

## ACTIVIDAD #2 MICROSERVICIOS CON FLASK

### OBJETIVO

Utilizar el servidor Flask para crear un microservicio y desplegarlo en un contenedor Docker.

### INSTRUCCIONES

Puedes utilizar el sistema operativo Linux o Windows

#### IMPORTAR EL MÓDULO FLASK

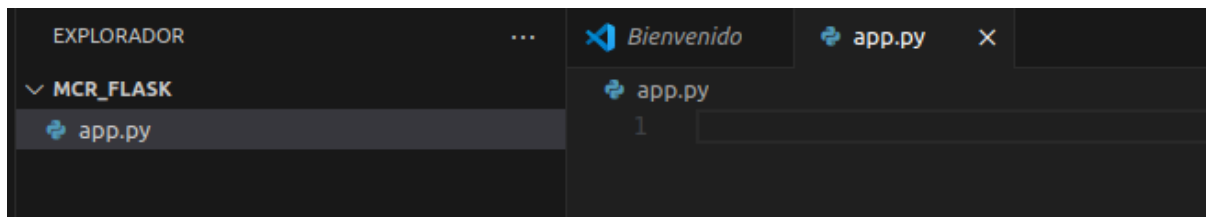
##### pip install flask

```
marianarmz@marianarmz:~/Documentos/giri4091-mcr/MCR_flask$ pip install flask
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: flask in /home/marianarmz/.local/lib/python3.10/site-packages (3.0.3)
Requirement already satisfied: Jinja2>=3.1.2 in /home/marianarmz/.local/lib/python3.10/site-packages (from flask) (3.1.4)
Requirement already satisfied: blinker>=1.6.2 in /home/marianarmz/.local/lib/python3.10/site-packages (from flask) (1.8.2)
Requirement already satisfied: Werkzeug>=3.0.0 in /home/marianarmz/.local/lib/python3.10/site-packages (from flask) (3.0.3)
Requirement already satisfied: click>=8.1.3 in /home/marianarmz/.local/lib/python3.10/site-packages (from flask) (8.1.7)
Requirement already satisfied: itsdangerous>=2.1.2 in /home/marianarmz/.local/lib/python3.10/site-packages (from flask) (2.2.0)
Requirement already satisfied: MarkupSafe>=2.0 in /home/marianarmz/.local/lib/python3.10/site-packages (from Jinja2>=3.1.2->flask) (2.1.5)
marianarmz@marianarmz:~/Documentos/giri4091-mcr/MCR_flask$
```

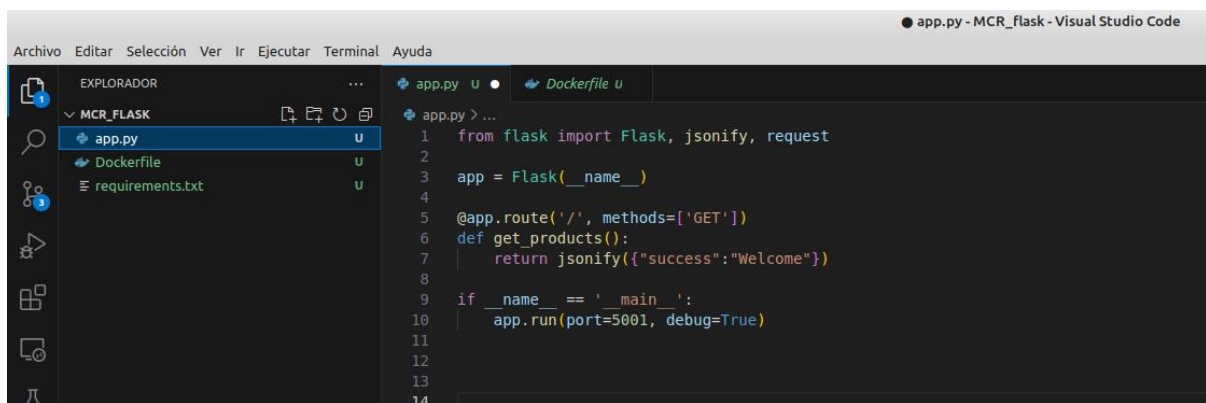
#### CREAR UN NUEVO PROYECTO Y NOMBRARLO MCR\_FLASK

```
marianarmz@marianarmz:~/Documentos/giri4091-mcr$ cd MCR_flask/
marianarmz@marianarmz:~/Documentos/giri4091-mcr/MCR_flask$
```

