Actividad 3 – Arquitectura Cliente-Servidor

Anyi Dayana Idrobo Parra

Docente Joaquín Sánchez

Corp. Universitaria Iberoamericana

Arquitectura de Software

Facultad de Ingeniería

Ingeniería de Software

Palmira, 2024

Actividad 1 - Arquitectura Cliente-Servidor

El objetivo es demostrar como en la codificación de un ejercicio con la estructura Cliente-Servidor, existe una interacción entre un cliente y un servidor, comúnmente se ejecuta desde una máquina externa hacía un servidor o desde la propia máquina a su servidor. Para el desarrollo de la actividad, se utiliza la herramienta Apache NetBeans IDE 14 con el lenguaje de programación JAVA, creando dos clases principales para ser ejecutadas en el mismo momento, una llamada Cliente y la otro Servidor, por medio de un protocolo de comunicación el cliente envía un mensaje y el servidor responde a ese mensaje.

Clase servidor

La siguiente imagen corresponde a las librerías importadas desde JAVA.

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
```

Se crea Servidor como clase principal. Se crean ServerSocket, Socket y se importan a la librería, a estas se les da el nombre de server y scliente. Se crea un puerto con una constante.

```
public class Servidor {
    public static void main (String[] arg){

        ServerSocket server = null;
        Socket scliente = null;
        DataInputStream in;
        DataOutputStream out;

        final int Puerto = 4500;
```

Se crea una excepción con un try catch con el mensaje de servidor iniciado, ya que debe estar atento a cualquier entrada del cliente, para esto se utiliza el método accept.

```
try {
    server = new ServerSocket (Puerto);
    System.out.println("Server started");

    while(true){
    scliente = server.accept();
```

Inicia la comunicación entre cliente y servidor, el servidor recibe el aviso desde el cliente y genera un aviso de respuesta. Al final se desconecta el servidor del cliente.

```
        System.out.println("Connected Customer");
    in = new DataInputStream (scliente.getInputStream());
    out = new DataOutputStream (scliente.getOutputStream());

    String aviso = in.readUTF();

        System.out.println(aviso);

        out.writeUTF("Message from the server");

        scliente.close();
        System.out.println("Disconnected customer");
```

Clase Cliente

Se crea Cliente como clase principal. Se crea la constante lhost indicando que se conectará a la propia máquina y se crea el mismo puerto del servidor.

```
public class Cliente {
    public static void main (String [] arg){

            final String lhost = "127.0.0.1";
            final int Puerto = 4500;
            DataInputStream in;
            DataOutputStream out;
```

Se crea una excepción con un try catch, un socket que trae el lhost, se utiliza el data stream del servidor, para generar el canal de comunicación entre las dos clases.

```
    try {
        Socket scliente = new Socket(lhost, Puerto);

        in = new DataInputStream (scliente.getInputStream());
        out = new DataOutputStream (scliente.getOutputStream());
```

El cliente genera un mensaje ya que el servidor está a la espera, recibe el mensaje del servidor y se desconecta.

```
        out.writeUTF("Message from the customer");

        String aviso = in.readUTF();
        System.out.println(aviso);

        scliente.close();
```

Ejecutar las clases

Se da click derecho en la clase servidor y click en Run file, luego en la clase cliente se repite la operación.

Referencias bibliográficas

DiscoDurodeRoer. (24 abril 2018). *Ejercicios Java - Sockets #1 - Conexión TCP cliente/servidor.*[Video]. Youtube. https://youtu.be/3wJTI9LMOsk?si=t4HMFF8EV0aSnTp-

Sánchez, J. (23 septiembre 2024). *Arquitectura Cliente Servidor*. [Tutoría] Curso Arquitectura de Software Corporación Universitaria Iberoamericana.