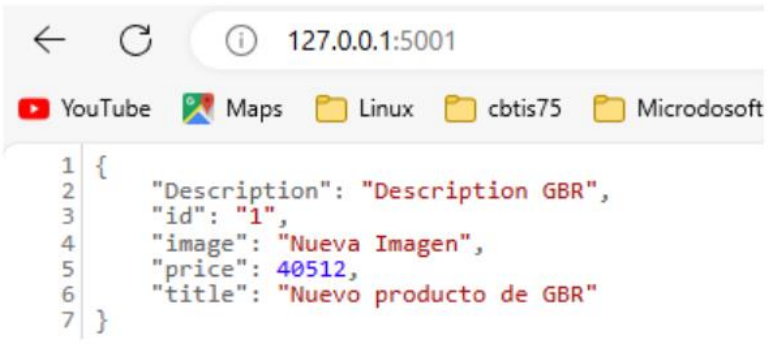
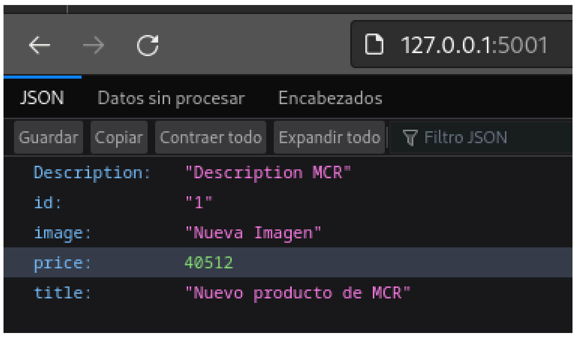
Desafío:

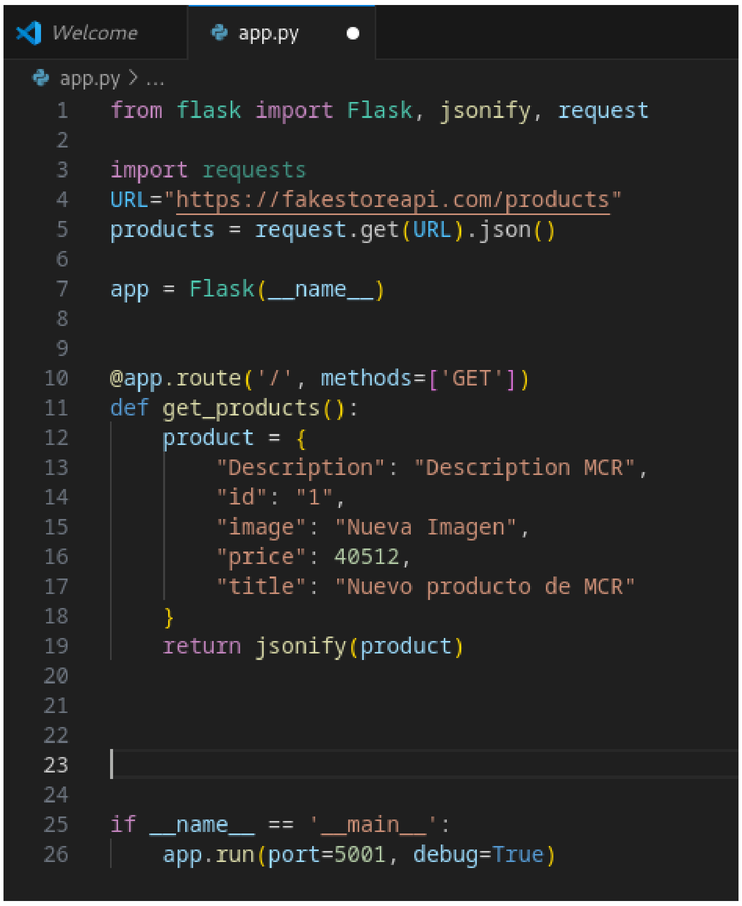
Haz que la salida sea parecida a la siguiente figura



```
{
    "Description": "Description GBR",
    "id": "1",
    "image": "Nueva Imagen",
    "price": 40512,
    "title": "Nuevo producto de GBR"
}
```



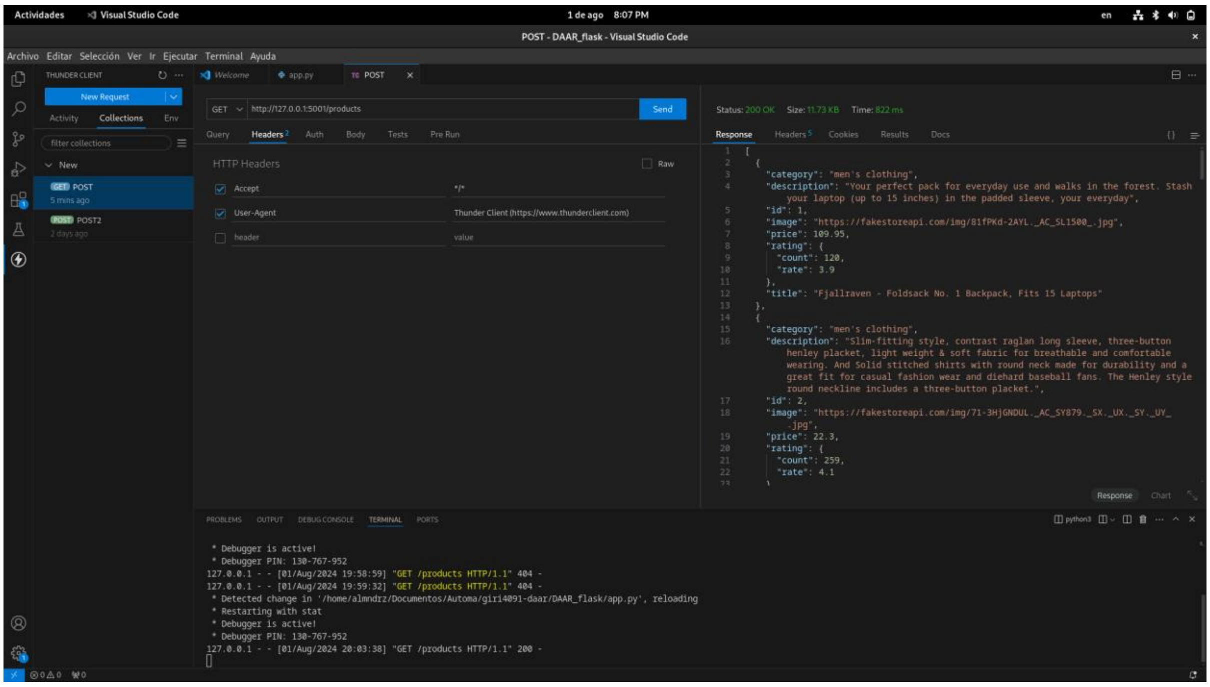
```
Description:    "Description MCR"
id:             "1"
image:          "Nueva Imagen"
price:          40512
title:          "Nuevo producto de MCR"
```



```python
from flask import Flask, jsonify, request

import requests
URL="https://fakestoreapi.com/products"
products = request.get(URL).json()

app = Flask(__name__)


@app.route('/', methods=['GET'])
def get_products():
    product = {
        "Description": "Description MCR",
        "id": "1",
        "image": "Nueva Imagen",
        "price": 40512,
        "title": "Nuevo producto de MCR"
    }
    return jsonify(product)




if __name__ == '__main__':
    app.run(port=5001, debug=True)
```

Funcionalidad de Listado de Productos

Ahora vamos a implementar el método que liste los productos de la API FAKE y manejarlos de manera local.



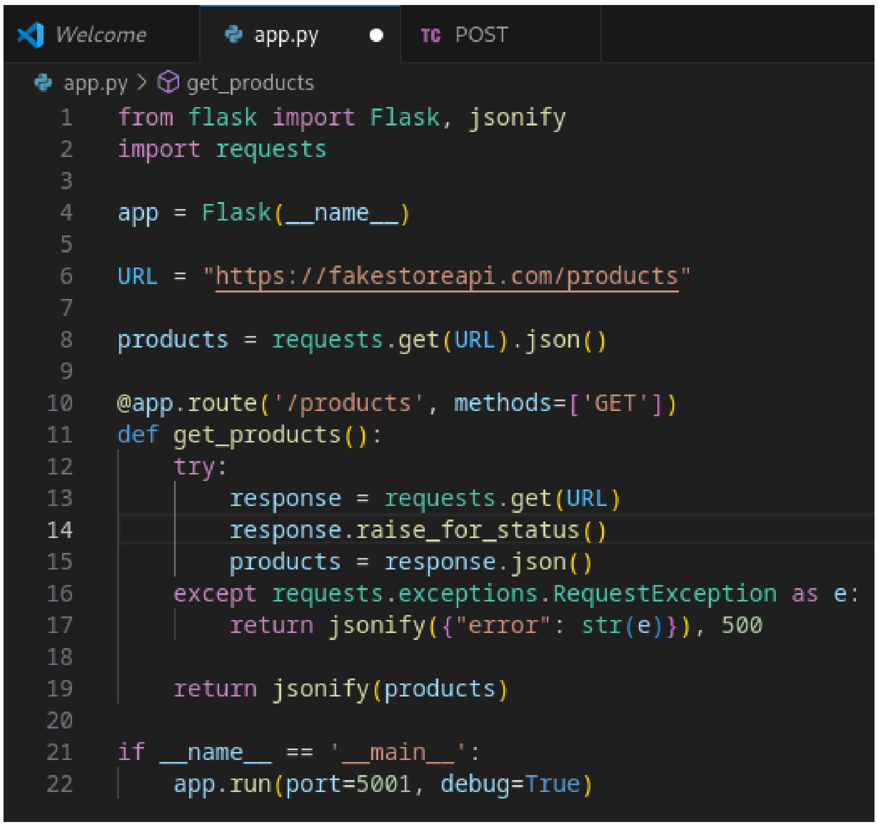
GET http://127.0.0.1:5001/products

HTTP Headers	
Accept	*/*
User-Agent	Thunder Client (https://www.thunderclient.com)

Status: 200 OK Size: 11.73 KB Time: 822 ms

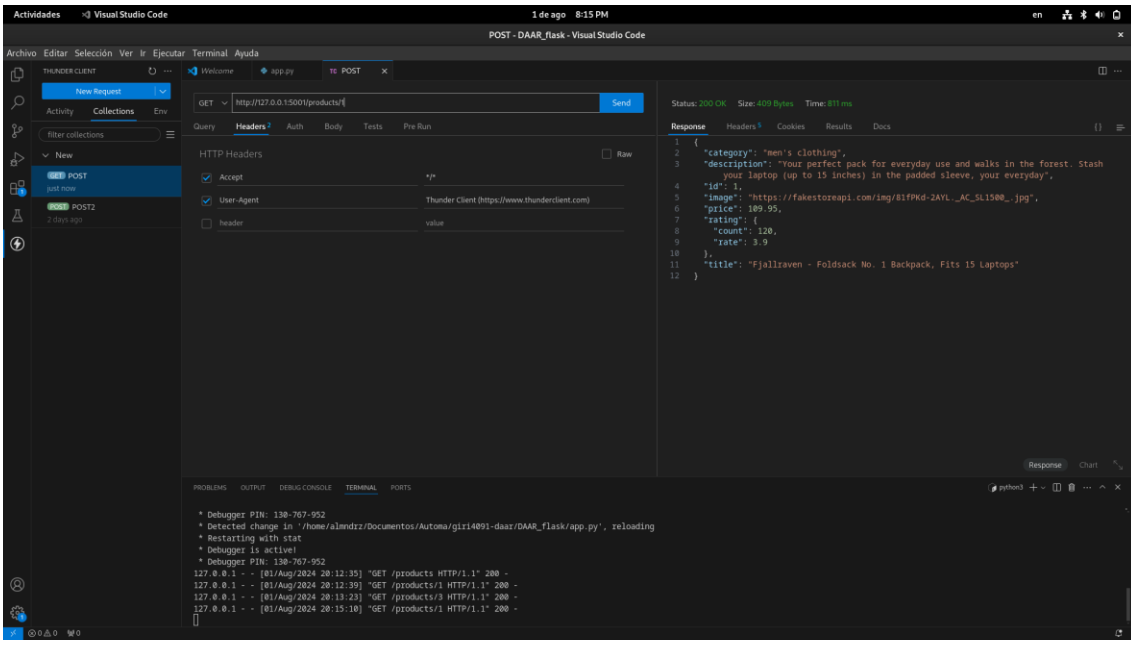
```
[
  {
    "category": "men's clothing",
    "description": "Your perfect pack for everyday use and walks in the forest. Stash
       your laptop (up to 15 inches) in the padded sleeve, your everyday",
    "id": 1,
    "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg",
    "price": 109.95,
    "rating": {
      "count": 120,
      "rate": 3.9
    },
    "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops"
  },
  {
    "category": "men's clothing",
    "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.",
    "id": 2,
    "image": "https://fakestoreapi.com/img/71-3HjGNDUL._AC_SY879._SX._UX._SY._UY_
       .jpg",
    "price": 22.3,
    "rating": {
      "count": 259,
      "rate": 4.1
```

```
 * Debugger is active!
 * Debugger PIN: 130-767-952
127.0.0.1 - - [01/Aug/2024 19:58:59] "GET /products HTTP/1.1" 404 -
127.0.0.1 - - [01/Aug/2024 19:59:32] "GET /products HTTP/1.1" 404 -
 * Detected change in '/home/almndrz/Documentos/Automa/giri4091-daar/DAAR_flask/app.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 130-767-952
127.0.0.1 - - [01/Aug/2024 20:03:38] "GET /products HTTP/1.1" 200 -
```



```python
from flask import Flask, jsonify
import requests

app = Flask(__name__)

URL = "https://fakestoreapi.com/products"

products = requests.get(URL).json()

@app.route('/products', methods=['GET'])
def get_products():
    try:
        response = requests.get(URL)
        response.raise_for_status()
        products = response.json()
    except requests.exceptions.RequestException as e:
        return jsonify({"error": str(e)}), 500

    return jsonify(products)

if __name__ == '__main__':
    app.run(port=5001, debug=True)
```

Funcionalidad de Buscar un producto por Id

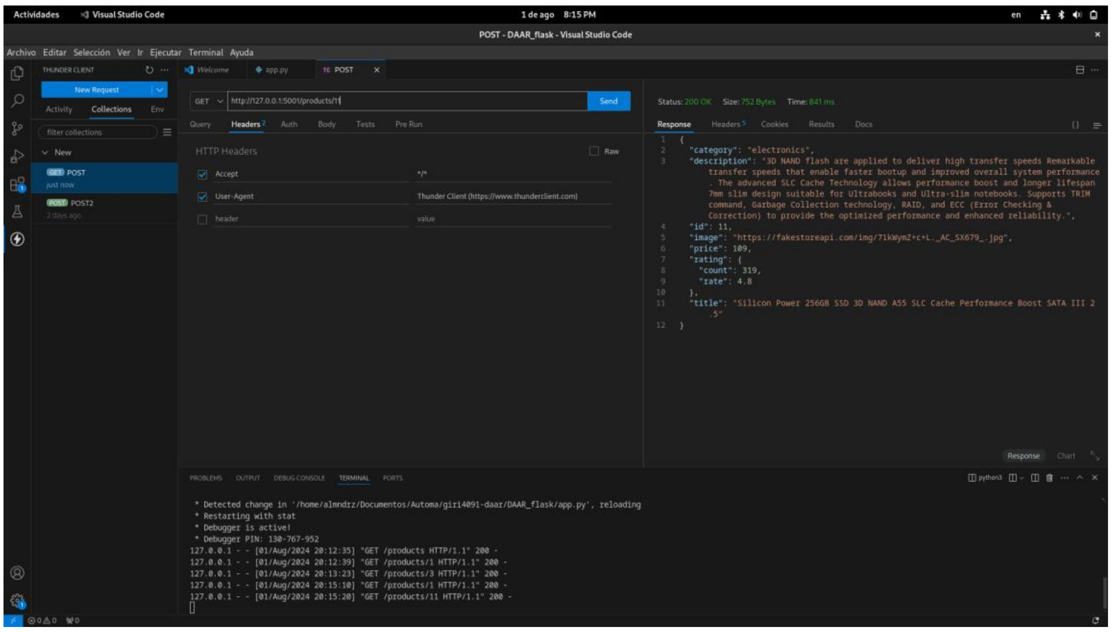


GET http://127.0.0.1:5001/products/1

Status: 200 OK Size: 409 Bytes Time: 811 ms

```
{
  "category": "men's clothing",
  "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",
  "id": 1,
  "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg",
  "price": 109.95,
  "rating": {
    "count": 120,
    "rate": 3.9
  },
  "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops"
}
```

```
 * Debugger PIN: 130-767-952
 * Detected change in '/home/almndrz/Documentos/Automa/giri4091-daar/DAAR_flask/app.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 130-767-952
127.0.0.1 - - [01/Aug/2024 20:12:35] "GET /products HTTP/1.1" 200 -
127.0.0.1 - - [01/Aug/2024 20:12:39] "GET /products/1 HTTP/1.1" 200 -
127.0.0.1 - - [01/Aug/2024 20:13:23] "GET /products/3 HTTP/1.1" 200 -
127.0.0.1 - - [01/Aug/2024 20:15:10] "GET /products/1 HTTP/1.1" 200 -
```