#### CREAR UN ARCHIVO APP.PY



#### AGREGAR LAS SIGUIENTES INSTRUCCIONES PARA CREAR UN SERVIDOR WEB CON FLASK

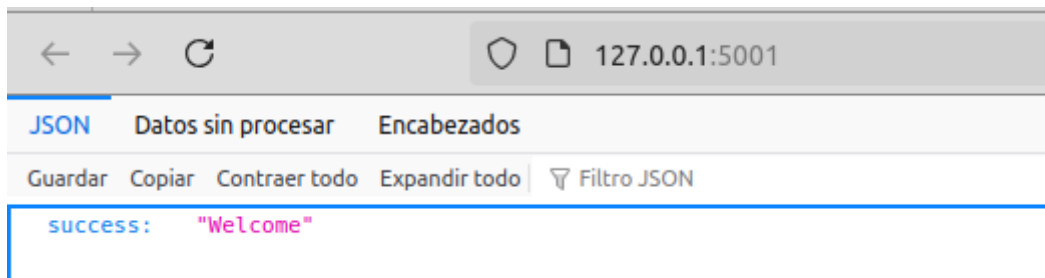


## EJECUTAR LA APLICACIÓN CON PYTHON APP.PY

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

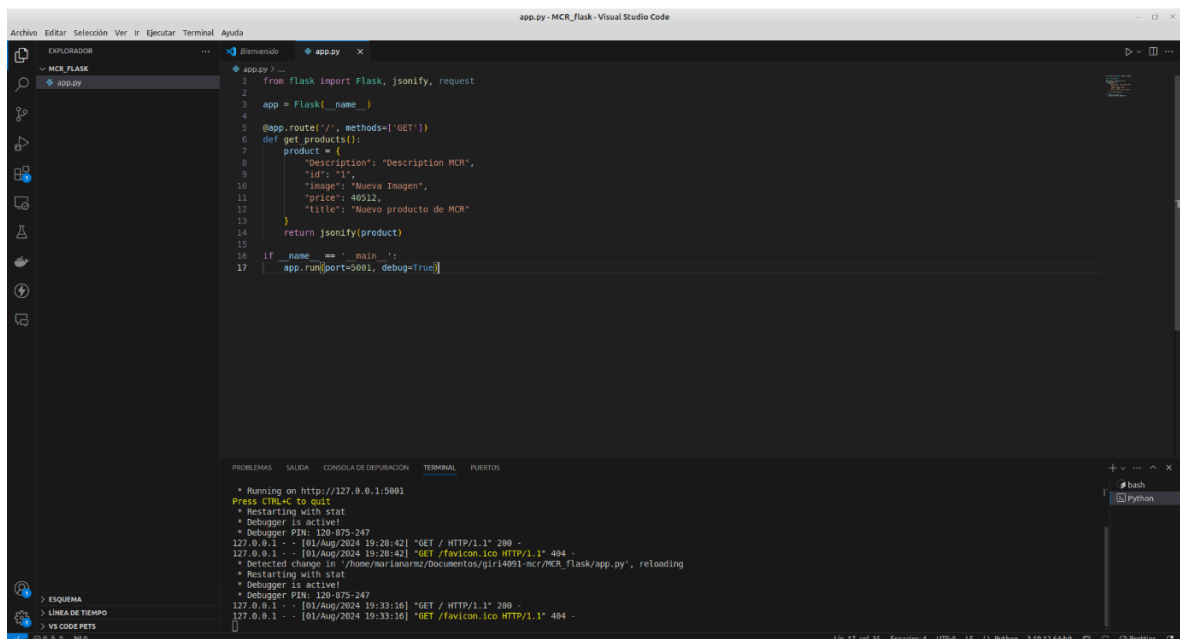
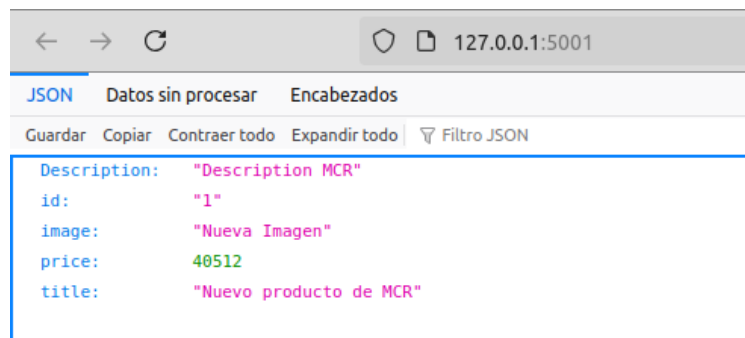
* Debugger is active!
* Debugger PIN: 410-885-161
^Cmarianarmz@marianarmz:~/Documentos/giri4091-mcr/MCR_flask$ /bin/python3 /home/marianarmz/Documentos/giri4091-mcr/MCR_flask/app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://192.168.1.82:5001
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 410-885-161
```

## VERIFICAR LA URL CON EL NAVEGADOR WEB DE TU PREFERENCIA

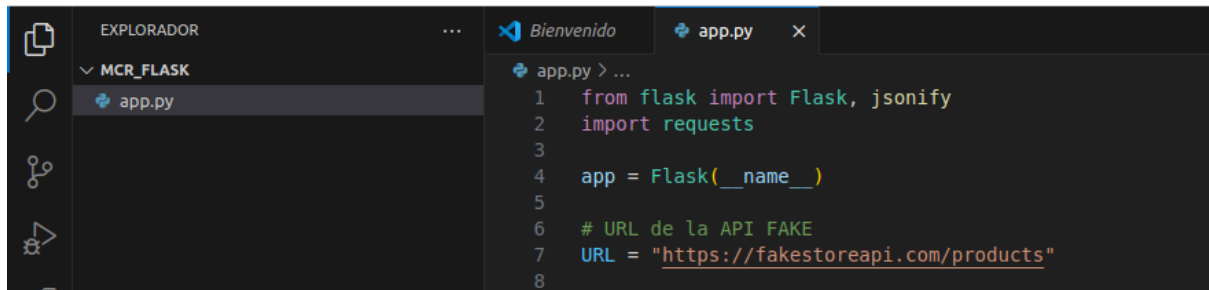


## DESAFÍO:

HAZ QUE LA SALIDA SEA PARECIDA A LA SIGUIENTE FIGURA



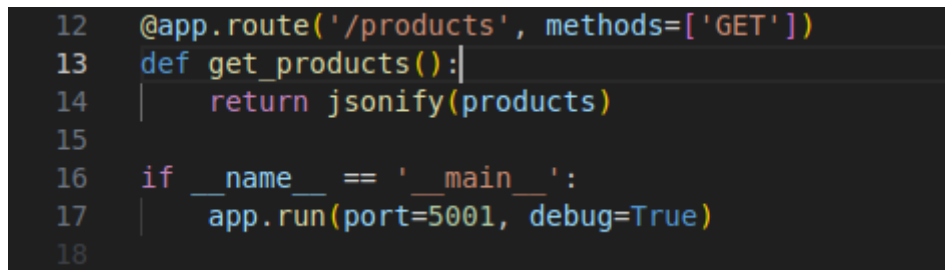
ENSEGUIDA VAMOS A ALMACENAR LOS DATOS DE PRODUCTOS DE LA API FAKE COMO LO MUESTRA LA SIGUIENTE FIGURA



