

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.

```
Selecciónar Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.18363.1016]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>pip install flask
Requirement already satisfied: flask in c:\python38\lib\site-packages (1.1.2)
Requirement already satisfied: Jinja2>=2.10.1 in c:\python38\lib\site-packages (from flask) (2.11.2)
Requirement already satisfied: itsdangerous>=0.24 in c:\python38\lib\site-packages (from flask) (1.1.0)
Requirement already satisfied: click>=5.1 in c:\python38\lib\site-packages (from flask) (7.1.2)
Requirement already satisfied: Werkzeug>=0.15 in c:\python38\lib\site-packages (from flask) (1.0.1)
Requirement already satisfied: MarkupSafe>=0.23 in c:\python38\lib\site-packages (from Jinja2>=2.10.1->flask) (1.1.1)
WARNING: You are using pip version 20.1.1; however, version 20.2.3 is available.
You should consider upgrading via the 'c:\python38\python.exe -m pip install --upgrade pip' command.

C:\Windows\system32>
```

- Posteriormente define lo mensajes de entrada y salida para la aplicación RSt utilice un archivo en formato JSON.

```
main.py  Empleados.json X
Empleados.json > {} 1
1  [
2      {
3          "nombre": "Emanuel",
4          "edad": 21,
5          "id": 0
6      },
7      {
8          "nombre": "Jesus",
9          "edad": 23,
10         "id": 1
11     },
12     {
13         "nombre": "Mario",
14         "edad": 26,
15         "id": 2
16     }
17 ]
18
```

- 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.

```
@server.route('/api/sps/helloworld/v1', methods=['GET'])
def obtenerEmpleados():
    archivo = leerArchivo('Empleados.json')

    Empleados = json.loads(archivo)
    return jsonify({'response': 'Hola Mundo'}, {'Empleados': 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.

```
@server.route('/api/sps/helloworld/v1/<string:nombreEmpleado>', methods=['GET'])
def obtenerEmpleado(nombreEmpleado):
    archivo = leerArchivo('Empleados.json')
    Empleados = json.loads(archivo)

    Empleado = [empleado for empleado in Empleados if empleado['nombre'].lower()
                 == nombreEmpleado.lower()]
    print(Empleado)
    if len(Empleado) > 0:
        return jsonify({'Empleado': Empleado})

    return jsonify({'error': 'Empleado no encontrado'})
```

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

```
C:\Users\Equipo\Documents\Microservicios>python main.py
* Serving Flask app "main" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 228-740-446
* Running on http://127.0.0.1:8090/ (Press CTRL+C to quit)
```

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

```
127.0.0.1:8090/api/sps/helloworld/ v1
[
  {
    "response": "Hola Mundo"
  },
  {
    "Empleados": [
      {
        "edad": 21,
        "id": 0,
        "nombre": "Emanuel"
      },
      {
        "edad": 23,
        "id": 1,
        "nombre": "Jesus"
      },
      {
        "edad": 26,
        "id": 2,
        "nombre": "Mario"
      }
    ]
  }
]
```

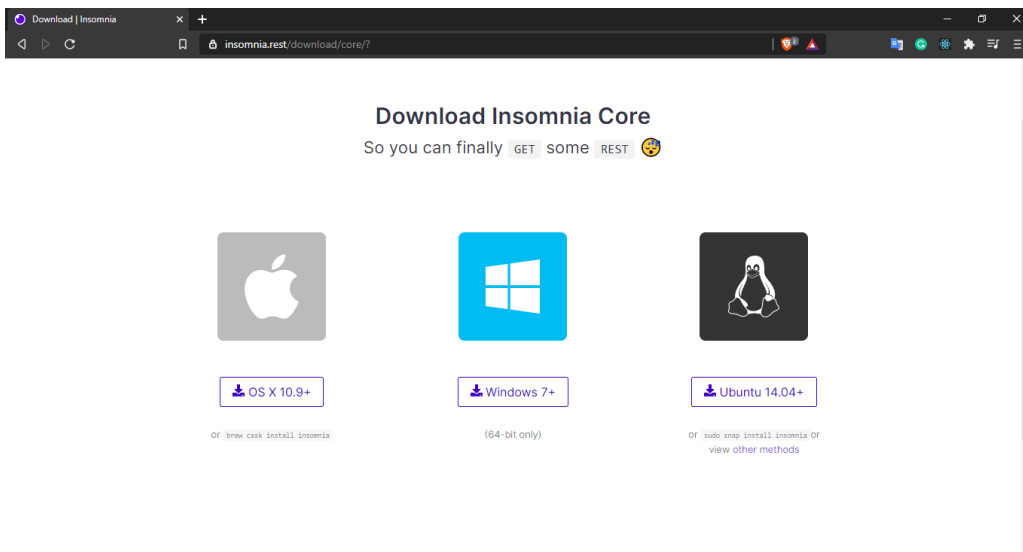
- Para probar los firentes EndPoint desde el navegador cambie la URL a <http://127.0.0.1:8090/api/sps/helloworld/v1/Jesus>

