



**INSTITUTO POLITÉCNICO NACIONAL.**

**ESCUELA SUPERIOR DE CÓMPUTO.**

**APLICACIONES PARA COMUNICACIONES  
EN RED.**

**PRÁCTICA 3.**

**PROFESOR: AXEL ERNESTO MORENO  
CERVANTES.**

**GRUPO 6CM1.**

**ALUMNA: SÁNCHEZ VALERIANO  
ALEXANDRA.**

**FECHA: 03/11/2025**



## INTRODUCCIÓN.

En la era de la comunicación digital, las aplicaciones de mensajería instantánea se han consolidado como herramientas indispensables. Sin embargo, más allá del simple intercambio de texto, la demanda de funcionalidades multimedia y la capacidad de gestionar interacciones complejas en tiempo real presentan desafíos técnicos significativos. Este proyecto aborda el desarrollo de un sistema de chat cliente-servidor en Java que va más allá de las funcionalidades básicas, implementando un entorno multisala dinámico y la capacidad de intercambiar mensajes de voz.

El principal objetivo es construir una aplicación robusta donde múltiples usuarios puedan conectarse, descubrir y unirse a diferentes salas de chat, ver la lista de participantes en tiempo real y comunicarse no solo a través de texto, sino también mediante clips de audio grabados al momento. Este enfoque introduce complejidades relacionadas con la gestión de estado distribuido, el uso eficiente de protocolos de red como unicast y multicast, la sincronización de la interfaz de usuario y el manejo de datos binarios (audio) a través de la red.

El sistema se basa en el protocolo UDP, utilizando un socket unicast para la comunicación directa del cliente al servidor y un socket multicast para la difusión de mensajes del servidor a todos los clientes. Esta arquitectura permite una distribución eficiente de la información, asegurando que las actualizaciones de mensajes, listas de usuarios y salas lleguen a todos los participantes de manera simultánea. El reporte detalla la arquitectura, el diseño de las clases principales y los mecanismos implementados para la transmisión y reproducción de audio.



## DESARROLLO.

### 1. SERVIDOR.JAVA - LA PARTE CENTRAL DEL CHAT.

La clase **Servidor** actúa como el backend de la aplicación, gestionando el estado de las salas, los usuarios y la distribución de mensajes. No posee interfaz gráfica; su función principal es recibir eventos de los clientes y difundir las actualizaciones correspondientes.

- **Gestión de Comunicación:** El servidor opera con dos sockets principales:
  - Un **DatagramSocket** (unicast) en el puerto **8000**, que escucha exclusivamente las peticiones de los clientes (unirse a una sala, crear una sala, enviar un mensaje, etc.).
  - Un **MulticastSocket** para enviar datos al grupo multicast en la dirección **230.0.0.0:8001**. Todos los mensajes y actualizaciones de estado (nuevos usuarios, listas de salas) se envían a través de este socket para que sean recibidos por todos los clientes suscritos.
- **Administración de Salas y Usuarios:** Utiliza una estructura de datos **Map<String, LinkedHashSet<String>>** para almacenar las salas y los usuarios dentro de cada una. Esto permite llevar un registro eficiente de qué usuarios están en qué sala. La lógica del servidor maneja la creación de salas, la unión de usuarios y su salida, ya sea de una sala específica o por desconexión total.
- **Protocolo de Aplicación:** El servidor procesa mensajes de los clientes basados en un protocolo simple con etiquetas tipo XML. Las acciones principales que maneja son:
  - **<join>**: Un usuario se une a una sala. El servidor actualiza su registro, notifica a todos los miembros de la sala sobre el nuevo integrante y reenvía la lista de usuarios actualizada de esa sala. Crucialmente, también reenvía la lista completa de salas a todos los clientes para mantener la sincronización.
  - **<leave> y <salir>**: Gestiona la salida de un usuario de una sala o del sistema por completo, notificando a los demás y actualizando las listas.
  - **<msg>, <privado>, <audio\_chunk>**: Recibe estos mensajes y, sin procesar su contenido, los retransmite inmediatamente al grupo multicast para que todos los clientes los reciban y procesen localmente.



- **<createRoom> y <listRooms>**: Administra la creación de nuevas salas y responde a las solicitudes de actualización de la lista de salas disponibles, difundiendo siempre la lista más reciente a todos.

## 2. CLIENTE.JAVA - LA INTERFAZ DE USUARIO Y EL CLIENTE DE RED.

El **Cliente** es la aplicación con la que el usuario interactúa. Es responsable de la interfaz gráfica, la comunicación con el servidor y el manejo de la lógica de grabación y reproducción de audio.

