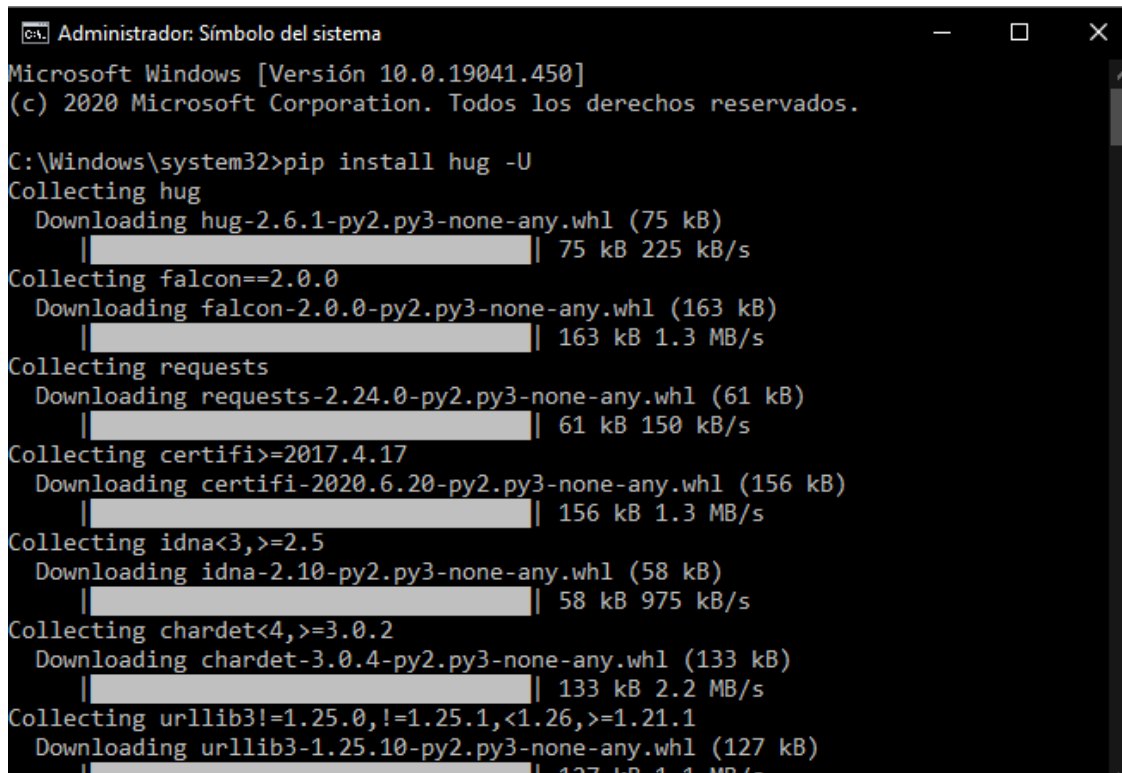


Microservicios y contenedores.

He optado por usar el *framework* Hug de Python para la creación de la API Rest. Primero instalé Hug desde la consola de Windows, bajo el comando “pip install hug -u”.



```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.450]
(c) 2020 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>pip install hug -U
Collecting hug
  Downloading hug-2.6.1-py2.py3-none-any.whl (75 kB)
    | 75 kB 225 kB/s
Collecting falcon==2.0.0
  Downloading falcon-2.0.0-py2.py3-none-any.whl (163 kB)
    | 163 kB 1.3 MB/s
Collecting requests
  Downloading requests-2.24.0-py2.py3-none-any.whl (61 kB)
    | 61 kB 150 kB/s
Collecting certifi>=2017.4.17
  Downloading certifi-2020.6.20-py2.py3-none-any.whl (156 kB)
    | 156 kB 1.3 MB/s
Collecting idna<3,>=2.5
  Downloading idna-2.10-py2.py3-none-any.whl (58 kB)
    | 58 kB 975 kB/s
Collecting chardet<4,>=3.0.2
  Downloading chardet-3.0.4-py2.py3-none-any.whl (133 kB)
    | 133 kB 2.2 MB/s
Collecting urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1
  Downloading urllib3-1.25.10-py2.py3-none-any.whl (127 kB)
    | 127 kB 1.1 MB/s
```

