

# **Arquitectura Cliente Servidor**

Estefanía Toro Bañol

Arquitectura de Software / Corporación Universitaria Iberoamericana

Joaquín Sánchez

Septiembre 2024

## Introducción

Al tener un conocimiento base de que es la arquitectura de software y sus tipos de arquitecturas, así como el tener conocimiento de los más utilizados en el mundo de la ingeniería de software, para esta actividad el objetivo es realizar una aplicación o programa utilizando la arquitectura Cliente/Servidor demostrando como dos partes principales interactúan entre sí. Sabemos que el cliente es el que realiza las solicitudes bien sea en un navegador web, una app o en un programa de una computadora para obtener o enviar una información y el servidor es el que recibe esas peticiones y responde a ellas, procesándolas y devolviendo los datos o resultados solicitados.

El proceso para este programa es que el cliente envíe una solicitud al servidor y que el servidor reciba, procese y que le devuelva una respuesta al cliente.

### Programa Cliente/Servidor para ingresar a la biblioteca

Ahora bien, el objetivo de este programa es que el servidor le permita al cliente el ingreso a la biblioteca y que le muestre los libros disponibles, pero antes de ello el sistema le pedirá que realice una autenticación de usuario, si el cliente no se encuentra registrado, el proceso de login ejecutara un error, pidiendo el registro de este.

Vamos a ver más a fondo el programa, demostrando desde la terminal cómo funciona el programa.

#### 1. Servidor

El primer paso que realizaremos será el crear una función en la cual me permitirá iniciar o correr el servidor, donde estaremos implementando de la librería de Python los socket, que es el que me permitirá una interacción local del programa, por lo tanto se deberá crear un socket de tipo TCP que me garantizará que los datos lleguen completos y en orden específico, luego asociaremos el socket en una dirección y puerto ("Localhost", 8080). Donde dará inicio y a la espera de que el cliente se conecte. Cuando el servidor se activa al ejecutar Servidor.py en la terminal, arrojará un mensaje "Servidor escuchando en el puerto 8080" donde está a la espera del cliente

```
2. def iniciar_servidor():
3.     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4.     server_socket.bind(('localhost', 8080))
5.     server_socket.listen(5)
6.     print("Servidor escuchando en el puerto 8080...")
7.
8.     while True:
9.         conn, addr = server_socket.accept()
10.        threading.Thread(target= manejar_cliente, args=(conn,
11.        addr)).start()
12. if __name__ == "__main__":
13.     iniciar_servidor()
```

```
C:\Windows\system32\cmd.exe - python Servidor.py
Microsoft Windows [Versión 10.0.19045.4894]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\USUARIO>cd OneDrive\Desktop\Biblioteca

C:\Users\USUARIO\OneDrive\Desktop\Biblioteca>python Servidor.py
Servidor escuchando en el puerto 8080...
```

Ahora, se crea una función que le permita al cliente visualizar un breve menú con los pasos que deberá realizar para dar ingreso a la biblioteca, cuando se confirma la conexión con el cliente, automáticamente en la terminal el servidor le mandará un menú al cliente.

```
54
55 def manejar_cliente(conn, addr):
56     print("Conexión establecida con {addr}")
57
58     while True:
59         # Enviar menú al cliente
60         menu = """
61         ¿Qué desea hacer?
62         1. Registrarse
63         2. Iniciar sesión
64         3. Salir
65         """
66         conn.send(menu.encode('utf-8'))
67
68         # Recibir opción del cliente
69         opcion = conn.recv(1024).decode('utf-8')
70
71         if opcion == "1":
72             Registrarse(conn)
73         elif opcion == "2":
74             login(conn)
75         elif opcion == "3":
76             conn.send("Hasta luego...".encode('utf-8'))
77             break
78         else:
79             conn.send("Opción inválida. Intente nuevamente.\n".encode('utf-8'))
80     conn.close()
81
```

Con la siguiente imagen se puede apreciar la conexión con el cliente y los datos enviados del servidor para el cliente que se acaba de conectar.