```
127.0.0.1:8090/api/sps/helloworld/ v1
127.0.0.1:8090/api/sps/helloworld/ v1/Jesus
{
  "Empleado": [
    {
      "edad": 23,
      "id": 1,
      "nombre": "Jesus"
    }
  ]
}
```

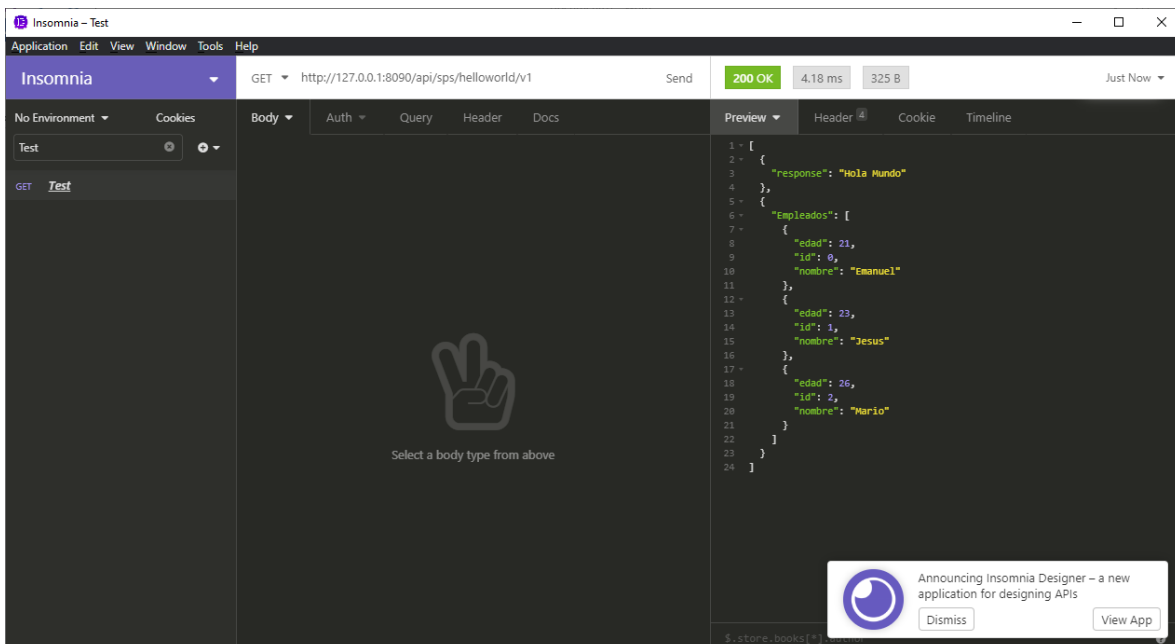
- En caso de que no exista el empleado en el archivo JSON el EndPoint mostrara lo siguiente

