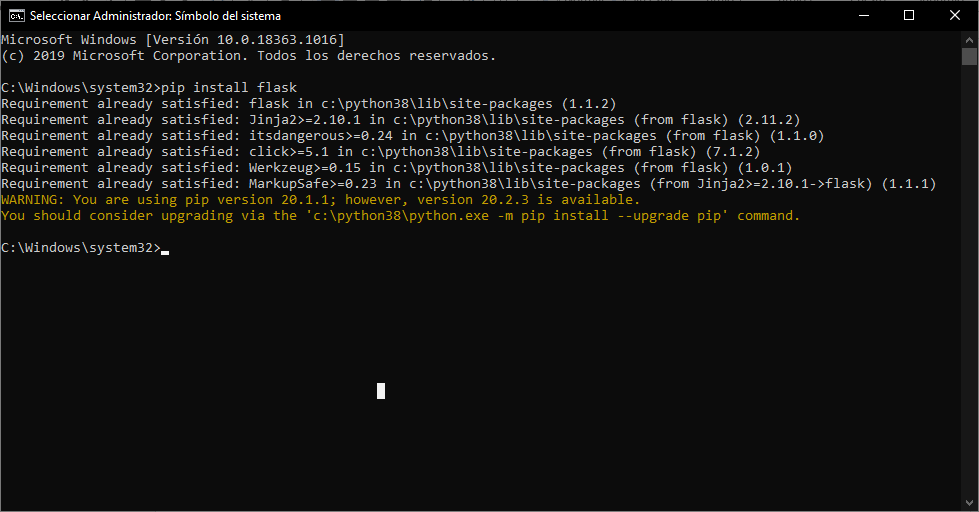
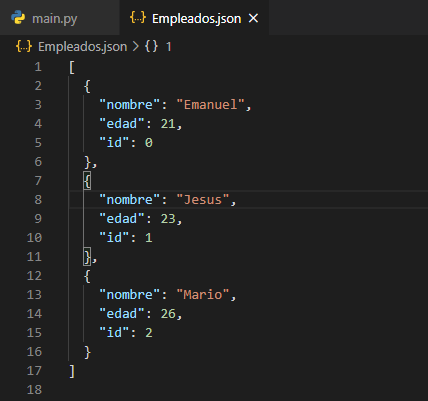
Practica Micro servicios y contenedores

Desarrollo

* Para el desarrollo de este ejercicio decidí utilizar python junto con el web framework flask, para instalar flask utilice el comando pip install flask.
* Posteriormente define lo menajes de entrada y salida para la aplicación RESt utilice un archivo en formato JSON.



* Antes de crear un EndPoint para la aplicación primero tuve que inicializar el servidor

**from** flask **import** Flask, jsonify, request

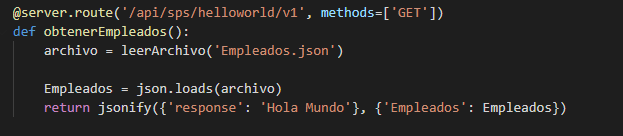
**import** json

server = Flask(\_\_name\_\_)

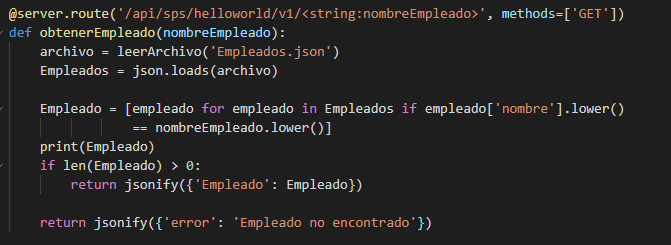
**if** \_\_name\_\_ == '\_\_main\_\_':

    server.run(debug=True, port=8090)

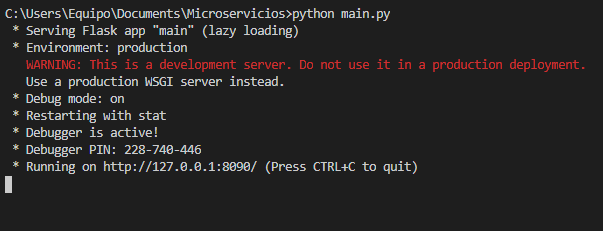
* Ahora para crear el EndPoint le indico al servidor la ruta en donde se mostrara los mensajes dependiendo de lo que se envié, en este EndPoint se mostrara toda la información que se tiene sobre los empleados.



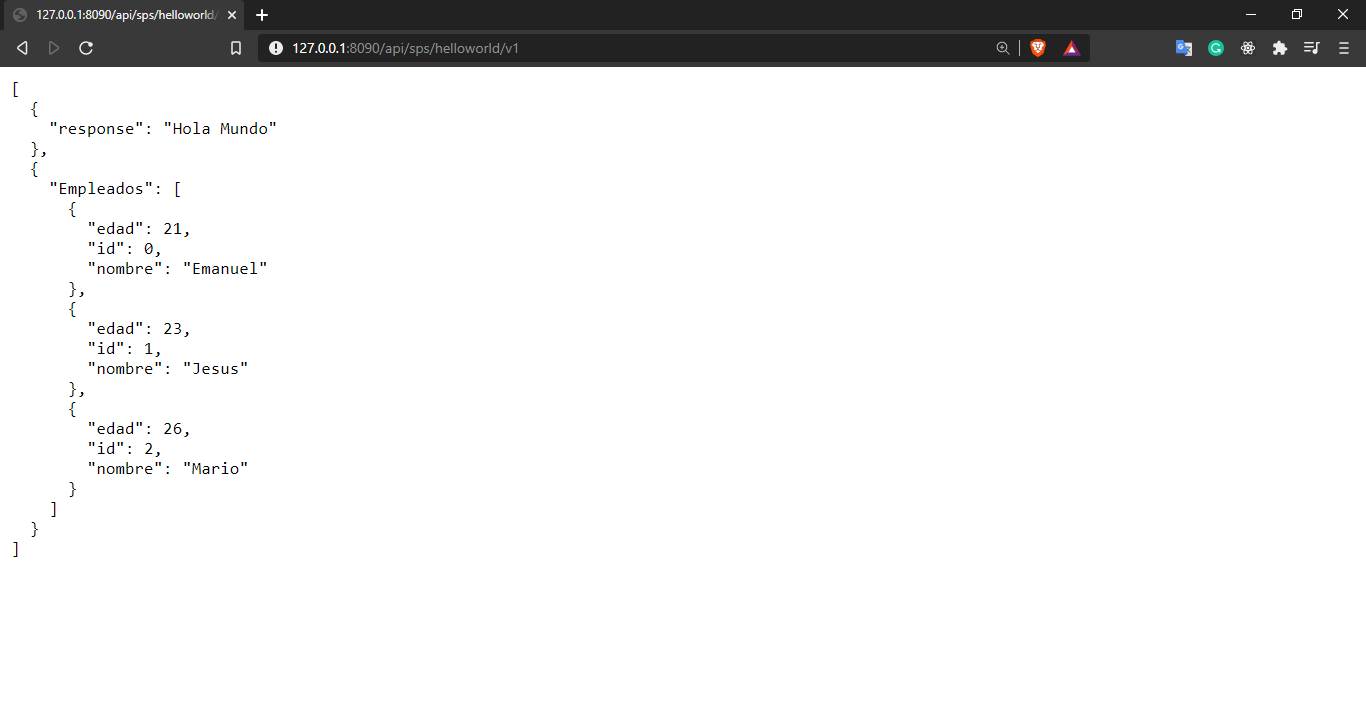
* Por ejemplo en este EndPoint se mostrara información sobre un empleado en específico en dado caso de que no exista tal empleado se mostrara que no existe tal empleado.

