Chat grupal con sockets

Julio Antonio Fresneda García y Santiago Vidal Martínez

November 22, 2017

¿Qué hace nuestro software?

Nuestro sistema es un chat en el cual se puede establecer una conexión entre clientes de la misma red. De esta manera podrá existir una ventana de chat con múltiples usuarios.

Primero comentaremos el código de nuestro sistema, que lo hemos dividido en tres archivos, uno que es lo que ejecutará nuestro servidor (el cual lanzará las hebras de los clientes), otro que contiene la funcionalidad de cada hebra y el último incluye el sistema que utilizarán los clientes.

Explicaremos esto con detalle a continuación.

El código del servidor es el siguiente:

```
import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

import java.net.ServerSocket;
import java.net.Socket;

import java.util.ArrayList;

// Clase servidor
public class ServidorChatGrupal
{

    static ArrayList<String> nicks = new
       ArrayList<String>();
    static ArrayList<PrintWriter> printWrites = new
       ArrayList<PrintWriter>();

    private static final int PUERTO = 8989;

    public static void main(String[] args) throws
       Exception
    {
      // Servidor lanzado
      System.out.println("Servidor OK");

      // Creamos el ServerSocket
      ServerSocket socketServicio = new ServerSocket(
         PUERTO );

      try
      {
          while( true )
          {
```

```
            new Hebrita( socketServicio.accept()
                ).start();
          }
      } catch( IOException e ){}

    }

}
```

Como se puede observar tenemos una lista de nicks y una lista de buffers (print-Writes)donde escribirá cada uno de los usuarios. Además por otro lado hemos establecido que el puerto donde va a escuchar nuestro servidor va a ser el 8989.

Nuestro servidor abre un socket que escucha dicho puerto, esto lo realiza con new ServerSocket (PUERTO) .

Una vez abierto, el servidor lanza una hebra cada vez que recibe una petición al socket. Esto lo realiza realizando el método start() de la instancia de Hebrita.

El siguiente código es el cliente:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;

import java.net.Socket;

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.JLabel;
import javax.swing.ImageIcon;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;


// Clase del cliente
public class ClienteChatGrupal
{
    JFrame jframe = new JFrame("Chat Grupal");

    JTextField jtextField = new JTextField(45);
    JTextArea jtextArea = new JTextArea(10, 45);

    BufferedReader inReader;
```

```
PrintWriter outPrinter;



public ClienteChatGrupal()
{
    jframe.getContentPane().add( new JLabel(new
       ImageIcon("./cabecera.png")), "North" );
    jframe.getContentPane().add( new JScrollPane(
       jtextArea ), "Center" );
    jframe.getContentPane().add( jtextField, "South"
       );
    jframe.pack();

    jtextArea.setEditable( false );

    jtextField.addActionListener(
      new ActionListener()
      {
        public void actionPerformed( ActionEvent e )
        {
            outPrinter.println( jtextField.getText()
               );
            jtextField.setText("");
        }
      }
    );


}



// Ejecutamos la hebra
private void run() throws IOException
{

  // Captamos la ip a la que nos vamos a conectar,
     escrita desde la interfaz
  String ip = JOptionPane.showInputDialog( jframe,
     "Introduce IP del servidor a conectar:",
     "Introduciendo IP:",
     JOptionPane.QUESTION_MESSAGE);

  // Creamos un socketConexion, lo conectamos al
     puerto 8989 (puerto del servidor)
  Socket socketConexion = new Socket(ip, 8989);

  outPrinter = new PrintWriter(
     socketConexion.getOutputStream(), true );
```

```
        inReader = new BufferedReader( new
            InputStreamReader(
            socketConexion.getInputStream() ) );


        while( true )
        {

            String readLine = inReader.readLine();

            // Si el mensaje recibido empieza con
                "enviarnick", se lanza una ventana pidiendo
                el nick
            if( readLine.startsWith("enviarnick"))
            {
                outPrinter.println(
                    JOptionPane.showInputDialog( jframe,
                    "Escribe tu nick:", "Introduciendo
                    nick:",  JOptionPane.PLAIN_MESSAGE) );
            }
            else if( readLine.startsWith("nickok") )
            {
                jtextField.setEditable(true);
            }
            // Si el mensaje recibido desde el servidor
                (el que haya enviado otro cliente) empieza
                con "mensaje", el mensaje se carga en la
                interfaz para que lo podamos ver
            else if (readLine.startsWith("mensaje"))
            {
                jtextArea.append(readLine.substring(8) +
                    "\n");
            }

