



“Informe de ejecución y explicación del Chat creado con Sockets en códigos JAVA”

Programación Avanzada



Ing. David Santiago Velásquez Cifuentes.¹

*Maestría en Ingeniería. Facultad de Ingeniería y Ciencias Básicas.
Fundación Universitaria Los Libertadores.*

1. Introducción

En esta actividad se ha desarrollado un sistema de chat virtual mediante el uso de sockets en el lenguaje de programación Java. Los sockets son puntos finales de conexión que permiten la comunicación bidireccional entre dos programas, facilitando la transmisión de datos a través de una red. En esencia, los sockets sirven como interfaz de programación para la comunicación entre procesos, ya sea en la misma máquina o a través de una red.

El objetivo principal de este informe es presentar una ejecución detallada y comprensiva del sistema de chat implementado, así como introducir los códigos creados y cargados al repositorio personal. Se ilustra por medio de capturas de pantalla la correcta ejecución de los códigos en la consola y, posteriormente, la carga exitosa de dichos códigos en el repositorio GitHub. El sistema de chat se basa en la utilización de sockets para establecer la conexión entre el servidor y los clientes. El servidor actúa como un intermediario que facilita la comunicación entre los diferentes clientes conectados. La interacción entre los usuarios se lleva a cabo mediante mensajes que son transmitidos a través de la red; la prueba del funcionamiento del chat se realizó con dos computadores conectados a la misma red Wifi.

2. Explicación del código del Servidor

```
import java.io.*;
import java.net.*;
import java.util.*;

public class Servidor {
    private static final int PUERTO = 6789;
    private static Set<ClienteHandler> clientHandlers = Collections.synchronizedSet(new HashSet<>());

    public static void main(String[] args) throws IOException {
        System.out.println("El servidor se ha inicializado y está en uso.");
        ServerSocket listener = new ServerSocket(PUERTO);

        try {
            while (true) {
                // Espera a que un cliente se conecte y crea un nuevo hilo para manejarlo
                Socket socket = listener.accept();
                ClienteHandler handler = new ClienteHandler(socket);
                clientHandlers.add(handler);
                handler.start();
            }
        } finally {
            listener.close();
        }
    }

    private static class ClienteHandler extends Thread {
        private Socket socket;
        private PrintWriter out;
        private BufferedReader in;
        private String nombreUsuario;
    }
}
```

Imagen 1 – Muestra del código creado para el servidor de la aplicación

¹ Email: dsvelasquezc@ulibertadores.edu.co

Este código implementa un servidor de chat básico en Java utilizando sockets. La clase principal es `Servidor`, que se encarga de aceptar conexiones de clientes mediante el uso de la clase `ServerSocket` en el puerto especificado (6789). Se crea un nuevo hilo (`ClienteHandler`) para cada cliente conectado, lo que permite manejar múltiples conexiones de manera concurrente.

La clase interna `ClienteHandler` extiende la clase `Thread` y se encarga de manejar la comunicación con un cliente específico. Cada instancia de `ClienteHandler` tiene flujos de entrada y salida para la comunicación bidireccional con el cliente. El servidor escucha mensajes del cliente y los difunde a todos los clientes conectados, permitiendo una comunicación en tiempo real entre ellos. Además, se manejan eventos como la entrada y salida de un usuario, mostrando mensajes informativos en la consola del servidor.

La implementación utiliza un conjunto sincronizado (`clientHandlers`) para almacenar las instancias de `ClienteHandler` que representan a los clientes conectados. Cuando un cliente se desconecta, se cierran los recursos asociados y se eliminan de este conjunto. El código maneja adecuadamente las excepciones de entrada/salida y utiliza la consola para informar sobre eventos importantes, como la entrada y salida de usuarios.

