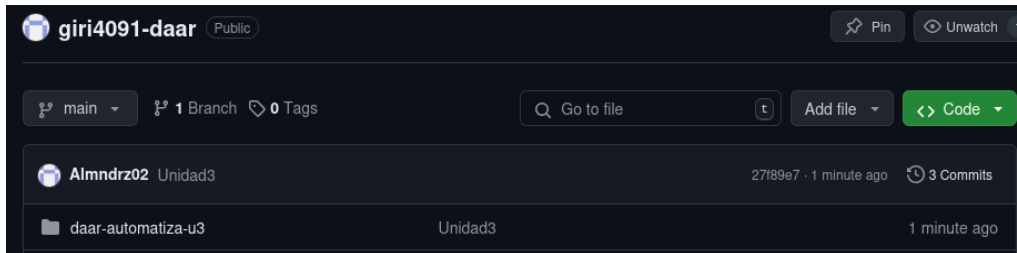
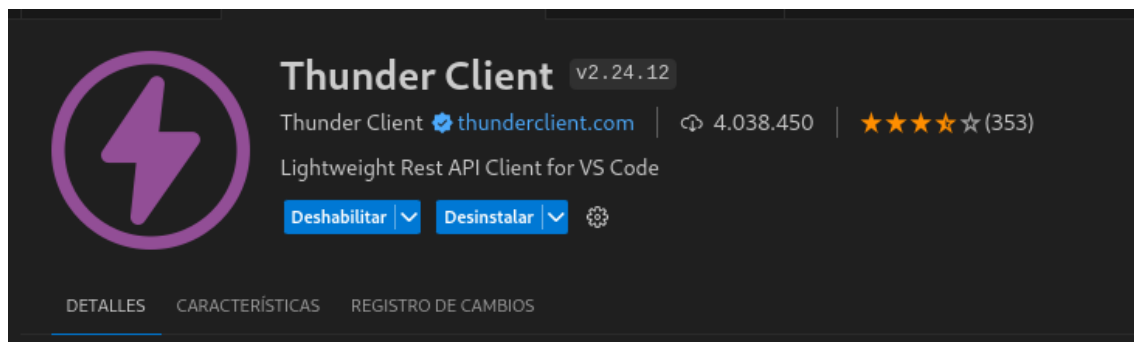


Python and REST APIs

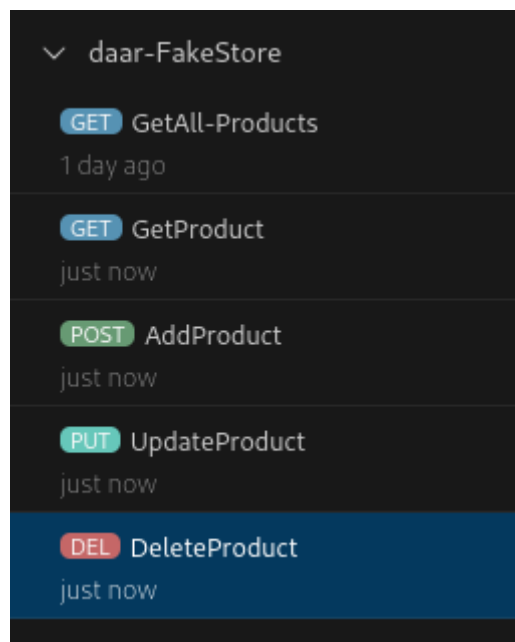
1. Crear un repositorio para la unidad 3 llamado [iniciales]-automatiza-u3.



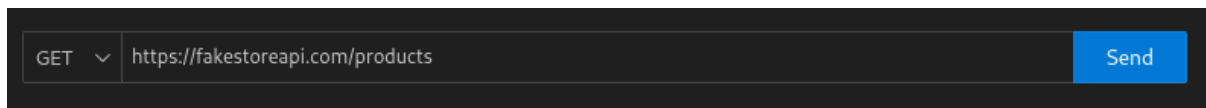
2. Crear un archivo en formato PDF llamado practica01_[iniciales].
3. Abrir Visual Studio Code e instalar el módulo Thunder Client para realizar peticiones.



