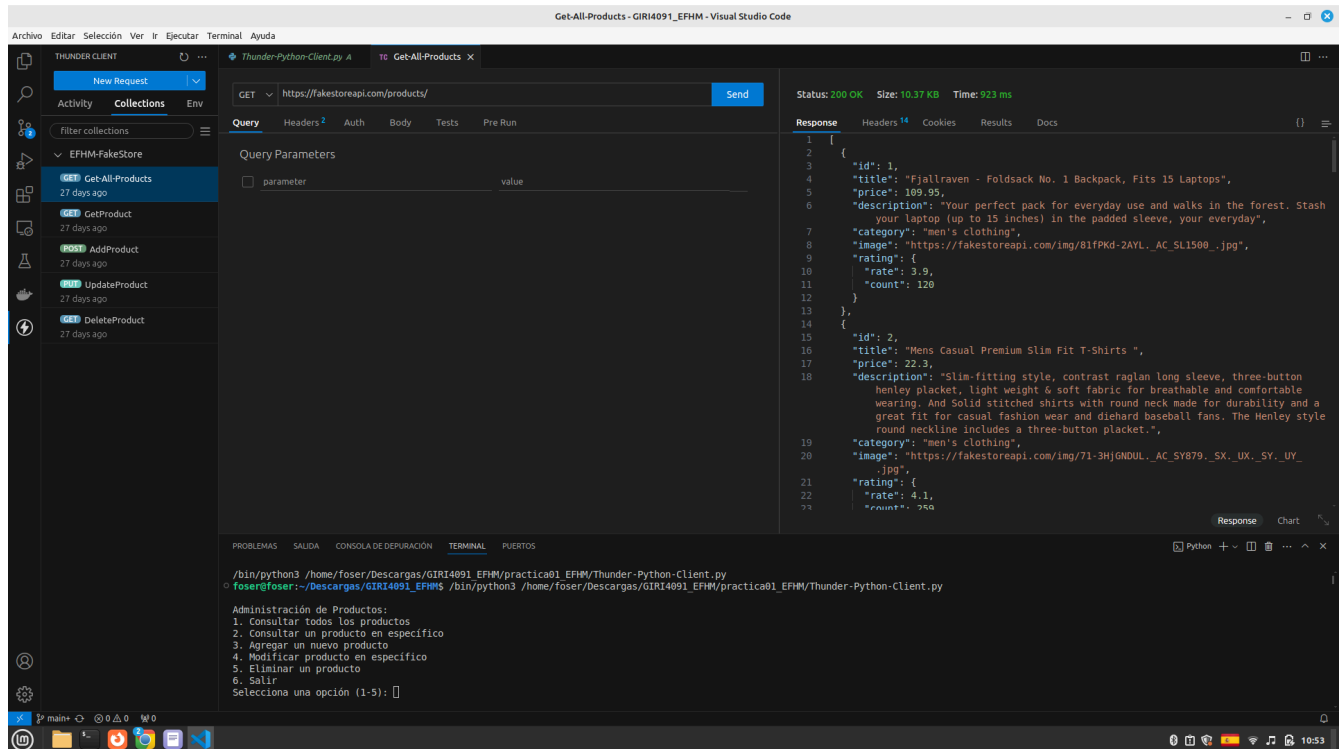
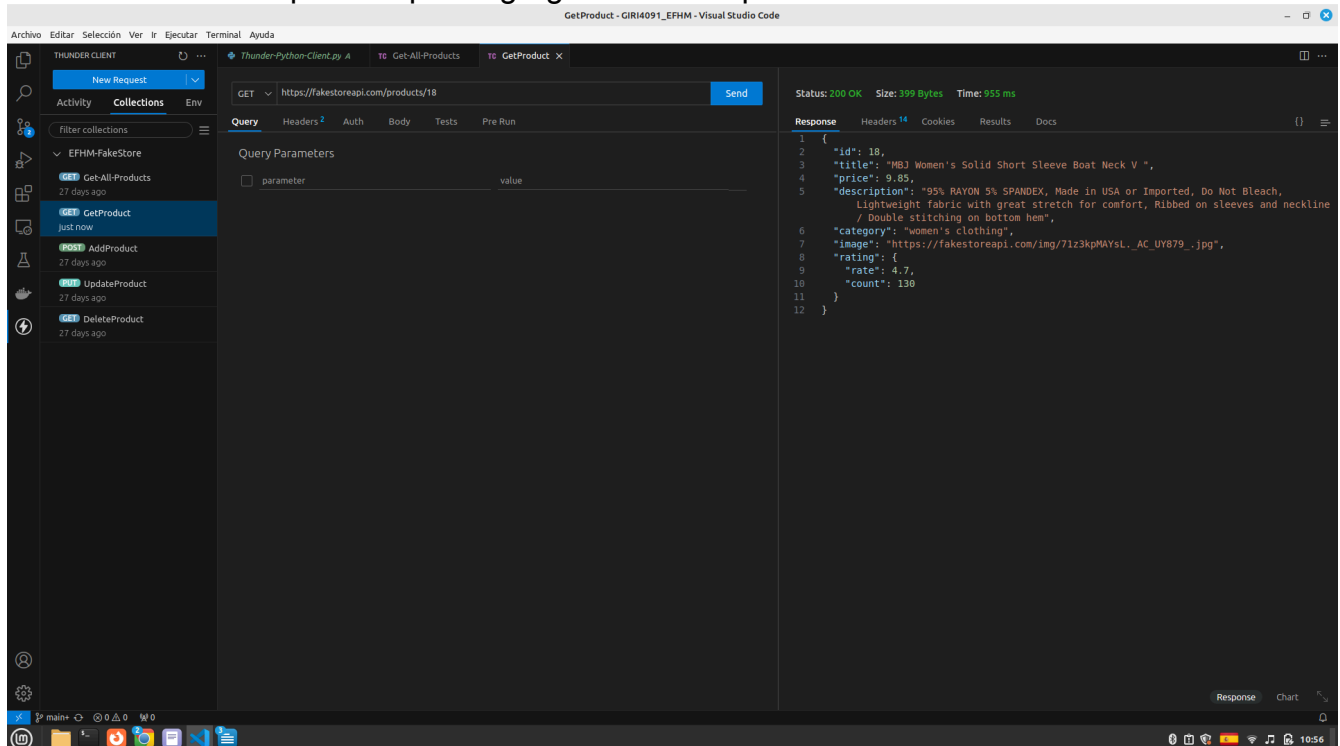