3. Explicación del código de la aplicación de Chat

A screenshot of a Java IDE window titled 'ChatApp'. The window has a menu bar with 'Archivo', 'Editar', and 'Ver'. The code is for a public class ChatApp. It has several private fields: JFrame frame, JTextPane textPaneChat, JTextField textFieldInput, JButton sendButton, PrintWriter out, Socket socket, StyledDocument doc, Map<String, Color> userColors, SimpleAttributeset userStyle, and String nombreUsuario. The ChatApp() constructor initializes userColors as a new LinkedHashMap, userStyle as a new SimpleAttributeset, and calls initializeGUI(), establishListeners(), and initiateConnection(). The initializeGUI() method configures the GUI: it creates a JFrame titled 'David's Chat Interface', a JTextPane, sets it to non-editable, gets its StyledDocument, creates a JTextField with the text 'ESCRIBA SU USUARIO', sets its foreground to GRAY, and adds a FocusListener that resets the text and foreground color to BLACK when the field gains focus. It also creates an 'Enter' button and a JPanel with a BorderLayout.

```
public class ChatApp {
    private JFrame frame;
    private JTextPane textPaneChat;
    private JTextField textFieldInput;
    private JButton sendButton;
    private PrintWriter out;
    private Socket socket;
    private StyledDocument doc;
    private Map<String, Color> userColors;
    private SimpleAttributeset userStyle;
    private String nombreUsuario;

    public ChatApp() {
        userColors = new LinkedHashMap<>();
        userStyle = new SimpleAttributeset();

        initializeGUI();
        establishListeners();
        initiateConnection();
    }

    private void initializeGUI() {
        // Configura la interfaz gráfica del cliente
        frame = new JFrame("David's Chat Interface");
        textPaneChat = new JTextPane();
        textPaneChat.setEditable(false);
        doc = textPaneChat.getStyledDocument();
        textFieldInput = new JTextField("ESCRIBA SU USUARIO");
        textFieldInput.setForeground(Color.GRAY);
        textFieldInput.addFocusListener(new FocusAdapter() {
            @Override
            public void focusGained(FocusEvent e) {
                if (textFieldInput.getText().equals("ESCRIBA SU USUARIO")) {
                    textFieldInput.setText("");
                    textFieldInput.setForeground(Color.BLACK);
                }
            }
        });
        sendButton = new JButton("Enter");

        JPanel panelInferior = new JPanel(new BorderLayout());
```

Imagen 2- Muestra del código creado para la interfaz de la aplicación creada

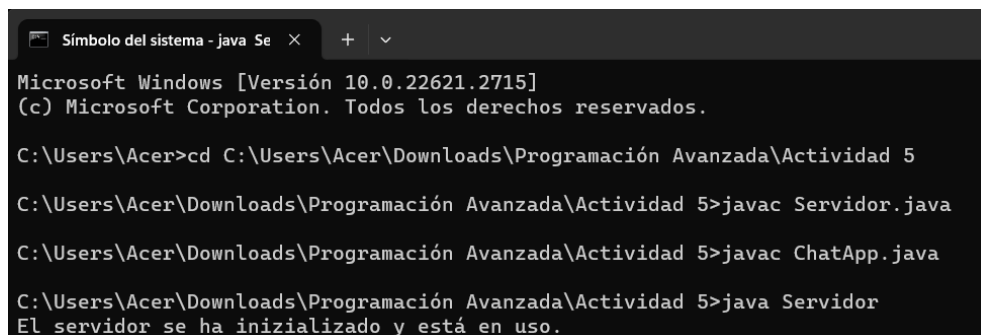
Actividad 5 – “Informe de ejecución y explicación del Chat creado con Sockets en códigos JAVA”

Este código Java implementa una aplicación de chat básica utilizando Swing y sockets. La interfaz gráfica permite a los usuarios ingresar un nombre de usuario, luego se conecta a un servidor a través de un socket para la comunicación en red. La aplicación utiliza hilos para manejar la entrada y salida de datos de manera asíncrona.

La clase `ChatApp` inicializa la interfaz gráfica del cliente, establece los listeners para el campo de entrada y el botón de enviar, y maneja la lógica de conexión al servidor. El programa utiliza colores para diferenciar a los usuarios y actualiza la interfaz gráfica en tiempo real cuando se reciben mensajes del servidor.

En el método `initiateConnection()`, se establece una conexión con el servidor a través de un socket y se configuran flujos de entrada/salida para la comunicación. Se inicia un hilo separado para escuchar mensajes del servidor y actualizar dinámicamente la interfaz de usuario cuando se reciben nuevos mensajes. La aplicación también maneja mensajes específicos, como la entrada de un nuevo usuario o la salida de un usuario, resaltando estos eventos en negrita en la interfaz gráfica.

