

SERVIDOR MULTIHILO SOCKETS

MARIO JIMÉNEZ MARSET

ÍNDICE

1.	ENUNCIADO – OBJETIVOS	3
	DESARROLLO – PROCEDIMIENTOS	
3.	RESULTADOS POR CONSOLA	7

1. ENUNCIADO - OBJETIVOS

En esta práctica se pedía realizar una aplicación Cliente-Servidor, cuyo servidor debía ser multihilo, con el objetivo de atender a múltiples clientes. Este hace ECO a partir de la información recibida del Cliente y se desconecta cuando recibe de este el mensaje "Adios".

Se mostrarán tres procesos clientes que comunican con el servidor, enviando mensajes.

2. DESARROLLO - PROCEDIMIENTOS

En este apartado se muestra, en primer lugar, el código y las explicaciones pertinentes.

```
Código Clase ClienteHiloMensajes:
package socketsSL;
import java.io.*;
import java.net.*;
public class ClienteHiloMensajes implements Runnable{
        //en primer lugar se declaran las variables a utilizar dentro del programa
       private boolean conexion=true;
       private Socket socket:
       private DataInputStream dis;
       private DataOutputStream dos;
       private String mensajeRecibido;
        //se crea el getter y setter perteneciente a la variable mensajeRecibido
       public String getMensajerecibido(){
                return mensajeRecibido;
       public void setMensajeRecibido(String mensajeRecibido){
                this.mensajeRecibido=mensajeRecibido;
        //se crea un método el cual permite leer el mensaje recibido
       public void leerMensaje() {
                try {
                        //se lee con el DataInputStream
                        mensajeRecibido = dis.readUTF();
                        String mensaje="";
                        //se crea un bucle for y un condicional if con el objetivo de que,
        si el usuario
                        //inserta dos puntos, se cree un espacio
                                for(int i=0;i<mensajeRecibido.length();i++) {
                                        if(mensajeRecibido.charAt(i)==':') {
       mensaje=mensajeRecibido.substring(i+1).trim();
                        System.out.println("El mensaje recibido es: "+mensaje);
                        //en el switch se indica que, si se recibe el mensaje Adiós, en
todas
                        //sus formas posibles, se desconectará del servidor
                                switch (mensaje) {
                                case "ADIOS":
                                        conexion=false:
                                break:
```

```
case "Adios":
                                        conexion=false;
                                        break;
                                case "adios":
                                        conexion=false;
                                        break;
                        System.out.println(mensajeRecibido);
                }catch (IOException e) {
                        System.out.println(e.getMessage());
                        cerrarSocket();
        //este método simplemente escribe en el DataOutputStream el texto recibido
       public void escribirLinea(String textoRecibido) {
                try {
                        dos.writeUTF(textoRecibido);
                }catch (IOException e) {
                        System.out.println(e.getMessage());
                        cerrarSocket();
       }
       //se invocan los dos métodos creados dentro de run
       //la línea escrita se recoge a través del getter anteriormente creado
       public synchronized void run() {
                while(conexion) {
                        this.leerMensaje();
                        this.escribirLinea(getMensajerecibido());
       //este método recoge la información del socket
       public ClienteHiloMensajes(Socket socketParametro) {
                socket = socketParametro;
                try {
                        dos = new DataOutputStream(socket.getOutputStream());
                        dis = new DataInputStream(socket.getInputStream());
                }catch (IOException e) {
                        System.out.println(e.getMessage());
                        cerrarSocket();
                }
       //este método simplemente cierra el Socket, el DataOutputStream y
DataInputStream
       public void cerrarSocket() {
                try {
                        conexion = false;
                        socket.close();
                        dis.close();
                        dos.close();
                }catch (IOException e) {
                        conexion = false;
                        System.out.println(e.getMessage());
               }
       }
}
```

```
Código Clase Servidor:
package socketsSL:
import java.io.*;
import java.net.*;
public class Servidor {
        public static void main(String[] args) {
                try {
                        //se crea el socket pasando en el constructor el puerto
                        ServerSocket socket=new ServerSocket(5000);
                        //bucle infinito
                                while(true) {
                                        //se espera a recibir la petición de un cliente
                                        Socket socketEnEspera = socket.accept();
                                        //se imprime quién se ha conectado
                                        System.out.println("Se ha conectado el usuario
cuya direccion IP es "+socketEnEspera.getInetAddress());
                                        //se invoca e inicia el hilo que llama a la clase
ClienteHiloMensajes
                                        Thread hilo=new Thread(new
ClienteHiloMensajes(socketEnEspera));
                                        hilo.start();
                }catch(IOException e) {
                        System.out.println(e.getMessage());
        }
}
```

```
Código Clase SocketCliente:
package socketsSL;
import java.io.*;
import java.net.*;
import java.util.*;
public class SocketCliente {
        //se declaran las variables a utilizar dentro del main
       static Scanner entrada;
       static String nombre, envio;
       public static void main(String[] args) {
                entrada=new Scanner(System.in);
                try {
                        //a través de scanner se introduce el nombre del Cliente
                        System.out.println("Introduce tu Nombre");
                        nombre=entrada.nextLine();
                        //el socket se conecta al localhost con el mismo puerto que en
el Servidor
                        Socket socket=new Socket("127.0.0.1", 5000);
                        while (socket.isConnected()) {
                                //se escribe el mensaje y se envía a la clase Servidor
                                DataOutputStream dos=new
DataOutputStream(socket.getOutputStream());
                                envio=entrada.nextLine();
                                dos.writeUTF(nombre+": "+envio);
                                //además se imprime para ver el contenido del mensaje
                                DataInputStream dis = new
DataInputStream(socket.getInputStream());
                                String recibido = dis.readUTF();
                                System.out.println("El mensaje enviado por "+nombre+"
es: "+recibido);
                        }
                        //se cierran las conexiones
                        socket.close();
                        entrada.close();
                } catch (IOException e) {
                        System.out.println(e.getMessage());
               }
       }
}
```