- **Interfaz Gráfica (GUI)**: Construida con Swing, la interfaz es dinámica y está organizada de la siguiente manera:
  - Un panel principal con una lista de salas disponibles, que permite a los usuarios unirse a ellas con un doble clic o crear nuevas.
  - Un **JTabbedPane** que gestiona las conversaciones. Cada sala a la que se une el usuario se abre en una nueva pestaña, permitiendo una fácil navegación entre chats.
  - Cada pestaña de sala contiene un área de chat principal, una lista de usuarios conectados a esa sala y los controles para enviar mensajes.
- **Comunicación en Red**:
  - Al iniciarse, el cliente crea un **DatagramSocket** (para enviar mensajes al servidor) y un **MulticastSocket** (para escuchar los mensajes del grupo).
  - Se une al grupo multicast para recibir todas las difusiones del servidor. Un hilo dedicado (**listenForMessages**) se encarga de recibir y procesar todos los mensajes entrantes (**<msg>**, **<listaUsrs>**, etc.) de forma asíncrona para no bloquear la interfaz gráfica.
- **Manejo de Mensajería de Voz**: Esta es la funcionalidad más compleja del cliente.
  - **Grabación**: Al pulsar el botón "Grabar", se utiliza la librería `javax.sound.sampled` para capturar audio del micrófono (`TargetDataLine`). Los bytes de audio se almacenan en un `ByteArrayOutputStream`.
  - **Fragmentación y Envío (Chunking)**: Una vez finalizada la grabación, el array de bytes de audio se divide en pequeños fragmentos (**chunks**) de 768 bytes. Cada fragmento se codifica en Base64 y se envía al servidor en un

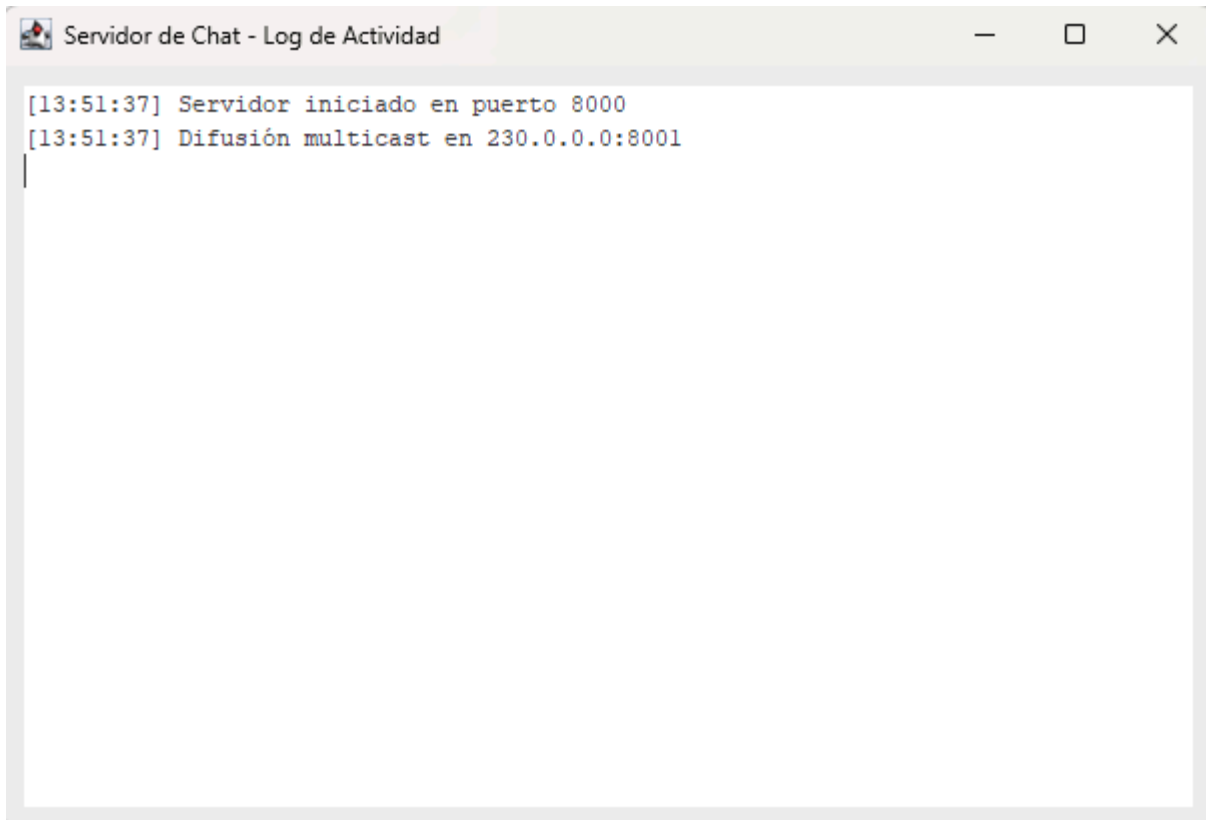
paquete **<audio\_chunk>** individual. Este paquete incluye metadatos clave: un ID único para el mensaje de voz, el número de fragmento y el total de fragmentos.

