

Trabajo práctico sobre **Demostración de Sockets utilizando Servidor y Cliente.**

Integrantes:

Marcos Flores Duarte.

Nicolas Daniel Arza Vega.

Sergio Ramón González Silva.

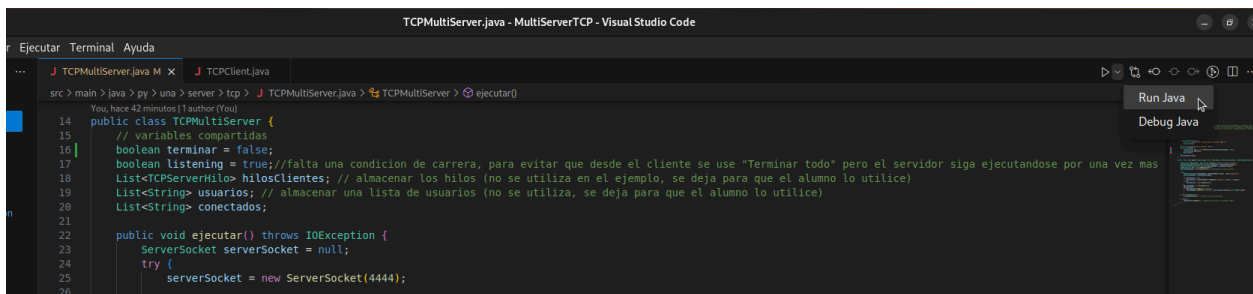
28 de septiembre del 2024

Como ejecutar:

Verifica que tenemos instalado y funcionando el servidor Redis.

```
sergio@sergio-Aspire-E5-573:~/Documentos/MultiServerTCP$ redis-cli
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> exit
```

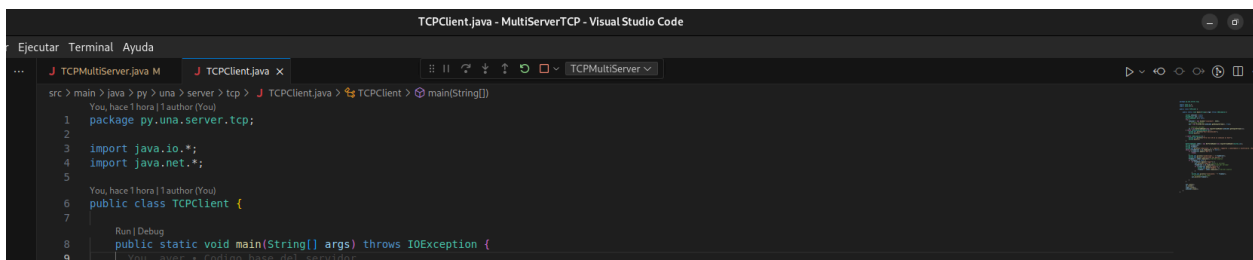
Ahora podemos ejecutar el TCPMultiServer.java.



```
src > main > java > py > una > server > tcp > TCPMultiServer.java > TCPMultiServer > ejecutar()
You, hace 42 minutos | author (You)
14 public class TCPMultiServer {
15     // variables compartidas
16     boolean terminar = false;
17     boolean listening = true; // falta una condicion de carrera, para evitar que desde el cliente se use "Terminar todo" pero el servidor siga ejecutandose por una vez mas
18     List<TCPServerHilo> hilosClientes; // almacenar los hilos (no se utiliza en el ejemplo, se deja para que el alumno lo utilice)
19     List<String> usuarios; // almacenar una lista de usuarios (no se utiliza, se deja para que el alumno lo utilice)
20     List<String> conectados;
21
22     public void ejecutar() throws IOException {
23         ServerSocket serverSocket = null;
24         try {
25             serverSocket = new ServerSocket(4444);
26
```

```
sergio@sergio-Aspire-E5-573:~/Documentos/MultiServerTCP$ /usr/bin/env /usr/lib/jvm
er
Puerto abierto: 4444.
```

