UNIVERSIDAD NACIONAL DE HURLINGHAM

REDES DE COMPUTADORAS

TRABAJO PRÁCTICO N.º 7

ALUMNOS

Sebastian Castignani DNI: 47.065.138

PROFESOR

Xoana Prior

GRUPO

N.º 5

- **Objetivo del TP:** Obtener los conocimientos necesarios para poder configurar un socket en Python para crear una conexión Cliente/Servidor

- **Materiales:** Compilador Python y el comando Netstat de Windows.

- **Escenario:** Generar 2 programas en Python, uno de un servidor y otro de un cliente. Realizar la conexión entre los dos y luego comprobar las conexiones abiertas mediante el comando netstat.

- **Actividad:**

1. Crear un programa de conexión de socket (servidor)

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("127.0.0.1", 7777))
s.listen(1)

print ("Servidor de Chat\n")

while True:
        print ("Esperando conexión del cliente...")
        sc, addr = s.accept()
        print ("Cliente conectado desde: ", addr)

        while True:
                recibido = sc.recv(1024)
                if recibido == "quit":
                        break
                print ("Recibido: ", recibido)

                nuestra_respuesta = "Hola cliente, yo soy el servidor"
                sc.send(nuestra_respuesta.encode('utf-8'))

print ("Adios")
sc.close()
s.close()
```

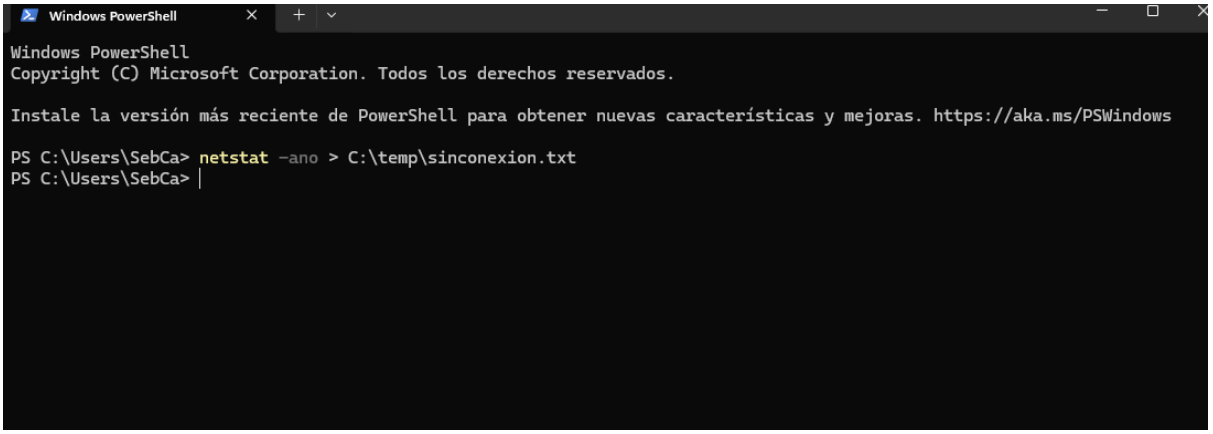
2. Crear un programa de conexión de socket (cliente).

```
import socket

socket_cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket_cliente.connect(("127.0.0.1", 7777))

while True:
        mensaje = str(input(">> "))
        socket_cliente.send(mensaje.encode('utf-8'))

        recibido = socket_cliente.recv(1024)
        print("Recibido: ", recibido)

