Desafío HTTP

Para validar tus conocimientos en este tema te proponemos resolver el siguiente desafío:

Tendrás que programar **2 servidores**, estos servidores expondrán los siguientes endpoints con las funciones especificadas a continuación

/ping

Este endpoint recibirá una petición de tipo GET y responderá con status code 200 OK y el JSON {“message” : ”Pong!”}

/forward

Este endpoint recibirá una petición de tipo GET con un parámetro url que hace referencia a una URL que devuelve un JSON la cual invocaremos con otra petición de tipo GET y retornaremos status code 200 y su JSON de respuesta.

En caso de error ejecutando la petición con el servicio externo se espera recibir una respuesta con status code 500 y un JSON con un mensaje de error como el siguiente:

{"message": "Error executing request"}

Ejemplo:

Petición

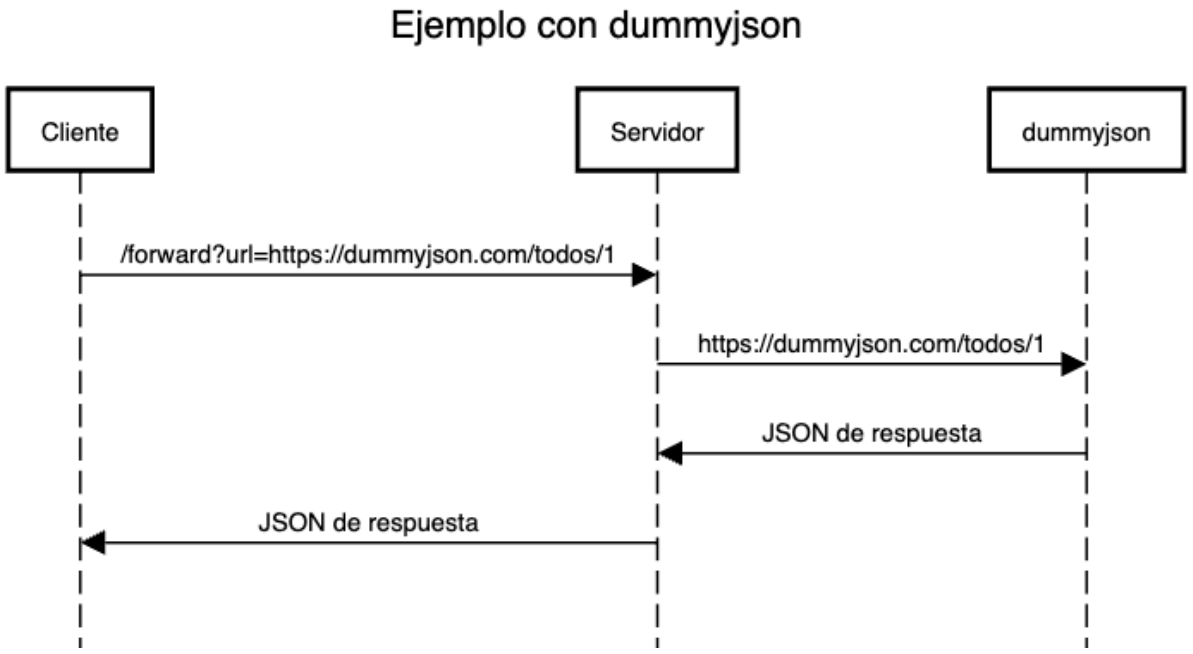
GET /forward?url=https://dummyjson.com/todos/1[1]

Respuesta


```
{
        "id": 1,
        "todo": "Do something nice for someone I care about",
        "completed": true,
        "userId": 26
}
```