```
1 from flask import Flask, jsonify
2 import requests
3
4 app = Flask(__name__)
5
6 # URL de la API FAKE
7 URL = "https://fakestoreapi.com/products"
8
```

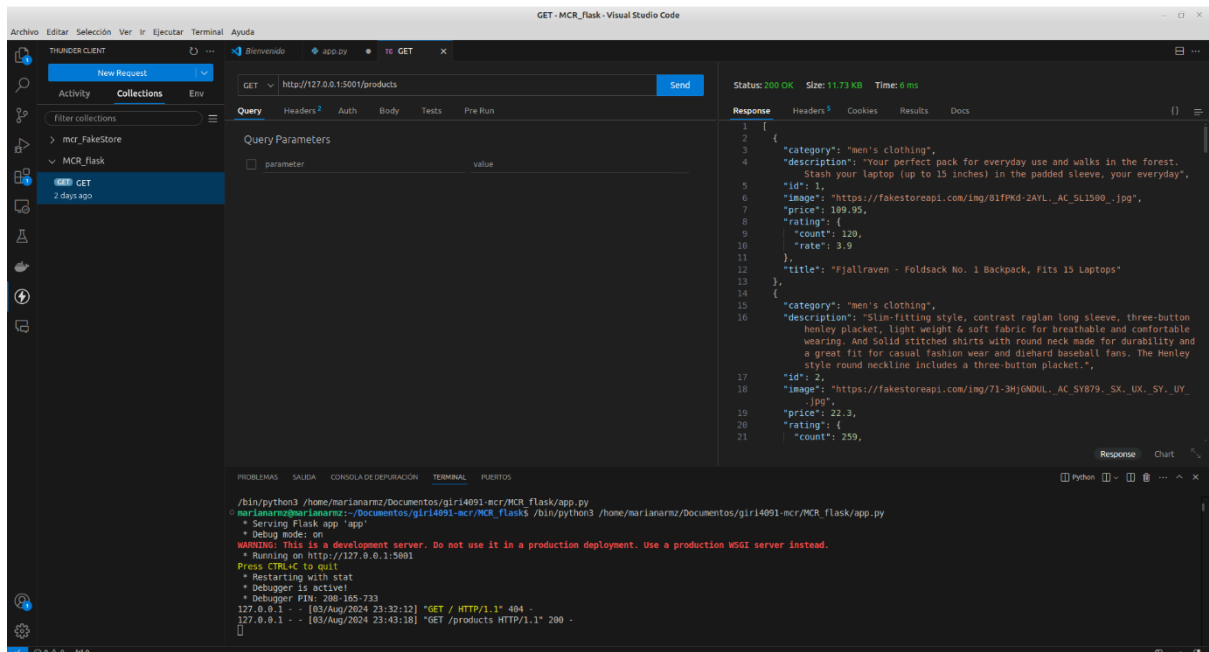
## FUNCIONALIDAD DE LISTADO DE PRODUCTOS

AHORA VAMOS A IMPLEMENTAR EL MÉTODO QUE LISTE LOS PRODUCTOS DE LA API FAKE Y MANEJARLOS DE MANERA LOCAL  
MODIFICAR LA FUNCIÓN GET\_PRODUCTS DE TAL MANERA QUE DESPLIEGUE LOS DATOS EN FORMATO JSON  
OBSERVA CÓMO SE LE ANTEPONE LA RUTA Y EL MÉTODO AL MÉTODO