Después cree el método local en el IDLE de Python, @hug.local, indica que será un servicio con ruta local solamente, se creo una función llamada say_hello, la cual saluda al usuario y le dice su año de nacimiento según su edad.

```
#Importamos lo modulos necesarios
import datetime
import hug

#Indica que será de solo acceso local
@hug.local()

#Definimos la función say_hello
def say_hello(name: hug.types.text, age: hug.types.number, hug_timer=3):#hug.typ
    year_of_birth = datetime.datetime.now().year - age
    return {#Decimos hola al usuario y calculamos su año de nacimiento
        'message': "Hola {0}, naciste el año {1}".format(name, year_of_birth),
        'took': float(hug_timer)
    }

if __name__ == '__main__':
    print(say_hello("panchito", 50))
```

Posteriormente cree el método HTTP, el cual funge para exponer la API como un servicio HTTP, se hizo uso en este caso `@hug.get`, existiendo la posibilidad de usar también PUT, POST, etc., la función de GET es solicitar información.

```
import datetime
import hug

@hug.get(examples="name=Jhon Doe&age=30")
@hug.local()
def say_hello(name: hug.types.text, age: hug.types.number, hug_timer=3):
    """Decimos hola al usuario y calculamos su año de nacimiento"""
    year_of_birth = datetime.datetime.now().year - age
    return {
        'message': "Hola {0}, naciste el año {1}".format(name, year_of_birth),
        'took': float(hug_timer)
    }

if __name__ == '__main__':
    print(say_hello("Juanito", 23))
```

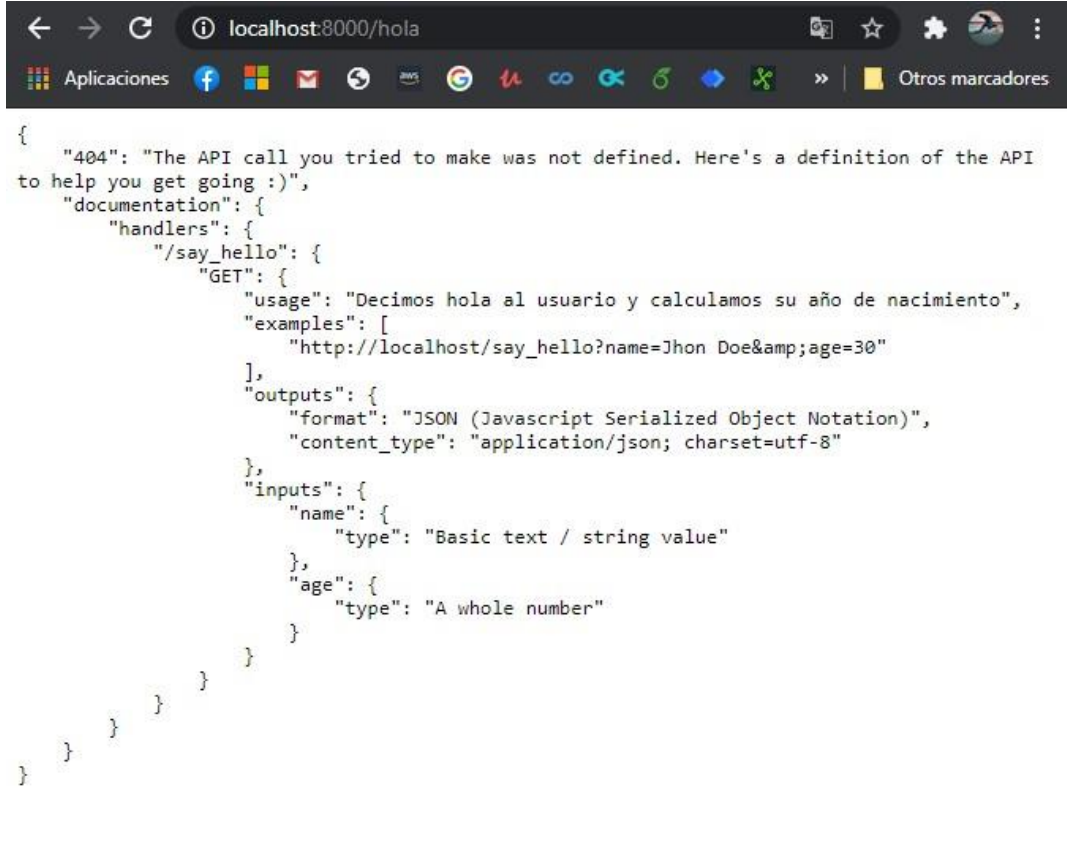
El método HTTP, lo guarde en la dirección `api\sps\helloworld\v1` y ejecute el método desde la consola de Windows.

```
C:\Users\KzyCk\Documents\Python\api\sps\helloworld\v1>hug -f AccesoHTTP.py
```

