



---

# EXAMEN UNIDAD TEMATICA

---

TECNM CAMPUS COLIMA



**5 DE SEPTIEMBRE DE 2023**  
**ULISES RAFAEL GUTIERREZ FLORES**  
**APPS PARA IOT**

## Introducción

En la era digital actual, la comunicación entre aplicaciones y servidores desempeña un papel crucial en el mundo de la informática. El código presente se centra en la creación de un servidor web Python personalizado que es capaz de manejar solicitudes POST y responder de manera dinámica a través del protocolo HTTP.

La implementación del servidor se basa en el lenguaje de programación Python, conocido por su simplicidad y versatilidad.

Se utiliza la biblioteca ``http. Server`` de Python para gestionar las solicitudes HTTP entrantes. Esta biblioteca proporciona una base sólida para crear un servidor web personalizado.

El formato JSON se emplea para el intercambio de datos entre el cliente y el servidor. Las solicitudes POST envían datos en formato JSON, que luego se analizan y manipulan en el servidor.

Para probar y realizar solicitudes POST al servidor, se utiliza Postman, una herramienta que simplifica la creación y el envío de solicitudes HTTP personalizadas.

El protocolo HTTP es fundamental en la comunicación entre clientes y servidores web. En este caso, las solicitudes y respuestas HTTP son la columna vertebral de la interacción entre Postman y el servidor Python.

Este código representa un ejemplo de cómo implementar un servidor web en Python capaz de recibir solicitudes POST, procesar datos en formato JSON y responder de manera eficaz.

En las siguientes imágenes se muestra el código que previamente se realizó y checo en clase, utilizando el lenguaje de programación Python y Visual Studio Code. Es un programa en el cual se utilizan acciones para descender o aumentar un numero almacenado en un contador.

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import json

contador = 11

class MyHTTPRequestHandler(BaseHTTPRequestHandler):
    def _set_response(self, content_type="text/plain"):
        self.send_response(200)
        self.send_header("Content-type", content_type)
        self.end_headers()

    def do_GET(self):
        self._set_response()
        respuesta = "El valor es: " + str(contador)
        self.wfile.write(respuesta.encode())

    def do_POST(self):
        content_length = int(self.headers["Content-length"])
        post_data = self.rfile.read(content_length)

        body_json = json.loads(post_data.decode())
        print(body_json['action'])

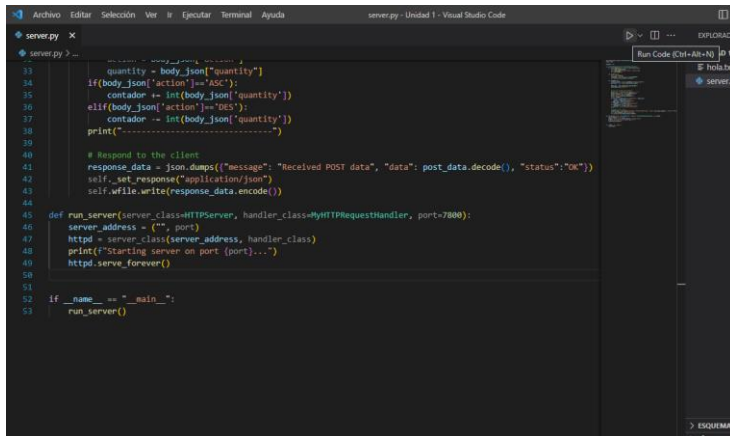
        # Print the complete HTTP request
        print("\n----- Incoming POST Request -----")
        print(f"Requestline: {self.requestline}")
        print(f"Headers: {self.headers}")
        print(f"Body: {post_data.decode()}")
        global contador
        if "action" in body_json and "quantity" in body_json:
            action = body_json["action"]

            quantity = body_json["quantity"]
            if body_json["action"] == "ASC":
                contador += int(body_json["quantity"])
            elif body_json["action"] == "DES":
                contador -= int(body_json["quantity"])
            print("-----")

        # Respond to the client
        response_data = json.dumps({"message": "Received POST data", "data": post_data.decode(), "status": "OK"})
        self._set_response("application/json")
        self.wfile.write(response_data.encode())