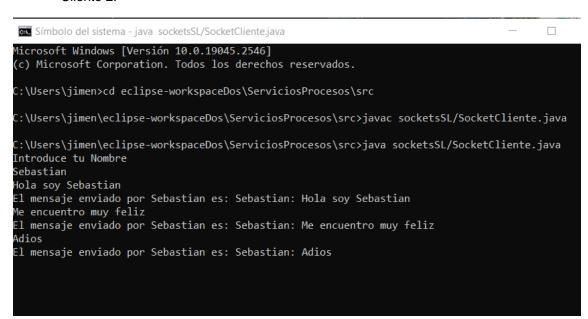
3. RESULTADOS POR CONSOLA

A través de capturas del cmd, se muestran la ejecución del Servidor y tres veces la clase SocketCliente, demostrando que se pueden conectar varios clientes al Servidor en su misma ejecución y que salgan sus mensajes:

Cliente 1:



Cliente 2:



Cliente 3:

```
Microsoft Windows [Versión 10.0.19045.2546]
(c) Microsoft Corporation. Todos los derechos reservados.

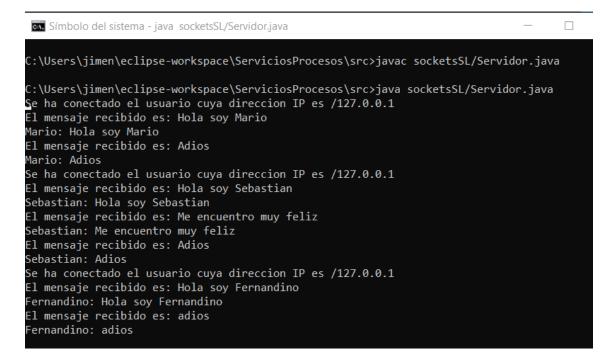
C:\Users\jimen>cd eclipse-workspaceDos\ServiciosProcesos\src

C:\Users\jimen\eclipse-workspaceDos\ServiciosProcesos\src>javac socketsSL/SocketCliente.java

C:\Users\jimen\eclipse-workspaceDos\ServiciosProcesos\src>java socketsSL/SocketCliente.java

Introduce tu Nombre
Fernandino
Hola soy Fernandino
El mensaje enviado por Fernandino es: Fernandino: Hola soy Fernandino
adios
El mensaje enviado por Fernandino es: Fernandino: adios
```

Servidor:



En conclusión, se ha aprendido cómo realizar un Servidor que atienda múltiples peticiones en local, a pesar de que con este mismo código también se podría conectar a otro ordenador con IP diferente.