En este momento la API ya esta expuesta, por medio del puerto 8000.

[illegible]

Visualizando desde el navegador se ve lo siguiente.



```
{
  "404": "The API call you tried to make was not defined. Here's a definition of the API
to help you get going :)",
  "documentation": {
    "handlers": {
      "/say_hello": {
        "GET": {
          "usage": "Decimos hola al usuario y calculamos su año de nacimiento",
          "examples": [
            "http://localhost/say_hello?name=Jhon Doe&age=30"
          ],
          "outputs": {
            "format": "JSON (Javascript Serialized Object Notation)",
            "content_type": "application/json; charset=utf-8"
          },
          "inputs": {
            "name": {
              "type": "Basic text / string value"
            },
            "age": {
              "type": "A whole number"
            }
          }
        }
      }
    }
  }
}
```

Y marcando los valores de nombre y edad, observamos que se despliega en la ventana la siguiente información.



```
{ "message": "Hola pepe perez, naciste el año 1990", "took": 0.0 }
```

Después cree el método CLI, el cual sirve como la interfaz de línea de comando.

```
import datetime
import hug

@hug.cli() #Se usa como decorador
@hug.get(examples="name=Jhon Doe&age=30")
@hug.local()

#Definimos la función say_hello
def say_hello(name: hug.types.text, age: hug.types.number, hug_timer=3): #hug.typ
    year_of_birth = datetime.datetime.now().year - age
    return {#Decimos hola al usuario y calculamos su año de nacimiento
        'message': "Hola {0}, naciste el año {1}".format(name, year_of_birth),
        'took': float(hug_timer)
    }

if __name__ == '__main__':
    print(say_hello("panchito", 50))
```

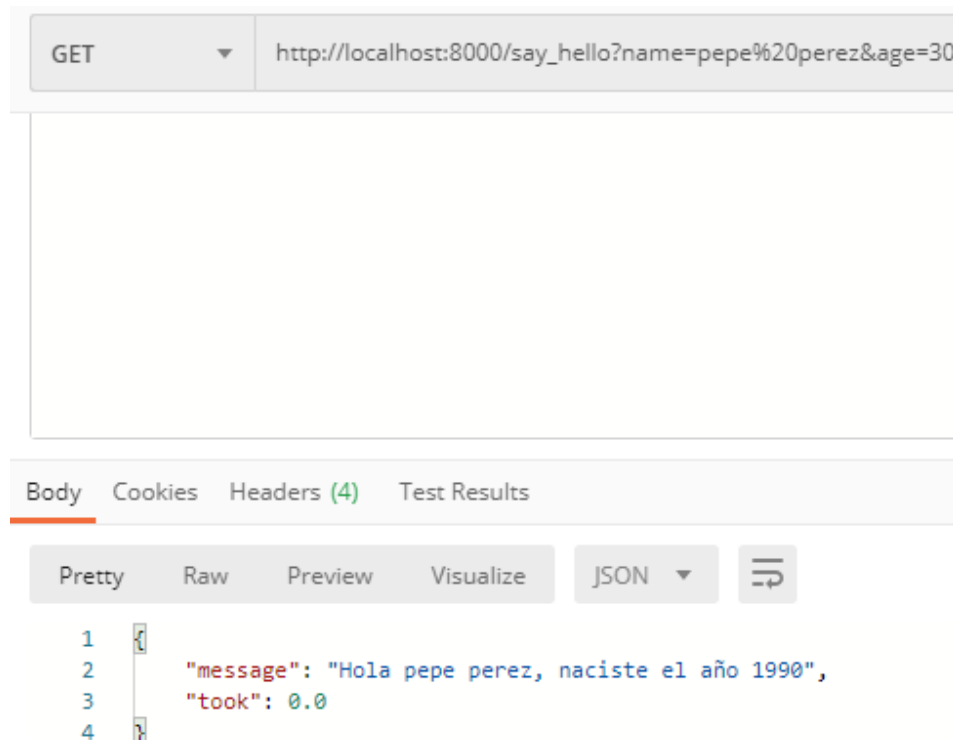
Luego ejecute desde la consola dicho método, donde se muestra la existencia del endpoint “say_hello”.

```
C:\Users\KzyCk\Documents\Python\api\sps\helloworld\v1>hug -f AccesoCli.py -c say_Hello "Panchito" 50
AccesoCli

Available Commands:

- say_hello
```

