ACTIVIDAD EVALUATIVA EJE 3

Laura Juliana Ramírez Barrera

Fundación Universitaria del Área Andina

Ingeniería De Sistemas – Virtual

Modelos De Programación II - IS - 202460-6a - 61

Deivys Morales

15 de agosto de 2024

INTRODUCCIÓN

Durante el desarrollo de diversas aplicaciones y sistemas de red, los hilos y sockets juegan un papel fundamental para garantizar un gran rendimiento y la capacidad de comunicación entre diferentes procesos. Mientras los hilos permiten la ejecución de múltiples tareas al tiempo los sockets establecen la comunicación entre distintos dispositivos en una red, logrando facilitar el intercambio de datos entre un cliente y un servidor. En este trabajo lograremos ver como funcionan mediante lenguaje de programación Python.

OBJETIVOS

El objetivo principal de este trabajo es entender cómo se implementan los hilos y sockets en lenguaje de programación Python mediante la construcción de un juego, entre el cliente y el servidor donde el cliente enviara los datos al servidor mediante la conexión de un socket.

TAREA

Construir una aplicación en python que permite aplicar el tema de hilos y socket, la aplicación consiste en:

• Construir un juego en el que se generan números aleatorios a través de hilos por parte del cliente.

• Establecer una comunicación entre el cliente y el servidor para que el servidor adivine los números generados por el cliente.

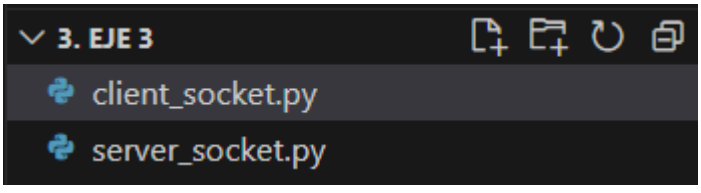
• Contabilizar los aciertos y desaciertos cuando finalice la aplicación.

• Cuando el servidor lleve 3 desaciertos seguidos debe salir un mensaje de “Perdiste”.

• Terminar la aplicación cuando el cliente envíe la palabra “terminar”

SOLUCIÓN

Primero creamos dos archivos, client_socket.py quien enviará los mensajes y server_socket.py quien recibirá los mensajes.



Dentro de client_socket.py comenzamos creando el socket del cliente, para posteriormente establecer conexión con el servidor.

```python
import socket

# Crear un socket de cliente
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Conectar el socket al servidor
server_address = ("localhost", 65432)
client_socket.connect(server_address)

```

Esto mismo lo hacemos en el archivo de server_socket.py que al definir el metodo .listen(5) hara que acepte las conexiones entrantes con un maximo de 5 conexiones.

```python
import socket
import random

# Crear un socket de servidor
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Vincular el socket a la dirección y puerto
server_address = ("localhost", 65432)
server_socket.bind(server_address)

# Escuchar conexiones entrantes
server_socket.listen(5)
print(f"Servidor escuchando en {server_address}")

```

Al ejecutar efectivamente el servidor inicia, esperando la conexión por parte del cliente.

```
PS C:\Users\julia\OneDrive - Fundacion Universitaria del Area Andi
na\1. Documentos Universidad\5. Quinto Semestre\MODELOS DE PROGRAM
ACIÓN\3. Eje 3> python server_socket.py
Servidor escuchando en ('localhost', 65432)
Esperando conexión de un cliente...
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

PS C:\Users\julia\OneDrive - Fundacion Universitaria del Area Andina\1
. Documentos Universidad\5. Quinto Semestre\MODELOS DE PROGRAMACIÓN\3.
 Eje 3> python client_socket.py
```

Se establece la conexión con el servidor.

```
40  while True:
41      # Esperar a que un cliente se conecte
42      print("Esperando conexión de un cliente...")
43      connection, client_address = server_socket.accept()
44
45      try:
46          print(f"Conexión establecida con: {client_address}")
```

```
PS C:\Users\julia\OneDrive - Fundacion Universitaria del Area Andi
na\1. Documentos Universidad\5. Quinto Semestre\MODELOS DE PROGRAM
ACIÓN\3. Eje 3> python server_socket.py
Servidor escuchando en ('localhost', 65432)
Esperando conexión de un cliente...
Conexión establecida con: ('127.0.0.1', 10092)
```