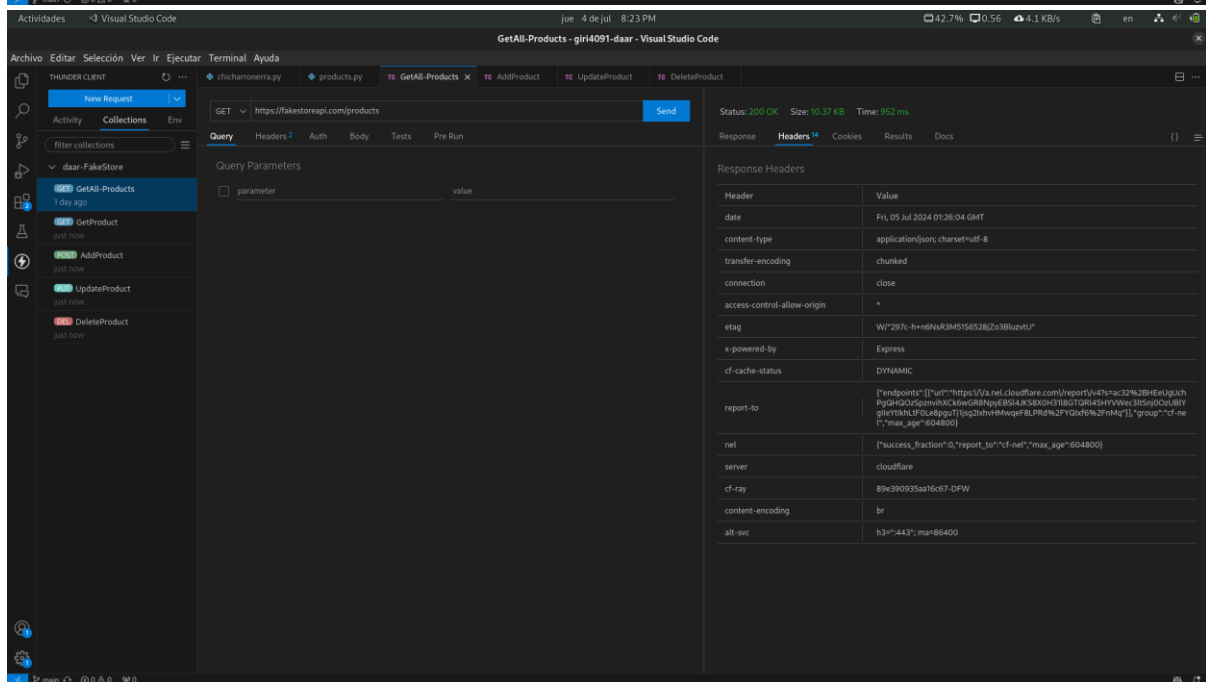
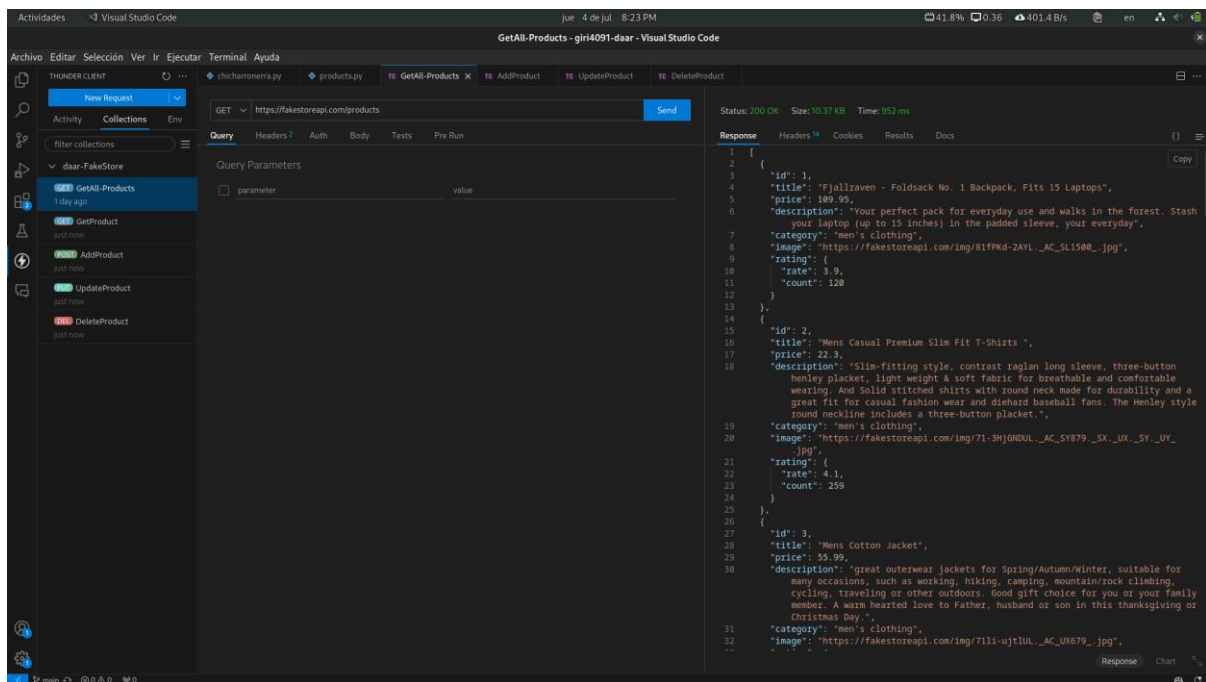
4. Visitar la URL [GitHub FakeStore](#) para conocer más sobre la API
5. Con la extensión **Thunder Client** en Visual Studio Code Crear una colección llamada [Iniciales]-FakeStore



