

# DOCUMENTACIÓN DEL PROYECTO QUASAR

1. Introducción	2
2. Peticiones HTTP disponibles en la API	2
3. Datos iniciales	5
4. Consideraciones	5
5. Guía de uso del cliente Front	5
a. Instalación y dependencias	5
b. Pantalla servicio 1	6
c. Pantalla servicio 2	8
d. Pantalla servicio 3	10

## Introducción

El proyecto está dividido en 2 proyectos, el servicio BACKEND y el servicio cliente (FRONT) quien es el que va a consumir dicho servicio.

Para acceder al servicio BACKEND se debe acceder al dominio en donde está deployado los servicios rest hechos con JAVA SPRINGBOOT.

<https://operacionfuegodequasarm1.herokuapp.com> -> En esta URL está publicada los servicios de la Api.

En cualquier caso, se puede probar de forma local al ejecutarla con Eclipse y se publicará en el host : LOCALHOST y el puerto: 8080

A partir de la misma se tiene los siguientes servicios HTTP de los cuales se explicará también cuales son sus funcionalidades, los parámetros que recibe y los que devuelve si salió correctamente o el mensaje de la excepción en caso de que no se lograra realizar la petición

## Peticiones HTTP disponibles en la API

- GET -> /

- Devuelve el string “PRUEBA” nada mas, solo está para poder visualizar si funciona el servidor, recomendado probar este primero para corroborar que esté levantado y que se conozca la URL a donde tiene que enviarse los datos de la petición.
- GET -> /coordenadas/
  - Devuelve los datos respecto a los satélites de los cuales devuelve las coordenadas de todos los satélites respecto a la posición X e Y, y además retorna el nombre de los satélites operativos
- Post -> /coordenadas/{nombre satélite}
  - Pasando por request body un json donde está la nueva posición (X, Y) y un nombre que está en el QUERYPARAM, entonces actualizará la posición del satélite
  - Puede ocasionar una excepción para los siguientes datos
    - No existe un satélite con ese nombre en específico, ya que no se van a crear nuevas entidades de satélites
    - Los datos ingresados en los campos de X, Y no corresponde a 1 numérico
- POST -> /topSecret/
  - Pasando como petición de body un array de satélites con los siguientes datos cada una
    - Name : Nombre del satélite
    - Distance : Distancia de la nave contra el satélite anteriormente nombrado
    - Message : Un array de palabras cargadas, de las cuales algunas están vacías ya que representa el desfase
  - FUNCIONALIDADES:
    - Debe ubicar primero que todos los nombres de los satélites existan, sino lanza una excepción de que no existe algún satélite nombrado
    - Mergeo y obtener mensaje original: Mergea todos los mensajes para ubicar el mensaje original cubriendo las palabras desfasadas. Sin embargo puede haber el error de que al terminar de mergearlo sigan habiendo palabras desfasadas o bien, en la misma posición del mensaje hay

diferentes palabras originando una excepción que no se pueda obtener el mensaje original

- Distance: Utilizando triangulación, con los datos de las distancias hacia los satélites y las posiciones de la misma, se obtendrá las coordenadas de los cuales para las 3 distancias y coordenadas, exista 1 punto en común entre los 3 que sería la posición de la nave. Sin embargo en caso de que no se logre dicho punto en común entonces devuelve una excepción.
- Debe devolver la posición original de la nave y el mensaje emitido.
- POST -> /topsecret\_split/{NOMBRE SATELITE}
  - Pasando una petición de la distancia de la nave al satélite nombrado y un array de palabras con posibles desfasajes, debe actualizar en las instancias de listaTransmisiones dicha información
  - Debe considerarse el nombre del satélite ya que si no existe dicho satélite, entonces sale excepción, lo mismo si la distancia no es numérica.
- GET -> /topsecret\_split/
  - En base a los datos cargados que es una lista de (Nombre satélite – distancia al satélite – array de palabras) tratará de obtener la información de la posición en X e Y a través de usar triangulación, además del mensaje original,
  - En caso de que no se pueda obtener las coordenadas de la nave ni el mensaje original emitirá el siguiente mensaje de excepción “NO HAY SUFICIENTE INFORMACIÓN”
- DELETE -> /topsecret\_split/
  - Vacía la lista de transmisiones
- GET -> /topsecret\_split/data
  - Obtendrá toda la información que existe respecto a las transmisiones realizadas

Por otra parte se creó un cliente FRONT en angular 8 del cual consumirá los servicios de la APIREST y se tendrá el siguiente instructivo

## DATOS INICIALES

La aplicación está inicializada con los siguientes datos:

1. Satélites:
  - a. Kenobi (posición X: 14, Posición Y: 45)
  - b. Skywalker (Posición X: 80, Posición Y:70)
  - c. Sato (Posición X: 71, Posición Y: 50)
2. Mediante triangulación se puede verificar que las coordenadas de la nave son X:50 – Y: 30 si las distancias entre nave y satélite son:
  - a. KENOBI -> 39.0
  - b. SKYWALKER -> 50.0
  - c. SATO -> 29.0
3. Los satélites no cuentan con transmisiones, las cuales son datos que pueden ser reemplazados o borrados así que no se tiene interés en su persistencia

## CONSIDERACIONES:

No se consideró la persistencia de los datos ya que no hay necesidad por parte de la API consultar a una base de datos para ir guardando posicionamiento de la nave y los mensajes que se van trasmitiendo constantemente además que los satélites siempre serán esos 3 y la única modificación que tendrán estas es cuando su posición cambia pero eso no amerita la necesidad de una base de datos relacional.