Comenzamos definiendo las variables en client_socket.py, en este caso como el juego lo que hará será adivinar el numero que el usuario ingrese. Para eso le pediremos que ingrese un número mínimo, un número máximo y el número que desea que el servidor adivine.

```python
try:
    # Enviar datos
    message = "Hola, servidor"
    minimun_number = int(input("Ingrese el número mínimo:"))
    maximun_number = int(input("Ingrese el número máximo" ":"))
    number_to_guess = int(input("Ingrese el número a adivinar" ":"))
    is_complete = input("Si no deseas jugar mas ingresa la palabra terminar de lo contrario dale enter para continuar" ":")
    print(f"Enviando: {message}")

    numbers = f"{minimun_number},{maximun_number},{number_to_guess},{is_complete}"
    client_socket.sendall(numbers.encode())
    # Esperar respuesta
    data = client_socket.recv(1024)
    print(f"Recibido del servidor: {data.decode()}")

finally:
    # Cerrar el socket
    client_socket.close()
```

Así mismo le preguntamos si desea continuar jugando, si no lo desea debe ingresar la palabra terminar.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

PS C:\Users\julia\OneDrive - Fundacion Universitaria del Area Andina\1.
Documentos Universidad\5. Quinto Semestre\MODELOS DE PROGRAMACIÓN\3. Eje
 3> python client_socket.py
Ingrese el número mínimo:1
Ingrese el número máximo:15
Ingrese el número a adivinar:3
Si no deseas jugar mas ingresa la palabra terminar de lo contrario dale
enter para continuar:
```

Los datos que ingrese el cliente pasaran al servidor para que inicie el juego

```
        # Recibe los datos
        data = connection.recv(1024)
        if data:
            received_message = data.decode()
            print(f"Recibido: {received_message}")

            # Separar los números usando el delimitador
            numbers_str = received_message.split(",")

            if len(numbers_str) >= 3:
                minimun_number = int(numbers_str[0])
                maximun_number = int(numbers_str[1])
                number_to_guess = int(numbers_str[2])
                print(f"Números recibidos: Min={minimun_number}, Max={maximun_number}, Adivinar={number_to_guess}")

                if adivinar_numero(minimun_number, maximun_number, number_to_guess):
                    cantidad_turnos_exitosos += 1
                else:
                    cantidad_turnos_fallidos += 1

                # Imprimir los contadores
                print(f"La cantidad de turnos exitosos es {cantidad_turnos_exitosos}")
                print(f"La cantidad de turnos fallidos es {cantidad_turnos_fallidos}")

                if numbers_str[3] == "terminar":
                    connection.close()
                    break

            else:
                print("No se recibieron suficientes números.")

            # Responder al cliente
            connection.sendall(b"Mensaje recibido")

```

Dentro de server_socket.py definimos la función adivinar número y por medio del condicional if empezara verificando si el numero aleatorio es igual al numero a adivinar si es menor o si es mayor. De igual forma al colocar el contador de desaciertos contara cuantos desaciertos lleva el servidor y mediante el bucle while, solo le estamos dando 3 oportunidades para que adivine el número.

```python
def adivinar_numero(minimo, maximo, numero_adivinar):
    coontador_desaciertos = 0

    while coontador_desaciertos <3:
        numero_aleatorio = random.randint(minimo, maximo)

        if numero_aleatorio == numero_adivinar:
            print(f"El número {numero_aleatorio} es el correcto")
            return True
        elif numero_aleatorio < numero_adivinar:
            print(f"El número {numero_aleatorio} es menor que el número que se quiere adivinar")
            minimo = numero_aleatorio +1
        elif numero_aleatorio > numero_adivinar:
            print(f"El número {numero_aleatorio} es mayor que el número que se quiere adivinar")
            maximo = numero_aleatorio -1

        coontador_desaciertos += 1
        print(f"Llevas {coontador_desaciertos} desacierto(s)")

    print(f"Perdiste, el número a adivinar era {numero_adivinar}")
    return False
```

En el primer “turno” del servidor podemos observar que no logro adivinar el número, por ende, nos imprime el número que debió adivinar y la cantidad de turnos fallidos que lleva.

```
Conexión establecida con: ('127.0.0.1', 10132)
Recibido: 1,15,3,
Números recibidos: Min=1, Max=15, Adivinar=3
El número 2 es menor que el número que se quiere adivinar
Llevas 1 desacierto(s)
El número 9 es mayor que el número que se quiere adivinar
Llevas 2 desacierto(s)
El número 5 es mayor que el número que se quiere adivinar
Llevas 3 desacierto(s)
Perdiste, el número a adivinar era 3
La cantidad de turnos exitosos es 0
La cantidad de turnos fallidos es 1
Esperando conexión de un cliente...
```

Lo intentamos nuevamente sin terminar aun el juego