- **Reensamblaje:** El cliente mantiene un **Map** de buffers de reensamblaje (**AudioReassemblyBuffer**). Cuando recibe un fragmento de audio, lo añade al buffer correspondiente a su ID. Cuando todos los fragmentos de un mensaje de voz han llegado, el buffer los une para reconstruir el array de bytes de audio original.
- **Reproducción Automática:** Una vez que un mensaje de audio se ha reensamblado correctamente, el sistema lo reproduce automáticamente usando una **SourceDataLine** (altavoces), siempre y cuando el mensaje no haya sido enviado por el propio usuario. Esto crea una experiencia de usuario fluida y natural.
- **Clases Internas de Modelo y Renderizado:**
  - **Message:** Una clase interna que modela los diferentes tipos de mensajes (sistema, texto, audio), guardando el contenido, el autor y los datos de audio si aplica.
  - **MessageCellRenderer:** Un renderizador de celda personalizado para la lista de chat, que dibuja "burbujas" de chat. Alinea los mensajes propios a la derecha y los de otros usuarios a la izquierda, y muestra el contenido de manera adecuada, indicando si un mensaje es de voz.
  - **AudioReassemblyBuffer:** Clase crucial para la lógica de audio. Almacena temporalmente los fragmentos recibidos y los reensambla. Incluye un temporizador de expiración para descartar mensajes de audio incompletos.