Practica 01

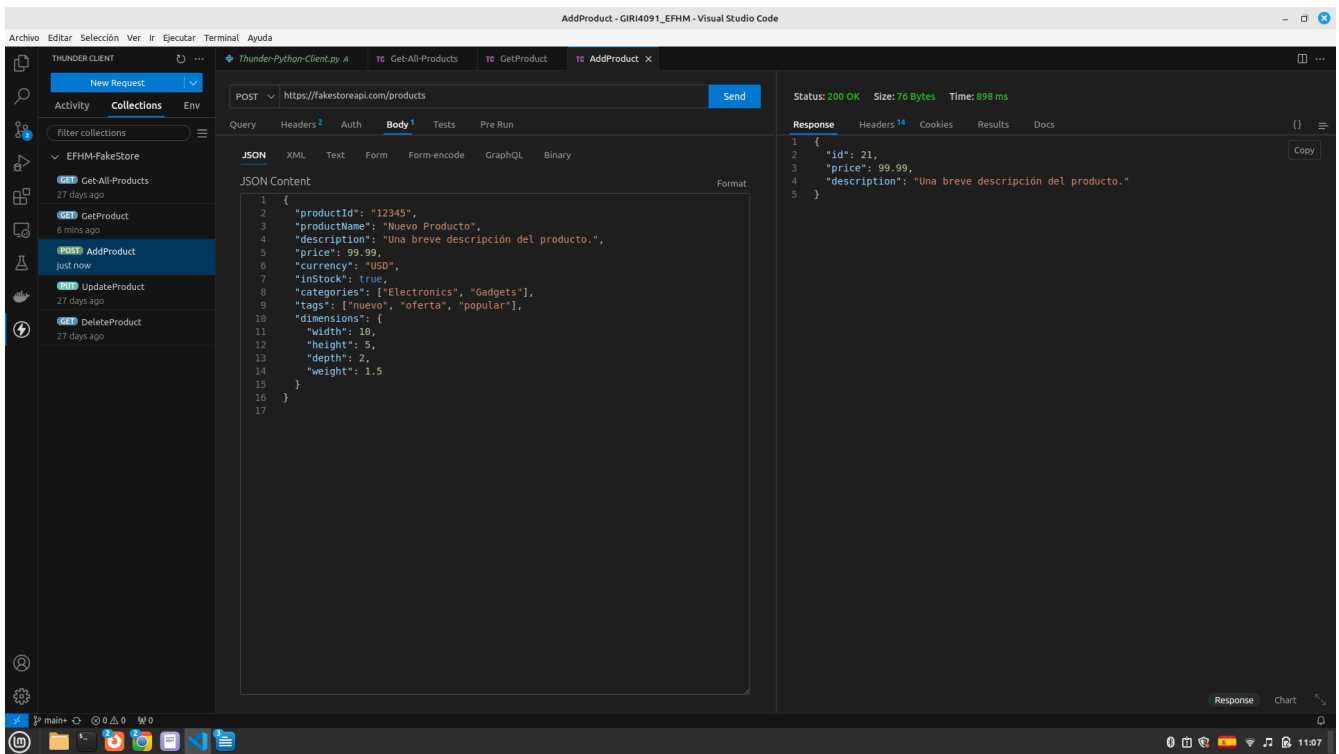
1. Agregar una nueva petición llamada GetAll-Products con el método GET.



