Arquitectura Cliente-Servidor

Sebastián Ricardo Alvarado Cely
Estudiante

Joaquín Sánchez
Docente

Corporación Universitaria Iberoamericana
Facultad de ingeniería
Ingeniería se software

Boyacá, Colombia
Septiembre,2024

Arquitectura Cliente-Servidor

En el mundo de las tecnologías de la información, la arquitectura cliente-servidor es un modelo fundamental que facilita la interacción y la comunicación entre estos dos tipos de componentes de software. Este tipo de arquitectura divide en dos las dos principales funcionalidades de un aplicativo: el cliente, que solicita servicios o recursos, y el servidor que se encarga de brindarle y proveerle esta solicitud. La separación clara de estas dos capas permite un funcionamiento más eficiente y también brinda una importante ayuda en el momento de la escalabilidad y el mantenimiento de software.

La arquitectura cliente-servidor me ha demostrado ser demasiado versátil, esta es capaz de adaptarse de una mera eficiente a desarrollo de sistemas y a bases de datos, principalmente, también es fundamental en aplicaciones alojadas en la red. La capacidad de este tipo de arquitectura para manejar varias solicitudes simultáneamente la convierte en una elección preferente para desarrolladores que buscan crear sistemas exitosos. Mediante el ejemplo del Juego de Adivinanza de Números pretendo mostrar básicamente el funcionamiento de este tipo de arquitectura y como se complementan entre sí el cliente y el servidor.

Enlace GitHub

https://github.com/sebaricardocely/juego_adivinanza

Código Juego de adivinanza Cliente

```python
#Sebastian Ricardo Alvarado Cely    CLIENTE
# importo biblioteca socket para la conexión
import socket
# Funcion para inciair conexión con el servidor
def iniciar_cliente(host='localhost', puerto=65434):
    cliente_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        cliente_socket.connect((host, puerto))
        print(f"[+] Conectado al servidor en {host}:{puerto}")

        while True:
            datos = cliente_socket.recv(1024)
            if not datos:
                print("[-] El servidor ha cerrado la conexión.")
                break
            mensaje = datos.decode('utf-8')
            print(mensaje, end='')
            # end='' para evitar doble salto de línea

            # Si el mensaje indica que el cliente adivinó correctamente, avisa y finaliza
            if "¡Correcto!" in mensaje:
                break

            # Solicitar al usuario que ingrese un intento
            intento = input("Tu intento: ")
            cliente_socket.sendall(intento.encode('utf-8'))

    except ConnectionRefusedError:
        print(f"[-] No se pudo conectar al servidor en {host}:{puerto}")
    except KeyboardInterrupt:
        print("\n[!] Cerrando cliente.")
    finally:
        cliente_socket.close()

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

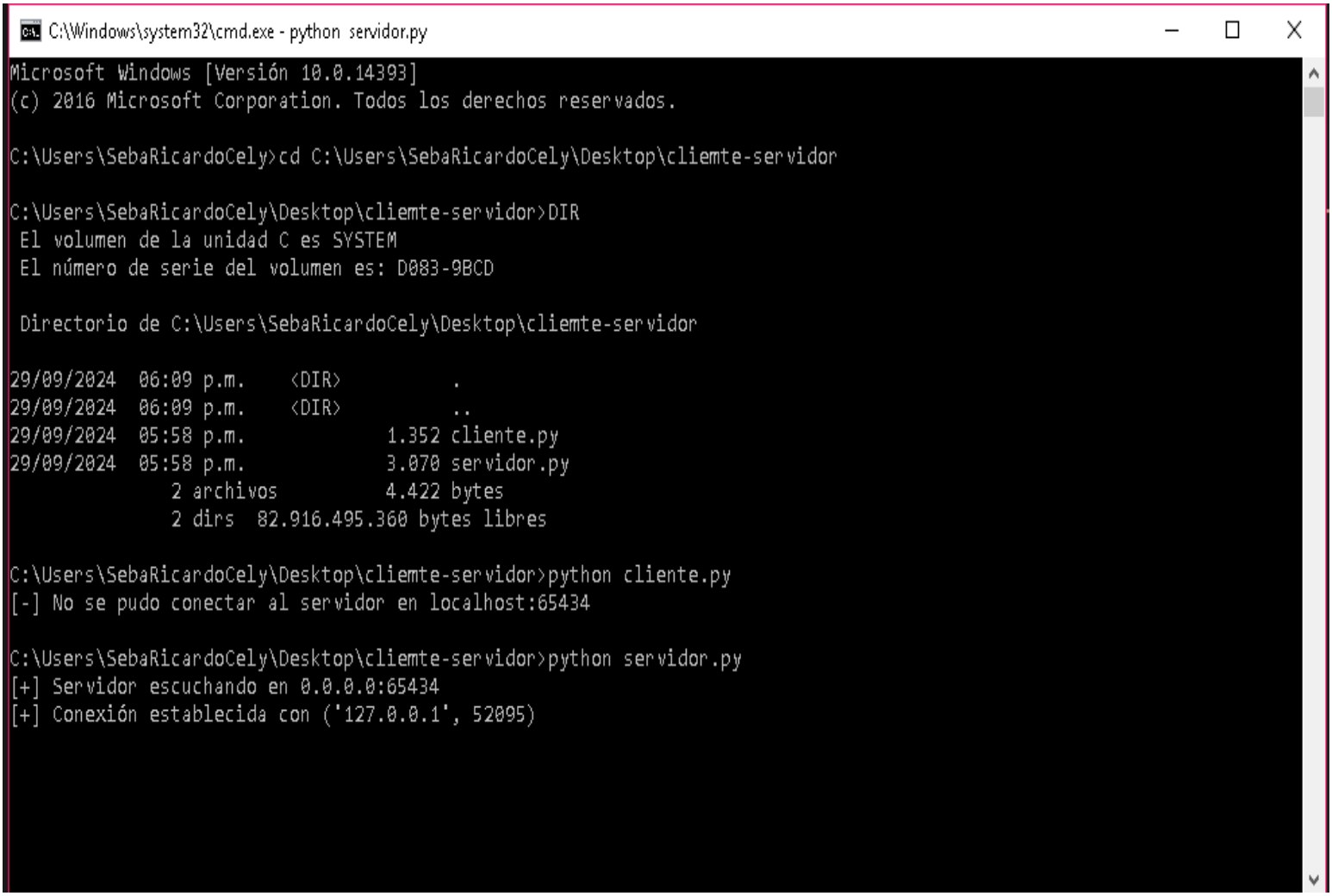
Código Juego de adivinanza Servidor