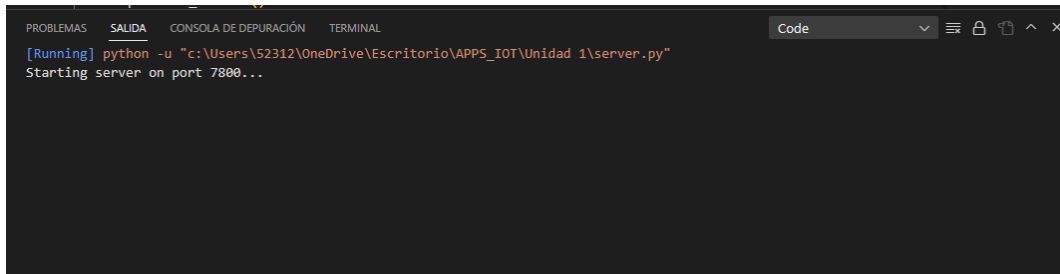
def run_server(server_class=HTTPServer, handler_class=MyHTTPRequestHandler, port=7800):
    server_address = ("", port)
    httpd = server_class(server_address, handler_class)
    print(f"Starting server on port {port}...")
    httpd.serve_forever()

if __name__ == "__main__":
    run_server()
```



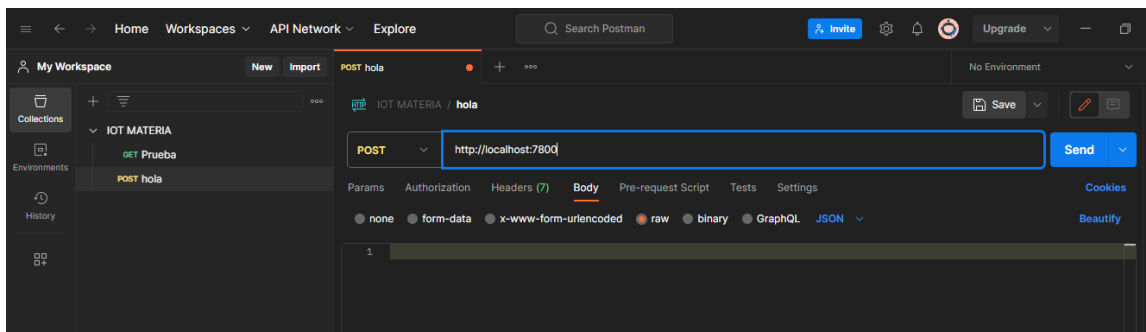
```
server.py: Unidad 1 - Visual Studio Code
server.py
34         quantity = body_json["quantity"]
35         if body_json["action"] == "ASC":
36             contador += int(body_json["quantity"])
37         elif body_json["action"] == "DES":
38             contador -= int(body_json["quantity"])
39         print("-----")
40
41         # Respond to the client