Por otra parte se consideró el desarrollo de un cliente front (que no pudo ser deployada y por lo tanto se usará de forma local como una DEMO) para una mejor interacción entre el usuario y la API.

Y por último, se tomó en cuenta la creación de nuevos servicios de la API para poder tener uso mas expandido de las funcionalidades y también la consideración de diversas excepciones para poder obtener la suficiente información sobre cuales son los problemas que se han encontrado.

## GUÍA DE USO DEL FRONT.

Para utilizar el front se requerirán en la carpeta del front

/front/frontQuasar:

1. Node.js <https://nodejs.org/es/> (8.x en adelante)
2. Angular CLI (`npm install -g @angular/cli`) (quitar el `-g` para que no sea global pero para que funcione debe hacerlo en la carpeta del front)
3. Typescript <https://www.typescriptlang.org/download>
4. Instalar las siguientes dependencias para el funcionamiento del proyecto de forma local:
  - a. BOOTSTRAP `npm install bootstrap@next`
  - b. SWEETALERT2 `npm install sweetalert2`
  - c. ANGULAR MATERIAL `ng add @angular/material`

Una vez instalados ir desde la consola y ejecutar NPM START

```
quasar (mas c  
$ npm start
```

Esperamos que se levante e ir a la dirección “localhost:4200/” para empezar a interactuar con el cliente, deberá aparecer la siguiente pantalla:

The screenshot displays a web application interface with the following sections:

- Servicio 1:** A header with the text "servicio 1, enviar esta información y obtener la ubicación de la nave en el caso que se pueda y el mensaje original". Below it is a form with input fields for "Nombre de satélite", "distancia", and "Mensajes a enviar", followed by an "Agregar datos de consulta" button.
- Table 1:** A table with 4 columns: "#", "Nombre satélite", "Distancia", and "Mensaje recibido". It contains 3 rows of data:

#	Nombre satélite	Distancia	Mensaje recibido
0	kenobi	100.0	[este_mensaje]
1	skywalker	100.0	[un_secreto]
2	sato	100.0	[es_mensaje]
- Buttons:** Below the table are two buttons: "Borrar todos los elementos de la lista" (red) and "Enviar consulta" (green).
- Servicio 2:** A header with the text "Servicio 2, visualizar y actualizar la posición de los satélites".
- Table 2:** A table with 3 columns: "Nombre satélite", "Posición X", and "Posición Y". It contains 3 rows of data:

Nombre satélite	Posición X	Posición Y
kenobi	14	45
skywalker	80	70
sato	71	50
- Form:** Below the table is a form with input fields for "Nombre de satélite", "Posicion en X", and "Posicion en Y", followed by an "Actualizar posición" button.
- Servicio 3:** A header with the text "Servicio 3, cargar para 1 satélite la distancia y el mensaje emitido por la nave".
- Form:** Below the header is a form with input fields for "Nombre de satélite", "distancia", and "Mensajes a enviar", followed by an "Agregar datos" button.
- Table 3:** A table with 3 columns: "Nombre satélite", "Distancia", and "Mensaje recibido".
- Buttons:** At the bottom are two buttons: "Borrar elementos almacenados" (red) and "Enviar consulta" (green).

## SERVICIO 1

Servicio 1, enviar esta información y obtener la ubicación de la nave en el caso que se pueda y el mensaje original

kenobi	125	hola como - estas - mio	Agregar datos de consulta
--------	-----	-------------------------	---------------------------

#	Nombre satélite	Distancia	Mensaje recibido
0	kenobi	100.0	[ este,,mensaje, ]
1	skywalker	100.0	[ ,,un,,secreto ]
2	sato	100.0	[ ,,es,,mensaje, ]
3	kenobi	125	[ hola,como,,estas,,mio ]

Para agregar un nuevo pedido que se enviaría a la API, el mensaje debe tomar en cuenta que las palabras desfasadas se representan con el carácter “-” tal como se muestra en la pantalla. Además quedará representado como vacía la misma.

Usa la siguiente petición HTTP

- POST -> /topSecret/

Por otro lado, los ítems ya cargados no se pueden modificar por lo tanto se debe vaciar la lista cada vez que se quiera modificar o agregar nueva información.