```python
#Sebastian Ricardo Alvarado Cely    CLIENTE-SERVIDOR

# Importo socket que permitirá la comunicación entre servidor y cliente, threading permitirá manejar varios clientes simuteneamente

import socket
import threading
import random


#funcion manejar cliente en la cual, genero un numero secreto y le envia mensajes al cliente para que intente adivinarlo.

def manejar_cliente(cliente_socket, direccion):
    print(f"[+] Conexión establecida con {direccion}")
    numero_secreto = random.randint(1, 100)
    intentos = 0
    cliente_socket.sendall("¡Bienvenido al Juego de Adivinanza de Números!\n".encode('utf-8'))
    cliente_socket.sendall("Estoy pensando en un número entre 1 y 100.\n".encode('utf-8'))
    cliente_socket.sendall("Intenta adivinarlo.\n".encode('utf-8'))
#Creacion de condicionales
    while True:
        try:
            datos = cliente_socket.recv(1024)
            if not datos:
                print(f"[-] Conexión cerrada por {direccion}")
                break
            intento = datos.decode('utf-8').strip()
            if not intento.isdigit():
                mensaje = "Por favor, ingresa un número válido.\n"
                cliente_socket.sendall(mensaje.encode('utf-8'))
 #Si el cliente ingresa numero equivocado responde el anterior mensaje el servidor.