El servidor puede recibir varios clientes, pero solo habrá un servidor. Y a medida que el cliente elige opciones el servidor le enviara los datos necesarios a diligenciar.

```
C:\Windows\system32\cmd.exe - python Servidor.py
Microsoft Windows [Versión 10.0.19045.4894]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\USUARIO>cd OneDrive\Desktop\Biblioteca

C:\Users\USUARIO\OneDrive\Desktop\Biblioteca>python Servidor.py
Servidor escuchando en el puerto 8080...
Conexión establecida con {addr}
```

```
C:\Windows\system32\cmd.exe - python Cliente.py
Microsoft Windows [Versión 10.0.19045.4894]
(c) Microsoft Corporation. Todos los derechos reservados.

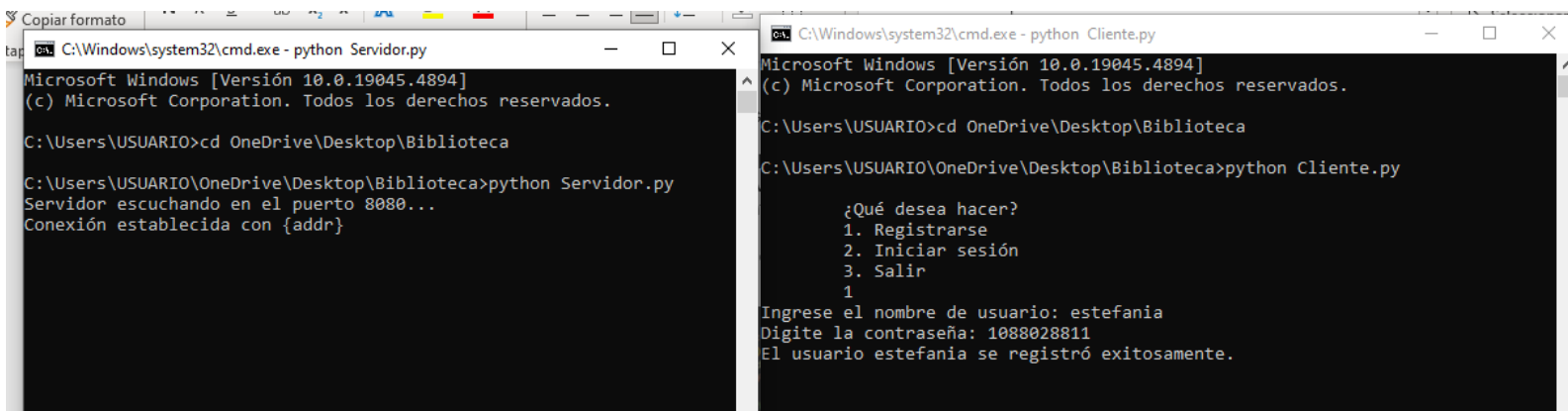
C:\Users\USUARIO>cd OneDrive\Desktop\Biblioteca

C:\Users\USUARIO\OneDrive\Desktop\Biblioteca>python Cliente.py

¿Qué desea hacer?
1. Registrarse
2. Iniciar sesión
3. Salir
```

Ahora miremos el código que nos permite el registro, así como el de iniciar sesión. En la función de Registro lo que hace es enviarle ciertos datos básicos al cliente como el nombre y la contraseña, la cual lo hacemos por medio de `.encode("utf-8")` y la información que el cliente envía lo recibe con `.decode("utf-8")`. La información dada por el cliente, el servidor lo almacena en dos variables, las cuales se determinaron al inicio del proyecto como `Lista_usuarios` y `Lista_contraseña`, luego el servidor le enviara un mensaje al cliente indicando que el registro fue exitoso.

```
30
31 def Registrarse(conn):
32     conn.send("Ingrese el nombre de usuario: ".encode('utf-8'))
33     nombre = conn.recv(1024).decode('utf-8')
34     conn.send("Digite la contraseña: ".encode('utf-8'))
35     contraseña = conn.recv(1024).decode('utf-8')
36
37     # Almacenar el nuevo usuario y contraseña
38     Lista_usuarios.append(nombre)
39     Lista_contraseña.append(contraseña)
40
41     # Confirmación al cliente
42     conn.send(f"El usuario {nombre} se registró exitosamente.\n".encode('utf-8'))
```