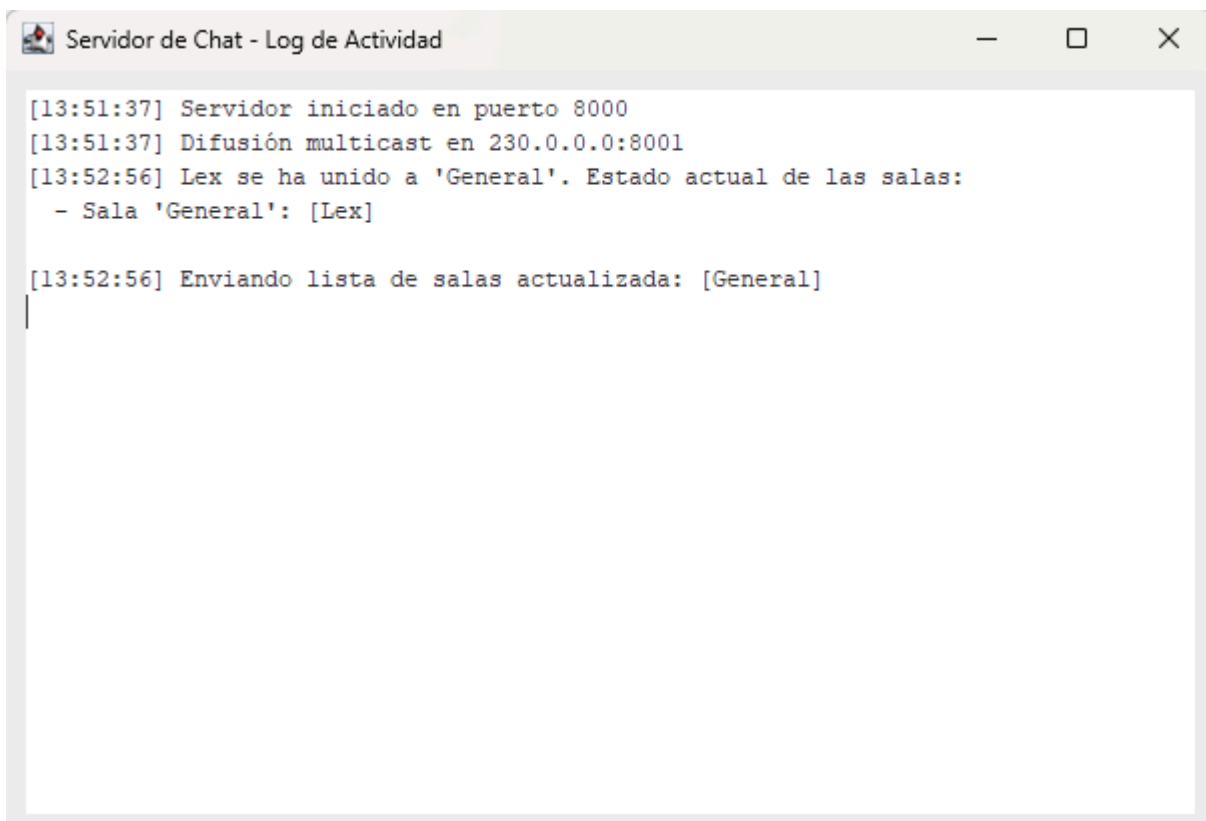


## PRUEBAS.

- **SERVIDOR.**



```
[13:51:37] Servidor iniciado en puerto 8000
[13:51:37] Difusión multicast en 230.0.0.0:8001
|
```



```
[13:51:37] Servidor iniciado en puerto 8000
[13:51:37] Difusión multicast en 230.0.0.0:8001
[13:52:56] Lex se ha unido a 'General'. Estado actual de las salas:
  - Sala 'General': [Lex]

[13:52:56] Enviando lista de salas actualizada: [General]
|
```



```
Servidor de Chat - Log de Actividad

[13:51:37] Servidor iniciado en puerto 8000
[13:51:37] Difusión multicast en 230.0.0.0:8001
[13:52:56] Lex se ha unido a 'General'. Estado actual de las salas:
  - Sala 'General': [Lex]

[13:52:56] Enviando lista de salas actualizada: [General]
[13:53:51] Matt se ha unido a 'General'. Estado actual de las salas:
  - Sala 'General': [Lex, Matt]

[13:53:51] Enviando lista de salas actualizada: [General]
[13:54:00] Cesar se ha unido a 'General'. Estado actual de las salas:
  - Sala 'General': [Lex, Matt, Cesar]

[13:54:00] Enviando lista de salas actualizada: [General]
```

```
Servidor de Chat - Log de Actividad

[13:51:37] Difusión multicast en 230.0.0.0:8001
[13:52:56] Lex se ha unido a 'General'. Estado actual de las salas:
  - Sala 'General': [Lex]

[13:52:56] Enviando lista de salas actualizada: [General]
[13:53:51] Matt se ha unido a 'General'. Estado actual de las salas:
  - Sala 'General': [Lex, Matt]

[13:53:51] Enviando lista de salas actualizada: [General]
[13:54:00] Cesar se ha unido a 'General'. Estado actual de las salas:
  - Sala 'General': [Lex, Matt, Cesar]

[13:54:00] Enviando lista de salas actualizada: [General]
[13:56:24] Sala creada: 'Sala2'
[13:56:24] Enviando lista de salas actualizada: [Sala2,General]
[13:56:26] Lex se ha unido a 'Sala2'. Estado actual de las salas:
  - Sala 'Sala2': [Lex]
  - Sala 'General': [Lex, Matt, Cesar]

[13:56:26] Enviando lista de salas actualizada: [Sala2,General]
```



```
Servidor de Chat - Log de Actividad

[13:53:51] Matt se ha unido a 'General'. Estado actual de las salas:
- Sala 'General': [Lex, Matt]

[13:53:51] Enviando lista de salas actualizada: [General]
[13:54:00] Cesar se ha unido a 'General'. Estado actual de las salas:
- Sala 'General': [Lex, Matt, Cesar]

[13:54:00] Enviando lista de salas actualizada: [General]
[13:56:24] Sala creada: 'Sala2'
[13:56:24] Enviando lista de salas actualizada: [Sala2,General]
[13:56:26] Lex se ha unido a 'Sala2'. Estado actual de las salas:
- Sala 'Sala2': [Lex]
- Sala 'General': [Lex, Matt, Cesar]

[13:56:26] Enviando lista de salas actualizada: [Sala2,General]
[14:00:04] Cesar se ha desconectado de 'General'.
[14:00:04] Estado de salas tras desconexión de Cesar:
- Sala 'Sala2': [Lex]
- Sala 'General': [Lex, Matt]
```

```
Servidor de Chat - Log de Actividad

[13:54:00] Cesar se ha unido a 'General'. Estado actual de las salas:
- Sala 'General': [Lex, Matt, Cesar]

[13:54:00] Enviando lista de salas actualizada: [General]
[13:56:24] Sala creada: 'Sala2'
[13:56:24] Enviando lista de salas actualizada: [Sala2,General]
[13:56:26] Lex se ha unido a 'Sala2'. Estado actual de las salas:
- Sala 'Sala2': [Lex]
- Sala 'General': [Lex, Matt, Cesar]

[13:56:26] Enviando lista de salas actualizada: [Sala2,General]
[14:00:04] Cesar se ha desconectado de 'General'.
[14:00:04] Estado de salas tras desconexión de Cesar:
- Sala 'Sala2': [Lex]
- Sala 'General': [Lex, Matt]

[14:00:55] Lex ha salido de 'Sala2'. Estado actual de las salas:
- Sala 'Sala2': (vacía)
- Sala 'General': [Lex, Matt]
```





```
Servidor de Chat - Log de Actividad

[13:56:24] Enviando lista de salas actualizada: [Sala2,General]
[13:56:26] Lex se ha unido a 'Sala2'. Estado actual de las salas:
- Sala 'Sala2': [Lex]
- Sala 'General': [Lex, Matt, Cesar]

[13:56:26] Enviando lista de salas actualizada: [Sala2,General]
[14:00:04] Cesar se ha desconectado de 'General'.
[14:00:04] Estado de salas tras desconexión de Cesar:
- Sala 'Sala2': [Lex]
- Sala 'General': [Lex, Matt]

[14:00:55] Lex ha salido de 'Sala2'. Estado actual de las salas:
- Sala 'Sala2': (vacía)
- Sala 'General': [Lex, Matt]

[14:01:52] Matt se ha desconectado de 'General'.
[14:01:52] Estado de salas tras desconexión de Matt:
- Sala 'Sala2': (vacía)
- Sala 'General': [Lex]
```

```
Servidor de Chat - Log de Actividad

[14:00:55] Lex ha salido de 'Sala2'. Estado actual de las salas:
- Sala 'Sala2': (vacía)
- Sala 'General': [Lex, Matt]

[14:01:52] Matt se ha desconectado de 'General'.
[14:01:52] Estado de salas tras desconexión de Matt:
- Sala 'Sala2': (vacía)
- Sala 'General': [Lex]

[14:02:47] Lex se ha unido a 'Sala2'. Estado actual de las salas:
- Sala 'Sala2': [Lex]
- Sala 'General': [Lex]

[14:02:47] Enviando lista de salas actualizada: [Sala2,General]
[14:02:52] Lex se ha desconectado de 'Sala2'.
[14:02:52] Lex se ha desconectado de 'General'.
[14:02:52] Estado de salas tras desconexión de Lex:
- Sala 'Sala2': (vacía)
- Sala 'General': (vacía)
```



```
Servidor de Chat - Log de Actividad

[14:01:52] Estado de salas tras desconexión de Matt:
- Sala 'Sala2': (vacía)
- Sala 'General': [Lex]

[14:02:47] Lex se ha unido a 'Sala2'. Estado actual de las salas:
- Sala 'Sala2': [Lex]
- Sala 'General': [Lex]

[14:02:47] Enviando lista de salas actualizada: [Sala2,General]
[14:02:52] Lex se ha desconectado de 'Sala2'.
[14:02:52] Lex se ha desconectado de 'General'.
[14:02:52] Estado de salas tras desconexión de Lex:
- Sala 'Sala2': (vacía)
- Sala 'General': (vacía)

[14:03:17] Marco se ha unido a 'General'. Estado actual de las salas:
- Sala 'Sala2': (vacía)
- Sala 'General': [Marco]

[14:03:17] Enviando lista de salas actualizada: [Sala2,General]
```