```
{
  "error": "Empleado no encontrado"
}
```

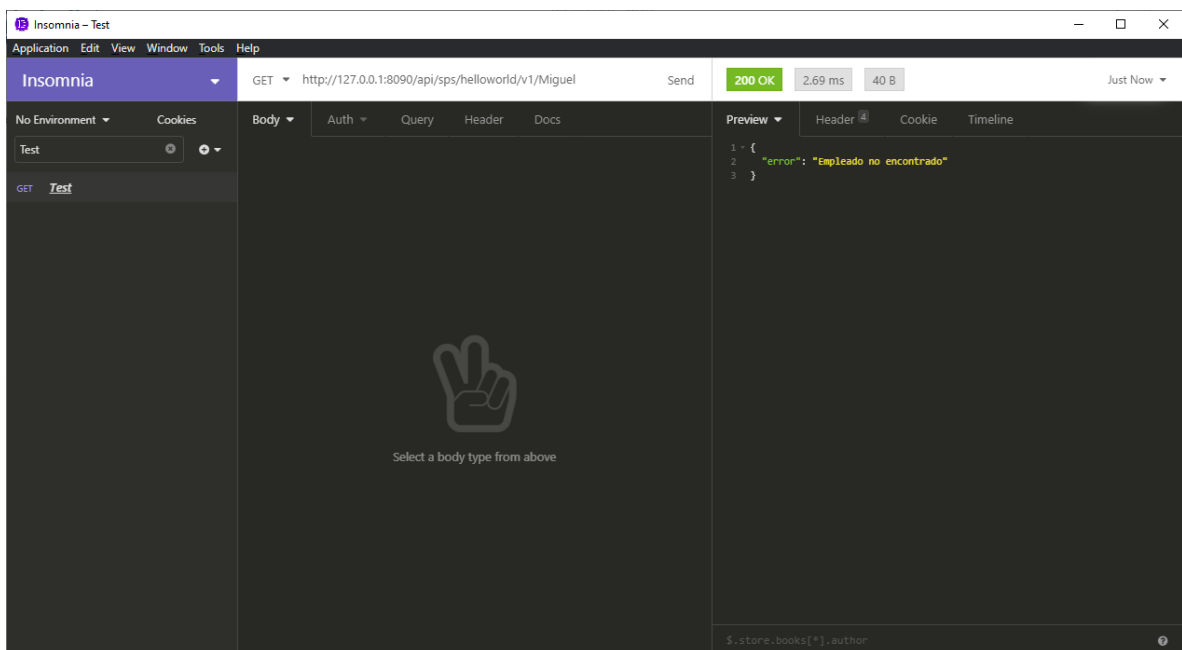
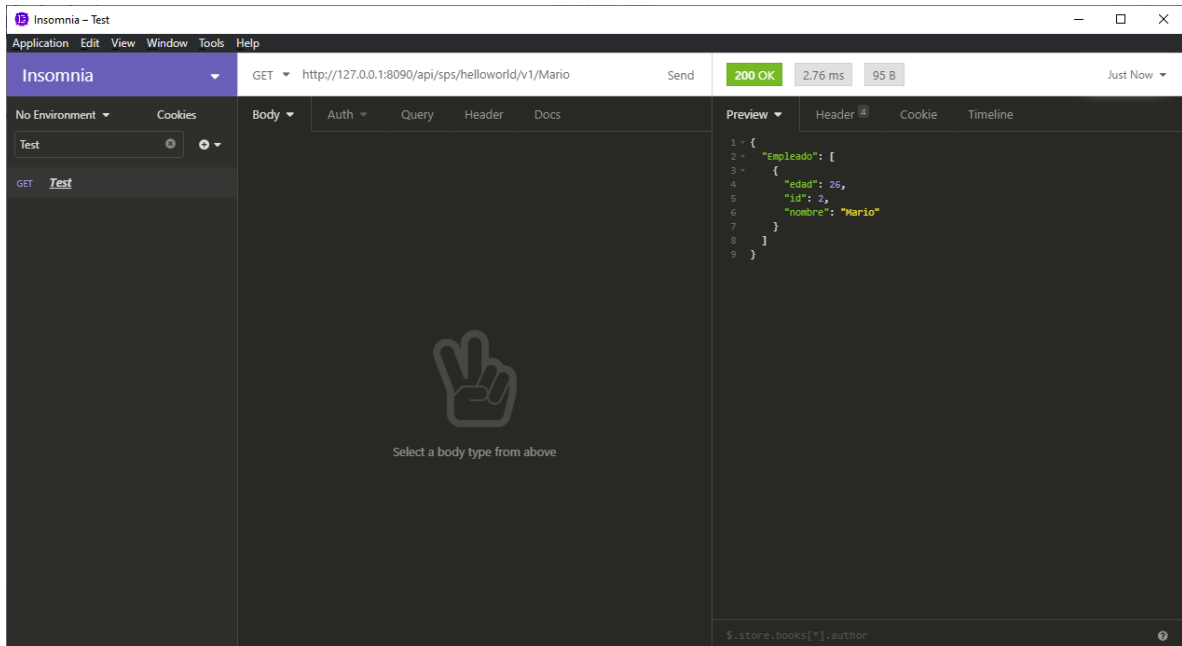
- Para probar la API Rest descargue el cliente REST Insomnia Core, en este caso descargue la versión para Windows.



- Una vez instalada creamos un entorno para probar nuestra API Rest, escribimos 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 contenerizar la aplicación para ello aislaremos la aplicación utilizando virtual env para instalarlo ejecutamos el siguiente comando pip install virtualenv

