Práctica 2-1: Chat Grupal

Andrea Sofía Pais Dos Santos

January 21, 2026

1 Índice

2 Introducción

Para la elaboración de un Chat Grupal entre varios usuarios y que entre ellos se vean los mensajes vamos a tener que a parte de implementar "puentes" (sockets) entre los usuarios y el servidor, vamos a tener que transpasar la información de los mensajes entre ellos.

Primero hay que elaborar una interfaz que cada vez que ejecutemos, el servidor funcione por terminal y permita que se vaya añadiendo usuarios que mediante un nombre lo guarde y sea su usuario para que los demás usuarios vean.

3 Tecnologías utilizadas

Para esta práctica las tecnologías utilizadas son JavaFx con IntelliJ usando SceneBuilder para diseñar la interfaz del usuario.

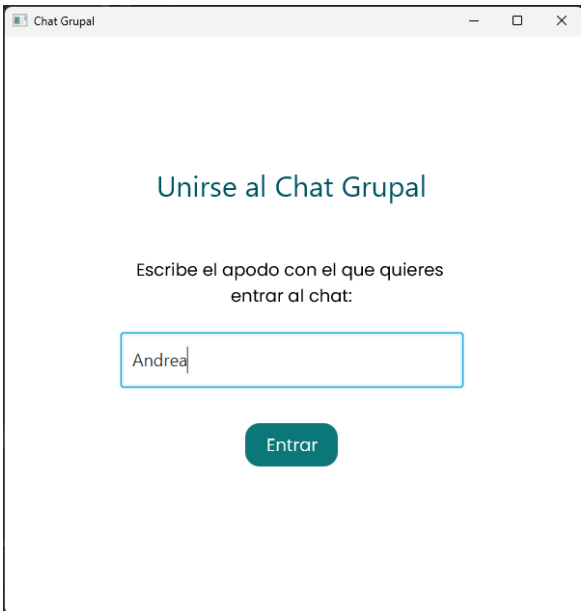
4 Diseño de la interfaz

Para una buena interfaz se va a realizar una vista para el usario sencilla, con primero un modal que aparezca al principo que te pide el nombre para usar en el chat, luego de darle al botón nos oculta el modal y vemos nuestro chat.

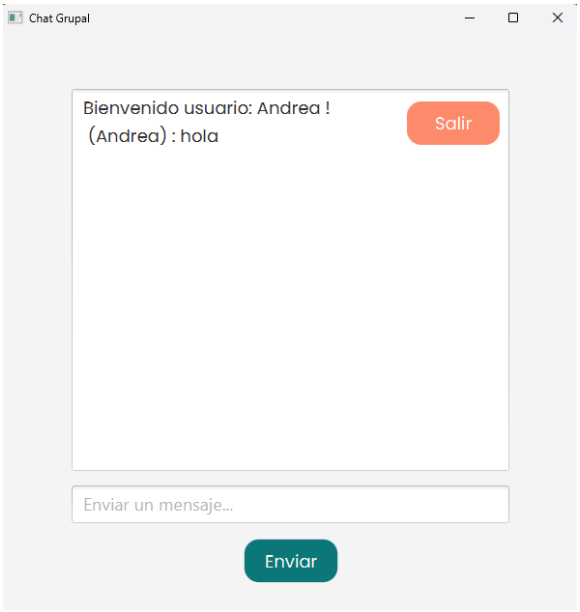
- Zona donde se ven todos los mensajes
- Espacio para enviar un mensaje nuevo
- Botones para enviar un mensaje y salir del chat desconectando el socket

EL servidor se ve por detrás, en la terminal y vemos todo el flujo de información entre usuarios.

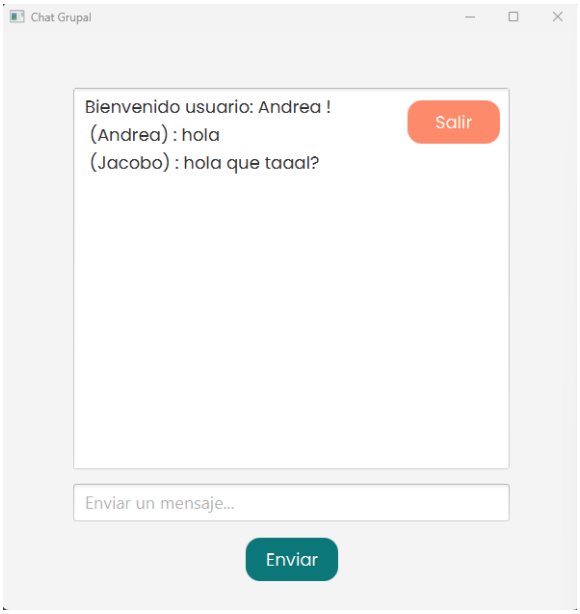
Aquí podemos ver la primera vista donde hay que seleccionar un nombre con el que nos van a ver en el Chat.



Luego ya accedemos al chat, en el que nos da la bienvenida y nos permite enviar mensajes.



Vemos que otra persona se ha conectado y recibimos los mensajes que ha enviado.



Luego vemos que tenemos un botón de salir que nos desconecta del servidor cerrando el socket.