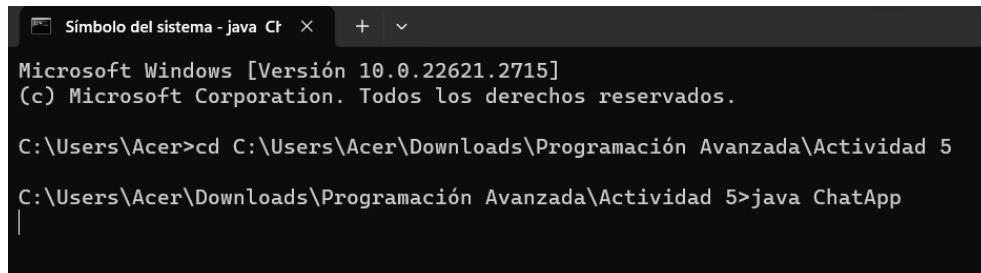
4. Ejecución de los ejemplos de excepciones en CMD



```
Símbolo del sistema - java Se X + v
Microsoft Windows [Versión 10.0.22621.2715]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Acer>cd C:\Users\Acer\Downloads\Programación Avanzada\Actividad 5
C:\Users\Acer\Downloads\Programación Avanzada\Actividad 5>javac Servidor.java
C:\Users\Acer\Downloads\Programación Avanzada\Actividad 5>javac ChatApp.java
C:\Users\Acer\Downloads\Programación Avanzada\Actividad 5>java Servidor
El servidor se ha inicializado y está en uso.
```

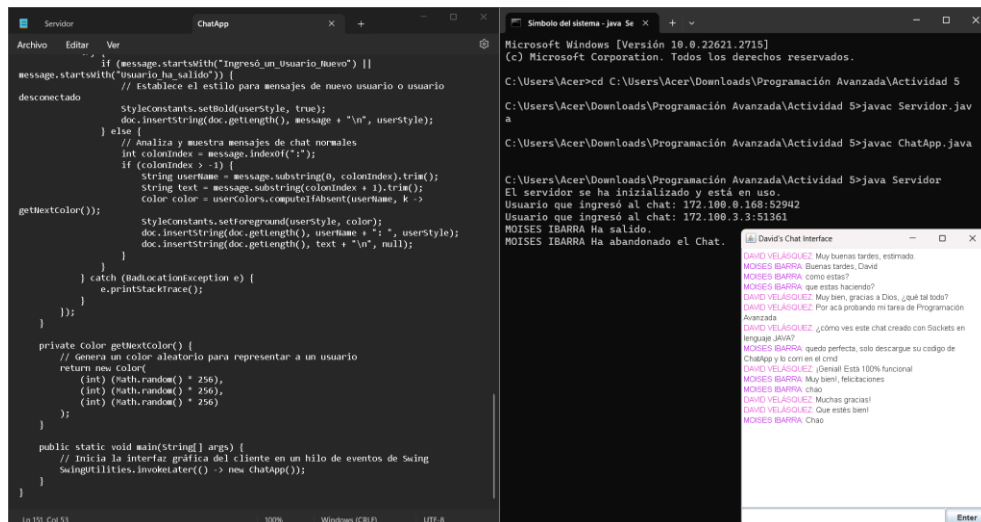
Imagen 3 - Compilación de los códigos y funcionamiento del servidor



```
Símbolo del sistema - java Ct X + v
Microsoft Windows [Versión 10.0.22621.2715]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Acer>cd C:\Users\Acer\Downloads\Programación Avanzada\Actividad 5
C:\Users\Acer\Downloads\Programación Avanzada\Actividad 5>java ChatApp
```