42         response_data = json.dumps({"message": "Received POST data", "data": post_data.decode(), "status": "OK"})
43         self._set_response("application/json")
44         self.wfile.write(response_data.encode())
45
46     def run_server(server_class=HTTPServer, handler_class=MyHTTPRequestHandler, port=7800):
47         server_address = ("", port)
48         httpd = server_class(server_address, handler_class)
49         print(f"Starting server on port {port}...")
50         httpd.serve_forever()
51
52     if __name__ == "__main__":
53         run_server()
```

Al ejecutar el programa se muestra en la consola que está corriendo o se inició con el servidor con el puerto 7800.

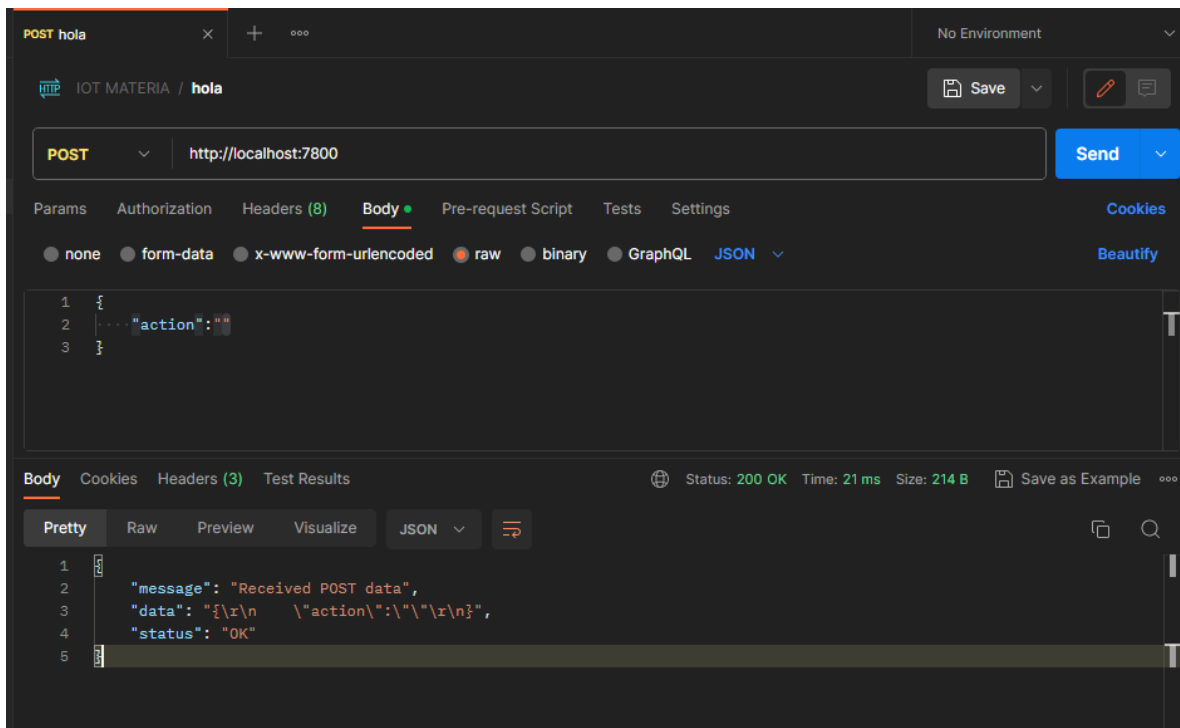