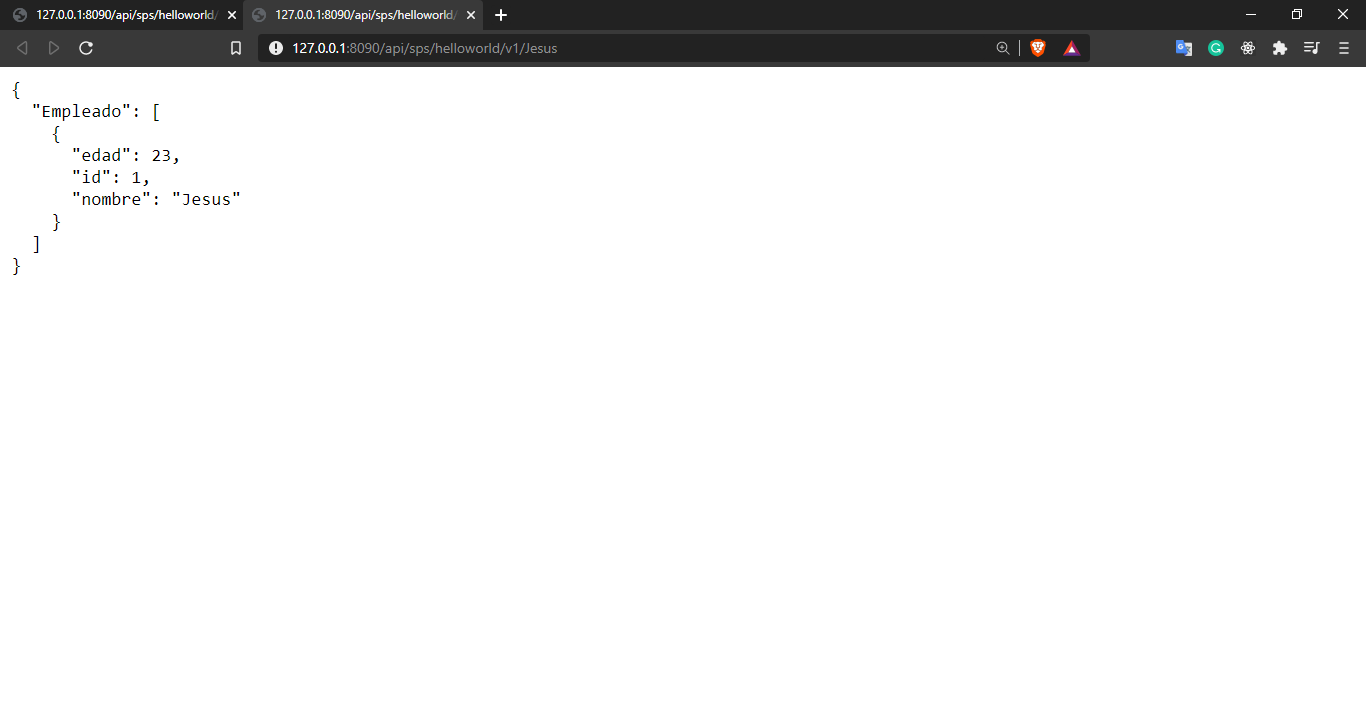


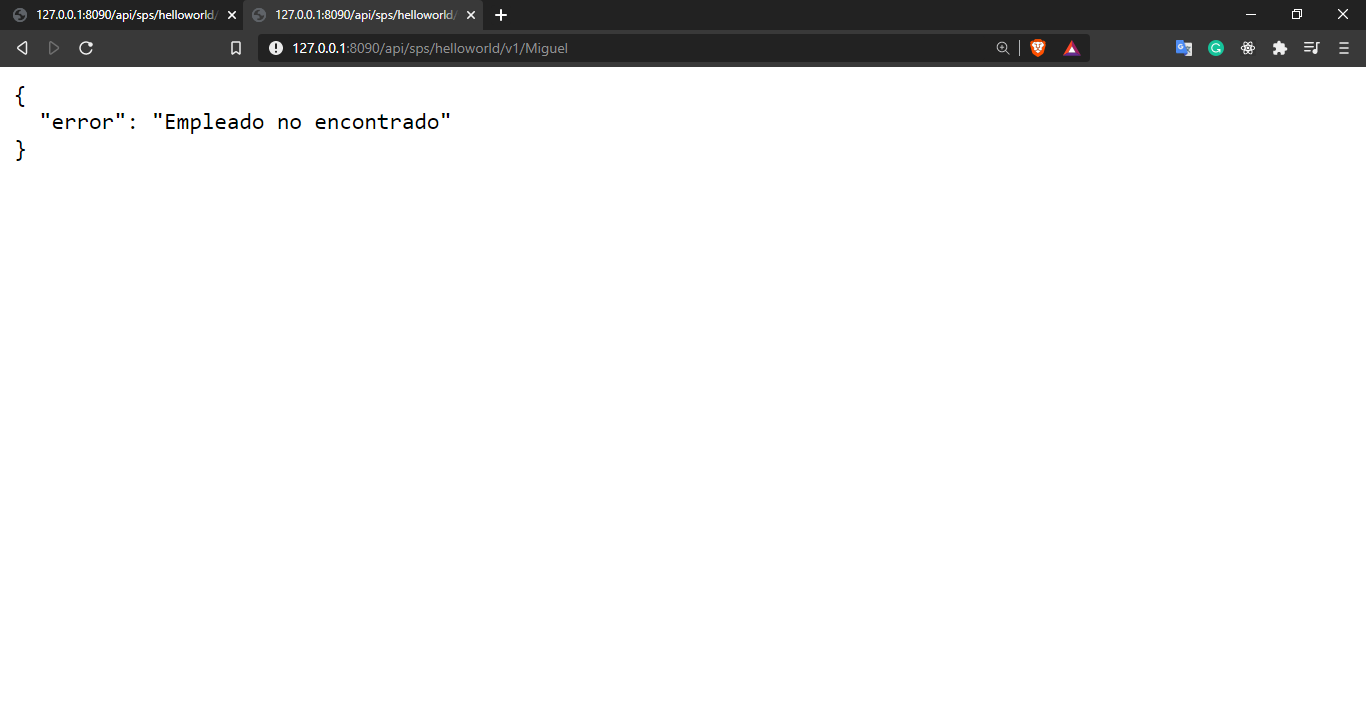
* Para desplegar el API Rest primero en la terminal ejecute el script de python