```
12 @app.route('/products', methods=['GET'])
13 def get_products():
14     return jsonify(products)
15
16 if __name__ == '__main__':
17     app.run(port=5001, debug=True)
18
```

VERIFICAR LA FUNCIONAL CON POSTMAN O THUNDER CLIENT  
HTTP://127.0.0.1:5001/PRODUCTS



```
1 {
2   {
3     "category": "men's clothing",
4     "description": "Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday",
5     "id": 1,
6     "image": "https://fakestoreapi.com/img/61PkD-2AYL_AC_SL1500_.jpg",
7     "price": 109.95,
8     "rating": {
9       "count": 120,
10      "rate": 3.9
11    },
12     "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops"
13   },
14   {
15     "category": "men's clothing",
16     "description": "Slim-fitting style, contrast raglan long sleeve, three-button henley placket, light weight & soft fabric for breathable and comfortable wearing. And Solid stitched shirts with round neck made for durability and a great fit for casual fashion wear and diehard baseball fans. The Henley style round neckline includes a three-button placket.",
17     "id": 2,
18     "image": "https://fakestoreapi.com/img/71-3HjONDLU_AC_S1879_SK_UX_SY_UY_.jpg",
19     "price": 22.3,
20     "rating": {
21       "count": 250,
```

## FUNCIONALIDAD DE BUSCAR UN PRODUCTO POR ID

IMPLEMENTAR EL MÉTODO QUE BUSCA PRODUCTO A TRAVÉS DE SU ID