                continue
```

```
def manejar_cliente(cliente_socket, direccion):
                continue
            intento = int(intento)
            intentos += 1
#El cliente ingresa numero correcto y el servidor le indicara si el numero es mas alto o bajo.
            if intento < numero_secreto:
                mensaje = "Demasiado bajo. Intenta de nuevo.\n"
            elif intento > numero_secreto:
                mensaje = "Demasiado alto. Intenta de nuevo.\n"
            else:
                mensaje = f"¡Correcto! Adivinaste el número en {intentos} intentos.\n"
                cliente_socket.sendall(mensaje.encode('utf-8'))
                print(f"[+] {direccion} Has adivinado el numero {numero_secreto} en {intentos} intentos.")
                break
            cliente_socket.sendall(mensaje.encode('utf-8'))
        except ConnectionResetError:
            print(f"[-] Conexión inesperadamente cerrada por {direccion}")
            break
    cliente_socket.close()

def iniciar_servidor(host='0.0.0.0', puerto=65434):
    servidor_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    servidor_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    servidor_socket.bind((host, puerto))
    servidor_socket.listen(5)
    print(f"[+] Servidor escuchando en {host}:{puerto}")

    try:
        while True:
            cliente_socket, direccion = servidor_socket.accept()
            hilo = threading.Thread(target=manejar_cliente, args=(cliente_socket, direccion))
            hilo.start()
    #se permite cerrar con el teclado el servidor
    except KeyboardInterrupt:
        print("\n[!] Cerrando servidor.")
    finally:
        servidor_socket.close()

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

Ejecución cliente-servidor

- Se ejecuta el servidor desde el terminal de comandos CMD



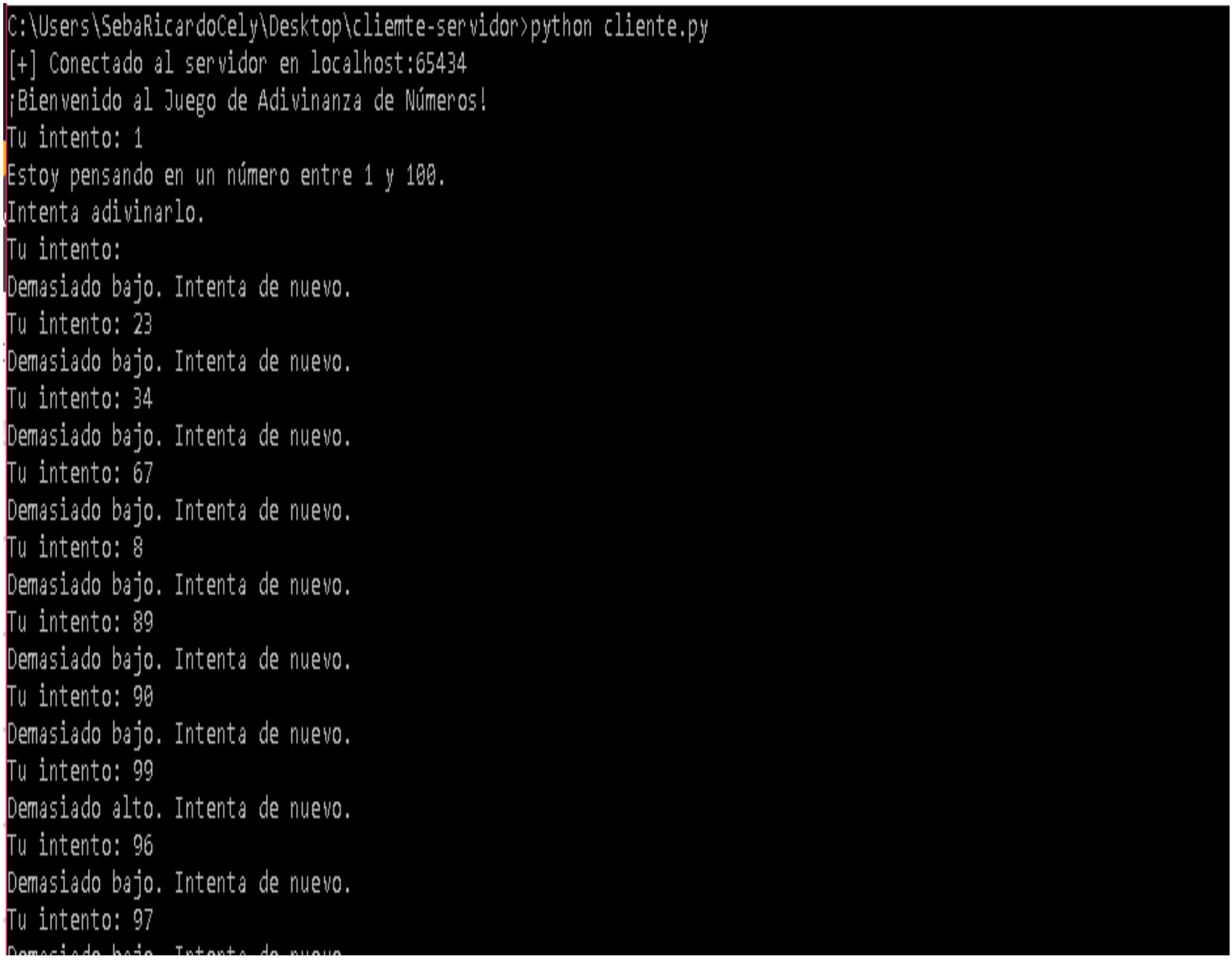
- Se inicia el cliente que inmediatamente se conecta al servidor

```
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\SebaRicardoCely>cd C:\Users\SebaRicardoCely\Desktop\cliemte-servidor

C:\Users\SebaRicardoCely\Desktop\cliemte-servidor>DIR
 El volumen de la unidad C es SYSTEM
 El número de serie del volumen es: D083-9BCD

 Directorio de C:\Users\SebaRicardoCely\Desktop\cliemte-servidor

29/09/2024  06:09 p.m.    <DIR>          .
29/09/2024  06:09 p.m.    <DIR>          ..
29/09/2024  05:58 p.m.             1.352 cliente.py
29/09/2024  05:58 p.m.             3.070 servidor.py
               2 archivos          4.422 bytes
               2 dirs  82.916.495.360 bytes libres

C:\Users\SebaRicardoCely\Desktop\cliemte-servidor>python cliente.py
[-] No se pudo conectar al servidor en localhost:65434

C:\Users\SebaRicardoCely\Desktop\cliemte-servidor>python servidor.py
[+] Servidor escuchando en 0.0.0.0:65434
[+] Conexión establecida con ('127.0.0.1', 52095)
```