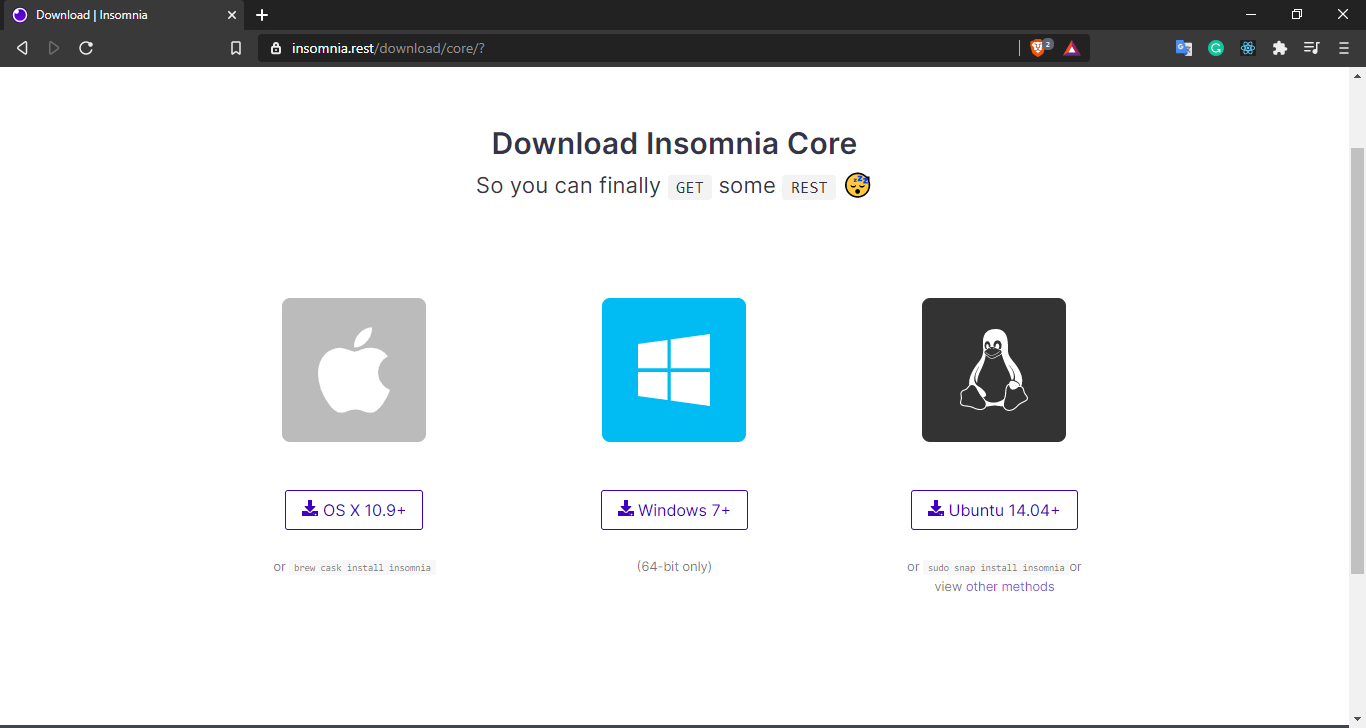
* Posteriormente entre a la dirección indicada por el servidor desde el navegador, entes EndPoint se muestra a todos los empleados.

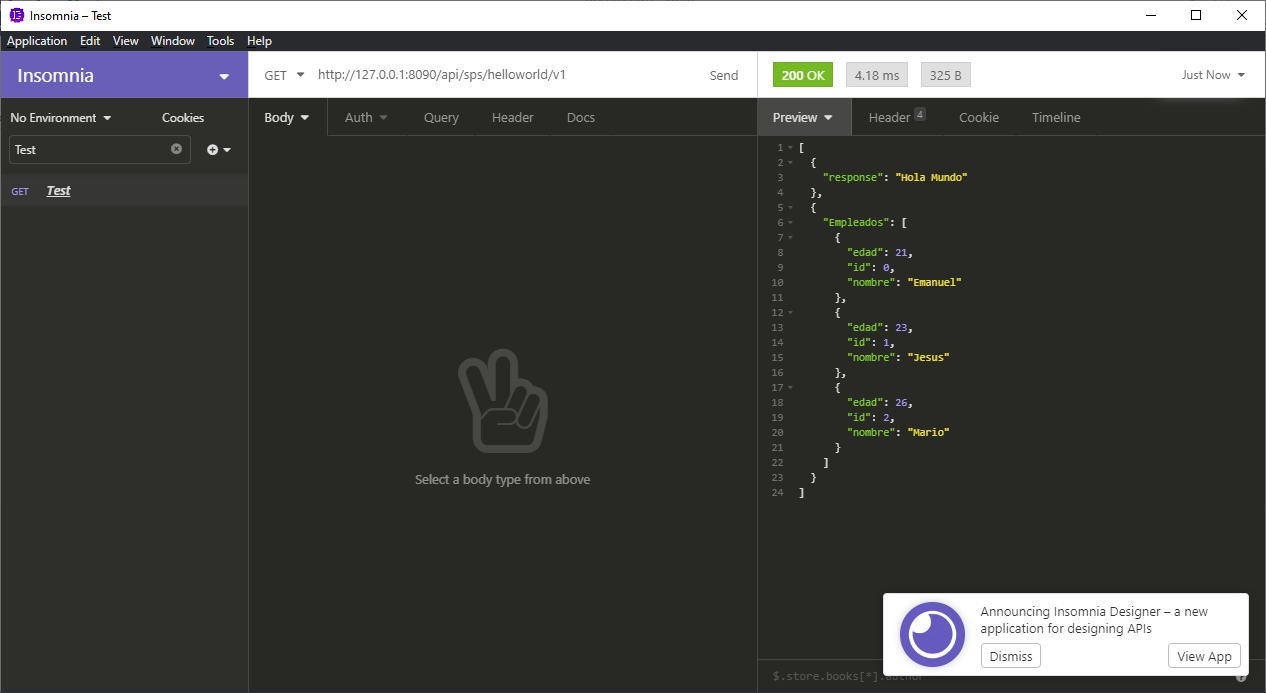


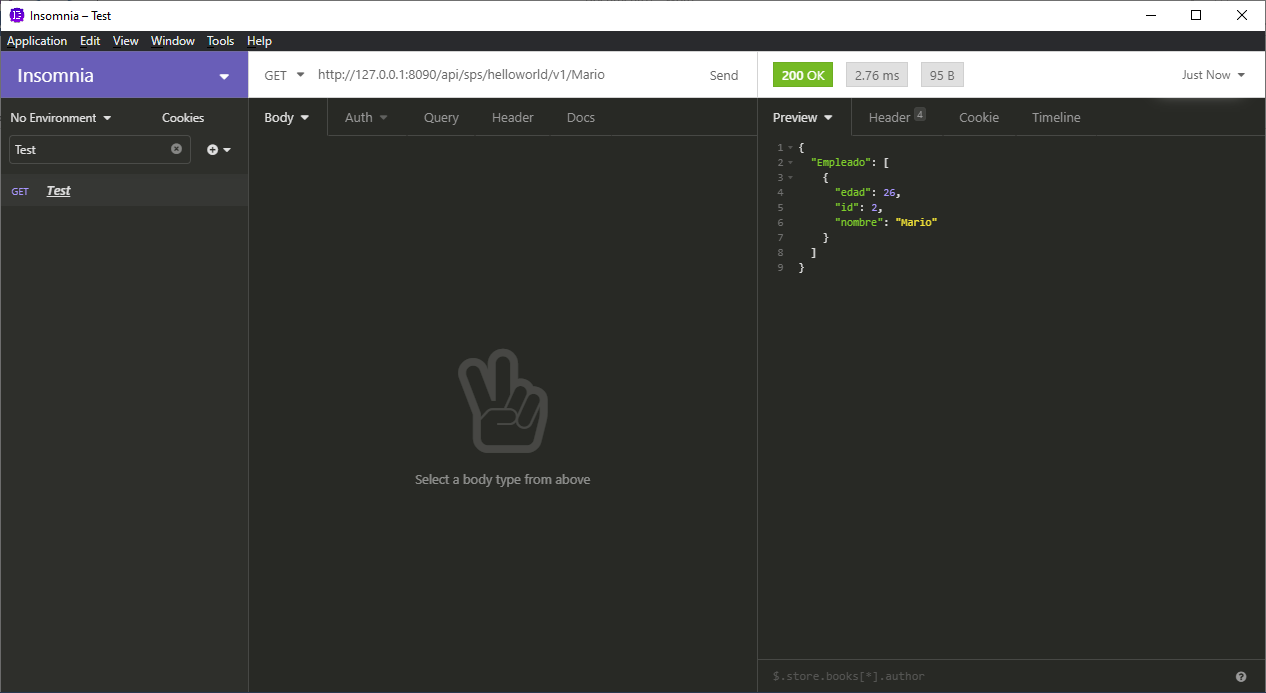
* Para probar los firentes EndPoint desde el navegador cambie la URL a <http://127.0.0.1:8090/api/sps/helloworld/v1/Jesus>
* En caso de que no exista el empleado en el archivo JSON el EndPoint mostrara lo siguiente