```
@app.route('/products/<int:product_id>', methods=['GET'])
```

```
def get_product(product_id):
```

```
    product = get_element(product_id)
```

```
    print(product)
```

```
    if product is None:
```

```
        return jsonify({"error": "Producto No encontrado"}), 404
```

```
    return jsonify(product)
```

OBSERVA CÓMO SE LE DEFINE EL PARÁMETRO PARA PASARLO AL MÉTODO, SEGUIMOS IGUALMENTE CON EL MÉTODO GET

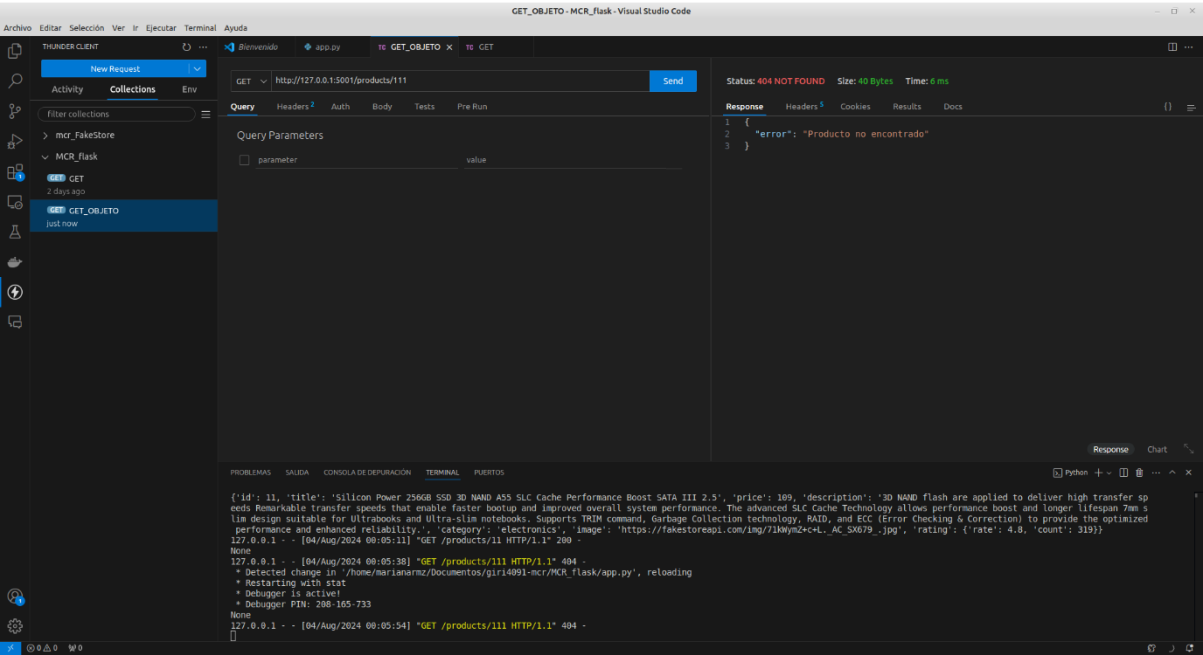
```
26 @app.route('/products/<int:product_id>', methods=['GET'])
27 def get_product(product_id):
28     # Llamar a la función get_element para obtener el producto
29     product = get_element(product_id)
30     print(product) # Imprimir el producto para depuración
31     if product is None:
32         return jsonify({"error": "Producto No encontrado"}), 404
33     return jsonify(product)
```

COMPLETAR EL MÉTODO QUE BUSCA UN PRODUCTO A TRAVÉS DE LA LISTA PRODUCTOS DE ACUERDO CON SU ID

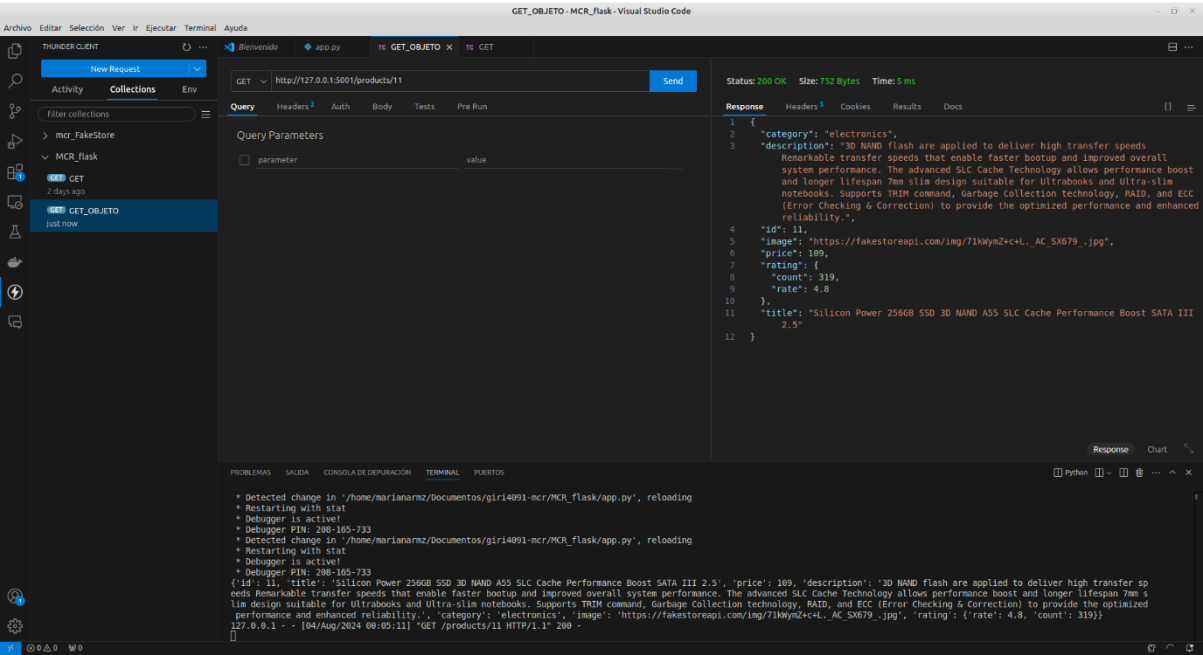
```
16 # Definir la función get_element
17 def get_element(product_id):
18     isFound = False
19     for product in products:
20         if product["id"] == product_id:
21             isFound = True
22             return product # Devuelve el producto si se encuentra
23     return None # Devuelve None si no se encuentra el producto
24
25
26 @app.route('/products/<int:product_id>', methods=['GET'])
27 def get_product(product_id):
28     # Llamar a la función get_element para obtener el producto
29     product = get_element(product_id)
30     print(product) # Imprimir el producto para depuración
31     if product is None:
32         return jsonify({"error": "Producto No encontrado"}), 404
33     return jsonify(product)
```

# PRUEBA LA FUNCIONALIDAD CON POSTMAN O THUNDER CLIENT

## PRODUCTO NO EXISTENTE



## PRODUCTO EXISTENTE



## FUNCIONALIDAD PARA AGREGAR PRODUCTO

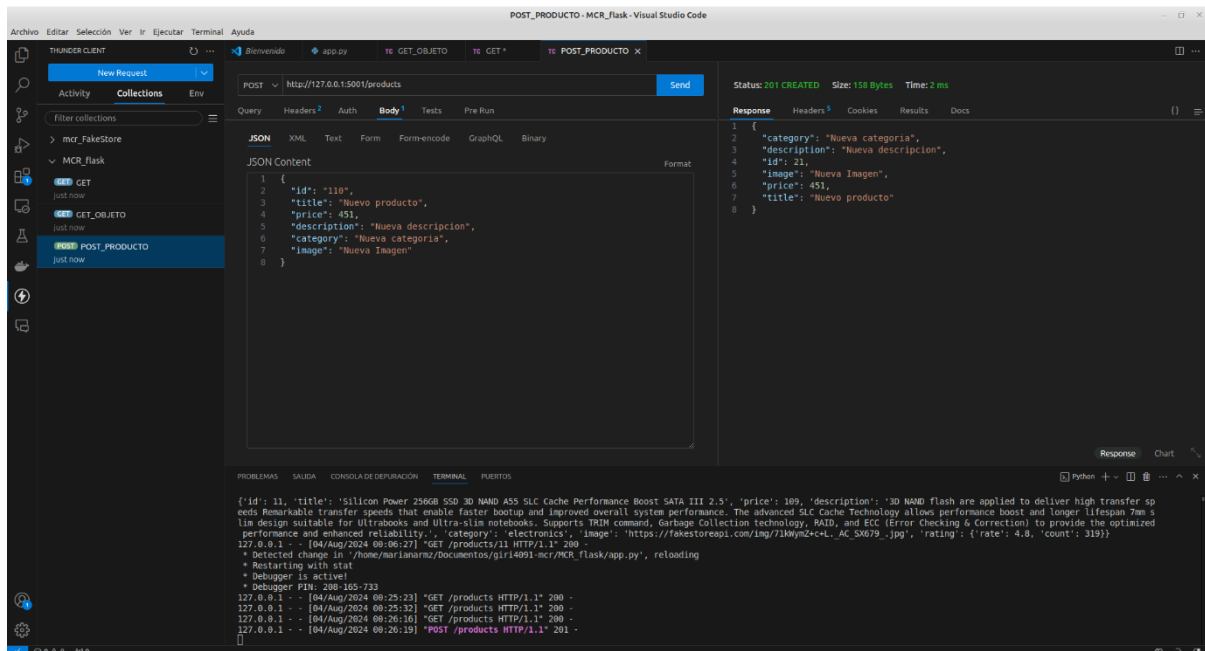
AGREGAR LA SIGUIENTE FUNCIONALIDAD PARA AGREGAR UN NUEVO PRODUCTO