Luego ejecutamos el TCPClient.java.



```
src > main > java > py > una > server > tcp > TCPClient.java > TCPClient > main(String[])
You, hace 1 hora | author (You)
1 package py.una.server.tcp;
2
3 import java.io.*;
4 import java.net.*;
5
6 You, hace 1 hora | author (You)
6 public class TCPClient {
7
8     Run | Debug
8     public static void main(String[] args) throws IOException {
9
10         // aqui poner el codigo base del servidor
```

```
sergio@sergio-Aspire-E5-573:~/Documentos/MultiServerTCP$ /usr/bin/env /usr/lib/jvm

Opciones:
» login
» logout
» conectados
» historial
» Bye
» Terminar todo

*Servidor: Bienvenido!

Seleccione una Opcion: █
```

Ya podemos ejecutar comandos utilizando el cliente.

Explicación del código del cliente.

Al ejecutar el cliente recibimos el siguiente mensaje en el terminal:

```
sergio@sergio-Aspire-E5-573:~/Documentos/MultiServerTCP$  
  
Opciones:  
  » login  
  » logout  
  » conectados  
  » historial  
  » Bye  
  » Terminar todo  
  
*Servidor: Bienvenido!  
  
Seleccione una Opcion: login
```

```
sergio@sergio-Aspire-E5-573:~/Documentos/MultiServerTCP$  
er  
Puerto abierto: 4444.
```

- **login:** Permite al usuario ingresar su nombre.

Al introducir nuestro nombre podemos elegir entre las opciones disponibles:

```
Seleccione una Opcion: login  
Nombre: Sergio  
  
*Cliente: Sergio  
  
*Servidor: Ha Iniciado Sesion
```

```
sergio@sergio-Aspire-E5-573:~/Documentos/MultiServerTCP$  
er  
Puerto abierto: 4444.  
Mensaje recibido: login
```

- **conectados:** lista los usuarios que están conectados en ese momento al servidor:

```
Seleccione una Opcion: conectados
*Cliente: conectados
*Servidor: Lista de usuarios:  » Sergio
```

- **Logout:** Cierra la sesión del usuario y quita al usuario de la lista de personas conectadas.

```
Seleccione una Opcion: logout
*Cliente: logout
*Servidor: Se Ha Cerrado la Sesion
```

```
o sergio@sergio-Aspire-E5-573:~/Documentos/MultiServerTCP$
er
Puerto abierto: 4444.
Mensaje recibido: login
Mensaje recibido: conectados
Mensaje recibido: logout
```

```
*Cliente: logout
*Servidor: Se Ha Cerrado la Sesion
Seleccione una Opcion: conectados
*Cliente: conectados
*Servidor: Lista de usuarios:
Seleccione una Opcion: █
```

- **historial:** Todos los usuarios que se han conectado al servidor:

```
Seleccione una Opcion: logout
*Cliente: logout
*Servidor: Se Ha Cerrado la Sesion
Seleccione una Opcion: login
Nombre: Marcos
*Cliente: Marcos
*Servidor: Ha Iniciado Sesion
Seleccione una Opcion: historial
*Cliente: historial
*Servidor: Historial de Usuarios:  » Sergio » Marcos
Seleccione una Opcion: □
```

```
○ sergio@sergio-Aspire-E5-573:~/Documentos/MultiServerTCP$
er
Puerto abierto: 4444.
Mensaje recibido: login
Mensaje recibido: conectados
Mensaje recibido: logout
Mensaje recibido: login
Mensaje recibido: historial
□
```

- **Comando incorrecto:** Al introducir un comando diferente a los establecidos el servidor responde con el mensaje de Comando no reconocido.

```
Seleccione una Opcion: otro
*Cliente: otro
*Servidor: Comando Desconocido
Seleccione una Opcion: █
```

```
sergio@sergio-Aspire-E5-573:~/Documentos/MultiServerTCP$
er
Puerto abierto: 4444.
Mensaje recibido: login
Mensaje recibido: conectados
Mensaje recibido: logout
Mensaje recibido: login
Mensaje recibido: historial
Mensaje recibido: logout
Mensaje recibido: conectados
Mensaje recibido: otro
█
```

Código del cliente:

```
J TCPMultiServer.java M J TCPClient.java x
src > main > java > py > una > server > tcp > J TCPClient.java > TCPClient > main(String[])
6 public class TCPClient {
8     public static void main(String[] args) throws IOException {
9
10         Socket unSocket = null;
11         PrintWriter out = null;
12         BufferedReader in = null;
13         try {
14             unSocket = new Socket("localhost", 4444);
15             // enviamos nosotros
16             out = new PrintWriter(unSocket.getOutputStream(), true);
17
18             // viene del servidor
19             in = new BufferedReader(new InputStreamReader(unSocket.getInputStream()));
20         } catch (UnknownHostException e) {
21             System.err.println("Host desconocido");
22             System.exit(1);
23         } catch (IOException e) {
24             System.err.println("Error de I/O en la conexion al host");
25             System.exit(1);
26         }
27
28         BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
29         String fromServer;
30         String fromUser;
31         System.out.println("\nOpciones: \n » login\n » logout\n » conectados\n » historial\n » Bye\n » Terminar todo");
32         while ((fromServer = in.readLine()) != null) {
33             if (fromServer.equals("Bye")) {
34                 break;
35             }
36             System.out.println("\n*Servidor: " + fromServer);
37             System.out.print("\nSeleccione una Opcion: ");
38             fromUser = stdIn.readLine();//lee del usuario
39             if (fromUser != null) {
40                 if (fromUser.equals("login")) {
41                     out.println("login");//envia al servidor
42                     fromServer = in.readLine();//lee del servidor
43                     if (fromServer.equals("login")) {
44                         System.out.print("Nombre: ");
45                         fromUser = stdIn.readLine();//lee del usuario
46                     }
47                 }
48                 System.out.println("\n*Cliente: " + fromUser);
49                 // escribimos al servidor
50                 out.println(fromUser);
51             }
52         }
53     }
```