print ("Adios")
socket_cliente.close()
```

3. Ejecutar el siguiente comando cmd -> “netstat –ano > C:\temp\sinconexion.txt”
4. Ejecutar primero el programa de servidor y luego el programa de cliente.
5. Verificar que la conexión se haya establecido de forma correcta.
6. Ejecutar el siguiente comando cmd -> “netstat –ano > C:\temp\conconexion.txt”
7. Comparar los TXT sinconexion y conconexion. ¿Qué diferencias encontramos?
8. Desarrollar una conclusión del trabajo realizado.

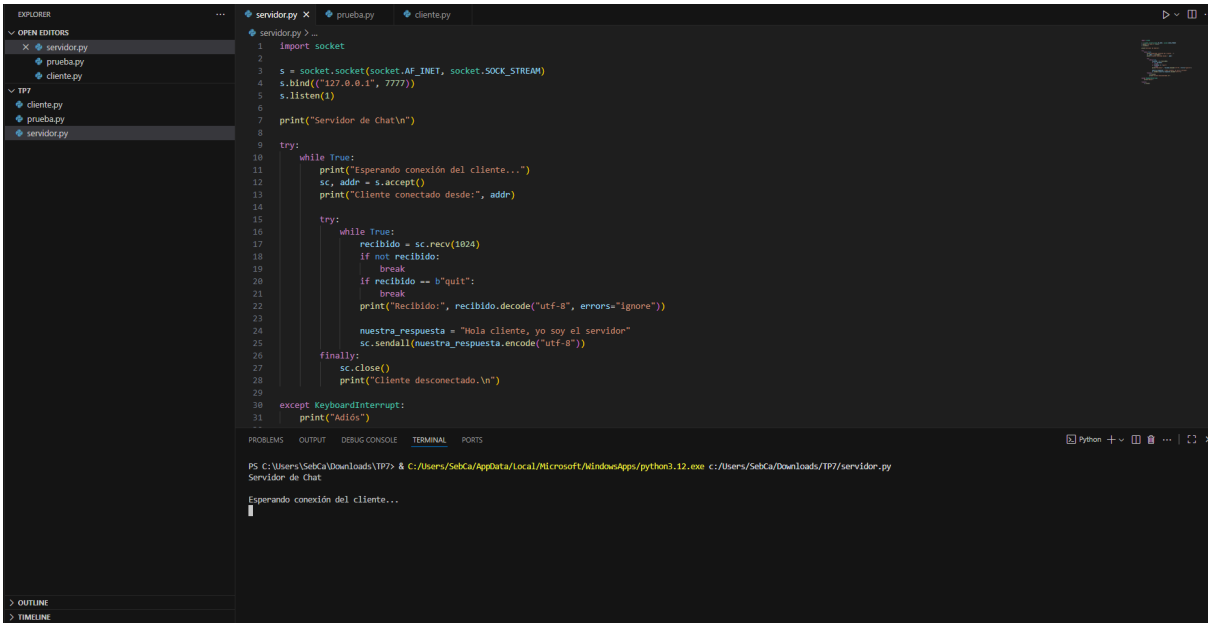
3.

primero creamos el archivo sin conexion en la carpeta temp , este nos genera un txt con muchas direcciones y conexiones

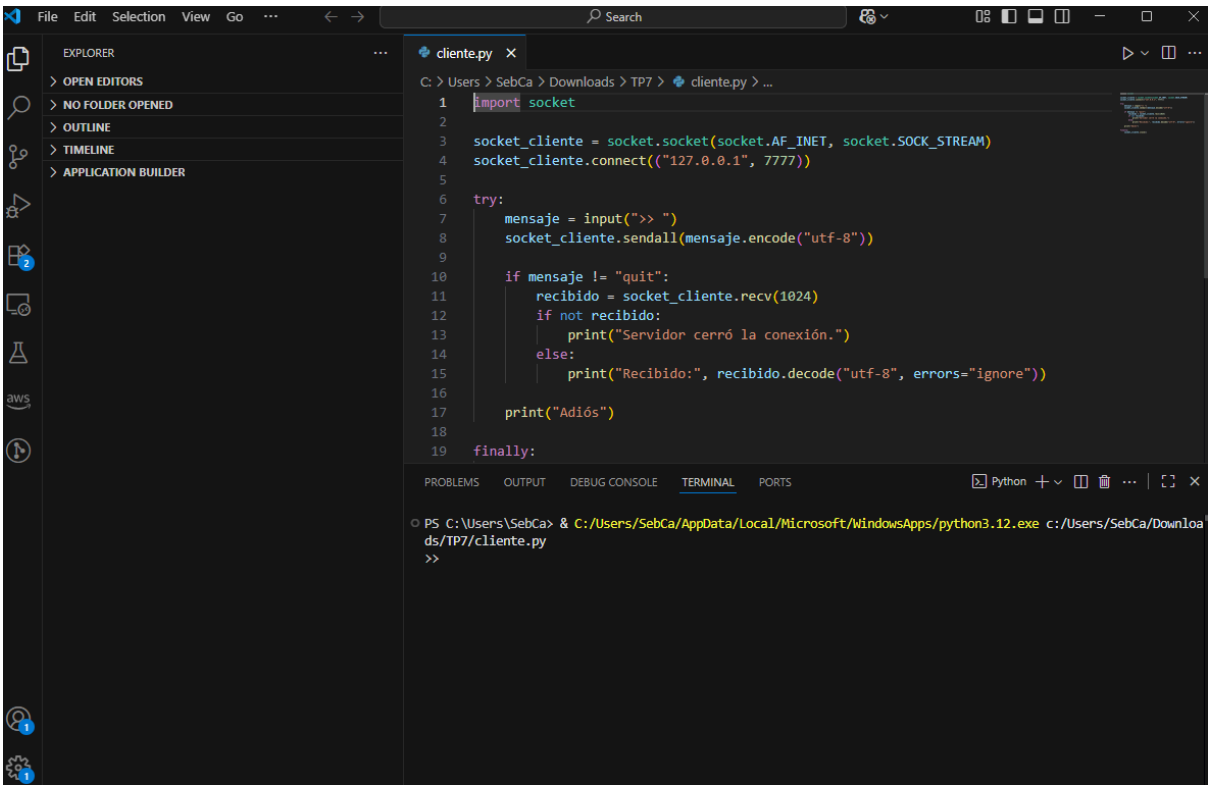


4

luego ejecutamos en la terminal de visual studio el codigo de python del servidor en el cual nos aparece que este esperando la conexión del cliente



luego en una terminal diferente ejecutamos el código de cliente , el cual se comunicara con el servidor



5

una vez ejecutado estos dos en la terminal del servidor nos dice que se establece exitosamente una conexión y se conectó el cliente apareciendo el siguiente texto

- Cliente conectado desde: ('127.0.0.1', 59475)

servidor.py