The image shows two terminal windows side-by-side. The left window is titled 'C:\Windows\system32\cmd.exe - python Servidor.py' and shows the following output: 'Microsoft Windows [Versión 10.0.19045.4894] (c) Microsoft Corporation. Todos los derechos reservados. C:\Users\USUARIO>cd OneDrive\Desktop\Biblioteca C:\Users\USUARIO\OneDrive\Desktop\Biblioteca>python Servidor.py Servidor escuchando en el puerto 8080... Conexión establecida con {addr}'. The right window is titled 'C:\Windows\system32\cmd.exe - python Cliente.py' and shows the following output: 'Microsoft Windows [Versión 10.0.19045.4894] (c) Microsoft Corporation. Todos los derechos reservados. C:\Users\USUARIO>cd OneDrive\Desktop\Biblioteca C:\Users\USUARIO\OneDrive\Desktop\Biblioteca>python Cliente.py ¿Qué desea hacer? 1. Registrarse 2. Iniciar sesión 3. Salir 1 Ingrese el nombre de usuario: estefania Digite la contraseña: 1088028811 El usuario estefania se registró exitosamente.'

Ahora para la función Login, lo que hace es enviar al cliente datos a diligenciar como el nombre y contraseña, y con ayuda de una condicional If en este caso, lo que hace es comparar si los nombres y contraseñas ingresadas estén dentro de la lista, de lo contrario arrojará un error.

Cuando el proceso de login procede exitosamente lo que hace el servidor automáticamente es darle la bienvenida a la biblioteca y mostrar una lista de los libros que están disponibles con el nombre del autor, y allí finaliza el proceso.

Con la función Biblioteca lo que hace es mostrarle o enviarle al usuario la lista de los libros que están guardados en una variable `lista_libros`

```
Servidor.py x Cliente.py
Servidor.py > manejar_cliente

1
2 import socket
3 import threading
4
5 Lista_usuarios = []
6 Lista_contrasena = []
7
8
9 def login(conn):
10     # Proceso de login
11     conn.send("Ingrese su nombre de usuario: ".encode('utf-8'))
12     username = conn.recv(1024).decode('utf-8')
13
14     conn.send("Ingrese su contraseña: ".encode('utf-8'))
15     password = conn.recv(1024).decode('utf-8')
16
17     # Validar información
18     # Verificar si el nombre de usuario está en la lista
19     if username in Lista_usuarios:
20         # Obtener el índice del usuario
21         index = Lista_usuarios.index(username)
22         # Verificar la contraseña usando el índice
23         if Lista_contrasena[index] == password:
24             conn.send("Login exitoso. Bienvenido a la biblioteca!\n".encode('utf-8'))
25             Biblioteca(conn)
26         else:
27             conn.send("Credenciales incorrectas. La contraseña no coincide.\n".encode('utf-8'))
28     else:
29         conn.send("El usuario no existe. Favor registrarse\n".encode('utf-8'))
```

```
43
44 def Biblioteca(conn):
45     Lista_Libros = ["Romeo y Julieta de William S.", "Cien Años de soledad de Gabriel G.M ", "Un mundo feliz de
46     conn.send("\n Gracias por visitar la biblioteca! \n Libros disponibles \n".encode('utf-8'))
47     for libro in Lista_Libros:
48         conn.send(f"- {libro}\n".encode('utf-8'))
49         #la f antes de las comillas indica que
50         #se esta utilizando un f-string o cadena de formato en python 3.6 y son una forma de
51         #formatear cadenas de texto, permitiendo incrustar expresiones dentro de cadenas
52         #de manera más legibles y concisa
53     conn.send("Fin de la lista\n".encode('utf-8'))
54
```