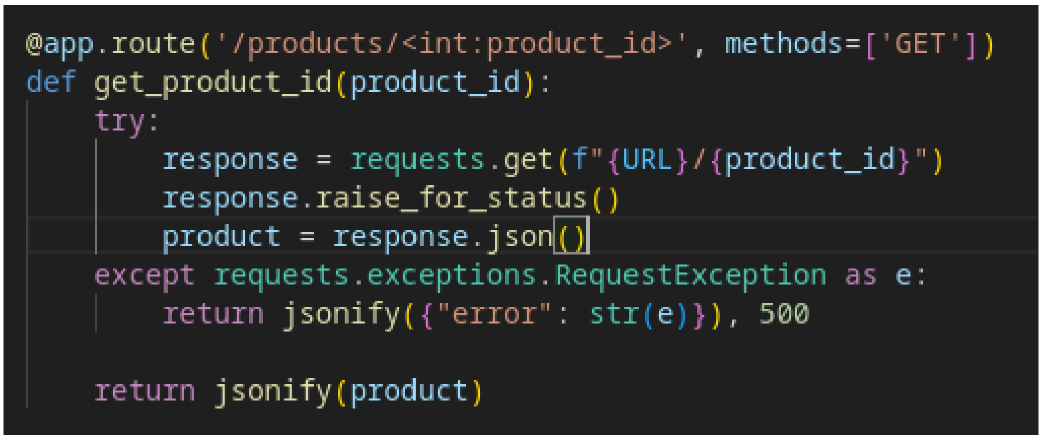


GET http://127.0.0.1:5001/products/11

Status: 200 OK Size: 752 Bytes Time: 841 ms

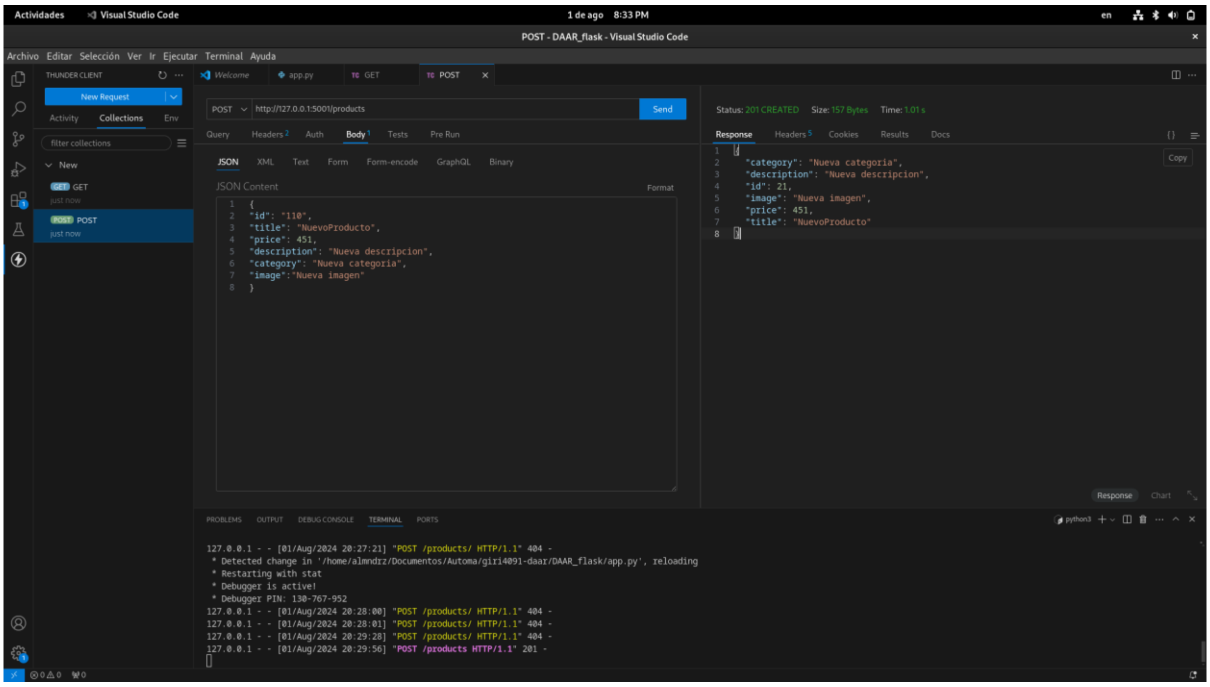
```
{
  "category": "electronics",
  "description": "3D NAND flash are applied to deliver high transfer speeds Remarkable
      transfer speeds that enable faster bootup and improved overall system performance
      . The advanced SLC Cache Technology allows performance boost and longer lifespan
      7mm slim design suitable for Ultrabooks and Ultra-slim notebooks. Supports TRIM
      command, Garbage Collection technology, RAID, and ECC (Error Checking &
      Correction) to provide the optimized performance and enhanced reliability.",
  "id": 11,
  "image": "https://fakestoreapi.com/img/71kWymZ+c+L._AC_SX679_.jpg",
  "price": 109,
  "rating": {
    "count": 319,
    "rate": 4.8
  },
  "title": "Silicon Power 256GB SSD 3D NAND A55 SLC Cache Performance Boost SATA III 2
      .5"
}
```

```
 * Detected change in '/home/almndrz/Documentos/Automa/giri4091-daar/DAAR_flask/app.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 130-767-952
127.0.0.1 - - [01/Aug/2024 20:12:35] "GET /products HTTP/1.1" 200 -
127.0.0.1 - - [01/Aug/2024 20:12:39] "GET /products/1 HTTP/1.1" 200 -
127.0.0.1 - - [01/Aug/2024 20:13:23] "GET /products/3 HTTP/1.1" 200 -
127.0.0.1 - - [01/Aug/2024 20:15:10] "GET /products/1 HTTP/1.1" 200 -
127.0.0.1 - - [01/Aug/2024 20:15:20] "GET /products/11 HTTP/1.1" 200 -
```



```python
@app.route('/products/<int:product_id>', methods=['GET'])
def get_product_id(product_id):
    try:
        response = requests.get(f"{URL}/{product_id}")
        response.raise_for_status()
        product = response.json()
    except requests.exceptions.RequestException as e:
        return jsonify({"error": str(e)}), 500

    return jsonify(product)
```

Funcionalidad para Agregar producto


Actividades
Visual Studio Code
1 de ago 8:33 PM
POST - DAAR_flask - Visual Studio Code
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
THUNDER CLIENT
New Request
Activity Collections Env
filter collections
New
GET GET
POST POST
Welcome
app.py
GET
POST
Send
Query Headers 2 Auth Body 1 Tests Pre Run
JSON XML Text Form Form-encode GraphQL Binary
JSON Content
Format
Response Headers 5 Cookies Results Docs
Copy
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

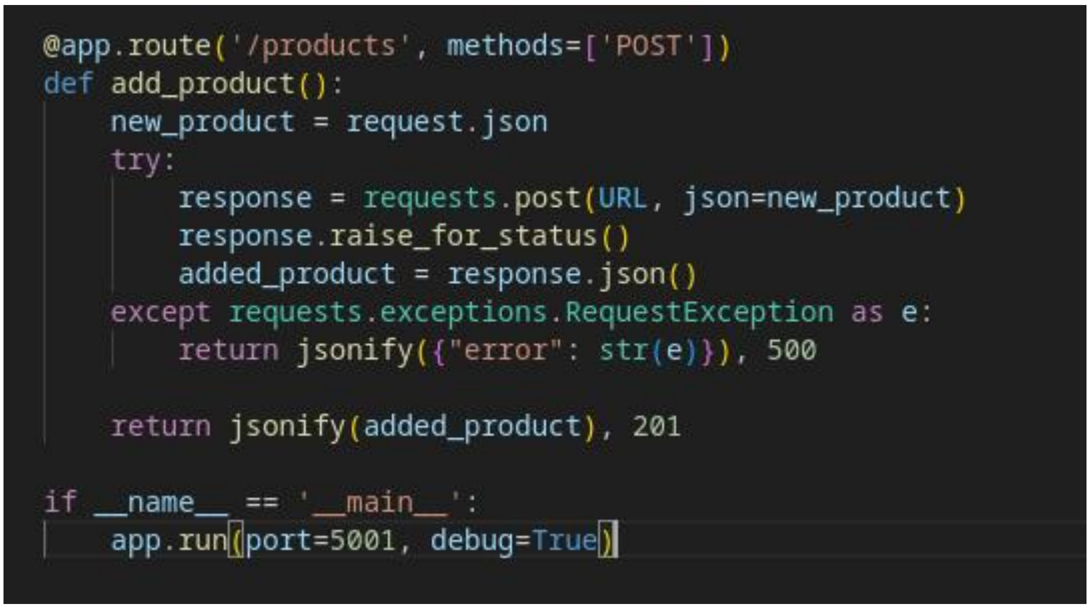
POST http://127.0.0.1:5001/products

```
{
"id": "110",
"title": "NuevoProducto",
"price": 451,
"description": "Nueva descripcion",
"category": "Nueva categoria",
"image":"Nueva imagen"
}
```

Status: 201 CREATED Size: 157 Bytes Time: 1.01 s

```
{
  "category": "Nueva categoria",
  "description": "Nueva descripcion",
  "id": 21,
  "image": "Nueva imagen",
  "price": 451,
  "title": "NuevoProducto"
}
```