```python
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)  # Para reutilizar la dirección
s.bind(("127.0.0.1", 7777))
s.listen(1)

print("Servidor de Chat\n")

try:
    while True:
        print("Esperando conexión del cliente...")
        sc, addr = s.accept()
        print("Cliente conectado desde:", addr)

        try:
            while True:
                recibido = sc.recv(1024)
                if not recibido:
                    break
                # Mejorar la comparación de "quit" - usar strip() para quitar saltos de línea
                if recibido.strip() == b"quit":
                    break
                print("Recibido:", recibido.decode("utf-8", errors="ignore"))

                nuestra_respuesta = "Hola cliente, yo soy el servidor"
                sc.sendall(nuestra_respuesta.encode("utf-8"))
        finally:
            sc.close()
            print("Cliente desconectado.\n")
```

```
PS C:\Users\SebCa\Downloads\TP7> & C:/Users/SebCa/AppData/Local/Microsoft/WindowsApps/python3.12.exe c:/Users/SebCa/Downloads/TP7/servidor.py
Servidor de Chat

Esperando conexión del cliente...
Cliente conectado desde: ('127.0.0.1', 59475)
```

6

creamos el archivo con conexión de la misma manera que habiamos creado el de sin conexión

```
PS C:\Users\SebCa> netstat -ano > C:\temp\sinconexion.txt
PS C:\Users\SebCa> netstat -ano > C:\temp\conconexion.txt
PS C:\Users\SebCa>
```

7

Al comparar ambos archivos, encontramos una diferencia que muestra lo que sucede cuando se ejecuta el código de servidor y de cliente .

En el archivo sin conexión.txt , el puerto 7777 no aparece en ninguna parte. Todas las conexiones que vemos son otras aplicaciones que estaban corriendo en ese momento.

En el archivo con conexion.txt , aparecen las siguientes líneas nuevas s:

TCP 127.0.0.1:7777 0.0.0.0:0 LISTENING 18760

TCP 127.0.0.1:7777 127.0.0.1:59475 ESTABLISHED 18760

La primera línea muestra que el servidor está "escuchando" en el puerto 7777, esperando que algún cliente se conecte. La otra línea muestra que en el momento de capturar las conexiones, había un cliente conectado: el servidor y el cliente establecieron una comunicación donde el cliente usó el puerto 59475 para conectarse al puerto 7777 del servidor.

Este trabajo nos permitió observar cómo funciona la comunicación entre un cliente y un servidor, viendo cómo se asigna un puerto específico para esta comunicación de manera similar a como sucede en otras aplicaciones de red. El ejercicio nos mostró los fundamentos de cómo las aplicaciones de nuestro sistema se comunican a través de la red, haciendo más fácil comprender estos procesos que normalmente no vemos.

La herramienta netstat resultó útil para verificar que nuestros programas funcionan correctamente a nivel de red, permitiendo confirmar que el servidor está efectivamente escuchando en el puerto designado y que las conexiones se establecen cuando el cliente se conecta.

En conclusión, este trabajo práctico demostró que cualquier conexión entre dos aplicaciones, ya sea una aplicación grande o los dos códigos que hicimos, el sistema operativo la gestiona de la misma forma, mostrando que el funcionamiento de las comunicaciones de red es el mismo para todos.