```
C:\Windows\system32>pip install virtualenv
Collecting virtualenv
  Using cached virtualenv-20.0.31-py2.py3-none-any.whl (4.9 MB)
Collecting distlib<1,>=0.3.1
  Using cached distlib-0.3.1-py2.py3-none-any.whl (335 kB)
Requirement already satisfied: appdirs<2,>=1.4.3 in c:\python38\lib\site-packages (from virtualenv) (1.4.4)
Collecting filelock<4,>=3.0.0
  Using cached filelock-3.0.12-py3-none-any.whl (7.6 kB)
Requirement already satisfied: six<2,>=1.9.0 in c:\users\equipo\appdata\roaming\python\python38\site-packages (from virtualenv) (1.15.0)
Installing collected packages: distlib, filelock, virtualenv
Successfully installed distlib-0.3.1 filelock-3.0.12 virtualenv-20.0.31
WARNING: You are using pip version 20.1.1; however, version 20.2.3 is available.
You should consider upgrading via the 'c:\python38\python.exe -m pip install --upgrade pip' command.
```

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

```
Microsoft Windows [Versión 10.0.18363.1016]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Equipo\Documents\Microservicios>virtualenv appenv
created virtual environment CPython3.8.3.final.0-64 in 8076ms
  creator CPython3Windows(dest=C:\Users\Equipo\Documents\Microservicios\appenv, clear=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\Equipo\AppData\Local\pypa\virtualenv)
    added seed packages: pip==20.2.2, setuptools==49.6.0, wheel==0.35.1
    activators BashActivator,BatchActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator

C:\Users\Equipo\Documents\Microservicios>cd appenv

C:\Users\Equipo\Documents\Microservicios\appenv>cd Scripts

C:\Users\Equipo\Documents\Microservicios\appenv\Scripts>activate.bat

(appenv) C:\Users\Equipo\Documents\Microservicios\appenv\Scripts>
```

- Instalamos Docker para contenerizar 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

```
Dockerfile > ...
1  #Instalamos el S.O en esta caso Alpine
2  FROM alpine:3.12
3
4  #Instalamos python3 junto con pip
5  RUN apk add --no-cache python3-dev \
6  && apk add py3-pip
7
```

- Crearemos la imagen usando el comando `docker build -t apiapp .`

```

6      && pip3 install --upgrade pip3

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Build an image from a Dockerfile