Imagen 4 - Ejecutando la aplicación de Chat desde CMD



```
Archivo Editar Ver
mensaje.startsWith("Ingresó un usuario nuevo") ||
mensaje.startsWith("usuario ha salido")) {
    // Establece el estilo para mensajes de nuevo usuario o usuario
    desconectado
        styleConstants.setBold(userStyle, true);
        doc.insertString(doc.getLength(), message + "\n", userStyle);
    } else {
        // Analiza y muestra mensajes de chat normales
        int colonIndex = message.indexOf(":");
        if (colonIndex > -1) {
            String username = message.substring(0, colonIndex).trim();
            String text = message.substring(colonIndex + 1).trim();
            color = userColors.computeIfAbsent(username, k ->
                styleConstants.setForeground(userStyle, color);
            doc.insertString(doc.getLength(), username + ": " + text, userStyle);
            doc.insertString(doc.getLength(), text + "\n", null);
        }
    }
} catch (BadLocationException e) {
    e.printStackTrace();
}
});
}

private Color getNextColor() {
    // Genera un color aleatorio para representar a un usuario
    return new Color(
        (int) (Math.random() * 256),
        (int) (Math.random() * 256),
        (int) (Math.random() * 256)
    );
}

public static void main(String[] args) {
    // Inicia la interfaz gráfica del cliente en un hilo de eventos de Swing
    SwingUtilities.invokeLater(() -> new ChatApp());
}
```

```
Símbolo del sistema - java Se X + v
Microsoft Windows [Versión 10.0.22621.2715]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Acer>cd C:\Users\Acer\Downloads\Programación Avanzada\Actividad 5
C:\Users\Acer\Downloads\Programación Avanzada\Actividad 5>javac Servidor.java
C:\Users\Acer\Downloads\Programación Avanzada\Actividad 5>javac ChatApp.java
C:\Users\Acer\Downloads\Programación Avanzada\Actividad 5>java Servidor
El servidor se ha inicializado y está en uso.
Usuario que ingresó al chat: 172.100.0.168:52942
Usuario que ingresó al chat: 172.100.3.3:51361
MOISES IBARRA Ha salido.
MOISES IBARRA Ha abandonado el Chat.
```

David's Chat Interface

DAVID VELAZQUEZ: Muy buenas tardes, estimado
MOISES IBARRA: Buenas tardes, David
MOISES IBARRA: como estas?
MOISES IBARRA: que estas haciendo?
DAVID VELAZQUEZ: Muy bien, gracias a Dios, ¿que tal todo?
DAVID VELAZQUEZ: Por acá probando mi tarea de Programación Avanzada
DAVID VELAZQUEZ: ¿cómo ves este chat creado con Sockets en lenguaje JAVA?
MOISES IBARRA: quedo perfecta, solo descargue su código de ChatApp y lo corré en el cmd
DAVID VELAZQUEZ: Genial! Está 100% funcional
MOISES IBARRA: Muy bien!, felicitaciones
MOISES IBARRA: chao
DAVID VELAZQUEZ: Muchas gracias!
DAVID VELAZQUEZ: Que estás bien!
MOISES IBARRA: Chao

Imagen 5 - Código creado, control desde la consola, y visualización de la interfaz de chat

Actividad 5 – “Informe de ejecución y explicación del Chat creado con Sockets en códigos JAVA”