- **CLIENTE.**

Bienvenido al Chat

✕

Ingresa tu nombre de usuario:

Lex

Aceptar

Cancelar

Chat - Lex

— □ ✕

Salas Disponibles

General

General

Chat

Usuarios

Lex

🔍

+

📎 Grabar

😊

→ Privado

Enviar



Chat - Lex

Salas Disponibles

General

General

Chat

*Lex se ha unido al chat.*

*Matt se ha unido al chat.*

*Cesar se ha unido al chat.*

Usuarios

Lex  
Matt  
Cesar

Grabar

→ Privado

Enviar

Chat - Matt

Salas Disponibles

General

General

Chat

*Matt se ha unido al chat.*

*Cesar se ha unido al chat.*

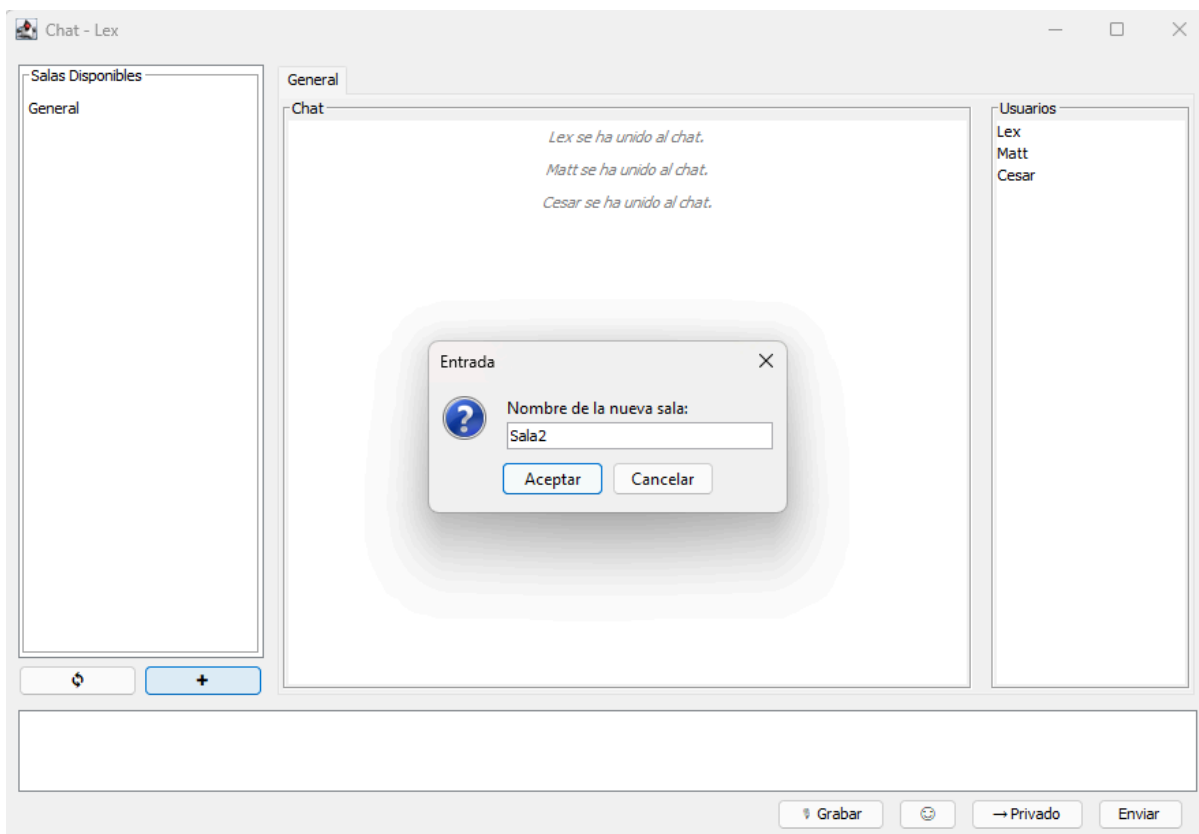
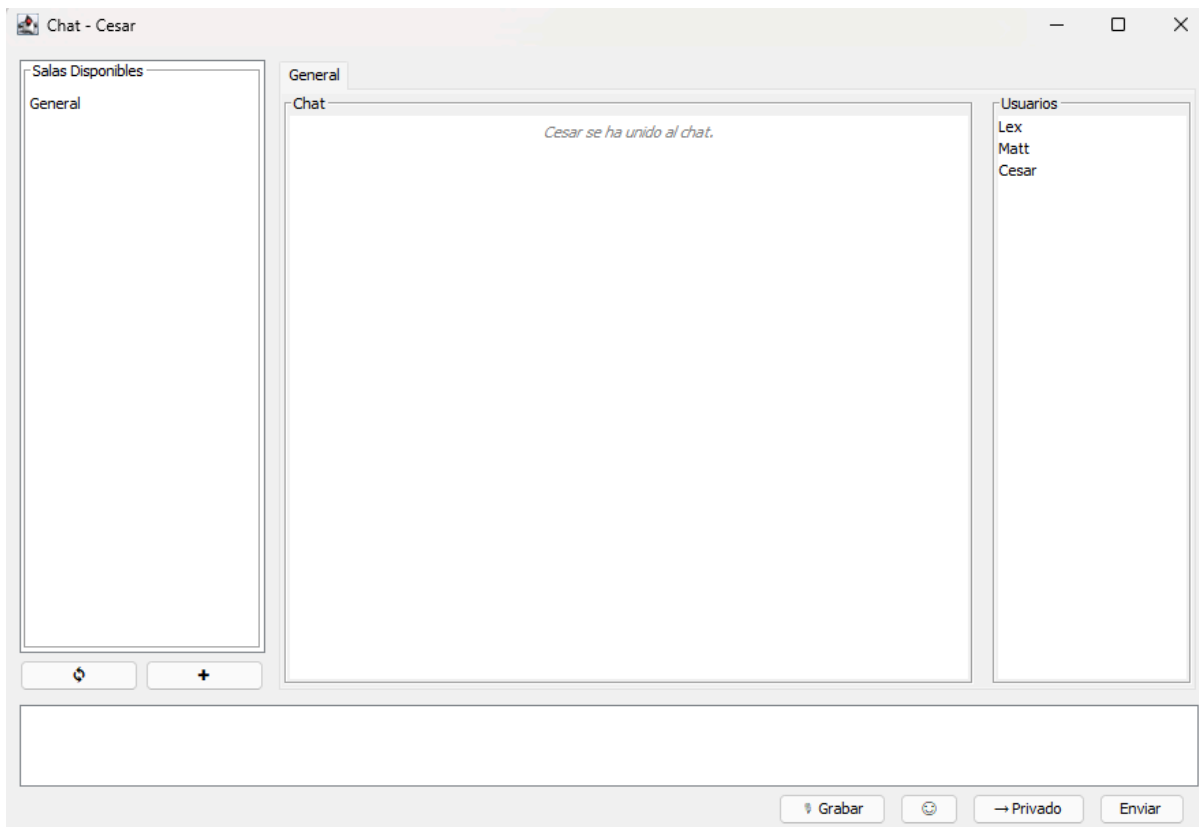
Usuarios

Lex  
Matt  
Cesar

Grabar

→ Privado

Enviar





Chat - Lex

Salas Disponibles

Sala2

General

General

Sala2

Chat

Lex se ha unido al chat.

Matt se ha unido al chat.

Cesar se ha unido al chat.

Lex

Hola 😊

Usuarios

Lex

Matt

Cesar

Hola Matt

Grabar

→ Privado

Enviar

Chat - Matt

Salas Disponibles

Sala2

General

General

Sala2

Chat

Matt se ha unido al chat.

Cesar se ha unido al chat.