6. Agregar una nueva petición llamada **GetAll-Products** con el método **GET**



7. Enviar la petición para extraer todos los productos
8. Verificar en la respuesta el código de respuesta, así como los Headers (tomar captura de pantalla)



The image displays two screenshots of the Visual Studio Code editor interface, showing a REST client and a Python script for interacting with a FakeStore API.

Top Screenshot: Thunder Client REST Client

The Thunder Client interface shows a REST client configuration for the URL `https://fakestoreapi.com/products/1`. The request is a GET method. The response status is 200 OK, with a size of 364 Bytes and a time of 881 ms. The response headers are displayed:

Header	Value
date	Mon, 08 Jul 2024 04:53:51 GMT
content-type	application/json; charset=utf-8
transfer-encoding	chunked
connection	close
access-control-allow-origin	*
etag	W/"9c-MM4dqV6N0sTieLdgt8eJ9eun"
x-powered-by	Express
cf-cache-status	DYNAMIC
report-to	[{"endpoints": [{"url": "https://ja.net.cloudflare.com/report?u47s=89556f657d7508gnJ9Ew1qAFrFkdLp5eUzj2Onth5hmVhV8052xhTUAJahwv58Rau158%2F%28Y2FheC00d0PTE35%2F068KVTGfG0C02w68%288B902z55G%2851kw"}], "group": "cf-nel", "max_age": 604800}]
nel	[{"success_fraction": 0, "report_to": "cf-nel", "max_age": 604800}]
server	cloudflare
cf-ray	895f790c28a0c46-DFW
content-encoding	br
alt-svc	h3="443"; ma=86400