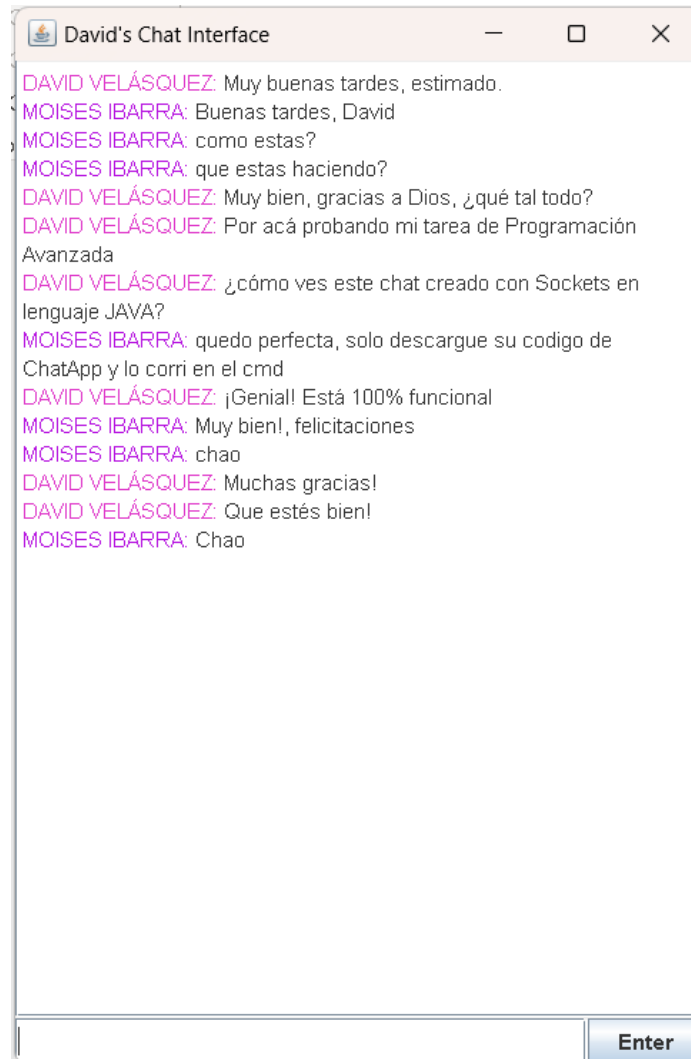


Imagen 6 - Interfaz GUI, prueba de la aplicación de Chat

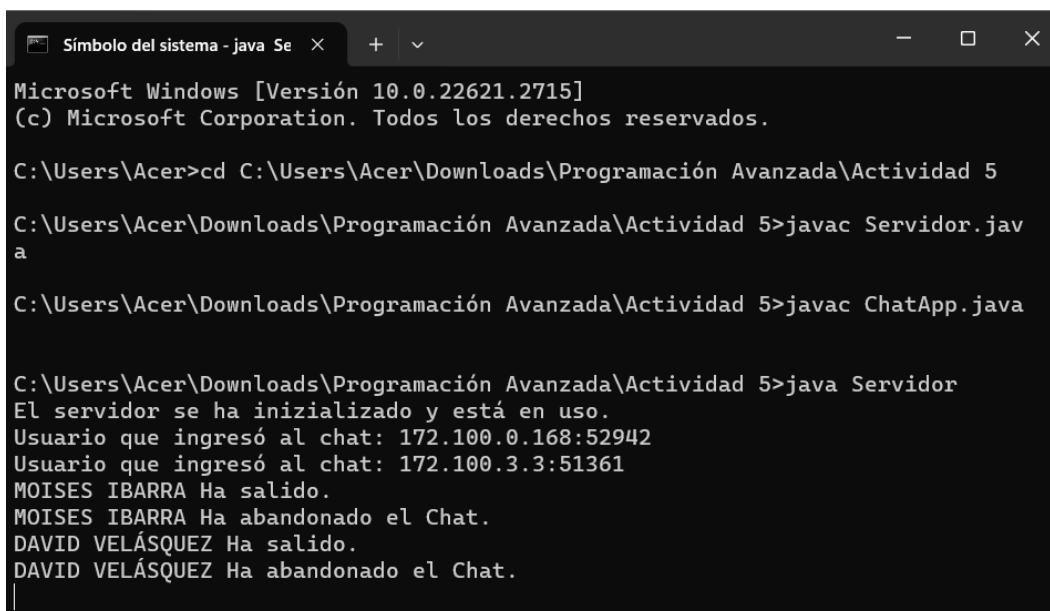
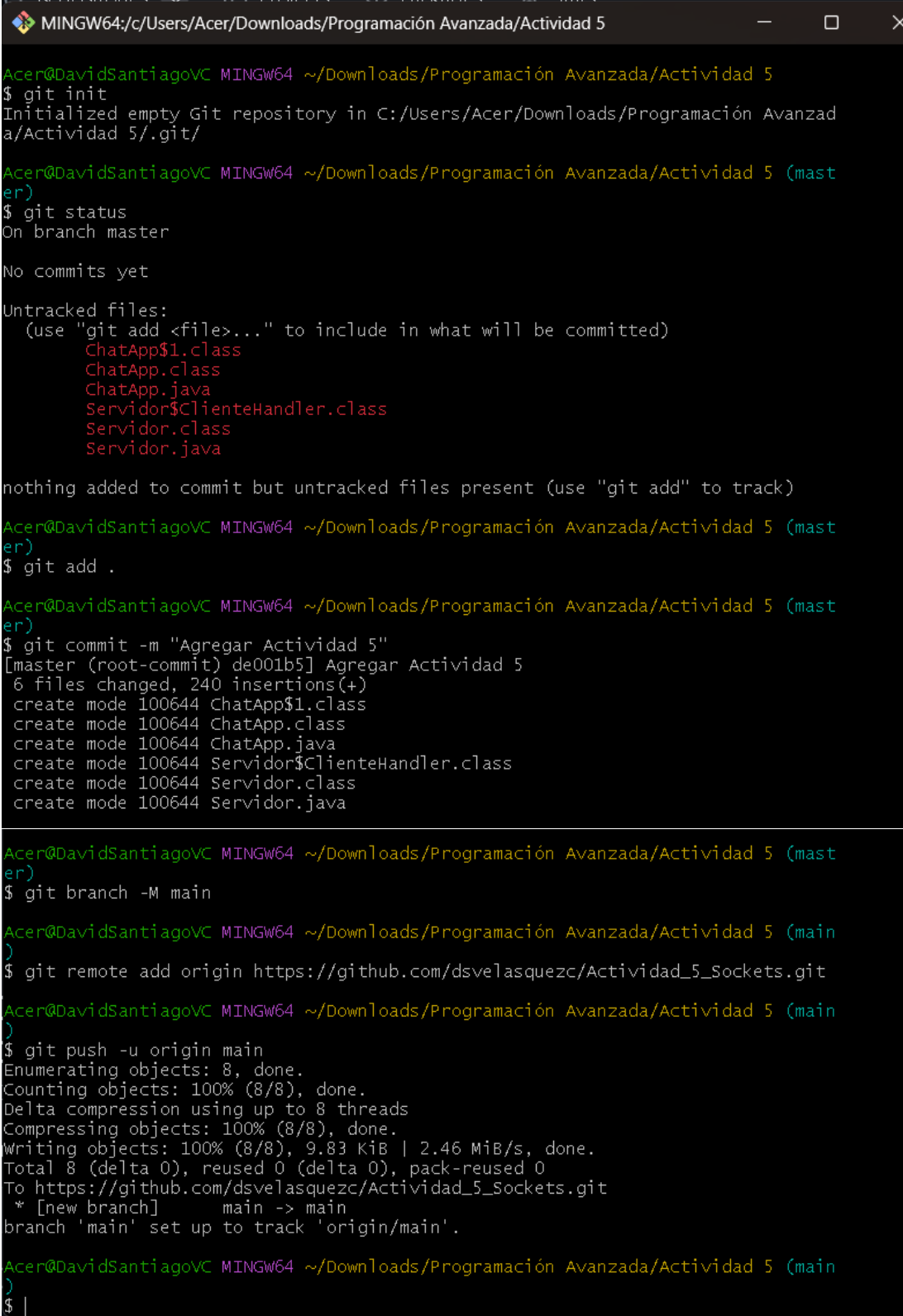