```
PS C:\Users\julia\OneDrive - Fundacion Universitaria del Area Andina\1.
Documentos Universidad\5. Quinto Semestre\MODELOS DE PROGRAMACIÓN\3. Eje
 3> python client_socket.py
Ingrese el número mínimo:2
Ingrese el número máximo:16
Ingrese el número a adivinar:13
Si no deseas jugar mas ingresa la palabra terminar de lo contrario dale
enter para continuar:
Enviando: Hola, servidor
Recibido del servidor: Mensaje recibido
PS C:\Users\julia\OneDrive - Fundacion Universitaria del Area Andina\1.
Documentos Universidad\5. Quinto Semestre\MODELOS DE PROGRAMACIÓN\3. Eje
```

Y el servidor sigue sin tener éxito en adivinar el número, al no terminar el juego nos contabiliza la cantidad de turnos fallidos que lleva el servidor

```
Conexión establecida con: ('127.0.0.1', 26143)
Recibido: 2,16,13,
Números recibidos: Min=2, Max=16, Adivinar=13
El número 14 es mayor que el número que se quiere adivinar
Llevas 1 desacierto(s)
El número 10 es menor que el número que se quiere adivinar
Llevas 2 desacierto(s)
El número 11 es menor que el número que se quiere adivinar
Llevas 3 desacierto(s)
Perdiste, el número a adivinar era 13
La cantidad de turnos exitosos es 0
La cantidad de turnos fallidos es 2
Esperando conexión de un cliente...
```

Continuamos jugando hasta que le damos la orden de terminar el juego.

```
PS C:\Users\julia\OneDrive - Fundacion Universitaria del Area Andina\1.
Documentos Universidad\5. Quinto Semestre\MODELOS DE PROGRAMACIÓN\3. Ej
 3> python client_socket.py
Ingrese el número mínimo:1
Ingrese el número máximo:15
Ingrese el número a adivinar:13
Si no deseas jugar mas ingresa la palabra terminar de lo contrario dale
enter para continuar:terminar
Enviando: Hola, servidor
Recibido del servidor:
```

Al decidir terminar el juego, el socket se desconecta perdiendo así la conexión entre el cliente y el servidor.

```
71
72          if numbers_str[3] == "terminar":
73              connection.close()
74              break
75
```

```
Conexión establecida con: ('127.0.0.1', 26232)
Recibido: 1,15,13,terminar
Números recibidos: Min=1, Max=15, Adivinar=13
El número 3 es menor que el número que se quiere adivinar
Llevas 1 desacierto(s)
El número 8 es menor que el número que se quiere adivinar
Llevas 2 desacierto(s)
El número 13 es el correcto
La cantidad de turnos exitosos es 2
La cantidad de turnos fallidos es 2
PS C:\Users\julia\OneDrive - Fundacion Universitaria del Area Andina\
1. Documentos Universidad\5. Quinto Semestre\MODELOS DE PROGRAMACIÓN\
3. Eje 3>
```

CONCLUSIONES

Mediante el desarrollo de este eje, me di cuenta de que el uso de hilos y sockets nos permiten desarrollar y mejorar significativamente la capacidad y el rendimiento de las aplicaciones, ya que podemos gestionar múltiples conexiones o procesos en paralelo sin bloquear el flujo principal del programa. A través de este eje comprendí que la integración de hilos y socket nos permite la creación no solo de juegos sino también de servidores web, algo fundamental hoy en día. Empresas como Uber utilizan sockets para la comunicación en tiempo real entre los conductores y los usuarios, en las referencias de este trabajo dejare un pequeño instructivo creado por esta compañía donde mencionan como realizan su proceso. Zoom es otra aplicación que utiliza sockets para hacer videollamada, siendo el socket la videollamada y el cliente los conferencistas que se conectaran a la videollamada, de igual forma la cantidad de usuarios dependen del plan adquirido con zoom esto también indica que utilizan sockets con un tope máximo de usuarios.

REFERENCIAS

Python. *Programación con Sockets.*

https://docs.python.org/es/3/howto/sockets.html septiembre 2024

W3Schools. *Random Module*

https://www.w3schools.com/python/module_random.asp

arm code, socket en python, cliente servidor

https://www.youtube.com/watch?v=4KBFG8TgCF4

Uber. *Uso de la app*

https://www.uber.com/co/es/about/how-does-uber-work/

Zoom. Reuniones virtuales

https://www.zoom.com/es/products/virtual-meetings/

Repositorio, Juliana Ramirez. EJE-3-HILOS-Y-SOCKET

https://github.com/JulianaRamirez/EJE-3-HILOS-Y-SOCKET