Después realice pruebas en el cliente Rest de Postman, y haciendo una solicitud tipo GET, obtenemos, lo siguiente.



El siguiente paso fue crear el Dockerfile, apoyándome en la documentación de Docker, lo realice desde un editor de texto, donde especifique el puerto 8090 como el puerto de salida.

```
Dockerfile
1 # Use the official image as a parent image.
2 FROM python:3
3
4 # Set the working directory.
5 WORKDIR /app
6
7 # Copy the file from your host to your current location.
8 COPY . /app
9
10 # Run the command inside your image filesystem.
11 RUN pip install --trusted-host pypi.python.org -r requirements.txt
12
13 # Add metadata to the image to describe which port the container is listening on at runtime.
14 EXPOSE 8090
15
16 # Run the specified command within the container.
17 CMD ["python", "AccesoLocal.py"]
18
19 # Copy the rest of your app's source code from your host to your image filesystem.
20 COPY . .
```

Una vez creado el Dockerfile, se ejecuta “docker build -t saludoyedad”, el cual creará el contenedor y lo nombrará como saludoyedad, a continuación se muestra el proceso de creación.

```
Administrador: Símbolo del sistema
C:\Users\KzyCk\Documents\Python\api\sps\helloworld\v1>docker build -t saludoyedad .
Sending build context to Docker daemon 11.78kB
Step 1/6 : FROM python:3
--> 28a4c88cdbbf
Step 2/6 : WORKDIR /app
--> Using cache
--> 8d75347f6bf0
Step 3/6 : COPY . /app
--> 3bbe5ec6f4b3
Step 4/6 : RUN pip install --trusted-host pypi.python.org -r requirements.txt
--> Running in a656f340a1
Collecting hug
  Downloading hug-2.6.1-py2.py3-none-any.whl (75 kB)
Collecting datetime
  Downloading DateTime-4.3-py2.py3-none-any.whl (60 kB)
Collecting requests
  Downloading requests-2.24.0-py2.py3-none-any.whl (61 kB)
Collecting falcon==2.0.0
  Downloading falcon-2.0.0-py2.py3-none-any.whl (163 kB)
Collecting pytz
  Downloading pytz-2020.1-py2.py3-none-any.whl (510 kB)
Collecting zope.interface
  Downloading zope.interface-5.1.0-cp38-cp38-manylinux2010_x86_64.whl (243 kB)
Collecting chardet<4,>=3.0.2
  Downloading chardet-3.0.4-py2.py3-none-any.whl (133 kB)
Collecting idna<3,>=2.5
  Downloading idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting certifi>=2017.4.17
  Downloading certifi-2020.6.20-py2.py3-none-any.whl (156 kB)
Collecting urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1
  Downloading urllib3-1.25.10-py2.py3-none-any.whl (127 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/site-packages (from zope.interface->datetime->-r requirements.txt (line 2)) (50.3.0)
Installing collected packages: chardet, idna, certifi, urllib3, requests, falcon, hug, pytz, zope.interface, datetime
Successfully installed certifi-2020.6.20 chardet-3.0.4 datetime-4.3 falcon-2.0.0 hug-2.6.1 idna-2.10 pytz-2020.1 requests-2.24.0 urllib3-1.25.10 zope.interface-5.1.0
Removing intermediate container a656f340a1
--> 3749a407190d
```

```
Administrador: Símbolo del sistema
Downloading DateTime-4.3-py2.py3-none-any.whl (60 kB)
Collecting requests
  Downloading requests-2.24.0-py2.py3-none-any.whl (61 kB)
Collecting falcon==2.0.0
  Downloading falcon-2.0.0-py2.py3-none-any.whl (163 kB)
Collecting pytz
  Downloading pytz-2020.1-py2.py3-none-any.whl (510 kB)
Collecting zope.interface
  Downloading zope.interface-5.1.0-cp38-cp38-manylinux2010_x86_64.whl (243 kB)
Collecting chardet<4,>=3.0.2
  Downloading chardet-3.0.4-py2.py3-none-any.whl (133 kB)
Collecting idna<3,>=2.5
  Downloading idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting certifi>=2017.4.17
  Downloading certifi-2020.6.20-py2.py3-none-any.whl (156 kB)
Collecting urllib3!=1.25.0,!>=1.25.1,<1.26,>=1.21.1
  Downloading urllib3-1.25.10-py2.py3-none-any.whl (127 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/site-packages (from zope.interface->datetime->-r requirements.txt (line 2)) (50.3.0)
Installing collected packages: chardet, idna, certifi, urllib3, requests, falcon, hug, pytz, zope.interface, datetime
Successfully installed certifi-2020.6.20 chardet-3.0.4 datetime-4.3 falcon-2.0.0 hug-2.6.1 idna-2.10 pytz-2020.1 requests-2.24.0 urllib3-1.25.10 zope
.interface-5.1.0
Removing intermediate container a656fbc340a1
--> 3749a407190d
Step 5/6 : EXPOSE 8090
--> Running in 90d287ce7fa4
Removing intermediate container 90d287ce7fa4
--> 2d458af4cf48
Step 6/6 : COPY . .
--> b9634e85cfbc
Successfully built b9634e85cfbc
Successfully tagged saludayedad:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context wi
ll have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.

C:\Users\KzyCk\Documents\Python\api\sps\helloworld\v1>
```