The image shows two terminal windows side-by-side. The left window is titled 'C:\Windows\system32\cmd.exe - python Servidor.py'. It shows the execution of a Python script that starts a server on port 8080. The output indicates that the server is listening and a connection has been established. The right window is titled 'C:\Windows\system32\cmd.exe - python Cliente.py'. It shows the execution of a Python script that connects to the server. The user is prompted to enter a username and password, and the script successfully registers the user 'estefania' with password '1088'. The script then displays a menu of books available in the library.

```
C:\Windows\system32\cmd.exe - python Servidor.py
Microsoft Windows [Versión 10.0.19045.4894]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\USUARIO>cd OneDrive\Desktop\Biblioteca
C:\Users\USUARIO\OneDrive\Desktop\Biblioteca>python Servidor.py
Servidor escuchando en el puerto 8080...
Conexión establecida con {addr}
```

```
C:\Windows\system32\cmd.exe - python Cliente.py
2. Iniciar sesión
3. Salir
1
Ingrese el nombre de usuario: estefania
Digite la contraseña: 1088
El usuario estefania se registró exitosamente.

¿Qué desea hacer?
1. Registrarse
2. Iniciar sesión
3. Salir
2
Ingrese su nombre de usuario: estefania
Ingrese su contraseña: 1088
Login exitoso. Bienvenido a la biblioteca!

Gracias por visitar la biblioteca!
Libros disponibles
- Romeo y Julieta de William S.
- Cien Años de soledad de Gabriel G.M
- Un mundo feliz de Aldoux H.
Fin de la lista

¿Qué desea hacer?
1. Registrarse
2. Iniciar sesión
3. Salir
```

### 13. Cliente

De la misma manera como en el servidor, se deberá crear una función la que nos permitirá conectarnos con un puerto y dirección con el servidor.

The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'BIBLIOTECA' containing two files: 'Cliente.py' and 'Servidor.py'. The code editor shows the implementation of the 'Cliente.py' file. The code defines a function 'iniciar\_cliente()' that connects to the server on 'localhost' port 8080. It then enters a loop where it receives messages from the server and sends responses. The code also includes a main block that calls the 'iniciar\_cliente()' function.

```
File Edit Selection View Go Run ...
BIBLIOTECA
  Cliente.py
  Servidor.py

def iniciar_cliente():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 8080))

    while True:
        # Recibe mensaje del servidor
        mensaje = client_socket.recv(1024).decode('utf-8')
        print(mensaje, end='') # end='' para evitar un salto de línea extra

        # Enviar respuesta (nombre de usuario o contraseña)
        respuesta = input()
        client_socket.send(respuesta.encode('utf-8'))

        if "login exitoso" in mensaje:
            break # Salir del bucle si el login fue exitoso

    while True:
        respuesta = client_socket.recv(1024).decode('utf-8')
        if not respuesta:
            break
        print(respuesta, end='')

    client_socket.close()

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

Como se muestra en la imagen anterior, para el Cliente también es necesario el uso del paquete de socket, pues este nos permitirá la conexión con el servidor. Cabe destacar que la dirección y el puerto deben ser las mismas que la del servidor para una mayor

eficacia en la conexión, de lo contrario al ejecutar el cliente, arrojará un error donde nos indica que el servidor no está aceptando conexiones.

Con la función de iniciar cliente se realiza el procedimiento de conexión con socket y un ciclo while que nos permitirá la verificación al momento de login recibiendo los datos del servidor y que a su vez el cliente pueda responder y otro ciclo en donde mostrará la respuesta del servidor. El if al final es el encargado de que la función iniciar cliente corra.

### **Requisitos para el desarrollo del proyecto**

Solo se necesitaron del uso de Visual Studio, la versión Python 3.9.0 y de una conexión local para la comunicación entre cliente y servidor

### **Explicación del funcionamiento**

**Servidor:** Se escucha en el puerto 8080, el servidor procesa comandos leer y guardar datos, así como le envía operaciones a realizar para el cliente

**Cliente:** Envía comandos al servidor para realizar operaciones como el de registrar usuario o inicio de sesión

### **Ejecución**

1. Se ejecuta primero el servidor en la terminal, para poder recibir a los clientes
2. Se ejecuta el cliente en otra terminal para que este se conecte con el Servidor
3. El cliente realiza las operaciones requeridas como ingresar nombre y contraseña para registrarse y así poder ingresar a la biblioteca