```
31
32 # Funcionalidad para agregar un nuevo producto
33 @app.route('/products', methods=['POST'])
34 def create_product():
35     data = request.get_json() # Obtener datos de la solicitud
36     product_id = max(p['id'] for p in products) + 1 # Asignar nuevo ID
37     data['id'] = product_id # Agregar ID al nuevo producto
38     products.append(data) # Agregar el nuevo producto a la lista
39     return jsonify(data), 201 # Devolver el producto creado |
40
```

IMPLEMENTAR EL MÉTODO MAX\_ID() PARA EXTRAER EL ID MÁXIMO QUE SE ENCUENTRA EN LA LISTA LLAMADO PRODUCTS

```
32 def max_id():
33     maximo = products[0]['id']
34     for product in products:
35         if product['id'] > maximo:
36             maximo = product['id']
37     return maximo
38
39 @app.route('/products', methods=['POST'])
40 def create_product():
41     data = request.get_json() # Obtener datos de la solicitud
42     product_id = max_id() + 1 # Asignar nuevo ID utilizando max_id()
43     data['id'] = product_id # Agregar ID al nuevo producto
44     products.append(data) # Agregar el nuevo producto a la lista
45     return jsonify(data), 201 # Devolver el producto creado con código 201
46
```

PRUEBA LA FUNCIONALIDAD CON POSTMAN O THUNDER CLIENT



## DESAFÍO:

IMPLEMENTAR Y PROBAR EL MÉTODO PARA ELIMINAR UN PRODUCTO A TRAVÉS DE SU ID