Verifique con el comando “docker image ls” los contenedores creados, y se aprecia que el contenedor de hasta arriba es saludoyedad.

```
Administrador: Símbolo del sistema

C:\Users\KzyCk\Documents\Python\api\sps\helloworld\v1>docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
saludayedad          latest             af74a88b7c6a       2 minutes ago      896MB
<none>               <none>            b9634e85cfbc       35 minutes ago     896MB
<none>               <none>            5d9f7dfde3aa       41 minutes ago     882MB
<none>               <none>            724d520efa24       47 minutes ago     882MB
<none>               <none>            574d5f0d476d       49 minutes ago     882MB
<none>               <none>            f3bfd8eb0b9c       50 minutes ago     882MB
<none>               <none>            fb2d0404b349       55 minutes ago     882MB
<none>               <none>            9356b6bdcc59       About an hour ago   882MB
<none>               <none>            9fa6b519278a       About an hour ago   882MB
python               3                 28a4c88cdbbf       25 hours ago       882MB
hello-world          latest            bf756fb1ae65       8 months ago       13.3kB

C:\Users\KzyCk\Documents\Python\api\sps\helloworld\v1>
```

Con el comando Docker ps -a visualizamos también los contenedores existentes pero con un despliegue de información más amplio.

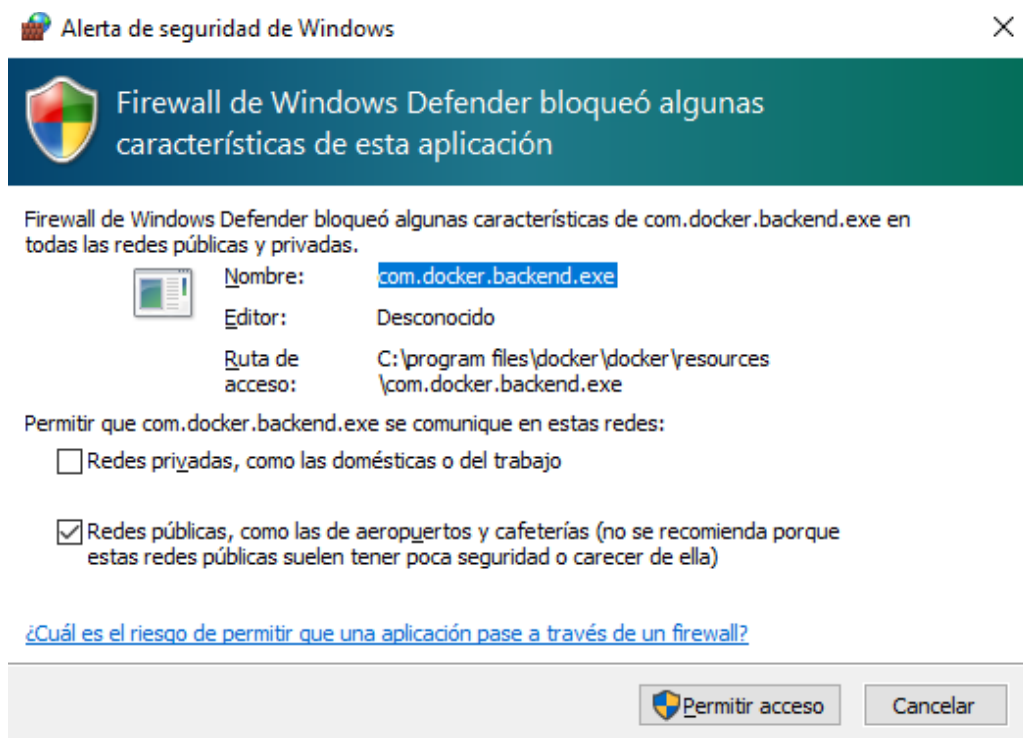
```
C:\Users\KzyCk\Documents\Python\api\sps\helloworld\v1>docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f03f53ef1676	934e3847c0e0	"/bin/sh -c 'pip ins..."	6 minutes ago	Exited (1) 3 minutes ago		festiv