5 Conexión con la interfaz

Para la conexión de nuestra práctica ya funcional por terminal a pasarla a una interfaz, en resumen es básicamente pasar los mensajes que se imprimen por terminal de cada usuario a uno elementos visuales.

Para la visualización del chat usaremos un TextArea y para los mensajes un TextField usando el HelloController que pasa los elementos gráficos que se van a rellenar y en las clases añadimos el contenido.

6 Clases e Interfaces

Explicación breve de cada clase e interfaz y las herramientas necesarias para llevar a cabo la aplicación de chat grupal. El sistema utiliza sockets para la comunicación y un pool de hilos para gestionar varios al mismo tiempo.

6.0.1 HelloController

- Variables FXML:
  1. TextField mensajeU1 -> Campo de texto para escribir el mensaje que quiera enviar el usuario.
  2. TextArea chat1 -> Área de texto donde se visualiza el historial de la conversación.
  3. Button botonEnviar1 -> Botón para mandar el mensaje al servidor.
  4. Pane pantallaNombre -> Panel superpuesto para solicitar el nombre de usuario antes de entrar al chat.
  5. TextField nombreUsuario -> Campo para introducir el nombre del usuario.
- Función initialize(): Se encarga de arrancar el servidor automáticamente al iniciar la aplicación y configurar la acción del botón entrar para capturar el nombre del usuario y habilitar la interfaz de chat.
- Función configurarUsuario(): Instancia la clase UsuarioInterfaz, inicia su hilo que corresponde y define las acciones de los botones enviar y desconectar.
- Función iniciarServidor(): Instancia y arranca el EchoServerMultihilo en un hilo separado para no bloquear la interfaz de usuario.

6.1 Clases en Java

El sistema se divide en la lógica del servidor (gestiona las conexiones) y la lógica del cliente (interacción con el usuario).

6.1.1 EchoServerMultihilo

- Atributos:
  1. PUERTO -> Constante que define el puerto de escucha: 8080.
  2. MAX CLIENTES -> Límite de hilos en el ExecutorService.
  3. Usuarios -> Lista estática de PrintWriter para realizar el broadcast de mensajes a todos los conectados.
- Función iniciarServidor(): Crea un ServerSocket y entra en un bucle infinito esperando conexiones. Cada vez que un cliente se conecta accept(), entrega el socket a un ManejadorUsuarioMultihilo ejecutado por el pool de hilos.

6.1.2 ManejadorUsuarioMultihilo

- Esta clase implementa Runnable y muestra la lógica del servidor para cada cliente individual.
- run():
  1. Guarda el flujo de salida del cliente en la lista global de usuarios.
  2. Lee la primera línea enviada por el cliente como su nombre de usuario.
  3. Entra en un bucle donde recibe mensajes y los reenvía (broadcast) a todos los clientes conectados.
  4. Utiliza Platform.runLater() para imprimir logs en la consola.
  5. En el bloque finally, elimina al usuario de la lista y cierra el socket al desconectarse.

6.1.3 UsuarioInterfaz

- Esta clase implementa Runnable y gestiona la conexión del lado del cliente.
- Atributos: Host, puerto, el nombre del usuario y referencias a los elementos como el TextArea y Button.
- run():
  1. Intenta establecer conexión con el servidor mediante un Socket.
  2. Envía el nombre de usuario como primer mensaje para identificarse.
  3. Mantiene un bucle activo escuchando los mensajes que llegan del servidor y los añade al areaChat usando Platform.runLater().
- Función enviarMensaje(): Envía un string a través del PrintWriter hacia el servidor.
- Función desconectar(): Cambia el estado de conexión y cierra el socket.

6.2 Hello View (FXML) y Launcher

- Vista: Define un Pane inicial para el login y un contenedor principal para el chat con TextArea y botones de envío.
- Launcher: Clase auxiliar que llama a Application launch para iniciar.

7 Casos de Usos

Caso de Uso	Descripción
Identificación de Usuario	El usuario ingresa su nombre en la pantalla inicial para identificarse ante el servidor antes de acceder al chat grupal.
Envío de Mensajes	Un usuario escribe un texto y, al pulsar enviar, el mensaje se transmite al servidor para que este lo distribuya a todos los clientes conectados.
Broadcast de Mensajes	El servidor recibe un mensaje de un cliente y lo reenvía automáticamente a todos.
Desconexión Voluntaria	El usuario pulsa el botón desconectar, lo que cierra su socket de comunicación y notifica al servidor para que lo elimine de la lista de difusión.

8 Historial de Usuarios

Identificación	Nombre	Tarea	Objetivo
HU1	Usuario	El usuario accede al servidor y se le da posibilidad de escribir el nombre o apodo que quiera.	Poder escoger mi nombre
HU3	Usuario	El usuario al acceder al chat y si hay más personas conectadas a ese servidor puede interactuar con la otra mediante mensajes.	Ver los mensajes de los otros usuarios
HU4	Gestión	El programa tiene que tener un flujo de usuarios y saber controlarlo, tanto limitarlo como tener usuarios infinitos si hiciera falta.	Que pueda administrar más de 1 persona y poner límites en el número de usuario
HU5	Usuario	Justo tras conectarse el usuarios y que el socket se abra, el usuario recibe un mensaje de bienvenida que también recibe el servidor.	Recibir un mensaje de bienvenida