(appenv) C:\Users\Equipo\Documents\Microservicios>docker build -t apiapp .
Sending build context to Docker daemon  9.174MB
Step 1/2 : FROM alpine:3.10
3.10: Pulling from library/alpine
21c83c524219: Pull complete
Digest: sha256:f0e9534a598e501320957059cb2a23774b4d4072e37c7b2cf7e95b241f019e35
Status: Downloaded newer image for alpine:3.10
--> be4e4bea2c2e
Step 2/2 : RUN apk add --no-cache python3-dev    && pip3 install --upgrade pip3
--> Running in bf4432f7321f
fetch http://dl-cdn.alpinelinux.org/alpine/v3.10/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.10/community/x86_64/APKINDEX.tar.gz
(1/12) Installing pkgconf (1.6.1-r1)
(2/12) Installing libbz2 (1.0.6-r7)
(3/12) Installing expat (2.2.8-r0)
(4/12) Installing libffi (3.2.1-r6)
(5/12) Installing gdbm (1.13-r1)
(6/12) Installing xz-libs (5.2.4-r0)
(7/12) Installing ncurses-terminfo-base (6.1_p20190518-r2)
(8/12) Installing ncurses-libs (6.1_p20190518-r2)
(9/12) Installing readline (8.0.0-r0)
(10/12) Installing sqlite-libs (3.28.0-r3)
(11/12) Installing python3 (3.7.7-r1)
(12/12) Installing python3-dev (3.7.7-r1)

```

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.

```

Traceback (most recent call last):
  File "main.py", line 1, in <module>
    from flask import Flask, jsonify, request
ModuleNotFoundError: No module named 'flask'

(appenv) C:\Users\Equipo\Documents\Microservicios\API>pip install flask
Collecting flask
  Using cached Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Collecting Werkzeug>=0.15
  Using cached Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
Collecting itsdangerous>=0.24
  Using cached itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting Jinja2>=2.10.1
  Using cached Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
Collecting click>=5.1
  Using cached click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting MarkupSafe>=0.23
  Using cached MarkupSafe-1.1.1-cp38-cp38-win_amd64.whl (16 kB)

```

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

