Guía de Pruebas de API: cURL y REST Client (VS Code)

Este documento sirve como referencia técnica para probar los endpoints de la API de FastAPI definida en server_modulo4.py.
Se cubren dos métodos: la herramienta de línea de comandos cURL y la extensión REST Client para Visual Studio Code.
Supuestos:

- El servidor FastAPI se está ejecutando localmente: uvicorn server_modulo4:app --reload
- La URL base de la API es: http://127.0.0.1:8000

Parte 1: Pruebas con cURL

cURL (Client URL) es una herramienta de línea de comandos para transferir datos mediante URLs. Es una forma estándar de interactuar con endpoints HTTP desde una terminal.

Sintaxis de Comandos Clave

- -X [VERBO]: Especifica el método HTTP (ej. -X POST, -X PUT, -X DELETE). Si se omite, el valor por defecto es GET.
- -H "[Cabecera]: [Valor]": Pasa una cabecera HTTP (ej. -H "Content-Type: application/json"). Esencial para POST y PUT.
- -d '[Datos]': Envía los datos (el *body*) en la petición. Para JSON, se deben usar comillas simples (') para envolver el JSON y comillas dobles (") en el interior del JSON.

Ejemplos de Peticiones (basados en server_modulo4.py)

1. GET /

Endpoint: read_root()
Propósito: Comprobar que la API está funcionando.
curl http://127.0.0.1:8000/

Respuesta Esperada:
{"status":"API funcionando","database_items":3}

2. GET /items/

Endpoint: read_items()
Propósito: Obtener la lista de todos los ítems.
curl http://127.0.0.1:8000/items/

Respuesta Esperada (basada en bd.json):

[
{"id":1,"name":"Laptop","description":"Un portátil de alto rendimiento","price":1299.99,"tax":129.99},
{"id":2,"name":"Teclado Mecánico","description":"Teclado con switches Cherry MX Red","price":150.0,"tax":31.5},
{"id":3,"name":"Monitor Ultrawide","description":null,"price":499.50,"tax":104.89}
]

3. GET /items/ (con Query Parameter)

Endpoint: read_items(max_price: ...)
Propósito: Filtrar ítems por precio máximo (ej. max_price=500).
Las comillas son importantes para que la terminal interprete la URL correctamente
curl "http://127.0.0.1:8000/items/?max_price=500"

Respuesta Esperada:

[
{"id":2,"name":"Teclado Mecánico","description":"Teclado con switches Cherry MX Red","price":150.0,"tax":31.5},
{"id":3,"name":"Monitor Ultrawide","description":null,"price":499.50,"tax":104.89}
]

4. GET /items/{item_id}

Endpoint: read_item(item_id: int)
Propósito: Obtener un ítem específico por su ID (ej. ID=2).
curl http://127.0.0.1:8000/items/2

Respuesta Esperada:

{"id":2,"name":"Teclado Mecánico","description":"Teclado con switches Cherry MX Red","price":150.0,"tax":31.5}

5. GET /items/{item_id} (Prueba de Error 404)

Endpoint: read_item(item_id: int)
Propósito: Verificar el manejo de errores HTTPException cuando un ID no existe (ej. ID=99).
curl http://127.0.0.1:8000/items/99

Respuesta Esperada:

{"detail":"Item no encontrado"}

6. POST /items/

Endpoint: create_item(item: ItemCreate)
Propósito: Crear un nuevo ítem. Requiere -X POST, cabecera Content-Type y un body JSON que coincida con el modelo ItemCreate.
curl -X POST "http://127.0.0.1:8000/items/" \
-H "Content-Type: application/json" \
-d '{"name": "Mouse Gamer", "description": "Un mouse con 12 botones", "price": 75.50, "tax": 15.86}'

Respuesta Esperada (Código 201 Created):
{"name":"Mouse Gamer","description":"Un mouse con 12
botones","price":75.5,"tax":15.86,"id":4}