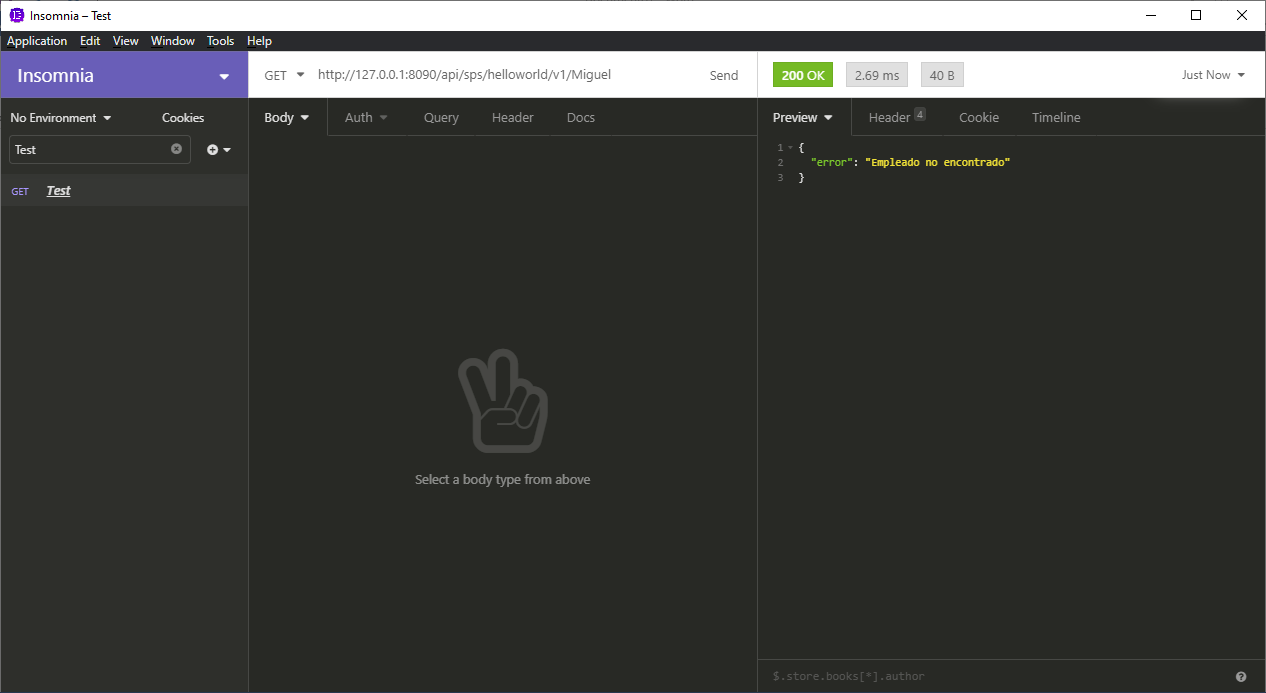


* Para probar la API Rest descarge el cliente REST Insomia Core, en este caso descargue la versión para Windows.

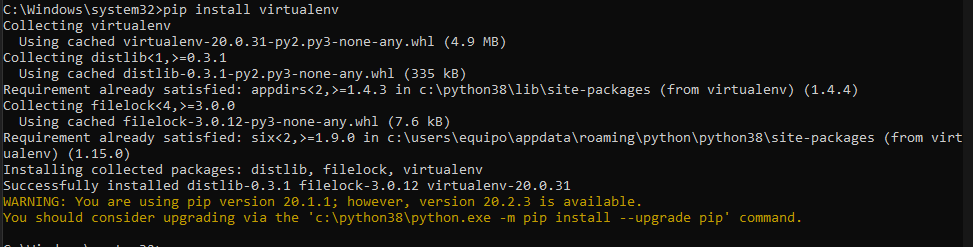


* Una vez instalada creamos un entorno para probar nuestra API Rest, escribirnos la URL y método que queremos usar en este caso usaremos el método GET.
* Para probar el otro EndPoint solo modificamos la URL y nos mostrara la información solicitada.

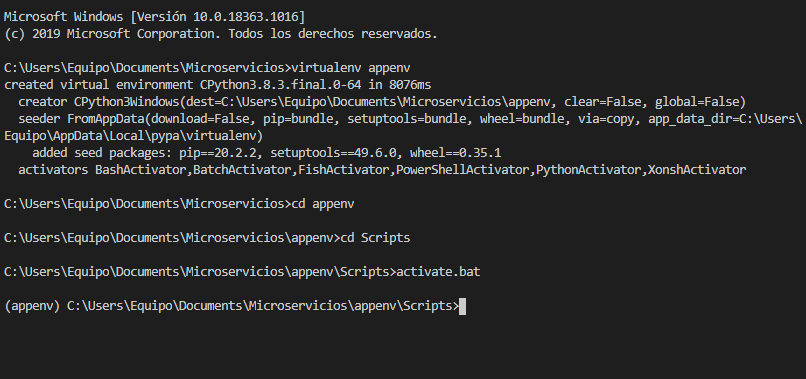




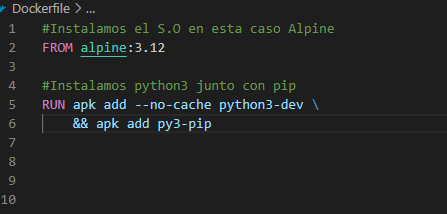
* Ahora procederemos a contenizar la aplicación para ello aislaremos la aplicación utilizando virtual env para instalarlo ejecutamos el siguiente comando pip install virtualenv

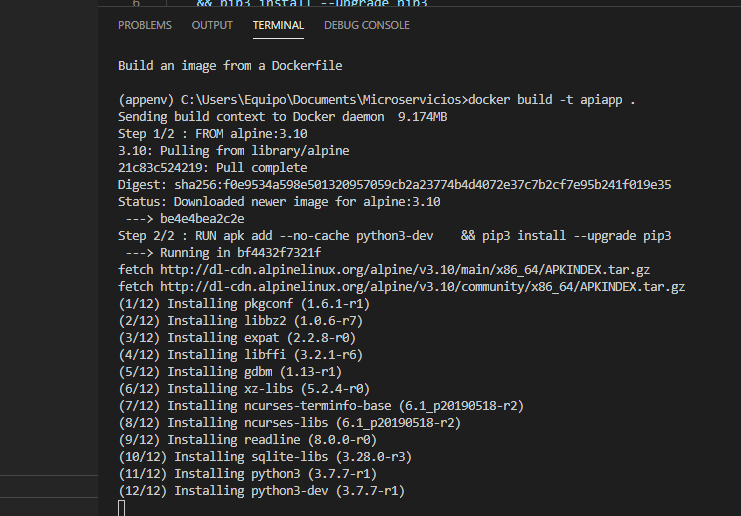


* Posteriormente creamos nuestro entorno virtual y ejecutamos el archivo actívate.bat que se encuentra en la carpeta Scripts

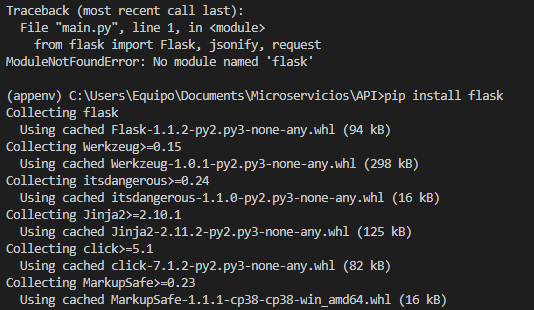


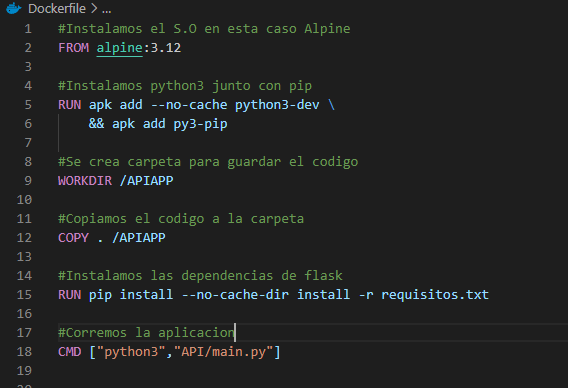
* Instalamos Docker para contenizar nuestra app y para ellos crearemos nuestra imagen en donde le especificaremos que es lo que necesita para poder ejecutar la aplicación.
* En este caso usaremos Alpine que es una versión minimalista Linux al cual le instalaremos python