```
Dockerfile > ...
1  #Instalamos el S.O en esta caso Alpine
2  FROM alpine:3.12
3
4  #Instalamos python3 junto con pip
5  RUN apk add --no-cache python3-dev \
6      && apk add py3-pip
7
8  #Se crea carpeta para guardar el codigo
9  WORKDIR /APIAPP
10
11 #Copiamos el codigo a la carpeta
12 COPY . /APIAPP
13
14 #Instalamos las dependencias de flask
15 RUN pip install --no-cache-dir install -r requisitos.txt
16
17 #Corremos la aplicacion
18 CMD ["python3","API/main.py"]
19
20
```

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

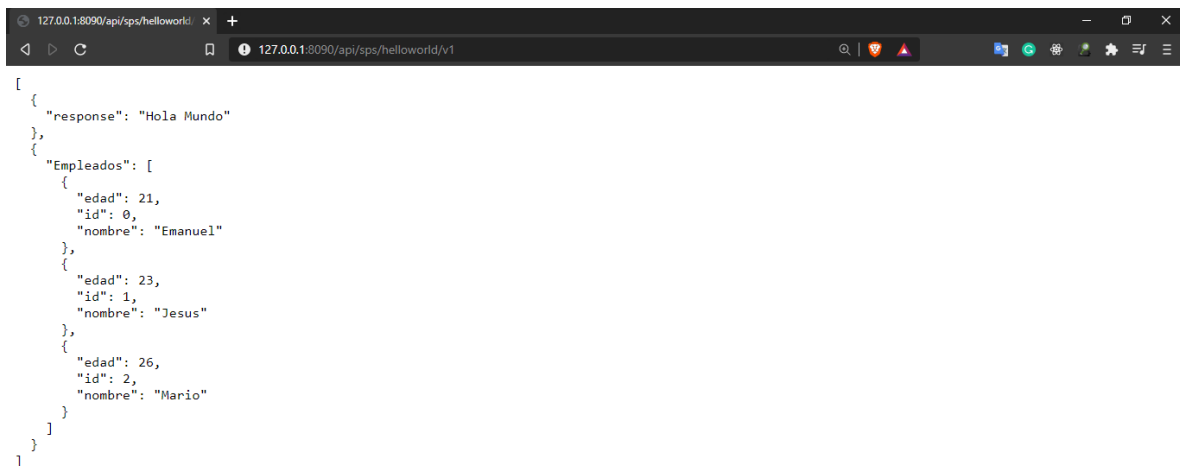
```
(appenv) C:\Users\Equipo\Documents\Microservicios>docker build -t apiapp .
Sending build context to Docker daemon 30.09MB
Step 1/6 : FROM alpine:3.12
--> a24bb4013296
Step 2/6 : RUN apk add --no-cache python3-dev && apk add py3-pip
--> Using cache
--> 0c479104bc80
Step 3/6 : WORKDIR /APIAPP
--> Using cache
--> 89758d3d6db7
Step 4/6 : COPY . /APIAPP
--> b4e5e4a54203
Step 5/6 : RUN pip install --no-cache-dir install -r requisitos.txt
--> Running in f4a82f8c0ca5
Collecting install
  Downloading install-1.3.3-py3-none-any.whl (3.1 kB)
Collecting click==7.1.2
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting Flask==1.1.2
  Downloading Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Collecting itsdangerous==1.1.0
  Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting Jinja2==2.11.2
  Downloading Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
Collecting MarkupSafe==1.1.1
  Downloading MarkupSafe-1.1.1.tar.gz (19 kB)
Collecting Werkzeug==1.0.1
  Downloading Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
```


Una vez creado nuestro contenedor lo ejecutamos con el siguiente comando

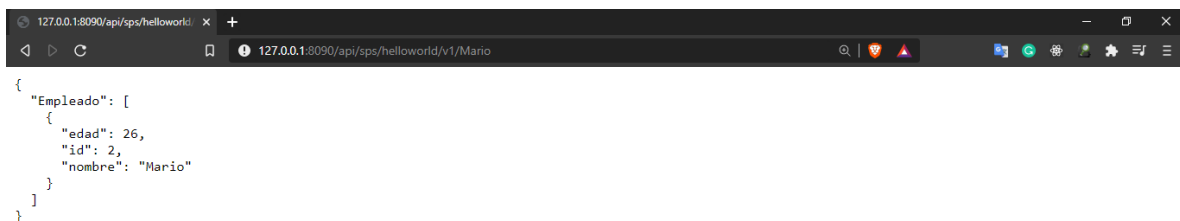
`docker run -it -p 8090:8080 apiapp` en este caso se corre en el Puerto 8090 ya que el Puerto 8080 no está disponible