Imagen 7 - Servidor desde la consola; permite evidenciar el acceso de usuarios remotos al chat.

5. Carga de la actividad al repositorio en GITHUB

Los archivos de esta actividad han sido añadidos al repositorio personal (dsvelasquezc) y están disponibles en: https://github.com/dsvelasquezc/Actividad_5_Sockets



```
MINGW64/c/Users/Acer/Downloads/Programación Avanzada/Actividad 5
Acer@DavidSantiagoVC MINGW64 ~/Downloads/Programación Avanzada/Actividad 5
$ git init
Initialized empty Git repository in C:/Users/Acer/Downloads/Programación Avanzada/Actividad 5/.git/

Acer@DavidSantiagoVC MINGW64 ~/Downloads/Programación Avanzada/Actividad 5 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ChatApp$1.class
        ChatApp.class
        ChatApp.java
        Servidor$ClienteHandler.class
        Servidor.class
        Servidor.java

nothing added to commit but untracked files present (use "git add" to track)

Acer@DavidSantiagoVC MINGW64 ~/Downloads/Programación Avanzada/Actividad 5 (master)
$ git add .

Acer@DavidSantiagoVC MINGW64 ~/Downloads/Programación Avanzada/Actividad 5 (master)
$ git commit -m "Agregar Actividad 5"
[master (root-commit) de001b5] Agregar Actividad 5
 6 files changed, 240 insertions(+)
 create mode 100644 ChatApp$1.class
 create mode 100644 ChatApp.class
 create mode 100644 ChatApp.java
 create mode 100644 Servidor$ClienteHandler.class
 create mode 100644 Servidor.class
 create mode 100644 Servidor.java

Acer@DavidSantiagoVC MINGW64 ~/Downloads/Programación Avanzada/Actividad 5 (master)
$ git branch -M main

Acer@DavidSantiagoVC MINGW64 ~/Downloads/Programación Avanzada/Actividad 5 (main)
$ git remote add origin https://github.com/dsvelasquezc/Actividad_5_Sockets.git

Acer@DavidSantiagoVC MINGW64 ~/Downloads/Programación Avanzada/Actividad 5 (main)
$ git push -u origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 9.83 KiB | 2.46 MiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/dsvelasquezc/Actividad_5_Sockets.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Acer@DavidSantiagoVC MINGW64 ~/Downloads/Programación Avanzada/Actividad 5 (main)
$ |
```

Imagen 8 - Proceso de subida de los archivos .java en el repositorio personal.

Actividad 5 – “Informe de ejecución y explicación del Chat creado con Sockets en códigos JAVA”

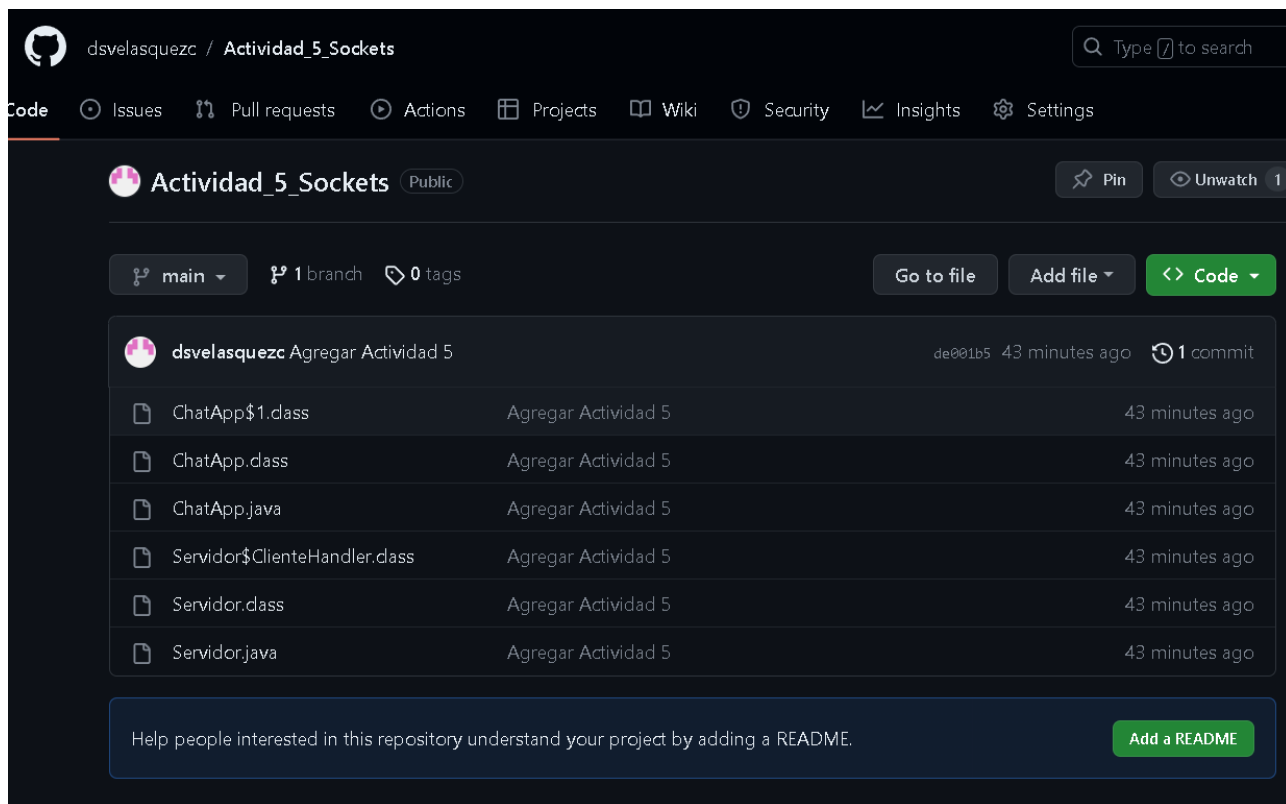


Imagen 9 - Confirmación de la creación del repositorio y la carga exitosa de los documentos