```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL Code
[Running] python -u "c:\Users\52312\OneDrive\Escritorio\APPS_IOT\Unidad 1\server.py"
Starting server on port 7800...
```

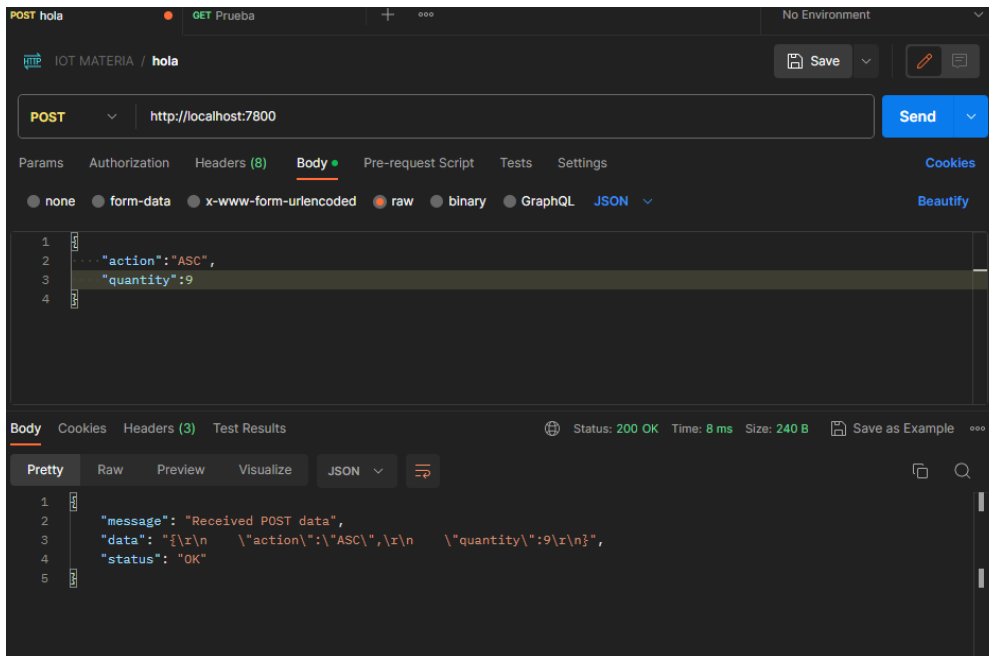
Al pasarnos a Postman ponemos la URL: `http://localhost:7800`



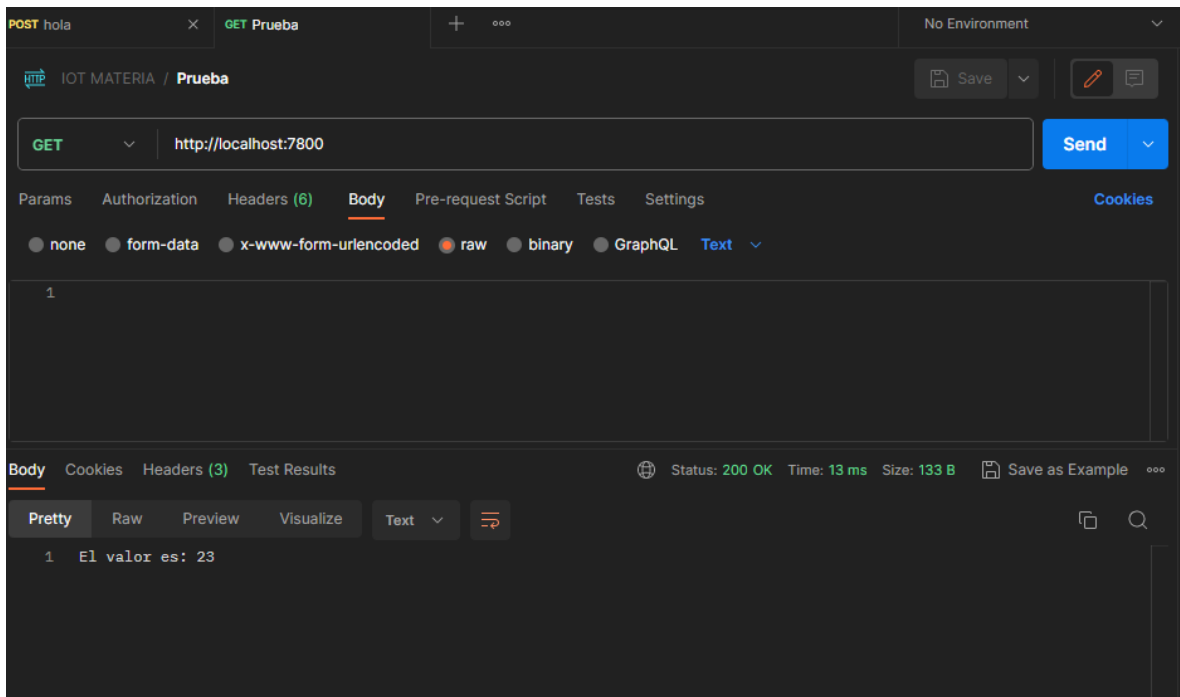
Al presionar el botón de Send aparece la acción y en la parte inferior nos indica que si se ha recibido o no la solicitud.



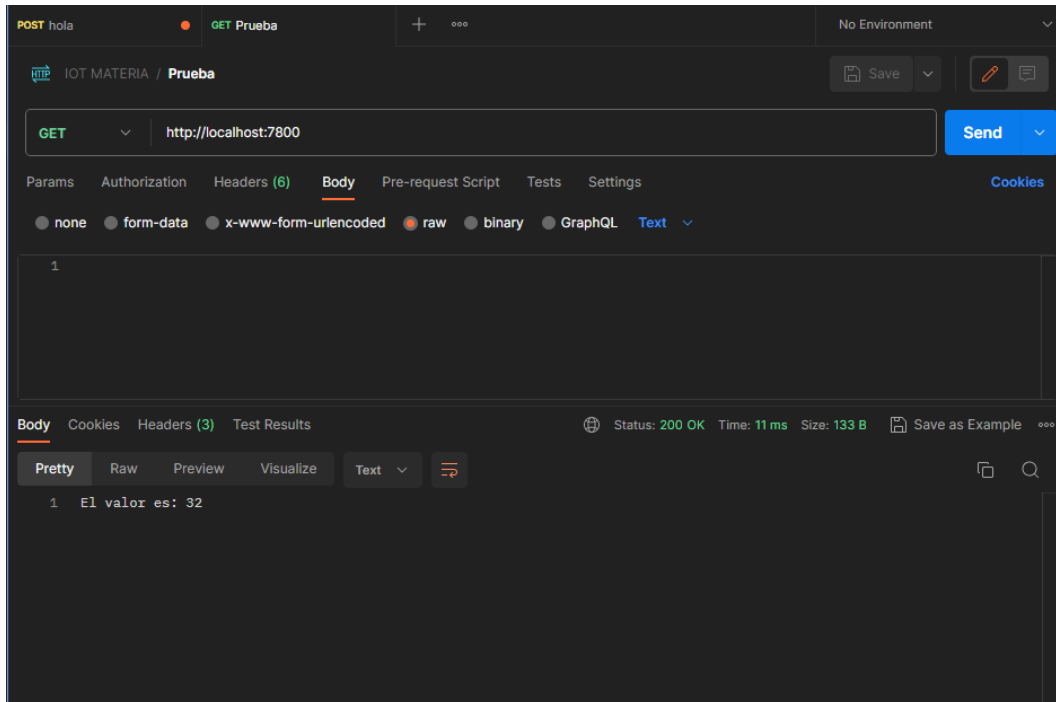
Al darle la acción ASC la cual tengo que es para aumentar el contador y el número 9, al presionar Send nos manda al contador el numero 9 sumándolo.



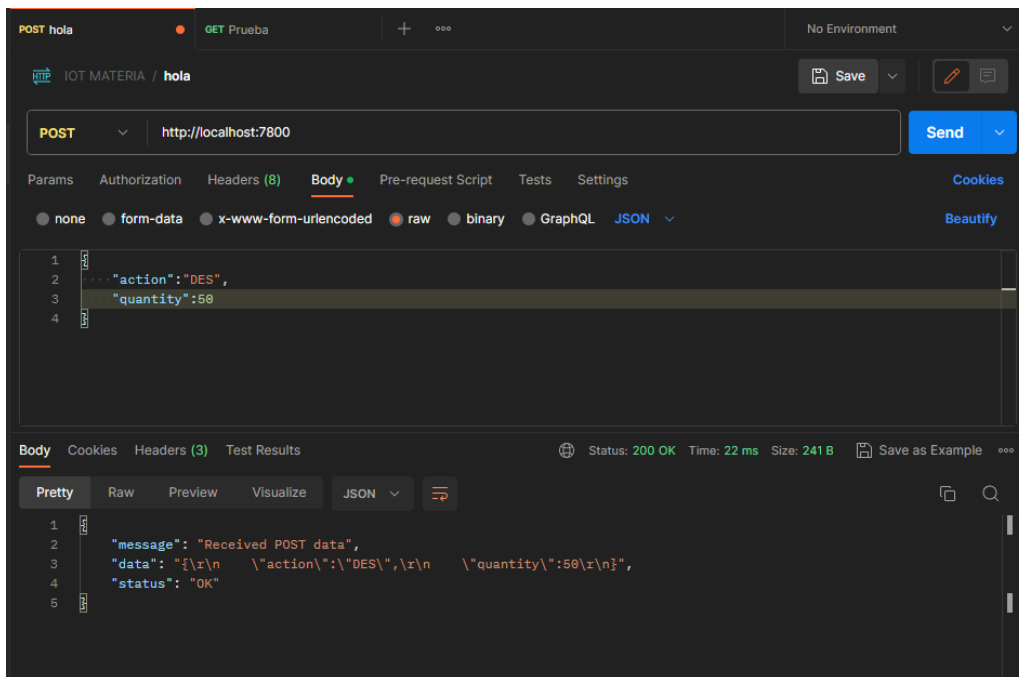
En esta parte se puede observar el contador antes de ser afectado por el POST que le ha sido mandado.