Bottom Screenshot: Python Script

The Python script defines a function `GetProduct()` that interacts with the FakeStore API. The script is as follows:

```
def GetProduct():
    url_base = 'https://fakestoreapi.com/products'
    print("Busqueda de producto")

    # Solicitar el ID del producto al usuario
    product_id = input("Ingrese el ID del producto que desea consultar: ")

    # Enviar la solicitud GET para obtener los detalles del producto
    url = f'{url_base}/{product_id}'
    response = requests.get(url)

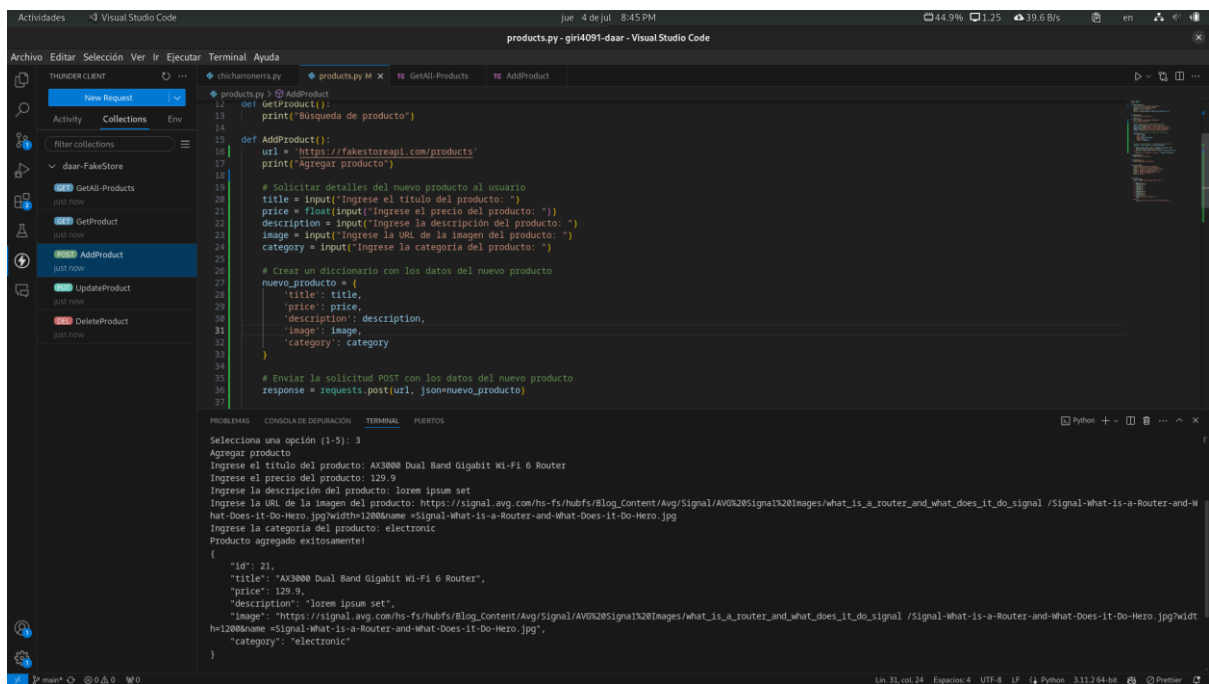
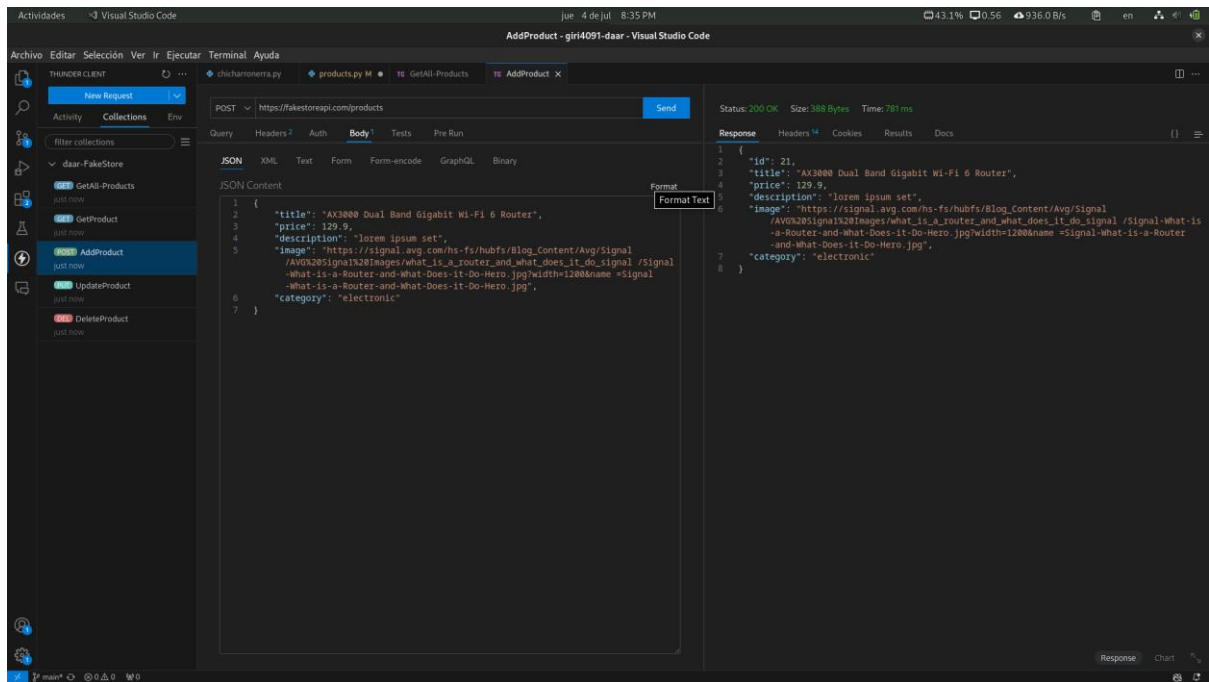
    if response.status_code == 200:
        producto = response.json()
        print("Detalles del producto")
        print("-----")
        print(json.dumps(producto, indent=4, ensure_ascii=False))
    else:
        print("Error al obtener el producto:", response.status_code)
```