7. PUT /items/{item_id}

Endpoint: update_item(item_id: int, item_update: ItemCreate)
Propósito: Reemplazar (actualizar) un ítem existente (ej. ID=1). Requiere -X PUT, cabecera Content-Type y un body ItemCreate.
curl -X PUT "http://127.0.0.1:8000/items/1" \
-H "Content-Type: application/json" \
-d '{"name": "Laptop PRO v2", "description": "Un portátil de ALTO rendimiento", "price": 1499.00, "tax": 314.79}'

Respuesta Esperada:
{"name":"Laptop PRO v2","description":"Un portátil de ALTO
rendimiento","price":1499.0,"tax":314.79,"id":1}

8. DELETE /items/{item_id}

Endpoint: delete_item(item_id: int)
Propósito: Eliminar un ítem por su ID (ej. ID=3).
curl -X DELETE "http://127.0.0.1:8000/items/3"

Respuesta Esperada:
{"detail":"Item eliminado correctamente","id":3}

Parte 2: Pruebas con "REST Client" (Plugin de VS Code)

Esta es una alternativa a Postman que se integra directamente en VS Code.

1. Nombre e Instalación del Plugin

- **Nombre:** REST Client
- **Autor:** Huachao Mao

Pasos de Instalación:

1. Abrir Visual Studio Code.
2. Navegar a la pestaña de **Extensiones** (Icono de cuadrados o Ctrl+Shift+X).
3. Buscar REST Client.
4. Seleccionar la extensión de Huachao Mao y hacer clic en **Instalar**.

2. Modo de Uso

REST Client funciona leyendo archivos de texto plano con la extensión .http o .rest.

1. Crear un nuevo archivo en el proyecto (ej. peticiones.http).
2. Escribir las peticiones en este archivo usando una sintaxis simple (ver ejemplos).
3. VS Code añadirá automáticamente un enlace de texto Send Request encima de cada petición.
4. Al hacer clic en Send Request, la respuesta del servidor se abrirá en un panel lateral.

3. Fichero de Ejemplo (peticiones.http)

Este bloque de código contiene todas las peticiones cURL anteriores, adaptadas al formato de REST Client. Se puede copiar y pegar en un archivo .http.

```
# Define la URL base de la API como una variable
@baseUrl = [http://127.0.0.1:8000](http://127.0.0.1:8000)

###
# 1. GET /
# Endpoint: read_root()
GET {{baseUrl}}/

###
# 2. GET /items/
# Endpoint: read_items()
GET {{baseUrl}}/items/

###
# 3. GET /items/ (con Query Parameter)
# Endpoint: read_items(max_price: ...)
GET {{baseUrl}}/items/?max_price=500

###
# 4. GET /items/{item_id}
# Endpoint: read_item(item_id: int)
GET {{baseUrl}}/items/2

###
```

```
# 5. GET /items/{item_id} (Prueba de Error 404)
# Endpoint: read_item(item_id: int)
GET {{baseUrl}}/items/99

###
# 6. POST /items/
# Endpoint: create_item(item: ItemCreate)
# (Nota: La cabecera Content-Type se añade automáticamente
#  cuando se detecta un body JSON)
POST {{baseUrl}}/items/
Content-Type: application/json

{
  "name": "Mouse Gamer",
  "description": "Un mouse con 12 botones",
  "price": 75.50,
  "tax": 15.86
}

###
# 7. PUT /items/{item_id}
# Endpoint: update_item(item_id: int, item_update: ItemCreate)
PUT {{baseUrl}}/items/1
Content-Type: application/json

{
  "name": "Laptop PRO v2",
  "description": "Un portátil de ALTO rendimiento",
  "price": 1499.00,
  "tax": 314.79
}

###
# 8. DELETE /items/{item_id}
# Endpoint: delete_item(item_id: int)
DELETE {{baseUrl}}/items/3
```