Por otro lado, se puede cargar en Distancia un valor alfanumérico pero generará un error desde la parte de la API que se mostrará debajo de la tabla

y los botones de BORRAR y CONSULTAR y por último los nombres de los satélites deben matchear si o si con los satélites existentes en la API, la cual se podrá ver en la siguiente pantalla.

Esta interfaz devolverá la información de la siguiente manera.

#	Nombre satélite	Distancia	Mensaje recibido
0	kenobi	100.0	[ este,,mensaje, ]
1	skywalker	100.0	[ ,,un,,secreto ]
2	sato	100.0	[ es,,mensaje, ]
3	kenobi		

Borrar todos los elementos de la lista

Enviar consulta


RESULTADO DE LA PETICIÓN

"Faltan algun satélite para consultar o están sobrando"

Servicio 2, visualizar y actualizar la posición de los satélites

Nombre satélite

kenobi



Error

No hay suficiente información o es incorrecta

OK


Debajo de RESULTADO se mostrará el mensaje JSON de la respuesta o en su defecto el mensaje de la excepción.

0	kenobi
1	skywalker
2	sato

Borrar todos los elementos de la lista

Envia

RESULTADO DE LA PETICIÓN



OK

La petición fue aceptada

También muestro los datos que dieron el caso de éxito



#	Nombre satélite	Distancia	Mensaje recibido
0	kenobi	39	[ este,,,mensaje, ]
1	skywalker	50	[ ,es,,,secreto ]
2	sato	29	[ ,,un,, ]

## SERVICIO 2

Es el servicio de consulta de las posiciones de los satélites, la misma muestra la posición actual de los satélites al igual que nos permite modificar dichas posiciones.

Servicio 2, visualizar y actualizar la posición de los satélites

Nombre satélite	Posición X	Posición Y
kenobi	14	45
skywalker	80	70
sato	71	50

Modificar la posición de 1 satélite indicando su nombre y la posición nueva

Nombre de satélite	Posicion en X	Posicion en Y	Actualizar posición
--------------------	---------------	---------------	---------------------

Hay que tomar una consideración y es que el nombre del satélite debe ser si o si 1 de las 3 que aparecen en pantalla ya que en caso de que se pida un nombre que no corresponde se obtendrá una excepción y la pantalla de error.

Servicio 2, visualizar y actualizar la posición de los satélites

Nombre satélite	
kenobi	
skywalker	
sato	

Modificar la posición de 1 satélite indicando su nombre

sarasa	150
--------	-----



Error

No hay suficiente información o es incorrecta

OK

Pero con una correcta información se actualizará los campos.

Este servicio envia la petición

- Post -> /coordenadas/{nombre satélite} (para actualizar posición)
- GET -> /coordenadas/ (para obtener la posición de todos los satélites)

## Servicio 3

Para finalizar con la guía de uso del FRONT, este es la vista que mas pedidos realiza a la API siendo las siguientes:

- POST -> /topsecret\_split/{NOMBRE SATELITE}
- GET -> /topsecret\_split/
- GET -> /topsecret\_split/data
- DELETE -> /topsecret\_split/

Servicio 3, cargar para 1 satélite la distancia y el mensaje emitido por la nave

Nombre de satélite	distancia	Mensajes a enviar	Agregar datos
--------------------	-----------	-------------------	---------------

Nombre satélite	Distancia	Mensaje recibido
sato	29	[ mensaje,,enviar ]

Borrar elementos almacenados    Enviar consulta

Para cargar un nuevo registro de transmisión, el nombre SI O SI debe existir entre los satélites ya existentes. La distancia y el mensaje se cargaran a continuación una vez que dicho campo del nombre se valide.

Por otro lado se puede borrar todos los elementos almacenados lo que realizará la petición DELETE -> /topsecret\_split/ que borrará toda la lista de transmisiones emitidas por la nave que recibieron los satélites.

Y por último ENVIAR CONSULTA consumirá el pedido GET -> /topsecret\_split/ que devolvería los siguientes resultados:

sato

Modificar la posición de 1 satélite indicando su nombre

Nombre de satélite	Posicion en X
sato	

Servicio 3, cargar para 1 satélite la distancia y el mensaje

Nombre de satélite	distancia
sato	

**Nombre satélite**

sato

Borrar elementos almacenados Enviar consulta

Resultado de la petición

'No hay suficiente información'

Al clickear se llena debajo la siguiente leyenda en caso de salir por excepción pero en el caso de que sea correcta sale lo siguiente:

Servicio 3, cargar para 1 satélite la distancia y el mensaje emitido por la nave

Nombre satélite	Distancia	Mensaje recibido
sato	29	[ mensaje,,enviar ]
kenobi	39	[ ,a, ]
skywalker	50	[ ,a,enviar ]

Borrar elementos almacenados Enviar consulta

Resultado de la petición

{"posicion":{"x":50,"y":30},"mensaje":"mensaje a enviar"}

Todo esto es lo que se necesita saber para manejar el cliente front.