        }
    }


    public static void main(String[] args) throws
        Exception
    {
        ClienteChatGrupal cliente = new
            ClienteChatGrupal();
        cliente.jframe.setVisible( true );
        cliente.run();
    }
}
```

El cliente tendrá la interfaz gráfica. El titulo del sistema (Chat grupal) estará colocado con un frame. Además tendrá dos áreas de texto uno para leer y otro para escribir.

Además se realiza la conexión con el socket del servidor conectandose a la ip que se le pasa por la interfaz y se almacena en la variable ip del método run() y atendiendo al puerto 8989 se abrirá una conexión a un socket. Esto se realizará con new Socket(ip,8989).

Cuando se establece la conexión se establecen la conexión con los flujos de entrada y salida. Si lee en la linea el estado enviarnick la linea que se escriba en la interfaz (habilitada una concreta aunque valdría en la de chat si no existiera la interfaz que hemos creado para dicha funcionalidad), pasará a ser el nick de nuestro cliente.

Si el nick es correcto accede al chat y si lo que lee es un mensaje lo lee del flujo de entrada.

Por último tenemos la hebra que ejecuta nuestro servidor:

```
import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

import java.net.Socket;


// Clase Hebrita
class Hebrita extends Thread
{
  private Socket socket;
  private String nick;

  private BufferedReader inReader;
  private PrintWriter outPrinter;

  // Constructor
  public Hebrita( Socket _socket )
  {
      socket = _socket;
  }


  public void run() {

    try
    {
```

```
        inReader = new BufferedReader( new
            InputStreamReader( socket.getInputStream() )
            );
        outPrinter = new PrintWriter(
            socket.getOutputStream(), true );


        while( true )
        {
          // Pedimos el nick
          outPrinter.println("enviarnick");

          // Leemos el nick que el cliente haya escrito
          nick = inReader.readLine();

          if( nick != null &&
              !ServidorChatGrupal.nicks.contains( nick ))
              {
               ServidorChatGrupal.nicks.add( nick );
               break;
          }

          if( nick == null ) return;

        }

        outPrinter.println("nickok");

        ServidorChatGrupal.printWrites.add(outPrinter);

        while( true )
        {
          // Leemos el mensaje de un cliente
          String input = inReader.readLine();

          if( input == null ) return;

          // Enviamos el mensaje recibido de ese cliente
             a cada uno de los clientes (incluido el
             mismo que lo ha enviado)
          for (PrintWriter pw :
              ServidorChatGrupal.printWrites) {
               pw.println("mensaje " + nick + ": " +
                   input);
          }