Código del hilo del servidor:

```
J TCPMultiServer.java M X J TCPClient.java
src > main > java > py > una > server > tcp > J TCPMultiServer.java > TCPMultiServer > ejecutar()

14 public class TCPMultiServer {
22     public void ejecutar() throws IOException {
26
27     } catch (IOException e) {
28         System.err.println("No se puede abrir el puerto: 4444.");
29         System.exit(1);
30     }
31     System.out.println("Puerto abierto: 4444.");
32     while (listening) { You, ayer * Código base del servidor.
33         TCPServerHilo hilo = new TCPServerHilo(serverSocket.accept(), this);
34         hilosClientes.add(hilo);
35         hilo.start();
36     }
37     serverSocket.close();
38 }
39

Run|Debug
40 public static void main(String[] args) throws IOException, ExecutionException, InterruptedException {
41
42     RedisClient redisClient = RedisClient.create(uri:"redis://localhost:6379");
43     StatefulRedisConnection<String, String> connection = redisClient.connect();
44     RedisAsyncCommands<String, String> asyncCommands = connection.async();
45     TCPMultiServer tms = new TCPMultiServer();
46     tms.hilosClientes = new ArrayList<>(); // Inicializar la lista de hilos
47
48     try {
49         RedisFuture<Long> existsFuture = asyncCommands.exists(...keys:"usuarios");
50         Long existsCount = existsFuture.get();
51
52         if (existsCount > 0) {
53             tms.usuarios = asyncCommands.lrange(key:"usuarios", start:0, -1).get();
54         } else {
55             tms.usuarios = new ArrayList<>();
56         }
57         tms.conectados = new ArrayList<>();
58         tms.ejecutar();
59         if (tms.usuarios.isEmpty() == false) {
60             asyncCommands.lpush(key:"usuarios", tms.usuarios.toArray(new String[0])).get();
61         }
62     } catch (Exception e) {
63         e.printStackTrace(); // Manejo básico de excepciones
64     } finally {
65         redisClient.shutdown(); // Asegurarse de cerrar la conexión a Redis
66     }
67 }
68
69 }
```

```
J TCPMultiServer.java M J TCPServerHilo.java X J TCPClient.java
src > main > java > py > una > server > tcp > J TCPServerHilo.java > TCPServerHilo > run()

You, hace 2 horas | 1 author (You)
7 public class TCPServerHilo extends Thread {
8
9     private Socket socket = null;
10
11     TCPMultiServer servidor;
12
13     public TCPServerHilo(Socket socket, TCPMultiServer servidor ) {
14         super("TCPServerHilo");
15         this.socket = socket;
16         this.servidor = servidor;
17     }
18
19     public void run() {
20
21         try {
22             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
23             BufferedReader in = new BufferedReader( new InputStreamReader( socket.getInputStream() ) );
24             out.println("Bienvenido!");
25             String inputLine, outputLine;
26             String user = null;
27             while ((inputLine = in.readLine()) != null) {
28                 System.out.println("Mensaje recibido: " + inputLine);
29
30                 //to-do: utilizar json
31                 if (inputLine.equals("Bye")) {
32                     outputLine = "Usted apago el hilo";
33                     if (user != null) {
34                         servidor.conectados.remove(user);
35                     }
36                     System.out.println(outputLine);
37                     break;
38
39                 } else if (inputLine.equals("Terminar todo")) {
40                     servidor.listening = false;    You, ayer * Código base del servidor.
41                     if (user != null) {
42                         servidor.conectados.remove(user);
43                     }
44                     outputLine = "Usted apago todo";
45                     System.out.println(outputLine);
46                     break;
47
48                 } else if (inputLine.equals("login")) {
49                     out.println("login");//envia al cliente
50                     String nombre = in.readLine();//lee del cliente
51                     if (false == servidor.conectados.contains(nombre)) { //pregunta si es que existe otro user
52                         servidor.conectados.add(nombre);
```

```
J TCPMultiServer.java M J TCPServerHilo.java X J TCPClient.java
src > main > java > py > una > server > tcp > J TCPServerHilo.java > TCPServerHilo > run()
7 public class TCPServerHilo extends Thread {
19 public void run() {
50 String nombre = in.readLine();//lee del cliente
51 if (false == servidor.conectados.contains(nombre)) { //pregunta si es que existe otro user
52     servidor.conectados.add(nombre);
53     user = nombre;
54     outputLine = "Ha Iniciado Sesion";
55     if (false == servidor.usuarios.contains(nombre)){
56         servidor.usuarios.add(nombre);
57     }
58 }else{
59     outputLine = "El Usuario ya Existe";
60 }
61 }
62
63 } else if (inputLine.equals("logout")) {
64     servidor.conectados.remove(user);
65     user = null;
66     outputLine = "Se Ha Cerrado la Sesion";
67
68 } else if (inputLine.equals("conectados")) {
69     outputLine = "Lista de usuarios: " ;
70     Iterator<String> iter = servidor.conectados.iterator();
71     while (iter.hasNext()) {
72         outputLine = outputLine + " » " + iter.next();
73     }
74
75 } else if (inputLine.equals("historial")) {
76     outputLine = "Historial de Usuarios: " ;
77     Iterator<String> iter = servidor.usuarios.iterator();
78     while (iter.hasNext()) {
79         outputLine = outputLine + " » " + iter.next();
80     }
81
82 }else{
83     outputLine = "Comando Desconocido";
84 }
85 out.println(outputLine);
86 }
87 out.close();
88 in.close();
89 socket.close();
90 System.out.println("Finalizando Hilo");
91
92 } catch (IOException e) {
93     e.printStackTrace();
94 }
```