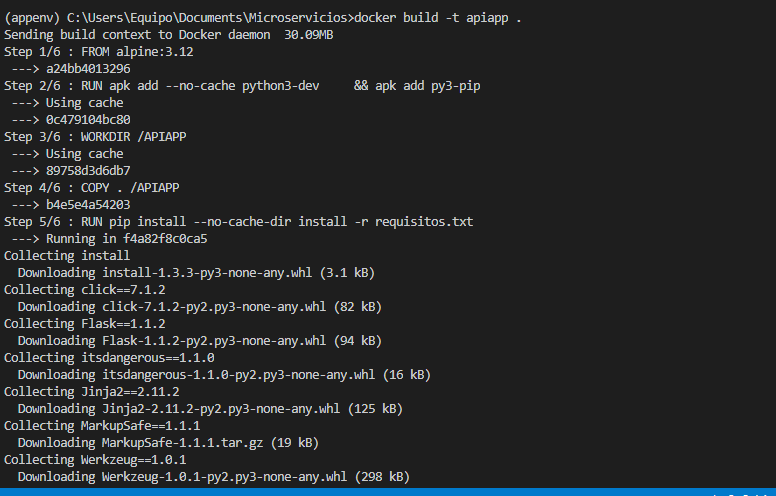
* Crearemos la imagen usando el comando docker build –t apiapp .

IMPORTANTE

Cuando se crea el entorno virtual es importante volver a ejecutar el comando pip install flask ya que de otra manera la API no funcionara.

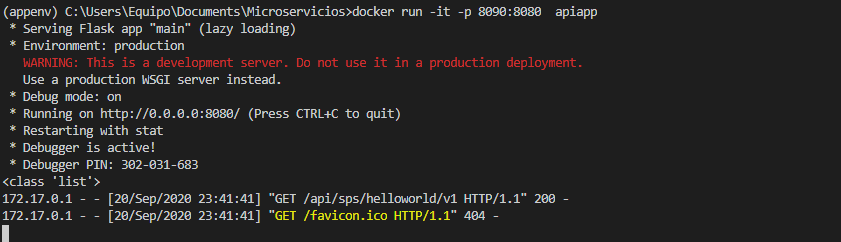
En el archivo docker especificamos lo que se tiene que instalar en el contenedor para que este se ejecute de manera correcta.

Volvemos a ejecutar el comando docker build –t apiapp . Para crear la imagen del contenedor

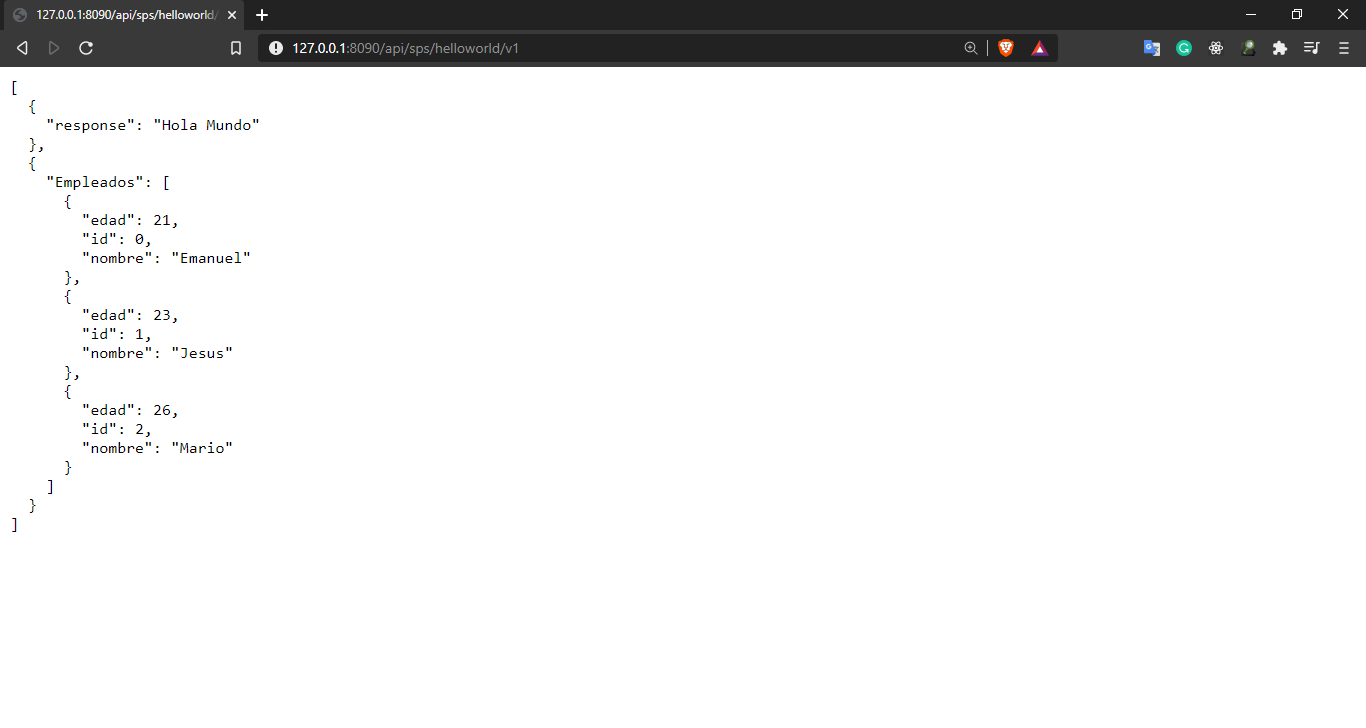


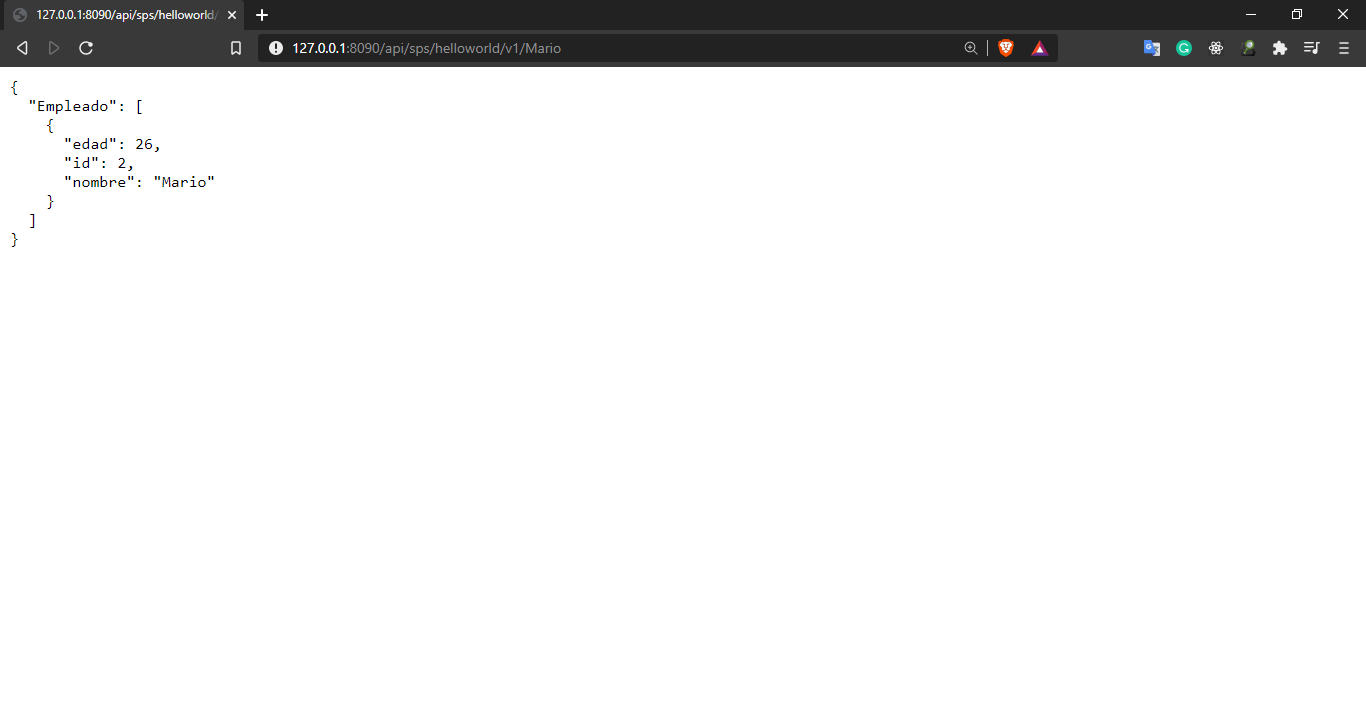
Una vez creado nuestro contenedor lo ejecutamos con el siguiente comando

docker run –it –p 8090:8080 apiapp en este caso se corre en el Puerto 8090 ya que el Puerto 8080 no está disponible

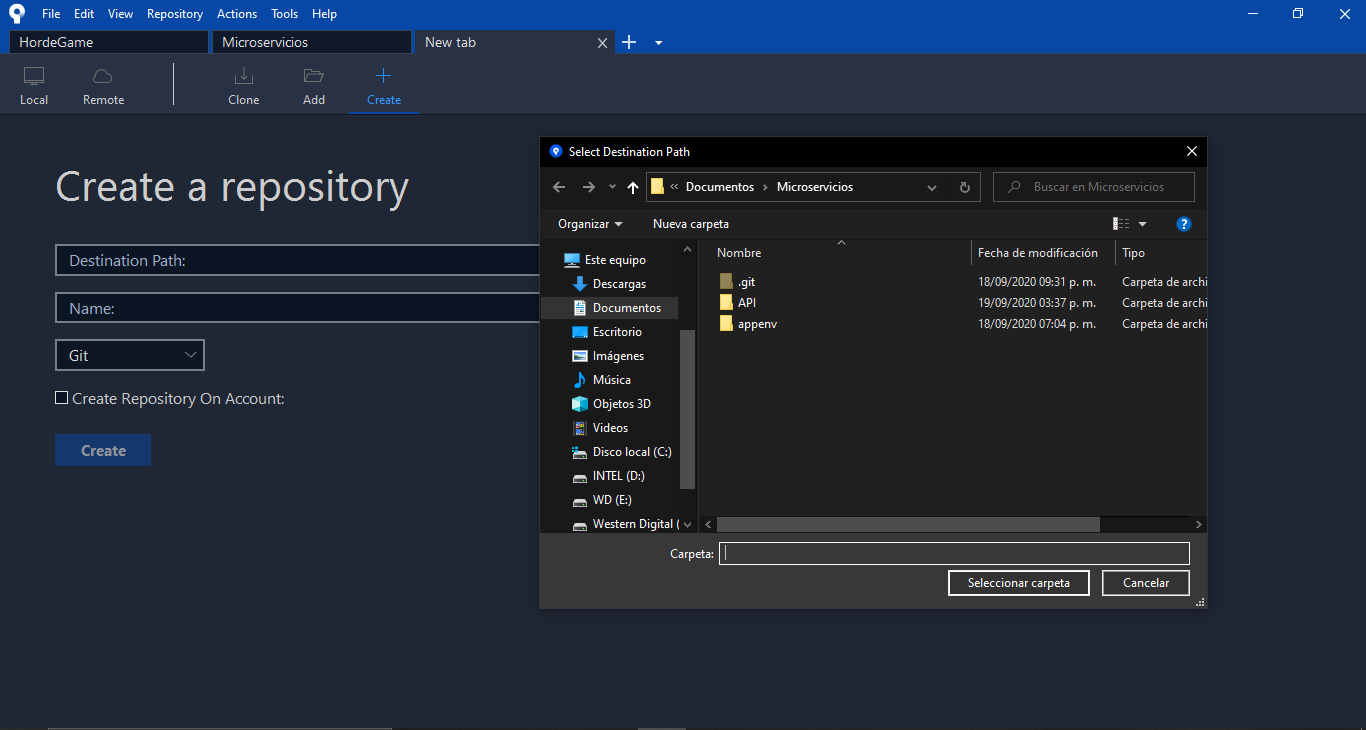


El contenedor se encuentra ejecutando en <http://127.0.0.1:8090/api/sps/helloworld/v1> desde el navegador

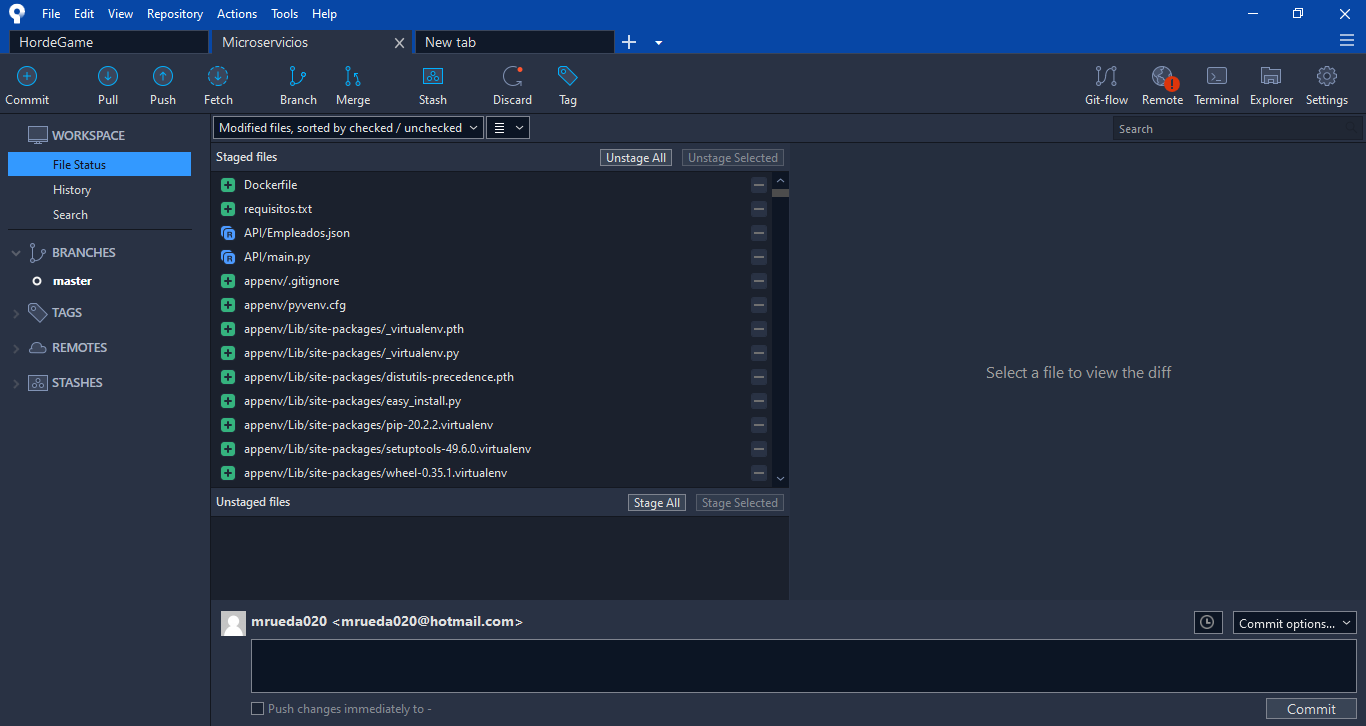




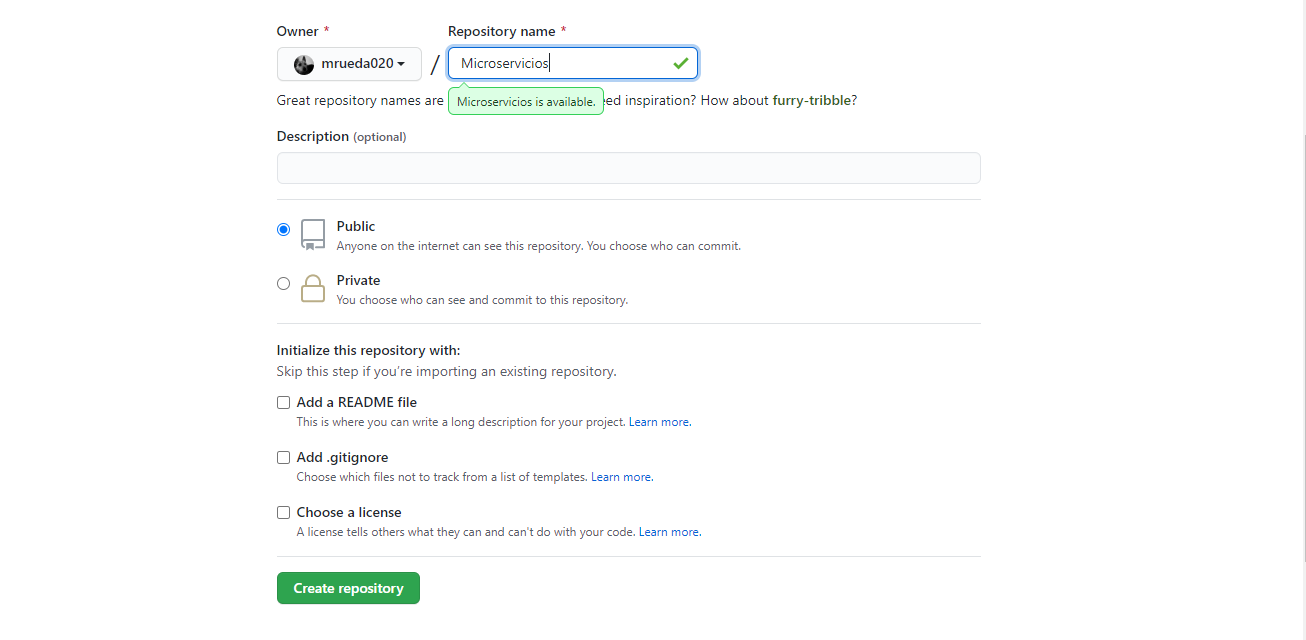
Ahora para subir el programa a un repositorio GIt utilize Source tree como repositorio local, para ello cree un nuevo repositorio en una nueva carpeta llamada microservicios



Una vez creada fui agrando los Scripts necesarios para la practica



Posteriormente cree un repositorio en GitHub



Posteriormente en la terminal ejecute los siguientes comandos para subir el repositorio a GitHub