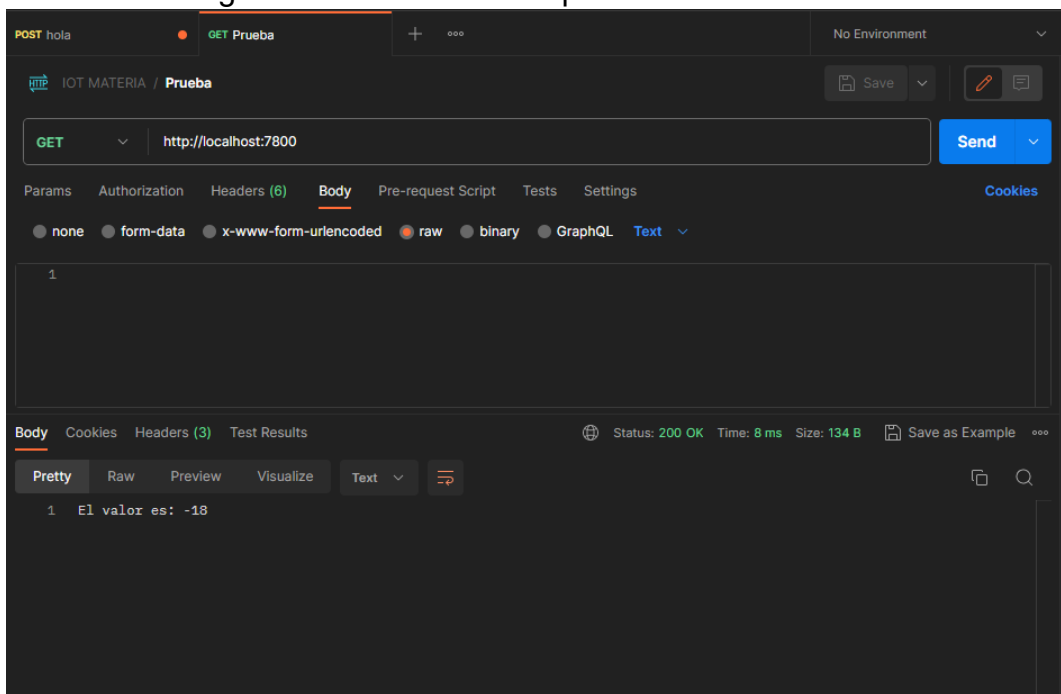
En esta imagen se muestra que efectivamente se ha sumado al contador el número deseado.



Al igual que con el ASC vamos a mandar otra acción, pero a diferencia de la anterior esta sirve para disminuir o descender el contador.



Al ver esta imagen nos damos cuenta que si ha sido afectado el contador.



## **Conclusión**

Hypertext Transfer Protocol es la infraestructura esencial que permite la navegación web, la transmisión de datos y la interacción en línea. Facilita la entrega de recursos, la solicitud de información y la actualización de contenido en tiempo real en la web. Su impacto es innegable, ya que sustenta las aplicaciones web, las API y la transferencia de datos en todo el mundo. La flexibilidad y la estandarización son fundamentales para el funcionamiento eficiente de internet.

Python, como lenguaje de programación, ofrece una excelente combinación de eficiencia. Su capacidad para desarrollar rápidamente aplicaciones web y servicios es una de sus características que lo han hecho popular entre los desarrolladores. se utiliza para crear un servidor web personalizado de manera elegante y eficiente, demostrando su capacidad para gestionar solicitudes http y JSON de manera efectiva.

Postman es una herramienta invaluable para los desarrolladores y probadores. Simplifica la interacción con servidores web, permitiendo la creación y el envío de solicitudes personalizadas de una manera fácil y comprensible. Su interfaz es muy intuitiva y su capacidad para automatizar pruebas son cualidades destacadas que agilizan el proceso de desarrollo y garantizan la calidad de aplicaciones.