e_noyce						
9da291ec153d	5d9f7dfde3aa	"/bin/sh -c 'pip ins..."	42 minutes ago	Exited (1) 42 minutes ago		inspir
ing_easley						
695a4552f0b7	724d520efa24	"/bin/sh -c 'npm ins..."	48 minutes ago	Exited (127) 48 minutes ago		quizzi
cal_noether						
b41fe94d09fe	574d5f0d476d	"/bin/sh -c 'npm ins..."	50 minutes ago	Exited (127) 50 minutes ago		nervou
s_diffie						
a7e072191cbe	fb2d0404b349	"npm start"	55 minutes ago	Created	0.0.0.0:4000->8090/tcp	pricel
ess_buck						
d839ee5cddb7	9356b6bdcc59	"npm start"	56 minutes ago	Created	0.0.0.0:4000->8090/tcp	vigila
nt_swirles						
3f640fde2051	9356b6bdcc59	"npm start"	59 minutes ago	Created	0.0.0.0:4000->8090/tcp	hungry
_satoshi						
5a09cee2a189	9fa6b519278a	"/bin/sh -c 'npm ins..."	About an hour ago	Exited (127) About an hour ago		amazon
g_darwin						
991b427a4dae	hello-world	"/hello"	3 hours ago	Exited (0) 3 hours ago		nifty_
joliot						

Para ejecutar el contenedor desde Docker se hace uso del siguiente comando, "docker run -p 4000:8090 saludayedad", donde el parámetro -p indica que se quiere abrir un puerto, 4000 indica el puerto interno y 8090 el puerto externo, finalmente se pone el nombre del contenedor que se desea correr.

```
C:\Users\KzyCk\Documents\Python\api\sps\helloworld\v1>docker run -p 4000:8090 saludayedad
```

Pide acceso para abrir el puerto, otorgamos permiso de administrador.



Y al ejecutarlo se muestra como se ve en la imagen, se ha ejecutado de manera exitosa mostrando el mensaje esperado.

```
C:\Users\KzyCk\Documents\Python\api\sps\helloworld\v1>docker run -p 4000:8090 saludayedad  
{'message': 'Hola panchito, naciste el año 1970', 'took': 0.0}
```