- El cliente ve el mensaje enviado por el servidor y envía el primer mensaje

```
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\SebaRicardoCely>cd C:\Users\SebaRicardoCely\Desktop\cliemte-servidor

C:\Users\SebaRicardoCely\Desktop\cliemte-servidor>python cliente.py
[+] Conectado al servidor en localhost:65434
¡Bienvenido al Juego de Adivinanza de Números!
Tu intento: 1
Estoy pensando en un número entre 1 y 100.
Intenta adivinarlo.
Tu intento: _
```

- El cliente ingresa los intentos necesarios, el servidor responde demasiado bajo o alto dependiendo del número.



- Al cliente ingresar el numero correcto, el servidor le envía el mensaje al cliente diciendo cuantas oportunidades necesitó para encontrar el correcto y se cierra.

```
C:\Users\SebaRicardoCely\Desktop\cliemte-servidor>python servidor.py
[+] Servidor escuchando en 0.0.0.0:65434
[+] Conexión establecida con ('127.0.0.1', 52095)
[+] ('127.0.0.1', 52095) Has adivinado el numero 98 en 11 intentos.
```

Conclusiones

En este trabajo comprendí más a profundidad la arquitectura cliente-servidor a través de la implementación de un juego de adivinanza de números, lo que ha permitido observar de forma práctica como funciona la interacción entre este modelo cliente-servidor. Este modelo aparte de optimizar la comunicación y el procesamiento de datos también brinda una estructura clara que facilita el desarrollo de aplicaciones. A medida que he avanzado en el desarrollo de software comprender este tema me facilita el aprendizaje general de carrera universitaria, ya que es fundamental para la creación de aplicaciones.