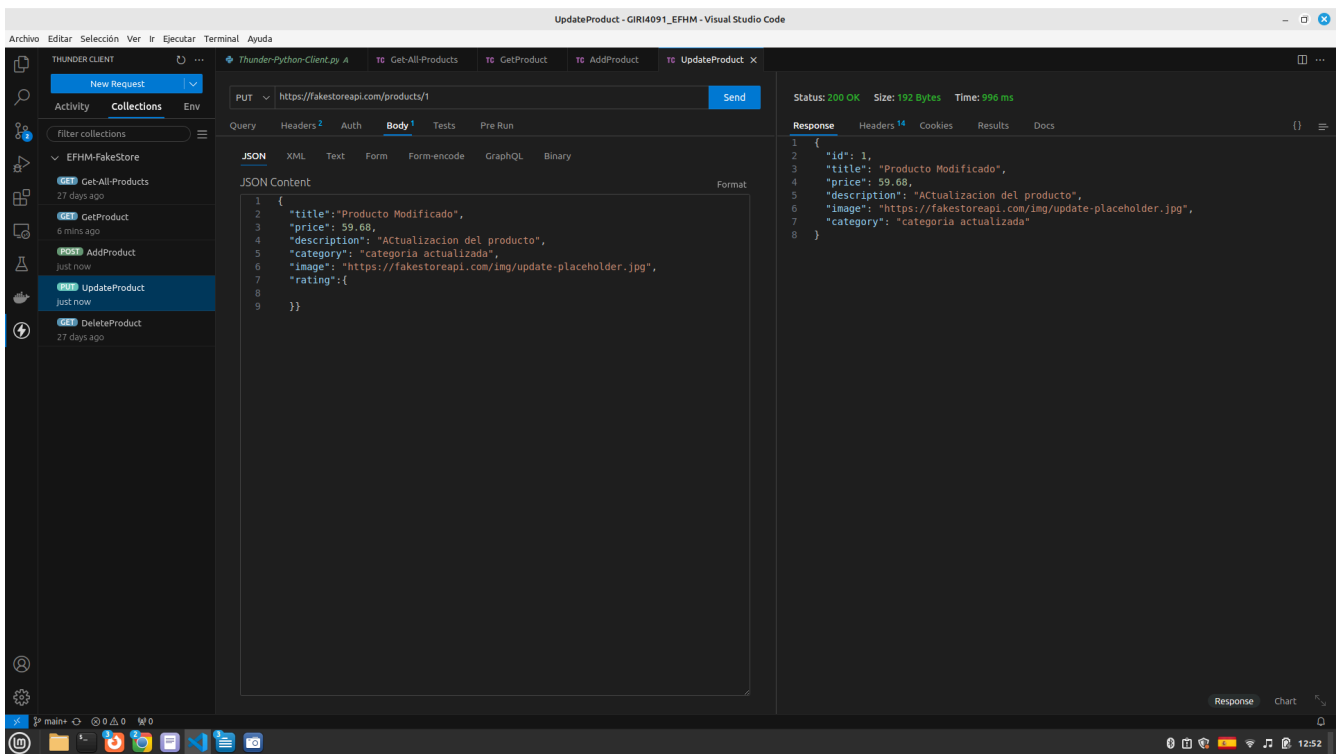
2. Crear una nueva petición para agregar un nuevo producto