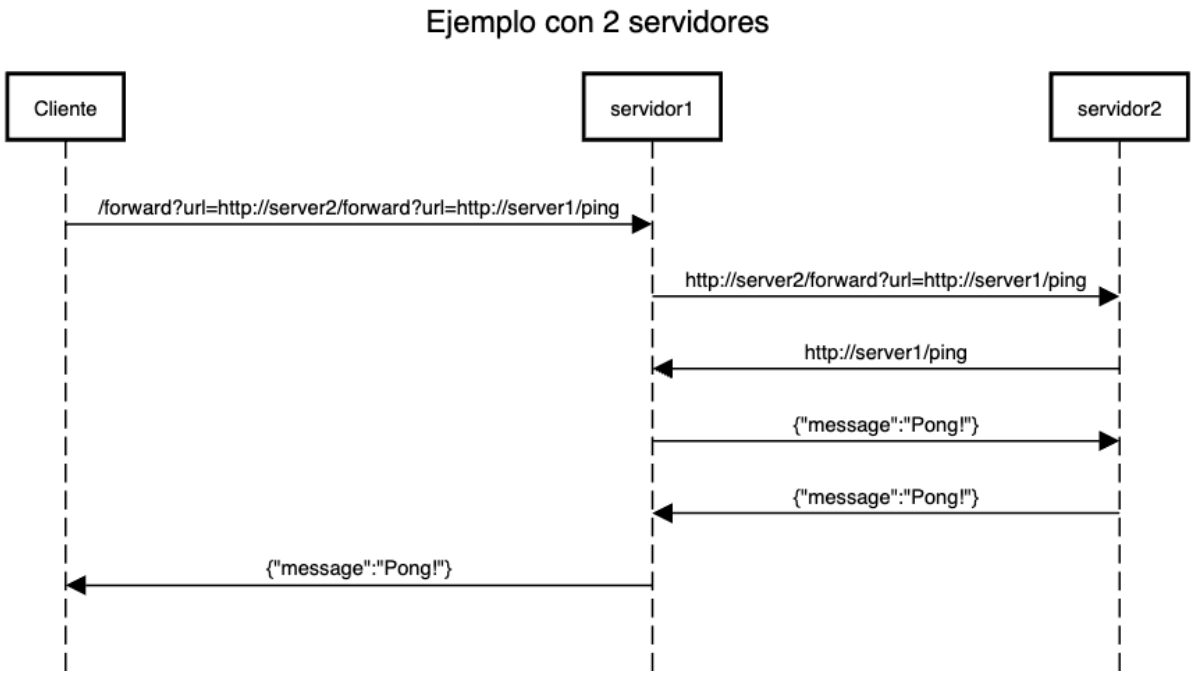

[1] https://dummyjson.com/ es una web que nos permite utilizar JSONs predefinidos para probar nuestros servicios

Funcionamiento:



Dado que vamos a tener 2 servidores con el mismo funcionamiento podríamos anidar las peticiones de tipo forward entre ellos, por ejemplo ejecutando la siguiente petición:

GET /forward?url=http://server2/forward?url=http://server1/ping



En el ejemplo anterior sucede lo siguiente:

1. El cliente ejecuta la petición contra el servidor1
2. el servidor1 interpreta la petición, extrae la URL y ejecuta una nueva petición contra el servidor2
3. El servidor2 interpreta esta petición y al ser un forward vuelve a extraer la URL y ejecuta la nueva petición, esta vez haciendo un ping contra el servidor1
4. El servidor1 responde con el mensaje de Pong al servidor2 como respuesta a la petición del punto 3
5. El servidor2 responde con el resultado del Pong al servidor1 como respuesta a la petición del punto 2
6. El servidor1 reenvía esta respuesta al cliente

Sugerencias:

- Puedes usar este repo como guía para ver ejemplos de distintas implementaciones de APIs REST
- Recuerda que dentro de la red interna de docker-compose puedes invocar los servicios utilizando el nombre de servicio, de esta forma no tendrás que saber cual es su IP
- Si expones los servicios en el puerto 80 y les llamas server1 y server2 las peticiones deberían quedar prácticamente iguales a los mencionados en este documento
- Puede ser de utilidad agregar logs en tu programa para entender que el funcionamiento es el esperado

Entrega:

Deberás entregar un archivo de **docker-compose.yml** junto con el código que contiene la implementación de ambos servidores.
A su vez deberás agregar un README.md que contenga los pasos a seguir para ejecutar y validar tu solución.

Criterios de evaluación:

50 puntos si resuelves el endpoint de ping
30 puntos si resuelves el endpoint de forward
20 puntos si lo resuelves utilizando 2 lenguajes de programación