

Implementa un sistema básico de chat en red utilizando sockets TCP en Python. El sistema debe constar de dos programas: un **servidor** y un **cliente**.

El **servidor** debe escuchar conexiones entrantes en la dirección 127.0.0.1 y el puerto 12345, aceptar múltiples clientes simultáneamente mediante hilos, almacenar los sockets de los clientes conectados y reenviar a todos ellos cualquier mensaje recibido de un cliente, excepto al que lo envió. Si un cliente se desconecta o ocurre un error, el servidor debe eliminar su socket y cerrar la conexión.

El **cliente** debe conectarse al servidor en la misma dirección y puerto, iniciar un hilo para recibir y mostrar en pantalla los mensajes enviados por otros clientes (a través del servidor), y permitir que el usuario envíe mensajes desde la consola. El cliente debe poder desconectarse escribiendo el comando "salir", cerrando su socket adecuadamente.

Debe incluirse el código fuente de cliente y servidor (**además se entregarán los ficheros fuente**).

```
SERVIDOR:

import socket
import threading

servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
servidor.bind(("127.0.0.1", 12345))
servidor.listen(5)
clientesConectados = []
print("servidor de chat iniciado en 127.0.0.1:12345")
print("esperando clientes...")
def manejarCliente(conn, addr):
    conn.sendall("bienvenido al chat! escribe 'salir' para
salir\n".encode("utf-8"))
    while True:
        try:
            mensaje = conn.recv(1024).decode("utf-8")
            #si el cliente se desconecta o escribe salir
            if not mensaje or mensaje.strip().lower() == "salir":
                break
            print(f"cliente {addr}: {mensaje}")
            #reenviar mensaje a todos los demás clientes
            for cliente in clientesConectados:
                if cliente != conn:
                    cliente.sendall(f"usuario {addr[1]}:
{mensaje}\n".encode("utf-8"))
        except:
            break
```

```

#remover cliente cuando se desconecta
if conn in clientesConectados:
    clientesConectados.remove(conn)
    conn.close()
    print(f"cliente {addr} desconectado")

try:
    while True:
        conn, addr = servidor.accept()
        clientesConectados.append(conn)
        print(f"nuevo cliente conectado: {addr}")

        hiloCliente = threading.Thread(target=manejarCliente, args=(conn,
addr))
        hiloCliente.daemon = True
        hiloCliente.start()

except KeyboardInterrupt:
    print("\ncerrando servidor...")
finally:
    #cerrar todas las conexiones
    for cliente in clientesConectados:
        cliente.close()
    servidor.close()

```

## cliente:

```

import socket
import threading

cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
cliente.connect(("127.0.0.1", 12345))

def recibirMensajes():
    while True:
        try:
            mensaje = cliente.recv(1024).decode("utf-8")
            #si no hay mensaje, servidor cerro conexion
            if not mensaje:
                break
            print(mensaje, end=' ')
        except:
            break

#hilo para recibir mensajes del servidor
hiloRecepcion = threading.Thread(target=recibirMensajes)
hiloRecepcion.daemon = True

```

```
hiloRecepcion.start()
print("conectado al servidor de chat!")
print("escribe tus mensajes:")
try:
    while True:
        mensaje = input()
        cliente.sendall(mensaje.encode("utf-8"))
        #salir si el usuario escribe salir
        if mensaje.lower() == "salir":
            break
except KeyboardInterrupt:
    print("\nsaliendo...")
finally:
    cliente.close()
```

Ejecútalo con un servidor y dos clientes, y muestra una captura con algunos mensajes enviados.

- PS D:\VSC\PY\.Sockets> & C:\Python314\python.exe d:/VSC/PY/Sockets/servidor.py  
servidor de chat iniciado en 127.0.0.1:12345  
esperando clientes...  
nuevo cliente conectado: ('127.0.0.1', 65310)  
nuevo cliente conectado: ('127.0.0.1', 65316)  
cliente ('127.0.0.1', 65310): Hola que tal?  
cliente ('127.0.0.1', 65316): Bien, haciendo una practica del ra3 de python  
cliente ('127.0.0.1', 65310): Genial, espero que la apruebes!  
cliente ('127.0.0.1', 65316): Eso espero!, adios!  
cliente ('127.0.0.1', 65316) desconectado  
[]

```
PS D:\VSC\PY\.Sockets> python cliente.py
conectado al servidor de chat!
escribe tus mensajes:
bienvenido al chat! escribe 'salir' para salir
Hola que tal?
usuario 65316: Bien, haciendo una practica del ra3 de python
Genial, espero que la apruebes!
usuario 65316: Eso espero!, adios!
[]
```

- PS D:\VSC\PY\.Sockets> python cliente.py
conectado al servidor de chat!
escribe tus mensajes:
bienvenido al chat! escribe 'salir' para salir
usuario 65310: Hola que tal?
Bien, haciendo una practica del ra3 de python
usuario 65310: Genial, espero que la apruebes!
Eso espero!, adios!
salir
○ PS D:\VSC\PY\.Sockets> []

## **Normas de entrega y calificación:**

La entrega deberá ser a través de Aula Virtual. Si todavía no se dispone de usuario de EducaMadrid, se permitirá excepcionalmente la entrega por email. Se entregará un documento word que contendrá el enunciado y las soluciones, con el siguiente nombre de fichero:

<GRUPO>\_<APELIDO1>\_<NOMBRE>.<extensión de fichero>

En cada parte delimitada con < y > se tendrá que sustituir según el valor correspondiente. Todo irá en mayúsculas (salvo la extensión) y sin tildes. Por ejemplo, si una persona llamada Ángel Pérez fuera a clase de 1DAW, sería:

1DAW\_PEREZ\_ANGEL.docx

También deberán entregarse los archivos fuente del servidor y cliente.

Rúbrica de evaluación:

Criterio	100%	75%	50%	25%	0%
Funcionamiento del servidor (5 puntos)	El servidor funciona bien, cumpliendo las especificaciones.	El servidor funciona, pero se ha ignorado alguna de las especificaciones.	El servidor funciona parcialmente, pero con fallos.	El programa funciona pero solo para un cliente.	El servidor no funciona o no se ejecuta.
Funcionamiento del cliente (5 puntos)	El cliente funciona bien, cumpliendo las especificaciones.	El cliente funciona, pero se ha ignorado alguna de las especificaciones.	El cliente funciona parcialmente, pero con fallos no bloqueantes.	El cliente funciona parcialmente, pero con fallos bloqueantes.	El cliente no funciona o no se ejecuta.

Además, se tendrán en cuenta los siguientes criterios de calificación adicionales:

- Fecha de entrega: Si la entrega no se realiza a tiempo, la calificación de la práctica será de 0 puntos.
- Adecuación al nombre del fichero: si no cumple el formato de nombre del fichero especificado, se restará 0.5 puntos.
- Entrega adecuada: si no se ha entregado el código fuente, la calificación de la práctica será de 0 puntos. Por cada captura de pantalla de resultado que no se haya incluido, se restará 0.5 puntos.

· Si se detecta copia o que la solución ha sido generada por inteligencia artificial o alguna herramienta similar no autorizada, la calificación será de 0 (tanto para el que ha copiado como para el que se ha dejado copiar, en el caso de copia). No compartas tu solución con nadie. Si tienes dudas sobre si se puede utilizar alguna herramienta, consúltalo con tu profesor antes de usarla. El profesor puede realizar una revisión oral de la práctica con los alumnos que considere que deben explicar su código para corroborar su autoría.