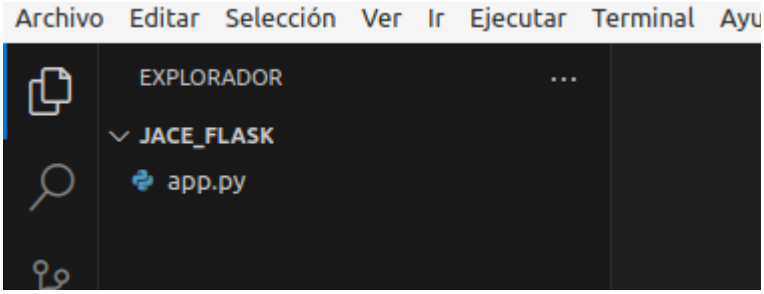
Importar modulo flask

```
jace@jace-ThinkPad-T580:~$ pip install flask
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: flask in ./.local/lib/python3.10/site-packages (3.0.3)
Requirement already satisfied: click>=8.1.3 in ./.local/lib/python3.10/site-packages (from flask) (8.1.7)
Requirement already satisfied: itsdangerous>=2.1.2 in ./.local/lib/python3.10/site-packages (from flask) (2.2.0)
Requirement already satisfied: blinker>=1.6.2 in ./.local/lib/python3.10/site-packages (from flask) (1.8.2)
Requirement already satisfied: Werkzeug>=3.0.0 in ./.local/lib/python3.10/site-packages (from flask) (3.0.3)
Requirement already satisfied: Jinja2>=3.1.2 in ./.local/lib/python3.10/site-packages (from flask) (3.1.4)
Requirement already satisfied: MarkupSafe>=2.0 in ./.local/lib/python3.10/site-packages (from Jinja2>=3.1.2->flask) (2.1.5)
jace@jace-ThinkPad-T580:~$
```

Crear un nuevo proyecto y un aprchivo .py




crear un servidor Web con Flask

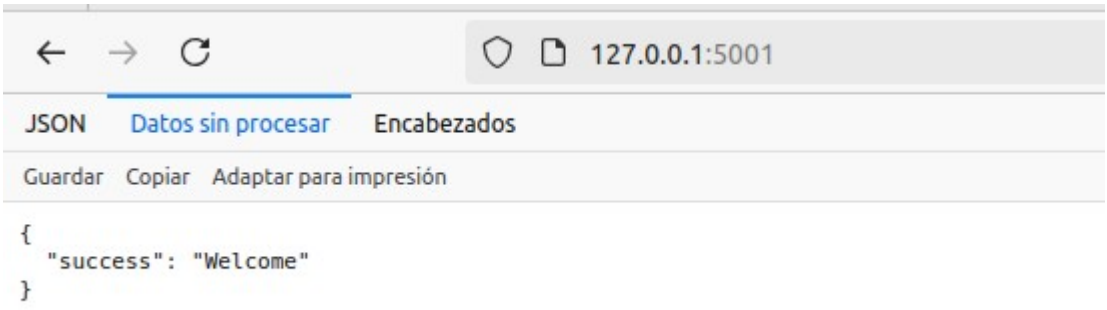
```
app.py > ...
from flask import Flask, jsonify, request
app = Flask(__name__)
@app.route('/', methods=['GET'])
def get_products(): return jsonify({"success":"Welcome"})

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

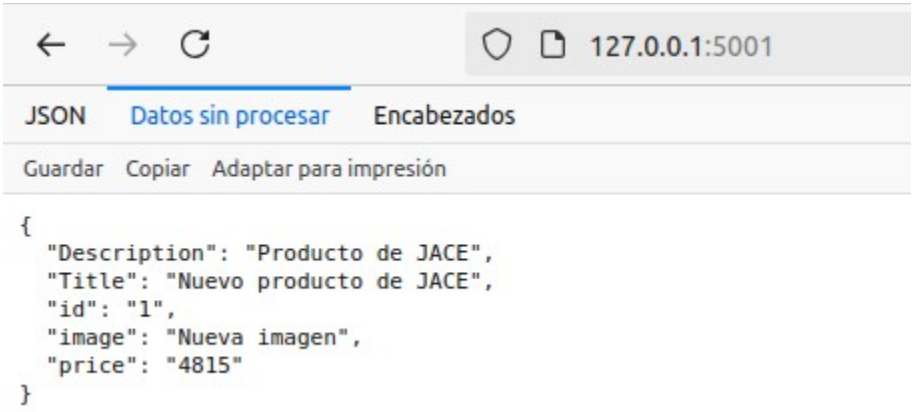
Ejecutar la aplicación con python app.py