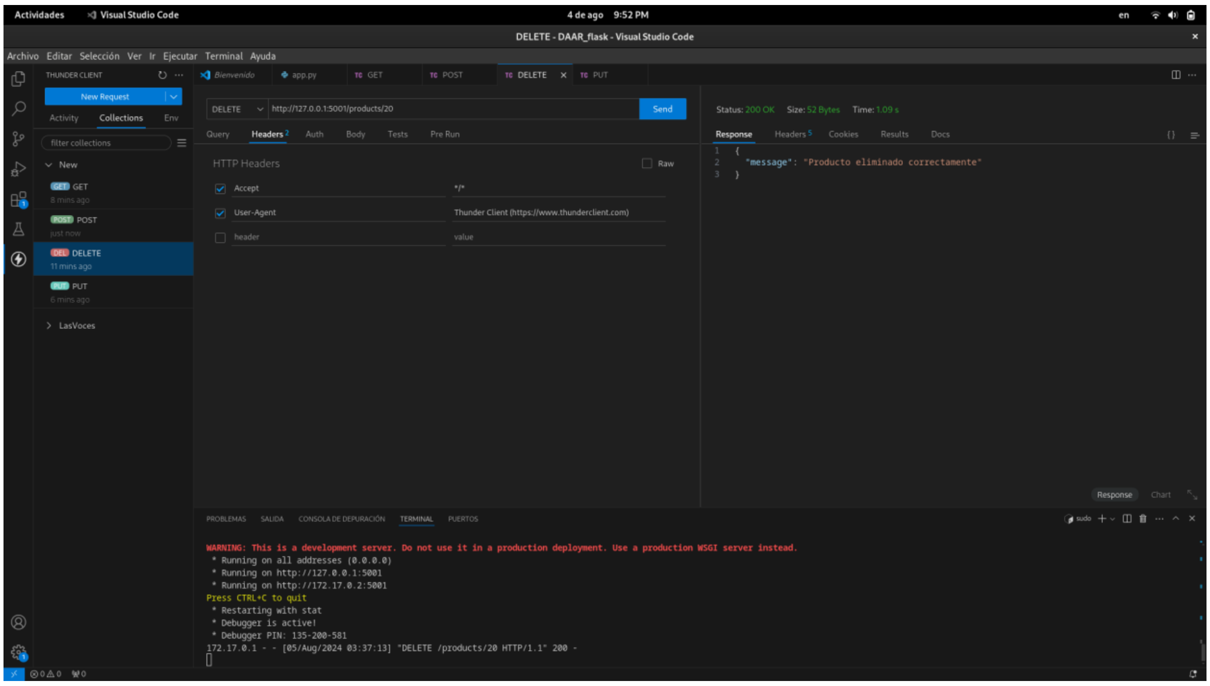
```
127.0.0.1 - - [01/Aug/2024 20:27:21] "POST /products/ HTTP/1.1" 404 -
 * Detected change in '/home/almndrz/Documentos/Automa/giri4091-daar/DAAR_flask/app.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 130-767-952
127.0.0.1 - - [01/Aug/2024 20:28:00] "POST /products/ HTTP/1.1" 404 -
127.0.0.1 - - [01/Aug/2024 20:28:01] "POST /products/ HTTP/1.1" 404 -
127.0.0.1 - - [01/Aug/2024 20:29:28] "POST /products/ HTTP/1.1" 404 -
127.0.0.1 - - [01/Aug/2024 20:29:56] "POST /products HTTP/1.1" 201 -
```



```python
@app.route('/products', methods=['POST'])
def add_product():
    new_product = request.json
    try:
        response = requests.post(URL, json=new_product)
        response.raise_for_status()
        added_product = response.json()
    except requests.exceptions.RequestException as e:
        return jsonify({"error": str(e)}), 500

    return jsonify(added_product), 201

if __name__ == '__main__':
    app.run(port=5001, debug=True)
```

Desafío:

- Implementar y probar el método para eliminar un producto a través de su ID\



DELETE - DAAR_flask - Visual Studio Code

```
DELETE  http://127.0.0.1:5001/products/20
```

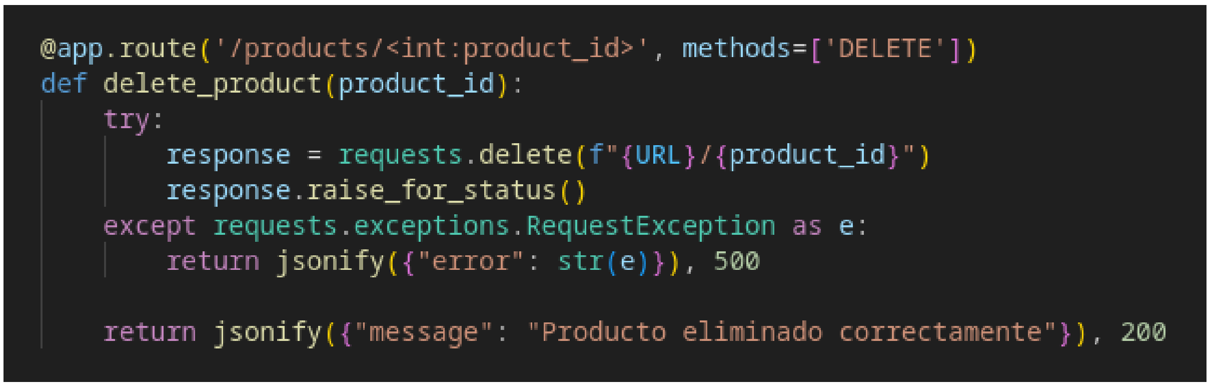
HTTP Headers

✓ Accept	*/*
✓ User-Agent	Thunder Client (https://www.thunderclient.com)
header	value

