CHAT

Alumno: Cristhian González
Curso: 2º DAM
Módulo: Programación de servicios
y procesos
Fecha: 25/02/2020

1. INTRODUCCIÓN

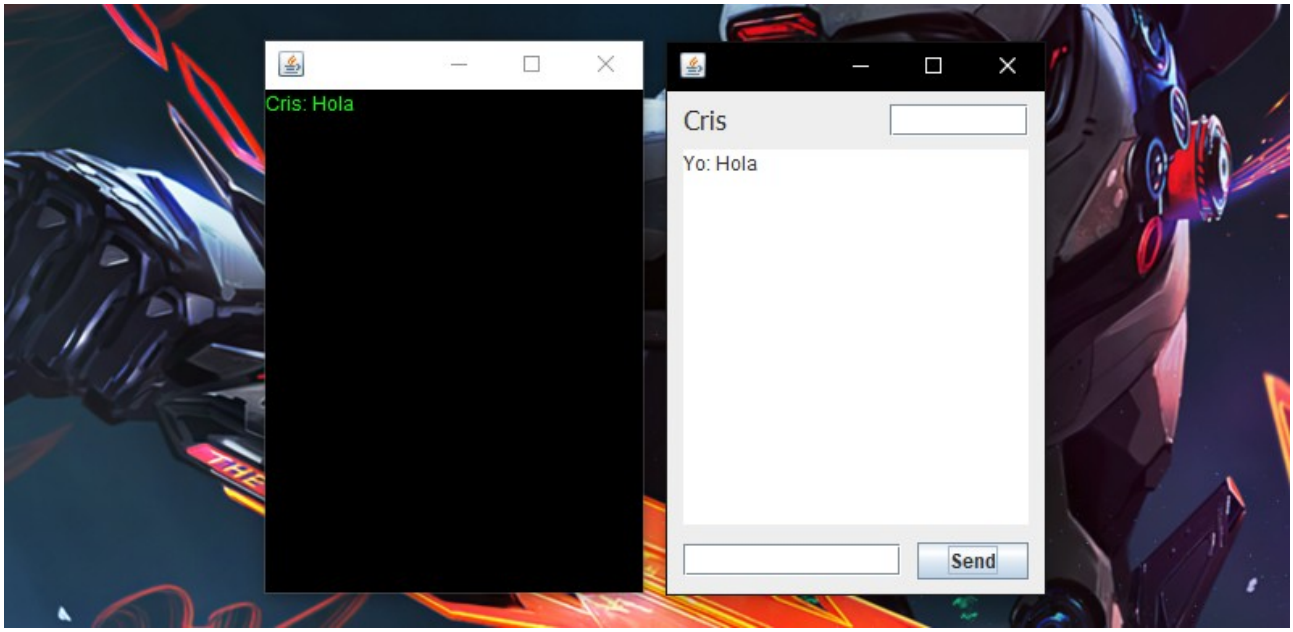
En esta práctica realizaré un proyecto divido en dos programas, uno de ellos actuará como servidor, permitiendo la comunicación entre varios interlocutores, y el segundo actuará como cliente, permitiendo a los usuarios comunicarse mediante una interfaz gráfica. Para realizar esta tarea utilizaré Eclipse como IDE, Java Swing para el desarrollo de ventanas.

2. DISEÑANDO CON SWING

Antes de comenzar a escribir todo lo código necesario para que mi chat funcionase lo primero que hice fueron las ventanas o interfaces que iba a necesitar para llevar a cabo mis pruebas una vez haya completado el funcionamiento del servidor y cliente.

Para el servidor me decidí por crear una simple ventana con un TextArea que vaya mostrando todos los mensajes que se reciben, le di un toque de consola. Por otro lado para el cliente necesitaba un Label que mostrará en todo momento el nombre de la

persona que chateba, dos TextField donde introducir los mensajes y la IP a la que queriamos enviarlo, además de un TextArea donde mostrarlos. Con esto ya hecho podía empezar a implementarle las funcionalidades necesarias para que funcionase y a la vez hacer las primeras pruebas.



3. CREANDO EL SERVIDOR

Lo primero a realizar será el servidor, ya que lo necesitaremos como punto intermedio para comunicar a los clientes, ya que es el que se encargará de gestionar el tráfico. Para ello utilizaré Sockets, algo que podríamos describir como un puente virtual que va a comunicar el servidor con un cliente por lo que podremos pasar información.

Para definir los Sockets he tenido que averiguar la dirección del servidor, en mi caso mi equipo, tanto como el puerto de recepción que es por donde se circulará la información, para ello solo he tenido que usar un ipconfig en la consola y asegurarme de que el puerto que utilizo no está ocupado.

Lo primero que he hecho es poner el servidor a la escucha permanentemente, el puerto recomendado a utilizar sería el 9999, esto mediante un hilo, ya que además de estar a la escucha tiene que mostrar en el TextArea lo que se está escribiendo en todo momento

por lo que mi clase consola debía implementar Runnable `public class Console extends JFrame implements Runnable.` además de crear un Thread y arrancarlo.

```
/** Nuevo hilo de ejecución que permanecerá a la escucha de nuevos mensajes */
private Thread thread;

 /** Arranco el hilo */

                thread = new Thread(this);
                thread.start();
```

Una vez arrancado el hilo en el metodo run, el cual me ha obligado a implementar Runnable es donde implementaré el socket para el servidor, el cual corresponde con la clase **ServerSocket** de JavaNET que crea un Socket de servidor.

4. CREANDO EL CLIENTE

Para el lado del cliente debía poner a la escucha al botón de envio para que al pulsar hiciese las operaciones pertinentes, la cual sería crear el Socket y así enviar los datos al servidor. En el cliente tambien debía crear la clase DataPacket, ya que al estar en proyectos diferentes esto no se comparte.

```
JButton btnSend = new JButton("Send");
                btnSend.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                Socket socket;
                                ObjectOutputStream oos;

                                if (!textField.equals("")) {
                                        textArea.append("Yo: " + textField.getText() + "\n");

                                        try {
                                                // Creamos el socket apuntando a la dirección del servidor
                                                socket = new Socket("192.168.1.39", 9999);
                                                // Además del Puerto que estará abierto

                                                // Mediante la clase DataPacket defino lo que será enviado
                                                DataPacket data = new DataPacket();

                                                data.setNickname(nickname);
                                                data.setIp(tfIp.getText());
                                                data.setMessage(textField.getText());

                                                oos = new ObjectOutputStream(socket.getOutputStream());
                                                oos.writeObject(data);
                                                oos.close();

                                                // Vuelvo a vaciar la caja de texto.
                                                textField.setText("");
                                        } catch (IOException ioe) {
                                                ioe.printStackTrace();
                                        }
                                }
                        }
                });
```

5. COMUNICANDO SERVIDOR Y CLIENTE

Con el ServerSocket creado ya podría empezar a enviar información a traves de los flujos de datos, pero claro esto es un chat por lo que necesitaba aparte del mensaje, saber el nombre de la persona que lo envía y la dirección IP de esta, por lo que tenía que empaquetar toda esta información en un modelo o una clase, a la cual he llamado DataPacket, que contiene el nombre, la ip y el mensaje, además de estar Serializada ya que vamos a enviar un objeto por la red.

Hasta aquí ya había conseguido comunicar un cliente con el servidor, pero lo que necesitabamos ahora es que esta información que nos envia el cliente atraves del servidor sea enviada a otro cliente.

Para esto necesitaba que el cliente tambien estuviese siempre a la escucha por lo que tendriamos que crear tambien un ServerSocket en el cliente y por otro lado el servidor debía ser capaz de enviar la información a la IP que le tocaba. Para ello e implementado el siguiente código en el servidor:

```
textArea.append(nickname + ": " + message + "\n");

                                    // Creo un nuevo socket que es el que se encargará de enviar
                                    // la información recibida al cliente deseado.
                                    socketReceiver = new Socket(ip, 9090);

                                    /** Obtengo un nuevo flujo de salida por donde escribir el
objeto recibido */
                                    oos = new
ObjectOutputStream(socketReceiver.getOutputStream());
                                    oos.writeObject(data);

                                    socketReceiver.close();
                                    socket.close();
                                    ois.close();
                                    oos.close();
```

Y por el otro lado, el del cliente he tenido que crear un ServerSocket y para esto he tenido que crear otro hilo, por lo que esta clase tambien implementa la interfaz Runnable.

```
@Override
	public void run() {
		ServerSocket serverSocket;
		Socket client;
		DataPacket data;
		ObjectInputStream ois;

		try {
			// Pone a la escucha al cliente en el 9090.
			serverSocket = new ServerSocket(9090);

			while (true) {
				// Socket por el que se recibe el paquete y acepta las conexiones.
				client = serverSocket.accept();
				ois = new ObjectInputStream(client.getInputStream());
				data = (DataPacket) ois.readObject();

				textArea.append(data.getNickname() + ": " + data.getMessage() + "\n");
			}

		} catch (IOException | ClassNotFoundException e) {
			e.printStackTrace();
		}
	}
```

5. SIMULACIÓN

Llegado a este punto mi aplicación ya debería funcionar así que me puse a simular un enterno en el que debía comunicar dos clientes. Una de las opciones era encender otro de mis ordenadores y usarlo de cliente, pero preferí instalar un entorno virtual de Windows 10 con Virtual Box, por lo que procedi a descargar el disco iso de evaluación de W10, fue una descarga bastante lenta. Una vez con mi máquina virtual ya podía simular un entorno de chat, ya que mi equipo hacía de servidor y a su vez podría abrir un cliente. Tuve que generar un archivo JAR del programa cliente y enviarselo a atraves de una carpeta compartida a la máquina virtual para empezar con las pruebas.

Finalizo la practica aquí, aunque hecho varias cosas más, pero están incompletas ya que no he tenido tiempo de seguir con ello, por lo que las clases de más están comentadas, o simplemente no les doy uso así que no interrumpen el funcionamiento básico del proyecto.

Los resultados finales estarán presentados en un video adjunto.