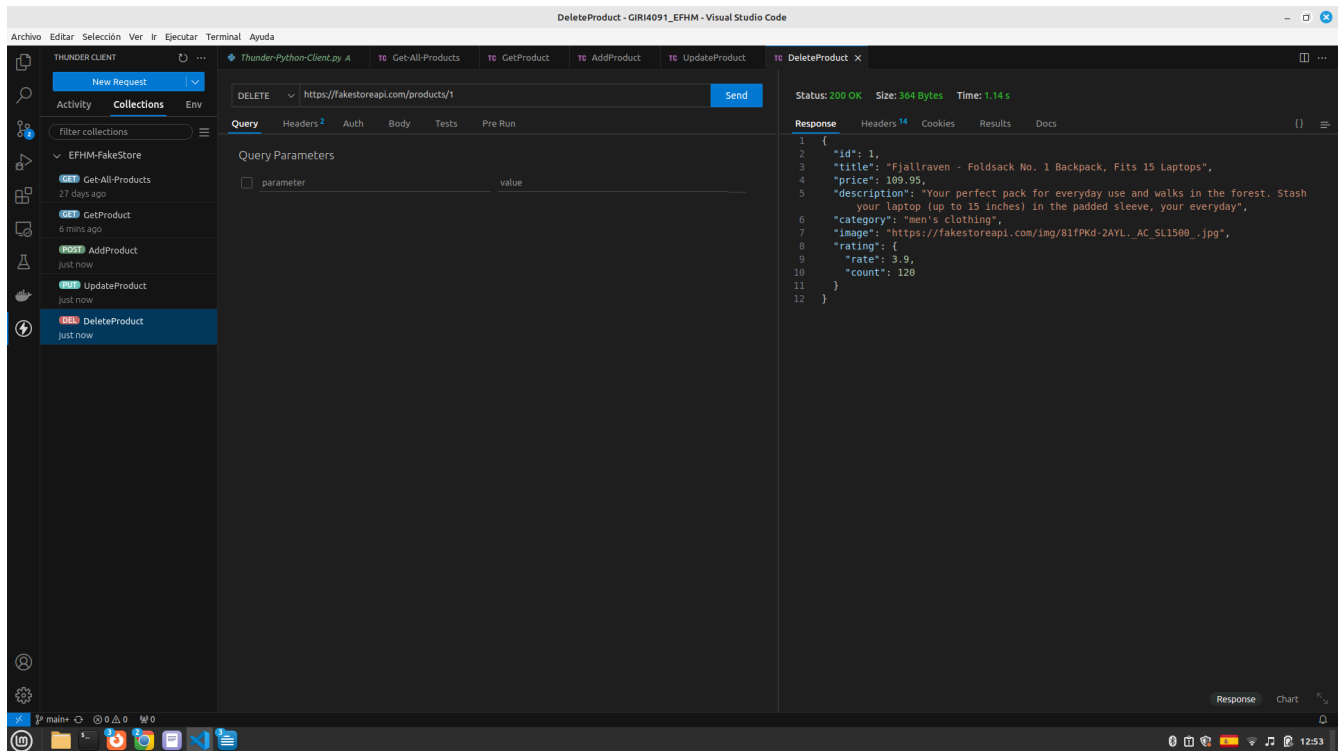
3. Modificando un nuevo producto



4. Modificar un Producto



5.- Eliminando un Producto.



6.- Programa python de api requests

Para mejorar tu código, te sugiero hacer algunos ajustes, incluyendo el uso de bloques `try-except` para manejar errores y asegurar que las operaciones de consulta, adición, modificación y eliminación de productos sean más robustas. Aquí tienes una versión mejorada del código:

```
python
import json
import requests
```

```
def show_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")
```

```
def GetAllProducts():
    try:
        url = "https://fakestoreapi.com/products"
        response = requests.get(url)
        response.raise_for_status()
        json_formateado = json.dumps(response.json(), indent=4, ensure_ascii=False)
```

```

    print("\nListado de productos:\n")
    print(json_formateado)
except requests.exceptions.RequestException as e:
    print(f"Error al consultar productos: {e}")