```
jace@jace-ThinkPad-T580:~/Escritorio/UTNG/JACE_flask$ /bin/python3 /home/jace/Escritorio/UTNG/JACE_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 http://127.0.0.1:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 225-740-905
```

Verificar la URL con el navegador web




Desafio 1

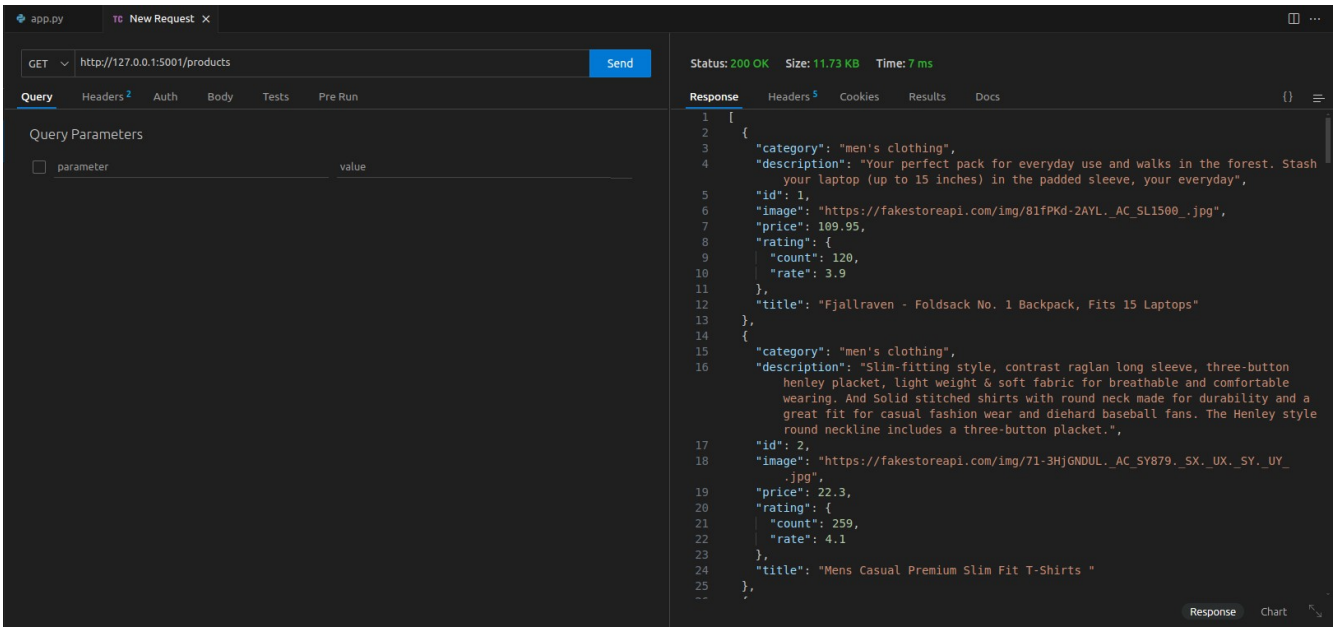



Extraer dato de API FAKE

```
#Extraer dato de API FAKE
import requests
URL = "https://fakestoreapi.com/products"
products = requests.get(URL).json()

@app.route('/products', methods=['GET'])
def get_products():
    return jsonify(products)
```

Verificacion de la funcionalidad con Tunderclient

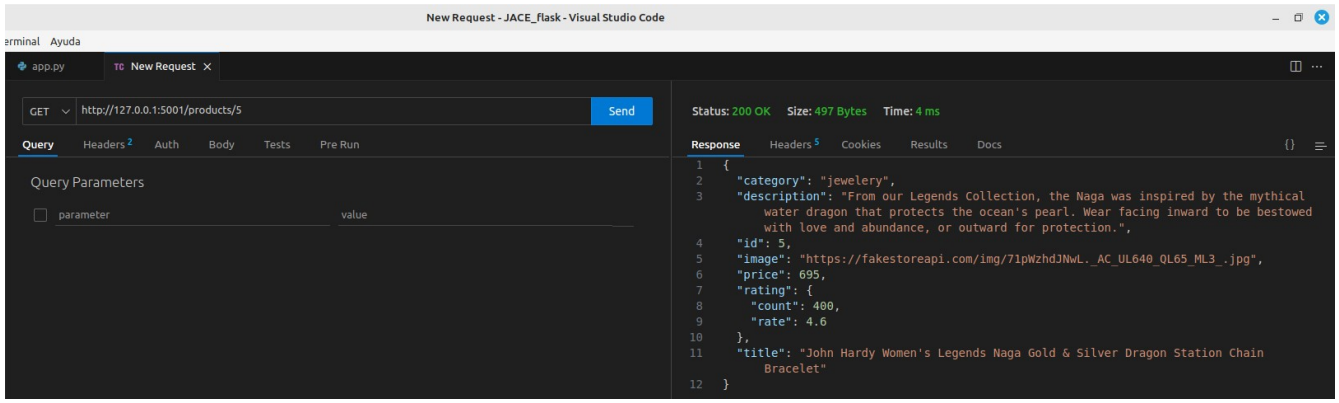



Busqueda de 1 producto por su ID

```python
def get_element(product_id):
    for product in products:
        if product["id"] == product_id:
            return product
    return None