```
(appenv) C:\Users\Equipo\Documents\Microservicios>docker run -it -p 8090:8080 apiapp
* Serving Flask app "main" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 302-031-683
<class 'list'>
172.17.0.1 - - [20/Sep/2020 23:41:41] "GET /api/sps/helloworld/v1 HTTP/1.1" 200 -
172.17.0.1 - - [20/Sep/2020 23:41:41] "GET /favicon.ico HTTP/1.1" 404 -
```

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

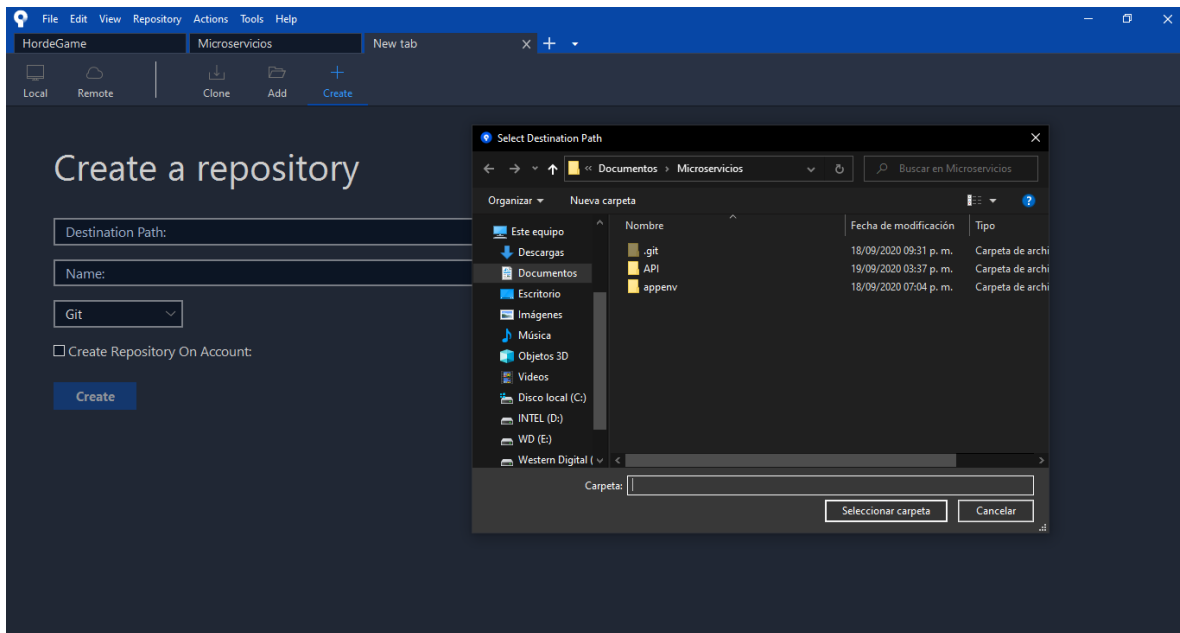


```
[
  {
    "response": "Hola Mundo"
  },
  {
    "Empleados": [
      {
        "edad": 21,
        "id": 0,
        "nombre": "Emanuel"
      },
      {
        "edad": 23,
        "id": 1,
        "nombre": "Jesus"
      },
      {
        "edad": 26,
        "id": 2,
        "nombre": "Mario"
      }
    ]
  }
]
```

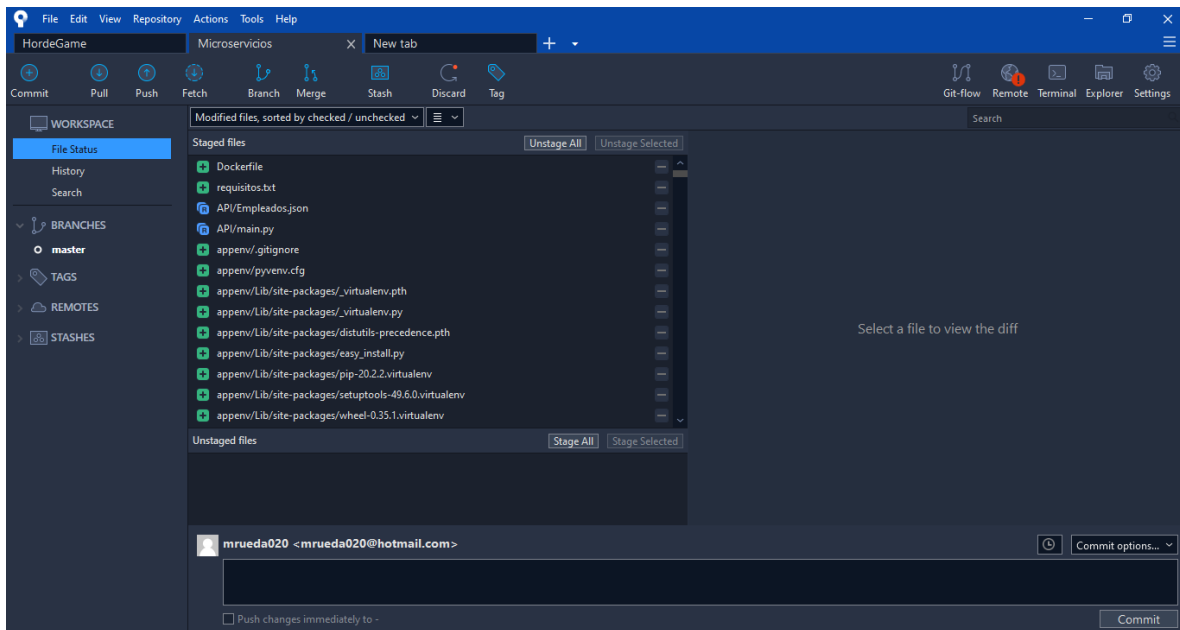