The terminal output shows the execution of the script, displaying the details of the product with ID 1:

```
4. Modificar producto en especifico
5. Eliminar un producto
6. Salir
Selecciona una opción (1-5): 2
Busqueda de producto
Ingrese el ID del producto que desea consultar: 1
Detalles del producto
-----
{
  "id": 1,
  "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",
  "price": 109.95,
  "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",
  "category": "men's clothing",
  "image": "https://fakestoreapi.com/img/81fPKd-2AYL_AC_SL1500_.jpg",
  "rating": {
    "rate": 3.9,
    "count": 120
  }
}
```

The terminal also shows the prompt "Administración de Productos: 1. Consultar todos los productos".

10. Crear una nueva petición para agregar un nuevo producto



11. Modificando un nuevo producto

The image displays two screenshots of the Visual Studio Code interface, showing the development of a REST client and the corresponding Python code for a product management API.

Top Screenshot: The interface shows the REST client for the `UpdateProduct` endpoint. The request is a `PUT` to `https://fakestoreapi.com/products/21`. The request body is a JSON object:

```
1 {
2   "title": "Router",
3   "price": 300,
4   "description": "Ruteador inalámbrico",
5   "image": "https://i.pinimg.com/564x/c9/43/82/c94382beb73e726658c383a3cd74acac.jpg",
6   "category": "dispositivo"
7 }
```

The response status is `200 OK`, with a size of `182 Bytes` and a time of `1.11 s`. The response body is a JSON object:

```
1 {
2   "id": 21,
3   "title": "Router",
4   "price": 300,
5   "description": "Ruteador inalámbrico",
6   "image": "https://i.pinimg.com/564x/c9/43/82/c94382beb73e726658c383a3cd74acac.jpg",
7   "category": "dispositivo"
8 }
```

The bottom panel shows the terminal output, which includes the command `curl -X PUT -H 'Content-Type: application/json' -d '{ "title": "Router", "price": 300, "description": "Ruteador inalámbrico", "image": "https://i.pinimg.com/564x/c9/43/82/c94382beb73e726658c383a3cd74acac.jpg", "category": "dispositivo" }' https://fakestoreapi.com/products/21` and the output `h=120&name=Signal-What-is-a-Router-and-What-Does-it-Do-Hero.jpg"`.