@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)
```

Prueba de funcionamiento con Tunderclient

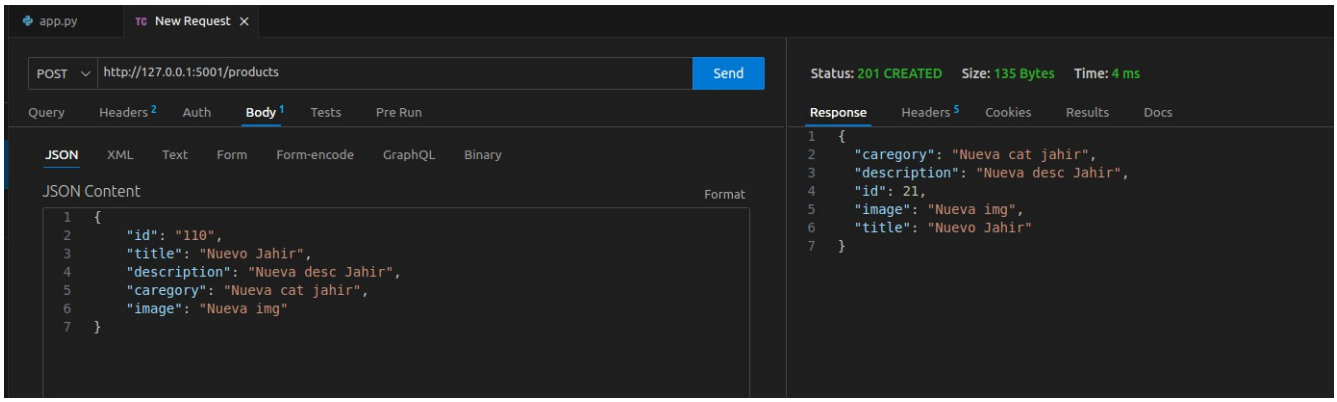



Prueba Agregar Producto:

```python
@app.route('/products', methods=['POST'])
def create_product():
    data = request.get_json()
    product_id = max_id() + 1
    data['id'] = product_id
    products.append(data)
    return jsonify(data), 201


def max_id():
    if not products:
        return 0
    maximo = products[0]['id']
    for product in products:
        if product['id'] > maximo:
            maximo = product['id']
    return maximo
```

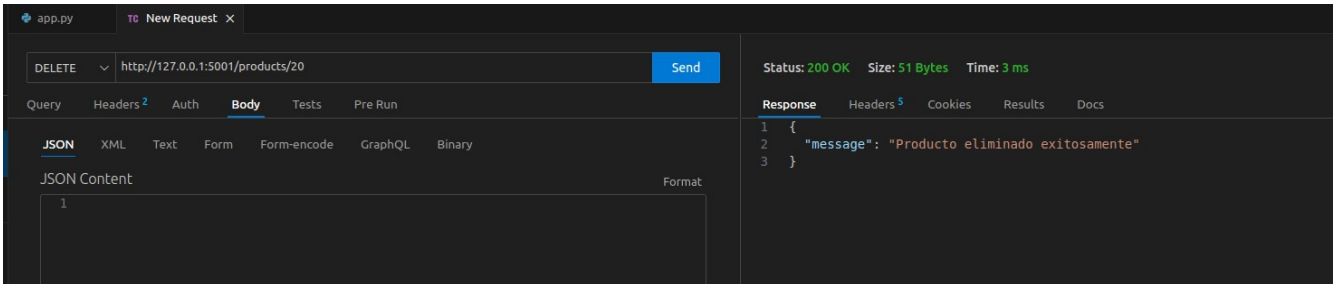
Prueba en TunderClient



Para eliminar un producto por su id

```python
@app.route('/products/<int:product_id>', methods=['DELETE'])
def delete_product(product_id):
    global products
    product_to_delete = next((product for product in products if product['id'] == product_id), None)
    if product_to_delete is None:
        return jsonify({"error": "Producto no encontrado"}), 404
    products = [product for product in products if product['id'] != product_id]
    return jsonify({"message": "Producto eliminado exitosamente"}), 200
```

Prueba con TunderClient




Para Actualizar un producto por su id

```python
@app.route('/products/<int:product_id>', methods=['PUT'])
def update_product(product_id):
    product = get_element(product_id)
    if product is None:
        return jsonify({"error": "Producto no encontrado"}), 404
    data = request.get_json()
    for id in data:
        product[id] = data[id]
    return jsonify(product)
```

Prueba con TunderClient

app.py | New Request

PUT http://127.0.0.1:5001/products/8 Send

Query | Headers 2 | Auth | Body 1 | Tests | Pre Run

JSON | XML | Text | Form | Form-encode | GraphQL | Binary

JSON Content Format

```
{
    "title": "ACt Jahir",
    "description": "Act Jahir",
    "caregory": "Act  jahir",
    "image": "Nueva img"
}
```

Status: 200 OK Size: 217 Bytes Time: 2 ms

Response | Headers 5 | Cookies | Results | Docs

```
{
  "caregory": "Act  jahir",
  "category": "jewelery",
  "description": "Act Jahir",
  "id": 8,
  "image": "Nueva img",
  "price": 10.99,
  "rating": {
    "count": 100,
    "rate": 1.9
  },
  "title": "ACt Jahir"
}
```