```
40
47 @app.route('/products/<int:product_id>', methods=['DELETE'])
48 def delete_product(product_id):
49     global products
50     product = get_element(product_id)
51     if product is None:
52         return jsonify({"error": "Producto no encontrado"}), 404
53     products = [p for p in products if p['id'] != product_id]
54     return jsonify({"message": "Producto eliminado"}), 200
55
56
```

## PRODUCTO NO ENCONTRADO

The screenshot shows the Visual Studio Code interface with the Thunder Client extension. A DELETE request is sent to `http://127.0.0.1:5001/products/150`. The response status is `404 NOT FOUND` with a size of 40 bytes and a time of 7 ms. The response body is `{ "error": "Producto no encontrado" }`. The terminal shows the application running and reloading after a change in the code.

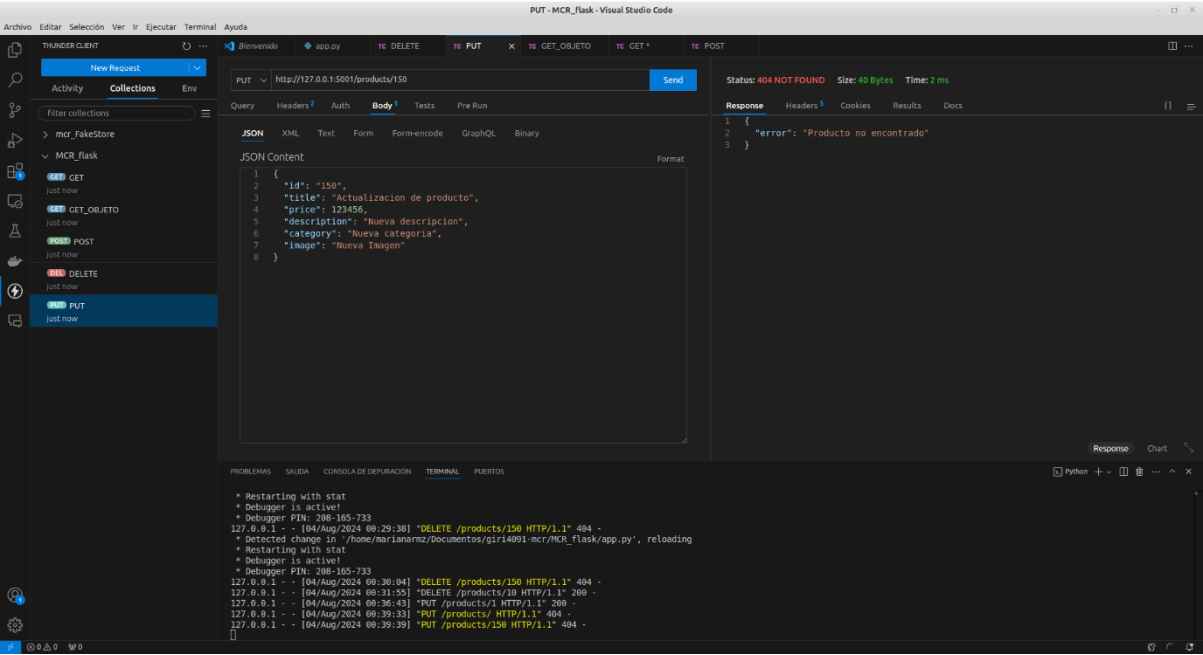
## PRODUCTO ENCONTRADO Y ELIMINADO CORRECTAMENTE

The screenshot shows the Visual Studio Code interface with the Thunder Client extension. A DELETE request is sent to `http://127.0.0.1:5001/products/10`. The response status is `200 OK` with a size of 38 bytes and a time of 4 ms. The response body is `{ "message": "Producto eliminado" }`. The terminal shows the application running and reloading after a change in the code.

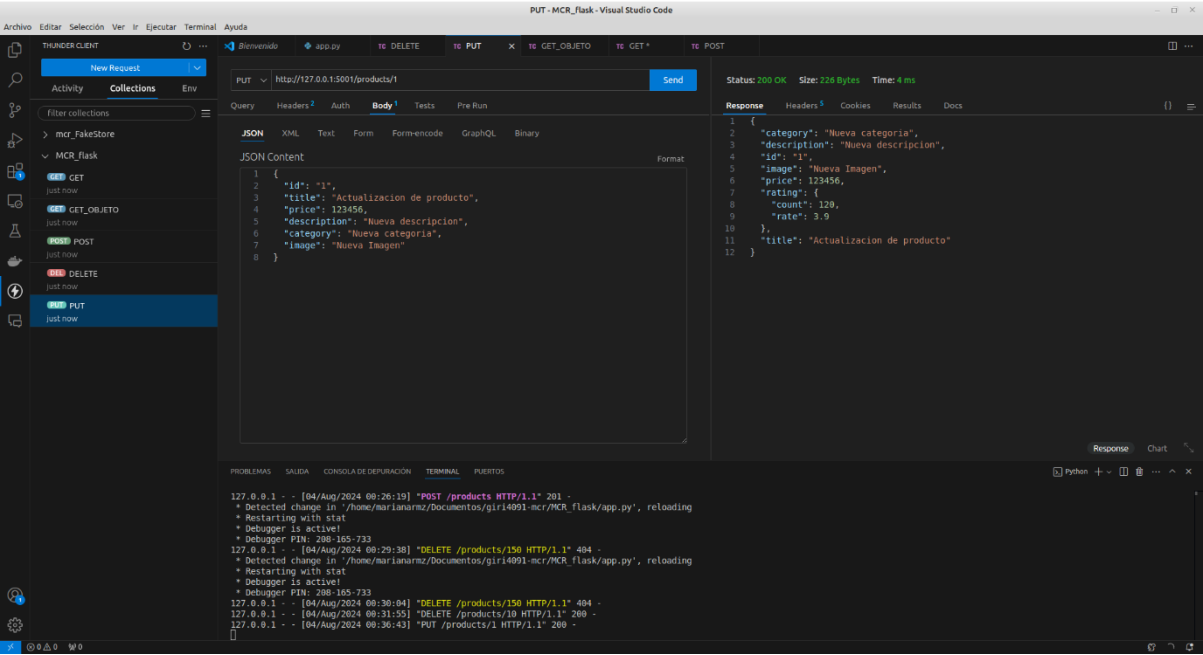
# IMPLEMENTAR Y PROBAR EL MÉTODO PARA MODIFICAR UN PRODUCTO A TRAVÉS DE SU ID

```
56
57 @app.route('/products/<int:product_id>', methods=['PUT'])
58 def update_product(product_id):
59     data = request.get_json()
60     product = get_element(product_id)
61     if product is None:
62         return jsonify({"error": "Producto no encontrado"}), 404
63     product.update(data)
64     return jsonify(product)
65
```

## PRODUCTO NO ENCONTRADO



## PRODUCTO ACTUALIZADO CORRECTAMENTE





## CREACIÓN DE ARCHIVO DOCKERFILE

```
marianarmz@marianarmz:~/Documentos/giri4091-mcr/MCR_flask$ sudo nano Dockerfile
marianarmz@marianarmz:~/Documentos/giri4091-mcr/MCR_flask$

marianarmz@marianarmz: ~/Documentos/giri4091-mcr/MCR_flask
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 6.2 Dockerfile *
# Usa una imagen base de Python
FROM python:3.9

# Establece el directorio de trabajo en /app
WORKDIR /app

# Copia el archivo requirements.txt al contenedor en el directorio de trabajo
COPY requirements.txt .

# Instala las dependencias necesarias
RUN pip install -r requirements.txt

# Copia el archivo app.py al contenedor en el directorio de trabajo
COPY app.py .

# Expone el puerto en el que correrá la aplicación
EXPOSE 5001

# Define el comando por defecto para correr la aplicación
CMD ["python", "app.py"]
```

## CREAR DOCUMENTO TXT DE REQUERIMIENTOS