Lex

Hola 😊

Lex

Hola Matt

Usuarios

Lex

Matt

Cesar

Grabar

→ Privado

Enviar



Chat - Cesar

Salas Disponibles

Sala2

General

General

Chat

Usuarios

Lex

Matt

Cesar

Hola 😊

Cesar se ha unido al chat.

Grabar

Enviar

Privado

Chat - Lex

Salas Disponibles

Sala2

General

General

Sala2

Usuarios

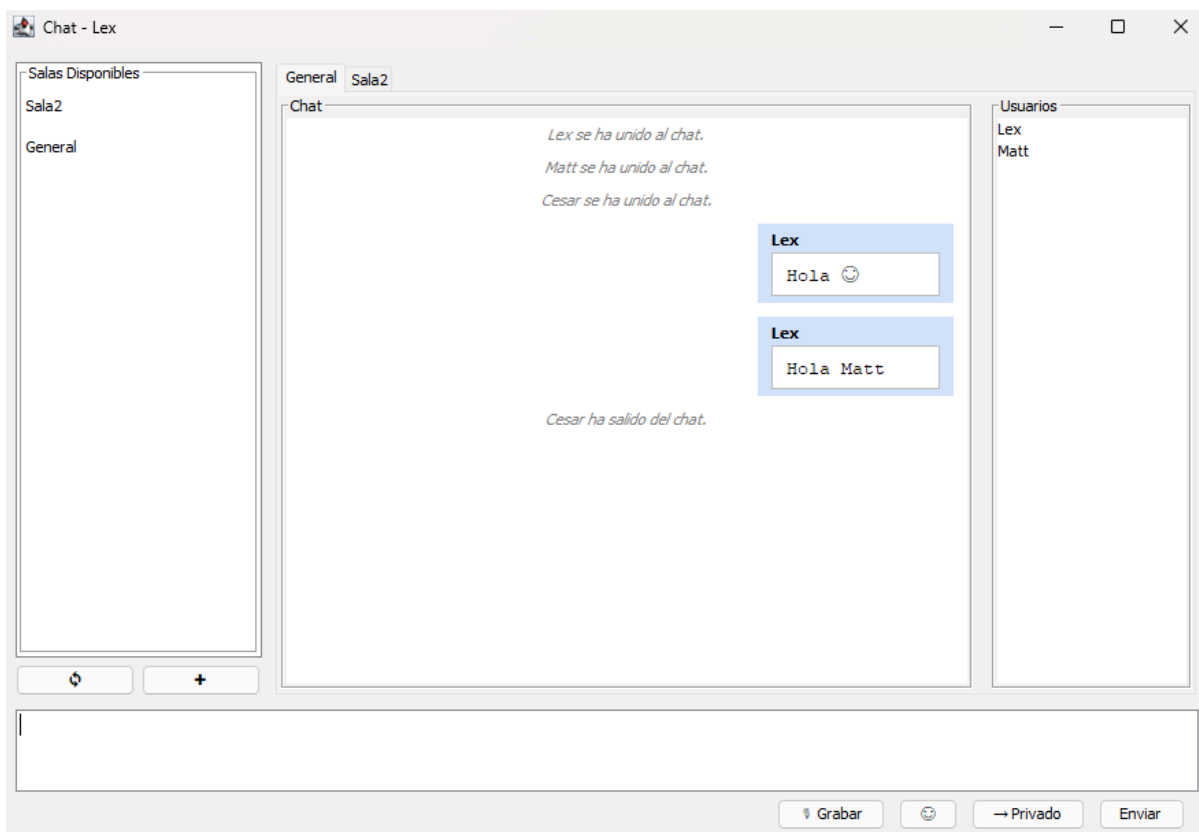
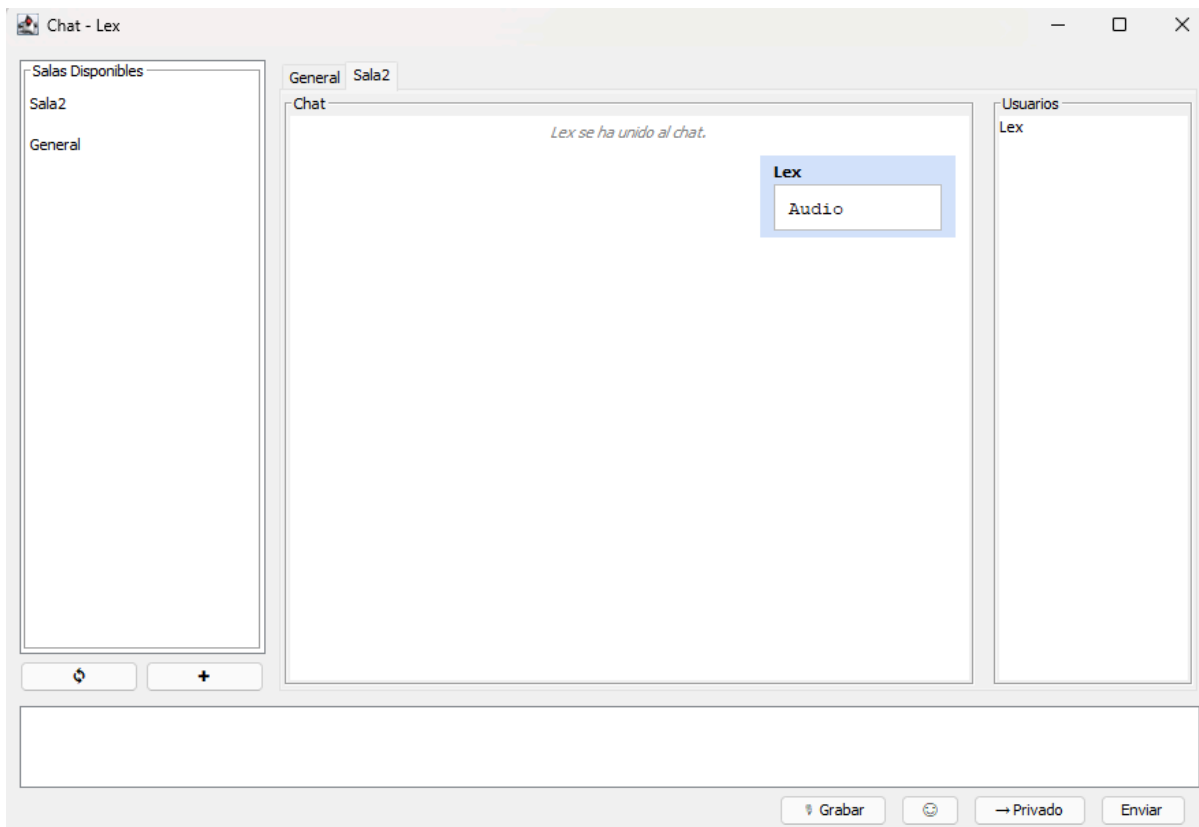
Lex