```

```

def GetProduct():
    noProduct = input("Ingresa el valor del número del producto: ")
    url = f"https://fakestoreapi.com/products/{noProduct}"
    try:
        response = requests.get(url)
        response.raise_for_status()
        json_formateado = json.dumps(response.json(), indent=4, ensure_ascii=False)
        print("\nListado de productos\n")
        print(json_formateado)
        print("\nProducto consultado exitosamente :D")
    except requests.exceptions.HTTPError:
        print("\nProducto no encontrado ;(")
    except requests.exceptions.RequestException as e:
        print(f"Error al consultar el producto: {e}")

```

```

def AddProduct():
    print("\nAgregar producto\n")
    titleProduct = input("Ingresa el título del producto:\n")
    priceProduct = input("Ingresa el precio del producto:\n")
    descriptionProduct = input("Ingresa la descripción del producto:\n")
    categoryProduct = input("Ingresa la categoría del producto:\n")

```

```

    payload = {
        "title": titleProduct,
        "price": priceProduct,
        "description": descriptionProduct,
        "category": categoryProduct,
        "image": "https://fakestoreapi.com/img/placeholder.jpg",
        "rating": {
            "rate": 4.5,
            "count": 10
        }
    }

```

```

headers = {"Content-Type": "application/json"}

```

```

try:
    response = requests.post("https://fakestoreapi.com/products", json=payload,
headers=headers)
    response.raise_for_status()
    print("\nProducto creado exitosamente :D")
    print(response.json())

```

```
except requests.exceptions.RequestException as e:  
    print(f"\nError al crear el producto: {e}")
```

```
def UpdateProduct():
```

```
    noProduct = input("\nIngresa el número de producto a cambiar:\n")  
    url = f"https://fakestoreapi.com/products/{noProduct}"
```

```
    titleProduct = input("Ingresa el título del producto:\n")  
    priceProduct = input("Ingresa el precio del producto:\n")  
    descriptionProduct = input("Ingresa la descripción del producto:\n")  
    categoryProduct = input("Ingresa la categoría del producto:\n")
```

```
    payload = {  
        "title": titleProduct,  
        "price": priceProduct,  
        "description": descriptionProduct,  
        "category": categoryProduct,  
        "image": "https://fakestoreapi.com/img/placeholder.jpg",  
        "rating": {  
            "rate": 4.5,  
            "count": 10  
        }  
    }  
}
```

```
headers = {"Content-Type": "application/json"}
```

```
try:
```

```
    response = requests.put(url, json=payload, headers=headers)  
    response.raise_for_status()  
    print("\nProducto actualizado exitosamente :D")  
    print(response.json())
```

```
except requests.exceptions.HTTPError:
```

```
    print("\nProducto no encontrado ;(")
```

```
except requests.exceptions.RequestException as e:
```

```
    print(f"\nError al actualizar el producto: {e}")
```

```
def DeleteProduct():
```

```
    noProduct = input("\nIngresa el número de producto a eliminar:\n")  
    url = f"https://fakestoreapi.com/products/{noProduct}"
```

```
try:
```

```
    response = requests.delete(url)  
    response.raise_for_status()  
    print("\nProducto eliminado exitosamente :D")  
    print(response.json())
```

```
except requests.exceptions.HTTPError:
```

```
    print("\nProducto no encontrado ;(")
```

```

except requests.exceptions.RequestException as e:
    print(f"\nError al eliminar el producto: {e}")

def main():
    while True:
        show_menu()
        choice = input("Seleccione una opción: ")
        if choice == '1':
            GetAllProducts()
        elif choice == '2':
            GetProduct()
        elif choice == '3':
            AddProduct()
        elif choice == '4':
            UpdateProduct()
        elif choice == '5':
            DeleteProduct()
        elif choice == '6':
            print("Saliendo...")
            break
        else:
            print("Opción no válida. Intente de nuevo.")

if __name__ == "__main__":
    main()
...

```

Mejoras realizadas:

1. **Errores manejados con `try-except`:** Se ha incluido el manejo de excepciones para capturar errores durante las solicitudes HTTP.
2. **Consistencia en las URLs:** La URL del API es ahora dinámica según el número de producto.
3. **Interfaz de usuario clara:** Un menú de opciones para navegar por las diferentes funcionalidades.
4. **Funciones organizadas y modulares:** Cada funcionalidad se encapsula en su propia función, lo que facilita la lectura y el mantenimiento del código.
5. **Validación de opciones del menú:** Se verifica que la opción elegida sea válida.

Con este código, puedes manejar de forma más segura los errores que puedan surgir al interactuar con la API.