Status: 200 OK Size: 52 Bytes Time: 1.09 s

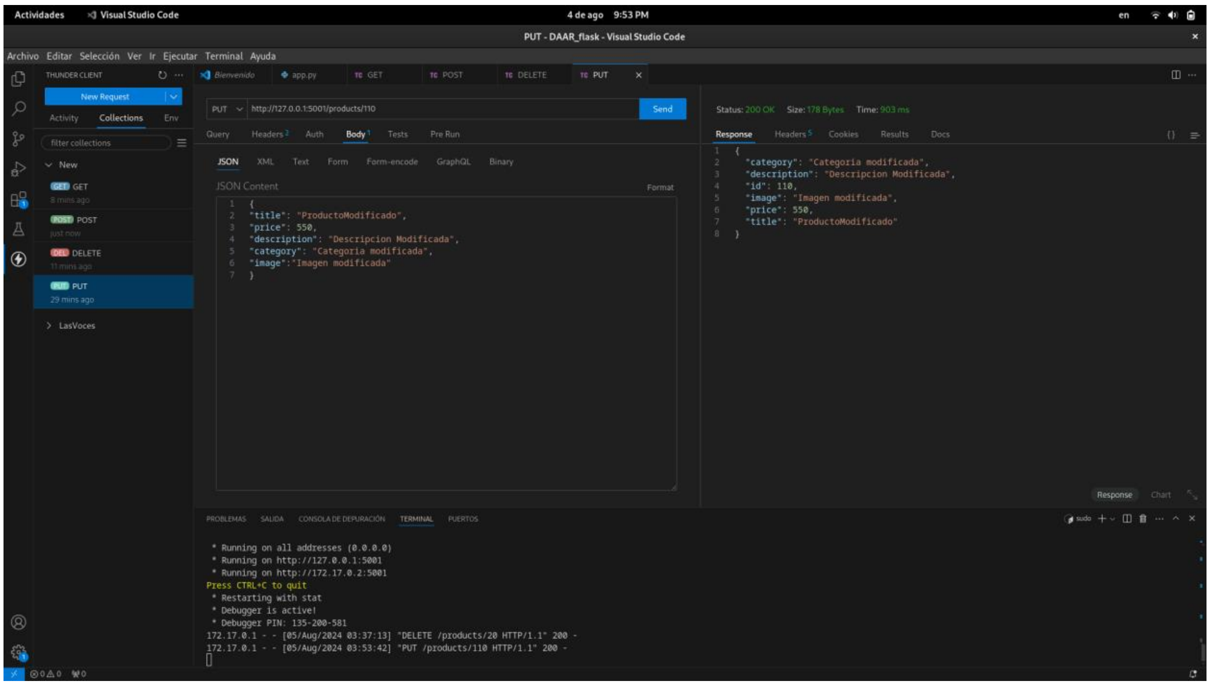
```
{
  "message": "Producto eliminado correctamente"
}
```

```
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://172.17.0.2:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 135-200-581
172.17.0.1 - - [05/Aug/2024 03:37:13] "DELETE /products/20 HTTP/1.1" 200 -
```



```python
@app.route('/products/<int:product_id>', methods=['DELETE'])
def delete_product(product_id):
    try:
        response = requests.delete(f"{URL}/{product_id}")
        response.raise_for_status()
    except requests.exceptions.RequestException as e:
        return jsonify({"error": str(e)}), 500

    return jsonify({"message": "Producto eliminado correctamente"}), 200
```

- Implementar y probar el método para modificar un producto a través de su ID


Actividades
Visual Studio Code
4 de ago 9:53 PM
PUT - DAAR_flask - Visual Studio Code
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
THUNDER CLIENT
New Request
Activity Collections Env
filter collections
New
GET 8 mins ago
POST just now
DELETE 11 mins ago
PUT 29 mins ago
LasVoces
Bienvenido app.py GET POST DELETE PUT
Query Headers Auth Body Tests Pre Run
JSON XML Text Form Form-encode GraphQL Binary
Response Headers Cookies Results Docs
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
PUT  http://127.0.0.1:5001/products/110  Send
```

JSON Content

```
{
"title": "ProductoModificado",
"price": 550,
"description": "Descripcion Modificada",
"category": "Categoria modificada",
"image":"Imagen modificada"
}
```

Status: 200 OK   Size: 178 Bytes   Time: 903 ms

```
{
  "category": "Categoria modificada",
  "description": "Descripcion Modificada",
  "id": 110,
  "image": "Imagen modificada",
  "price": 550,
  "title": "ProductoModificado"
}
```

```
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://172.17.0.2:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 135-200-581
172.17.0.1 - - [05/Aug/2024 03:37:13] "DELETE /products/20 HTTP/1.1" 200 -
172.17.0.1 - - [05/Aug/2024 03:53:42] "PUT /products/110 HTTP/1.1" 200 -
```

```python
@app.route('/products/<int:product_id>', methods=['PUT'])
def update_product(product_id):
    updated_product = request.json
    try:
        response = requests.put(f"{URL}/{product_id}", json=updated_product)
        response.raise_for_status()
        product = response.json()
    except requests.exceptions.RequestException as e:
        return jsonify({"error": str(e)}), 500

    return jsonify(product)
```