Lex se ha unido al chat.

Detener

Enviar

Privado







Chat - Matt

Salas Disponibles

Sala2

General

General

Chat

Usuarios

Lex

Matt

Lex

Hola 😊

Lex

Hola Matt

Matt se ha unido al chat.

Cesar se ha unido al chat.

Cesar ha salido del chat.

Grabar

😊

→ Privado

Enviar

Chat - Lex

Salas Disponibles

Sala2

General

General

Chat

Usuarios

Lex

Lex

Hola 😊

Lex

Hola Matt

Lex se ha unido al chat.

Matt se ha unido al chat.

Cesar se ha unido al chat.

Cesar ha salido del chat.

Matt ha salido del chat.

Grabar

😊

→ Privado

Enviar



Chat - Marco

Salas Disponibles

Sala2

General

General

Chat

Marco se ha unido al chat.

Usuarios

Marco

Grabar

→ Privado

Enviar



## CONCLUSIONES.

El desarrollo de esta práctica ha sido una experiencia sumamente enriquecedora que permitió aplicar conceptos teóricos de redes y concurrencia en un escenario práctico y complejo. A diferencia de una comunicación simple punto a punto, la gestión de un entorno multisala introdujo el reto de mantener un estado consistente a través de múltiples clientes distribuidos. La elección de una arquitectura con un servidor central que difunde cambios vía multicast demostró ser altamente efectiva para este propósito.

Uno de los aprendizajes más significativos fue el manejo de datos multimedia en tiempo real. La transmisión de audio no es tan simple como enviar texto; requiere un protocolo más sofisticado. La implementación de la fragmentación (chunking) y el reensamblaje de los datos de audio fue fundamental para superar las limitaciones de tamaño de los paquetes UDP y asegurar una transmisión fiable. El diseño de la clase **AudioReassemblyBuffer** fue clave para gestionar esta lógica de forma ordenada.

Asimismo, el proyecto reforzó la importancia de la concurrencia en aplicaciones con interfaz gráfica. El uso de un **ExecutorService** para manejar todas las operaciones de red y de audio en segundo plano fue crucial para mantener la GUI receptiva y fluida en todo momento. Acciones como escuchar mensajes, grabar audio o enviar paquetes se ejecutaron sin congelar la experiencia del usuario, un principio fundamental en el desarrollo de software de calidad.



## REFERENCIAS BIBLIOGRÁFICAS.

- Deitel, P., & Deitel, H. (2018). *Java: How to Program, Early Objects*. Pearson Education.
- Oracle. (s. f.). *Java SE Documentation*. Obtenido de <https://docs.oracle.com/javase/8/docs/api/index.htm>
- Kurose, J. F., & Ross, K. W. (2017). *Computer Networking: A Top-Down Approach*. Pearson.
- Oracle. (s. f.). *Java Sound API Tutorial*. Obtenido de <https://docs.oracle.com/javase/tutorial/sound/>