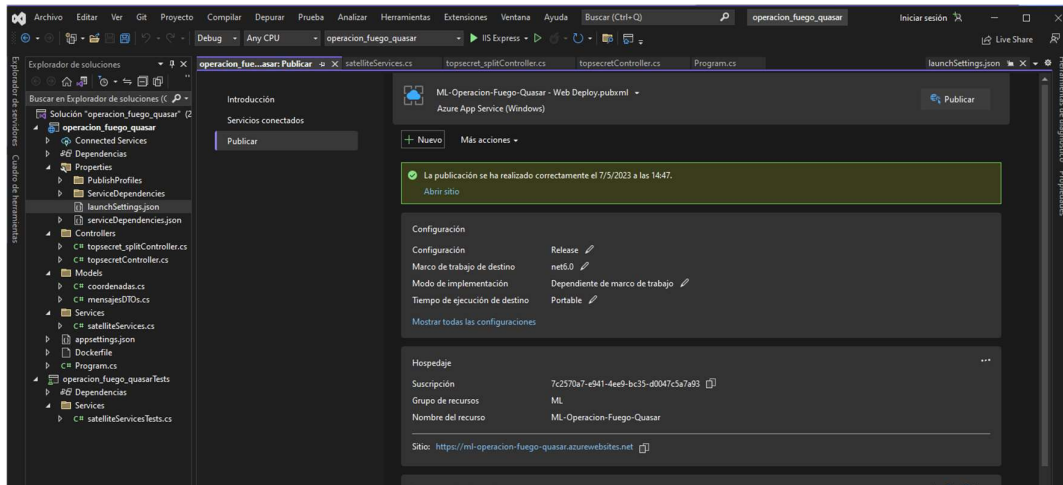
Planes de precios

El plan de tarifa de App Service determina la ubicación, las características, los costos y los recursos del proceso asociados a la aplicación. [Más información](#)

[Revisar y crear](#) [< Anterior](#) [Siguiendo: Implementación >](#)

Paso 5

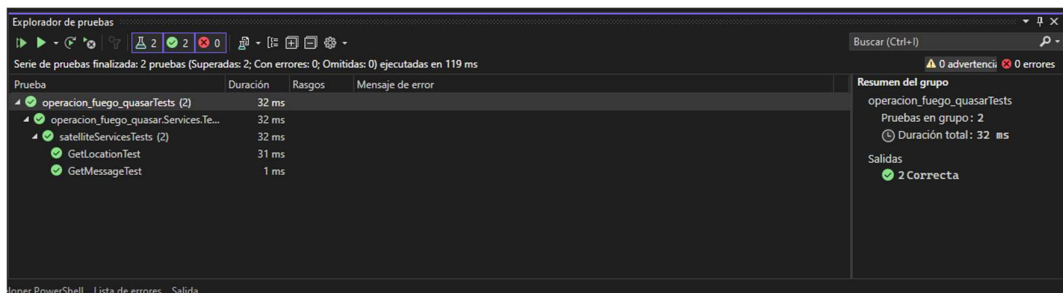
En Visual Studio seleccionamos en la parte superior Compilar, luego Publicar operación_fuego_quasar. Creamos un nuevo destino y seleccionamos Azure, luego seleccionamos Azure App Service (Windows) para este caso, debemos tener la sesión iniciada con cuenta de Azure buscamos el App Service que creamos anteriormente y damos en finalizar.



Luego damos en Publicar y al terminar nos iniciara la url de la aplicación.

Ejecutar Pruebas Unitarias C#

En Visual Studio vamos a la parte superior seleccionamos Prueba, luego ejecutar todas las pruebas.



Ejecutar el programa NodeJs (Debug)

Se debe tener en consideración los siguientes requisitos:

- NodeJs instalado
- Se utilizará Visual Studio Code para el ejemplo
- Instalado npm

Paso 1

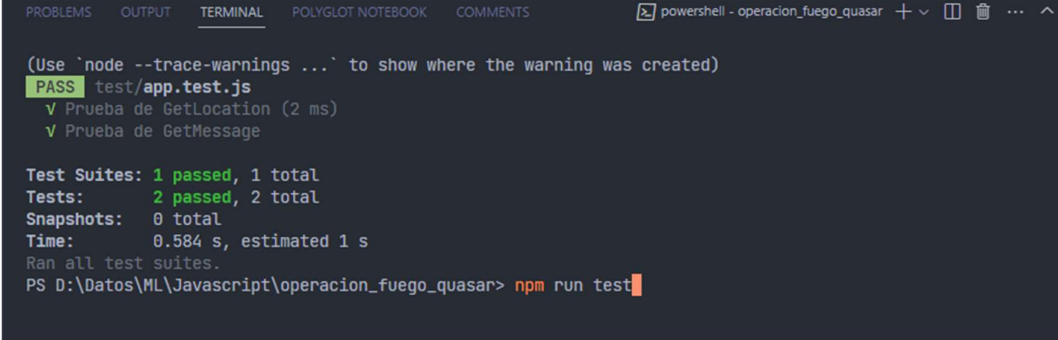
Desde consola en la carpeta del proyecto, ejecutar el comando “npm install” para descargar las dependencias del proyecto.

Paso 2

Ejecutar el comando “npm run start” para iniciar el servicio en el puerto 3000, el puerto puede ser modificado desde el archivo index.js.

Ejecutar pruebas unitarias NodeJs

Mismos requisitos que en el paso anterior, solo debemos ejecutar en consola el siguiente comando: “npm run test”.



```
PROBLEMS  OUTPUT  TERMINAL  POLYGLOT NOTEBOOK  COMMENTS  powershell - operacion_fuego_quasar + v [] [x] ... ^

(Use `node --trace-warnings ...` to show where the warning was created)
PASS test/app.test.js
  ✓ Prueba de GetLocation (2 ms)
  ✓ Prueba de GetMessage

Test Suites: 1 passed, 1 total
Tests:      2 passed, 2 total
Snapshots:  0 total
Time:       0.584 s, estimated 1 s
Ran all test suites.
PS D:\Datos\ML\Javascript\operacion_fuego_quasar> npm run test
```