```
marianarmz@marianarmz:~/Documentos/giri4091-mcr/MCR_flask$ sudo nano requirements.txt
marianarmz@marianarmz:~/Documentos/giri4091-mcr/MCR_flask$ !c
root@marianarmz: /home/marianarmz/Documentos/giri4091-mcr/MCR_flask

Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 6.2 requirements.txt
flask==2.0.3
requests==2.26.0
Werkzeug==2.0.3
```

## CONSTRUIR IMAGEN DOCKER CON EL SIGUIENTE COMANDO:

**docker build -t flask-api .**

```
marianarmz@marianarmz:~/Documentos/giri4091-mcr/MCR_flask$ sudo docker build -t flask-api .
[+] Building 455.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 558B
=> [internal] load metadata for docker.io/library/python:3.9
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/python:3.9@sha256:96daeb1dce644e6dd313489b27a83fbd6d1ccd81d6993e8079f78e4485ba24f7
=> => resolve docker.io/library/python:3.9@sha256:96daeb1dce644e6dd313489b27a83fbd6d1ccd81d6993e8079f78e4485ba24f7
=> => sha256:83a59ab1a4811d0d1b135849e5071eff4d461a56def17589bd1b2f093aeeb5a1 7.31kB / 7.31kB
=> => sha256:10d643a5fa823cd013a108b2076f4d2edf1b2a921f863b533e83ea5ed8d09bd4 64.14MB / 64.14MB
=> => sha256:9972540d93856f9ca3eff2cf803ffb472bf1687cd1a91365cc803a539281900b 2.52kB / 2.52kB
=> => sha256:ca4e5d6727252f0dbc207fbf283cb95e278bf562bda42d35ce6c919583a110a0 49.55MB / 49.55MB
=> => sha256:30b93c12a9c9326732b35d9e3ebe57148abe33f8fa6e25ab76867410b0ccf876 24.05MB / 24.05MB
=> => sha256:96daeb1dce644e6dd313489b27a83fbd6d1ccd81d6993e8079f78e4485ba24f7 10.35kB / 10.35kB
=> => sha256:d6dc1019d7935fe82827434da11bf96cf14e24979f8155e73b794286f10b7f05 211.24MB / 211.24MB
=> => sha256:c7d45ab0573c09f3315112fe3e8d273f4b54dab9e8c3f315810afb743e794a28 6.16MB / 6.16MB
=> => sha256:564d1c451ea70670b349d1250f5c0577416f873f6ee7b5cb33dafeb21c2c40a4 15.82MB / 15.82MB
=> => sha256:ddf5b0ba1977e47749619886799b60da9f2a856fca3270ccb051d2f326489bd5 233B / 233B
=> => sha256:91b87d81d4c8d2b201b71e0a5b07fe01ea4e6d1be30cdc8c30f96653b6663df3 2.70MB / 2.70MB
=> => extracting sha256:ca4e5d6727252f0dbc207fbf283cb95e278bf562bda42d35ce6c919583a110a0
=> => extracting sha256:30b93c12a9c9326732b35d9e3ebe57148abe33f8fa6e25ab76867410b0ccf876
=> => extracting sha256:10d643a5fa823cd013a108b2076f4d2edf1b2a921f863b533e83ea5ed8d09bd4
=> => extracting sha256:d6dc1019d7935fe82827434da11bf96cf14e24979f8155e73b794286f10b7f05
=> => extracting sha256:c7d45ab0573c09f3315112fe3e8d273f4b54dab9e8c3f315810afb743e794a28
=> => extracting sha256:564d1c451ea70670b349d1250f5c0577416f873f6ee7b5cb33dafeb21c2c40a4
=> => extracting sha256:ddf5b0ba1977e47749619886799b60da9f2a856fca3270ccb051d2f326489bd5
=> => extracting sha256:91b87d81d4c8d2b201b71e0a5b07fe01ea4e6d1be30cdc8c30f96653b6663df3
=> [internal] load build context
=> => transferring context: 2.29kB
=> [2/5] WORKDIR /app
=> [3/5] COPY requirements.txt .
=> [4/5] RUN pip install -r requirements.txt
=> [5/5] COPY app.py .
=> => exporting to image
=> => exporting layers
=> => writing image sha256:aafc5247538d634d317245a2ea14812062c2aa1e4d5a3e1542b6bf9ad040a480
=> => naming to docker.io/library/flask-api
marianarmz@marianarmz:~/Documentos/giri4091-mcr/MCR_flask$
```

EJECUTAMOS EL CONTENEDOR DOCKER CON EL SIGUIENTE COMANDO:

**docker run -p 5001:5001 flask-api**

```
root@marianarmz:/home/marianarmz/Documentos/giri4091-mcr/MCR_flask# sudo docker run -p 5001:5001 flask-api
* Serving Flask app 'app' (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://127.0.0.1:5001/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 798-986-770
```