Bottom Screenshot: The interface shows the Python code for the `UpdateProduct` endpoint. The code is as follows:

```
43 def UpdateProduct():
44     url_base = 'https://fakestoreapi.com/products'
45     print("Modificar producto")
46
47     # Solicitar el ID del producto a modificar
48     product_id = input("Ingrese el ID del producto que desea modificar: ")
49
50     # Solicitar los nuevos detalles del producto al usuario
51     title = input("Ingrese el nuevo título del producto: ")
52     price = float(input("Ingrese el nuevo precio del producto: "))
53     description = input("Ingrese la nueva descripción del producto: ")
54     image = input("Ingrese la nueva URL de la imagen del producto: ")
55     category = input("Ingrese la nueva categoría del producto: ")
56
57     # Crear un diccionario con los datos actualizados del producto
58     producto_actualizado = {
59         'title': title,
60         'price': price,
61         'description': description,
62         'image': image,
63         'category': category
64     }
65
66     # Enviar la solicitud PUT con los datos actualizados del producto
67
```

The bottom panel shows the terminal output, which includes the command `python products.py > UpdateProduct` and the output `Selecciona una opción (1-5): 4`.

12. Eliminando un Producto

Visual Studio Code interface showing a REST client request and response for a DELETE operation.

Request:

```
DELETE https://fakestoreapi.com/products/20
```

Response:

```
{
  "id": 20,
  "title": "DANVOUY Womens T Shirt Casual Cotton Short",
  "price": 12.99,
  "description": "95%Cotton,5%Spandex, Features: Casual, Short Sleeve, Letter Print,V-Neck,Fashion Tees, The fabric is soft and has some stretch., Occasion: Casual /Office/Beach/School/Home/Street. Season: Spring,summer,Autumn,Winter.",
  "category": "women's clothing",
  "image": "https://fakestoreapi.com/img/61pAEJ4MML_AC_UX679_.jpg",
  "rating": {
    "rate": 3.6,
    "count": 143
  }
}
```

Visual Studio Code interface showing a Python script for a REST client and its output in the terminal.

Python Script:

```
def DeleteProduct():
    url_base = "https://fakestoreapi.com/products"
    print("Eliminación de producto")

    # Solicitar el ID del producto a eliminar
    product_id = input("Ingrese el ID del producto que desea eliminar: ")

    # Enviar la solicitud DELETE
    url = f"{url_base}/{product_id}"
    response = requests.delete(url)

    if response.status_code == 200:
        print("Producto eliminado exitosamente!")
    else:
        print("Error al eliminar el producto:", response.status_code)

def mostrar_menu():
    print("\nAdministración de Productos:")
    print("1. Consultar todos los productos")
    print("2. Consultar un producto en específico")
    print("3. Agregar un nuevo producto")
    print("4. Modificar producto en específico")
    print("5. Eliminar un producto")
    print("6. Salir")
    print("Selecciona una opción (1-5): ")

if __name__ == "__main__":
    mostrar_menu()
    DeleteProduct()
    mostrar_menu()
```

Terminal Output:

```
Administración de Productos:
1. Consultar todos los productos
2. Consultar un producto en específico
3. Agregar un nuevo producto
4. Modificar producto en específico
5. Eliminar un producto
6. Salir
Selecciona una opción (1-5): 5
Eliminación de producto
Ingrese el ID del producto que desea eliminar: 20
Producto eliminado exitosamente!

Administración de Productos:
1. Consultar todos los productos
2. Consultar un producto en específico
3. Agregar un nuevo producto
4. Modificar producto en específico
5. Eliminar un producto
6. Salir
Selecciona una opción (1-5):
```