```
{
  "Empleado": [
    {
      "edad": 26,
      "id": 2,
      "nombre": "Mario"
    }
  ]
}
```

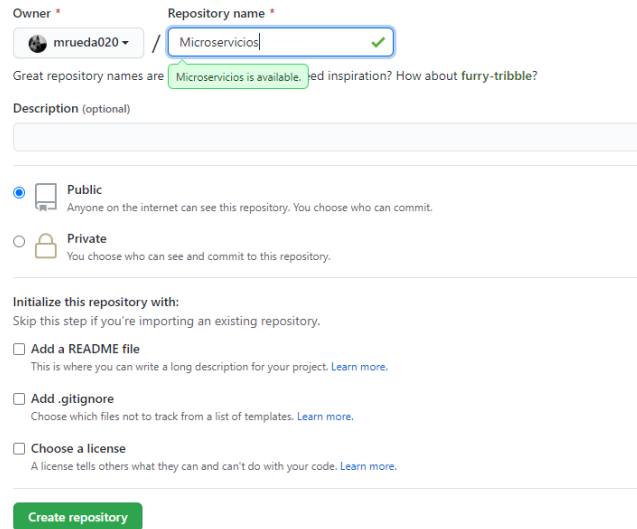
Ahora para subir el programa a un repositorio Git utilice Source tree como repositorio local, para ello cree un nuevo repositorio en una nueva carpeta llamada microservicios



Una vez creada fui agrando los Scripts necesarios para la practica



Posteriormente cree un repositorio en GitHub



The screenshot shows the GitHub repository creation interface. At the top, the 'Owner' is set to 'mrueda020' and the 'Repository name' is 'Microservicios', which is marked as available. Below this, there is a text input for a 'Description (optional)'. The 'Public' option is selected, indicating that anyone on the internet can see the repository. Under the 'Initialize this repository with' section, three checkboxes are present: 'Add a README file', 'Add .gitignore', and 'Choose a license', all of which are currently unchecked. A green 'Create repository' button is located at the bottom of the form.

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

```
MINGW32:/c/Users/Equipo/Documents/Microservicios
Equipo@DESKTOP-97UR9VV MINGW32 ~/Documents/Microservicios (master)
$ git branch -M master

Equipo@DESKTOP-97UR9VV MINGW32 ~/Documents/Microservicios (master)
$ git push -u origin maste
error: src refspec maste does not match any
error: failed to push some refs to 'https://github.com/mrueda020/Microservicios.git'

Equipo@DESKTOP-97UR9VV MINGW32 ~/Documents/Microservicios (master)
$ git push -u origin master
Enumerating objects: 1282, done.
Counting objects: 100% (1282/1282), done.
Delta compression using up to 4 threads
Compressing objects: 100% (1223/1223), done.
Writing objects: 100% (1282/1282), 11.48 MiB | 49.59 MiB/s, done.
Total 1282 (delta 67), reused 0 (delta 0)
remote: Resolving deltas: 100% (67/67), done.
To https://github.com/mrueda020/Microservicios.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Equipo@DESKTOP-97UR9VV MINGW32 ~/Documents/Microservicios (master)
$ |
```

master 1 branch 0 tags

Go to file

Add file

Code



Aplicacion Contenizada

963dd31 11 minutes ago 3 commits



API

Aplicacion Contenizada

11 minutes ago



appenv

Aplicacion Contenizada

11 minutes ago



Dockerfile

Aplicacion Contenizada

11 minutes ago



requisitos.txt

Aplicacion Contenizada

11 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README