        }

    } catch (IOException e) {}
```

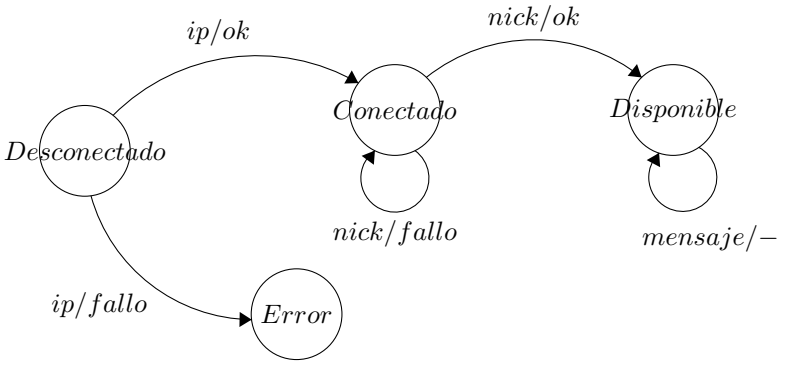
```
  }
}
```

Las funciones de esta hebra es pedir el nick y leer el que nos pasa el cliente. Nos aseguramos que el nick no esté ya dentro del chat. Si no existe el nick introducido el nick está en estado ok.

A partir de aqui procede a leer o escribir y la forma de realizarlo es tan sencilla como cuando recibe un mensaje de un cliente se distribuye al resto de los clientes.

El mensaje que se envía incluye la secuencia mensaje <nick>: <mensaje>.

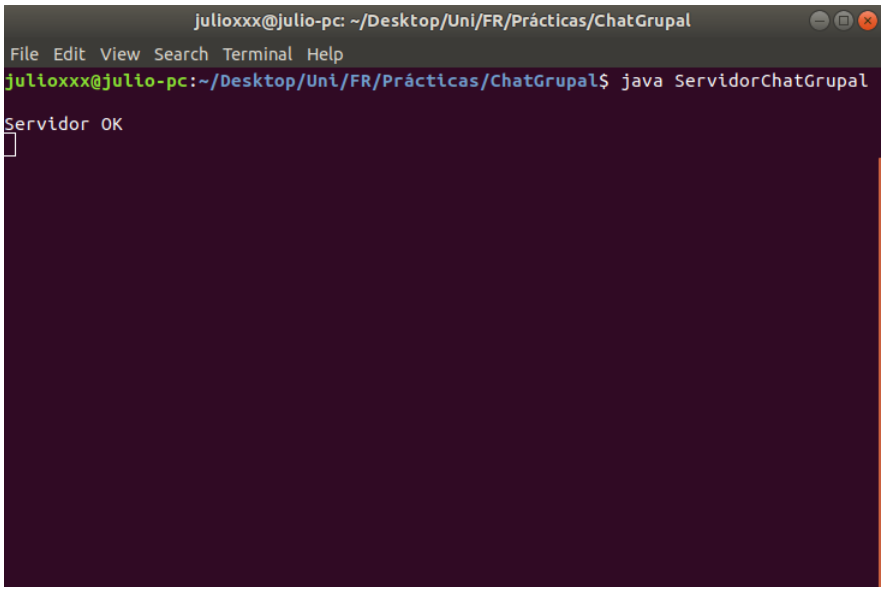
Diagrama de estados



Mensajes

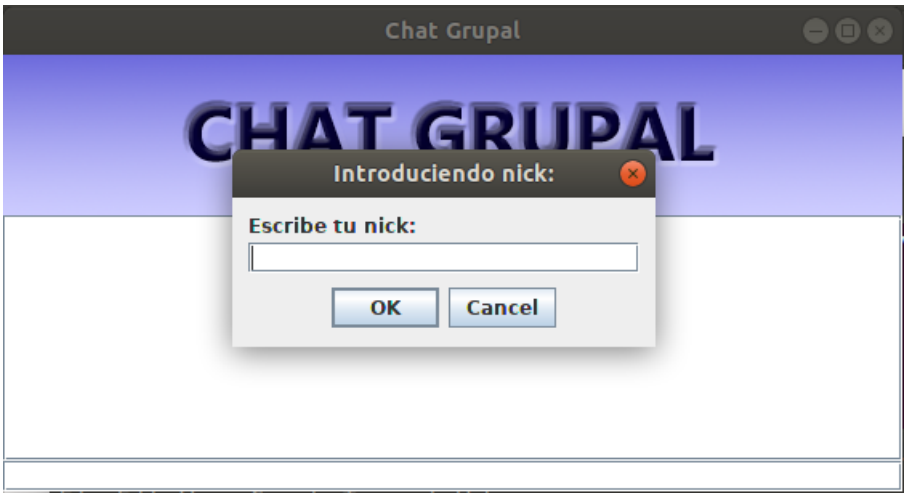
Mensaje	Campos	Respuesta
IP	dirección ip del servidor	0 o 1 (conexión realizada o fallida)
Nick	Nombre de usuario	0 o 1 (se ha podido crear o no el usuario)
Mensaje	Información a transmitir por el chat	No devuelve nada

Evaluación de la aplicación



Podemos observar que al ejecutar el programa desde nuestro servidor, este se queda en estado OK a la espera de recibir peticiones de los clientes.

A cada cliente le saldrá la ventana anterior, es imprescindible utilizar una ip válida (la del servidor) y a la cual estaremos conectado para comunicarnos con otros clientes.

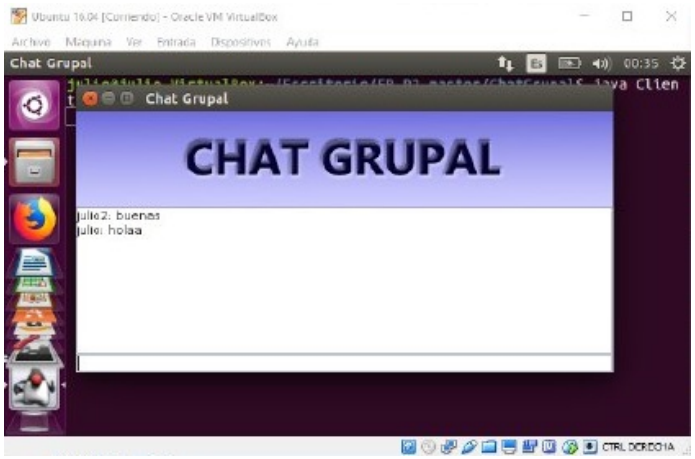


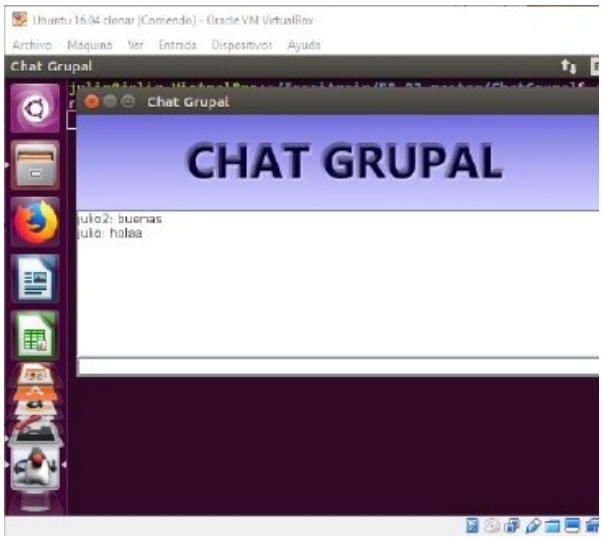
A continuación, se tendrá que introducir en la siguiente interfaz un nick que no esté ya dentro de la ventana de chat. Tras este paso ya se podrá proceder a chatear salvo que el nick ya exista.

Como se puede observar la caja inferior es para escribir el mensaje que queremos enviar a través de nuestro chat.

Una vez enviado el mensaje se observará como en la foto previa. Para otros clientes se verá de forma diferente.

A continuación vamos a mostrar el chat conectado al mismo servidor entre dos clientes:





Por último veremos la conexión entre cuatro clientes: