CHAT UDP

ALGORITMOS PRINCIPALES

CHAT SERVIDOR:

El código del servidor se encarga de gestionar las conexiones entrantes de los clientes y distribuir los mensajes entre ellos. Aquí hay una breve descripción de sus componentes clave:

- **Método `run()`**:
    - Este método se ejecuta cuando se inicia el servidor como un hilo.
    - Espera la llegada de paquetes UDP, los procesa y maneja según su contenido.
- **Gestión de Conexiones**:
    - Cuando recibe un paquete que comienza con "/nickname", interpreta el contenido como un intento de conexión de un cliente.
    - Verifica si el nickname propuesto está disponible y, si lo está, registra al cliente con su dirección IP y puerto asociados.
    - Envía mensajes de bienvenida y notificaciones de conexión a los otros clientes.
- **Broadcast de Mensajes**:
    - Cuando recibe un mensaje normal (no un intento de conexión), lo reenvía a todos los clientes conectados.
    - Utiliza un bucle para enviar el mensaje a cada cliente individualmente.
- **Almacenamiento del Historial de Mensajes**:
    - Guarda una cola de mensajes enviados para que los nuevos clientes puedan recibir mensajes anteriores al unirse.
    - Envía el historial de mensajes al cliente recién conectado.

CHAT CLIENTE:

El código del cliente se encarga de enviar y recibir mensajes del servidor, además de gestionar la interfaz gráfica para el usuario. Aquí hay una explicación rápida de su funcionamiento:

- **Método `initializeNetworking()`**:
    - Configura el socket UDP para enviar y recibir mensajes.
    - Envía el nickname del usuario al servidor para registrarse.
- **Envío de Mensajes**:
    - Cuando el usuario escribe un mensaje y lo envía, se convierte en un paquete UDP y se envía al servidor.
- **Recepción de Mensajes**:

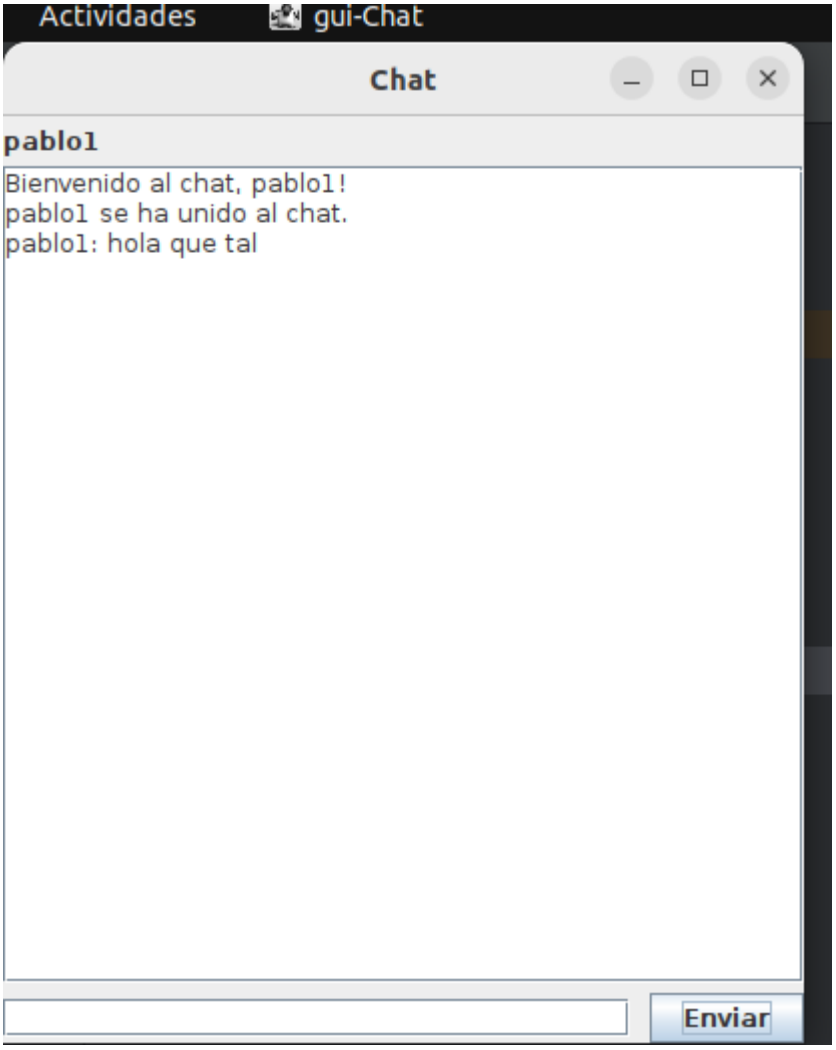
    - El cliente permanece en un bucle infinito esperando la llegada de paquetes UDP.
    - Cuando llega un paquete, se procesa y muestra el mensaje en la interfaz gráfica del usuario.
- **Interfaz Gráfica**:
    - Proporciona una interfaz amigable donde el usuario puede escribir mensajes y ver el historial de mensajes.
    - Muestra mensajes de error si el nickname elegido ya está en uso.

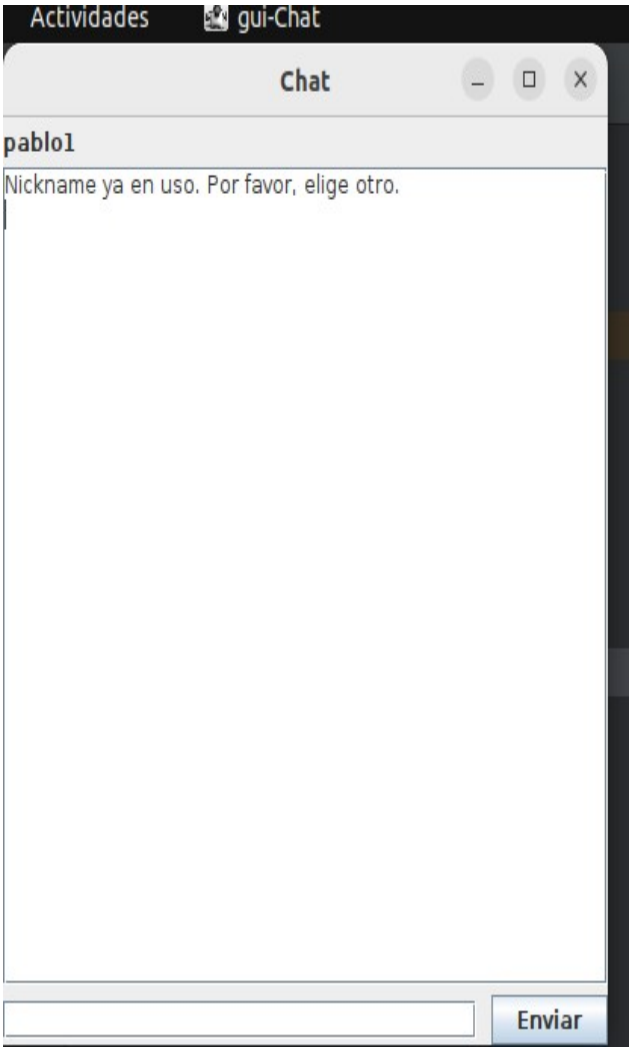
FUNCIONALIDADES

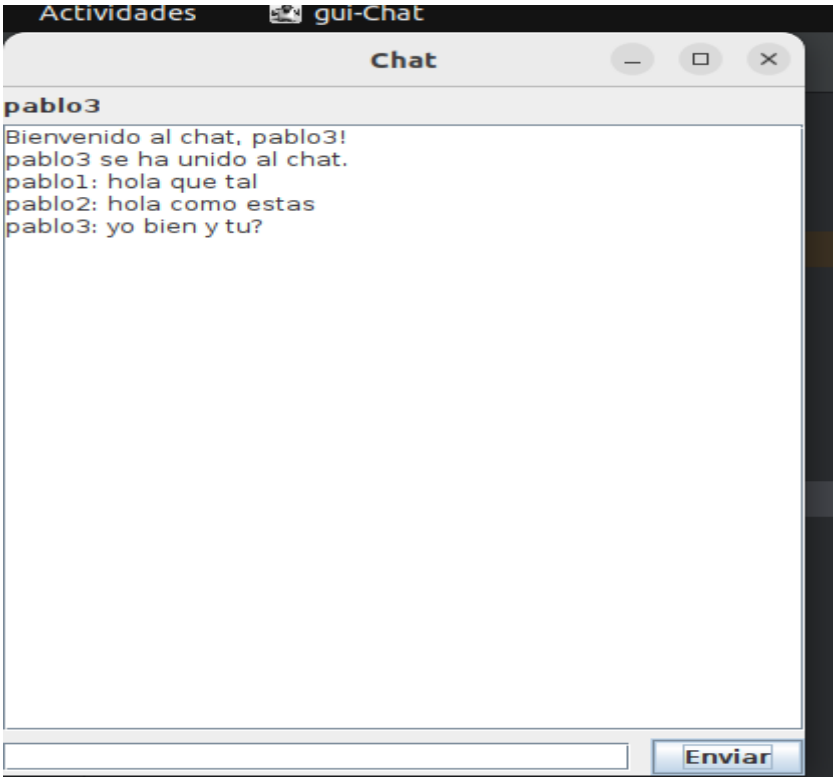
La funcionalidad del proyecto de chat UDP es facilitar la comunicación en tiempo real entre múltiples usuarios a través de una red, utilizando el protocolo de datagramas UDP. Aquí hay una descripción de sus funcionalidades principales:

1. **Conexión Simultánea de Múltiples Usuarios**:
    - El servidor puede manejar conexiones de varios clientes de forma simultánea.
    - Los usuarios pueden enviar y recibir mensajes al mismo tiempo, lo que permite una comunicación fluida entre todos los participantes.
2. **Gestión de Nicknames Únicos**:
    - El servidor garantiza que cada usuario tenga un nickname único.
    - Al conectarse, los usuarios eligen un nickname que será utilizado para identificarlos en el chat.
3. **Envío y Recepción de Mensajes**:
    - Los usuarios pueden enviar mensajes de texto a todos los demás participantes del chat.
    - Los mensajes enviados por un usuario son recibidos instantáneamente por todos los demás usuarios conectados.
4. **Visualización de Mensajes Anteriores**:
    - Opcionalmente, los usuarios pueden ver mensajes enviados antes de unirse al chat.
    - El servidor mantiene un historial de mensajes que puede ser enviado a los nuevos usuarios cuando se conectan.
5. **Interfaz Gráfica Intuitiva**:
    - El cliente proporciona una interfaz gráfica amigable que facilita la entrada de mensajes y la visualización del chat.
    - Los usuarios pueden escribir mensajes en un cuadro de texto y ver el historial de mensajes en una ventana dedicada.

Pruebas


Actividades
gui-Chat
Chat
pablo1
Bienvenido al chat, pablo1!
pablo1 se ha unido al chat.
pablo1: hola que tal
Enviar


Actividades
gui-Chat
Chat
pablo1
Nickname ya en uso. Por favor, elige otro.
Enviar


Actividades
gui-Chat
Chat
pablo3
Bienvenido al chat, pablo3!
pablo3 se ha unido al chat.
pablo1: hola que tal
pablo2: hola como estas
pablo3: yo bien y tu?
Enviar

CHAT TCP

ALGORITMOS PRINCIPALES

CLASE CHAT SERVIDOR:

El código del servidor se encarga de gestionar las conexiones entrantes de los clientes y distribuir los mensajes entre ellos. Aquí hay una breve descripción de sus componentes clave:

1. Método run():
    - Este método se ejecuta cuando se inicia el servidor y se encarga de aceptar conexiones de clientes.
    - Utiliza un ServerSocket para esperar conexiones entrantes en un puerto específico.
    - Cuando un cliente se conecta, se crea un nuevo hilo (Handler) para manejar esa conexión.
2. Método exit():
    - Este método se llama cuando el servidor necesita cerrarse.
    - Establece una bandera para indicar que el servidor debe salir del bucle principal.
    - Cierra el ServerSocket para dejar de aceptar nuevas conexiones.
    - Cierra todas las conexiones existentes con los clientes.

CLASE CHAT CLIENTE:

1. Constructor Chat(String nickname):
    - Este constructor inicializa la interfaz de usuario del chat con el apodo del usuario proporcionado.
    - Configura un ActionListener para el botón de enviar mensajes (sendButton). Cuando se hace clic en este botón, se recupera el texto del campo de entrada de mensajes (messageTextField). Si el mensaje no está vacío, se envía al servidor a través de un PrintWriter asociado al socket de cliente (out). Si el mensaje está vacío, se muestra un mensaje de advertencia al usuario.
    - Crea un socket para conectar el cliente al servidor. También envía el apodo del usuario al servidor a través del socket.
    - Inicia un hilo (Thread) para leer continuamente los mensajes del servidor.
2. Método addMessages():
    - Este método se ejecuta en un hilo separado y se encarga de leer continuamente los mensajes del servidor.

- Utiliza un BufferedReader para leer mensajes del servidor a través del socket de cliente (clientSocket.getInputStream()).
- Los mensajes recibidos se agregan al cuadro de chat (chatBox) en la interfaz de usuario del cliente.

3. Método main(String[] args):
    - Este método es la entrada principal del programa del cliente.
    - Solicita al usuario que ingrese un apodo.
    - Crea una instancia de la clase Chat con el apodo proporcionado.
    - Configura y muestra la interfaz de usuario del chat.

CLASE HANDLER (Manejador de Clientes en el Servidor):

1. Constructor Handler(Socket client):
    - Este constructor crea un nuevo manejador para un cliente dado.
    - Guarda el socket asociado con el cliente.
2. Método run():
    - Este método se ejecuta cuando se inicia el hilo para manejar un cliente.
    - Lee continuamente los mensajes del cliente a través del socket utilizando un BufferedReader.
    - Cuando recibe un mensaje, lo retransmite a todos los clientes conectados, incluyendo el apodo del remitente.
    - Si el mensaje indica que el cliente se desconecta (/quit), envía un mensaje de desconexión a todos los clientes y cierra la conexión con el cliente.
3. Método broadcastMessage(String message):
    - Este método envía un mensaje a todos los clientes conectados.
    - Itera a través de todos los manejadores de clientes y llama al método sendMessage() de cada uno para enviar el mensaje.
4. Método sendMessage(String message):
    - Este método envía un mensaje al cliente asociado utilizando un PrintWriter asociado con el socket del cliente.
5. Método exit():
    - Este método se llama cuando se necesita cerrar la conexión con el cliente.
    - Cierra los flujos de entrada y salida asociados con el cliente.
    - Cierra el socket asociado con el cliente.

FUNCIONALIDADES

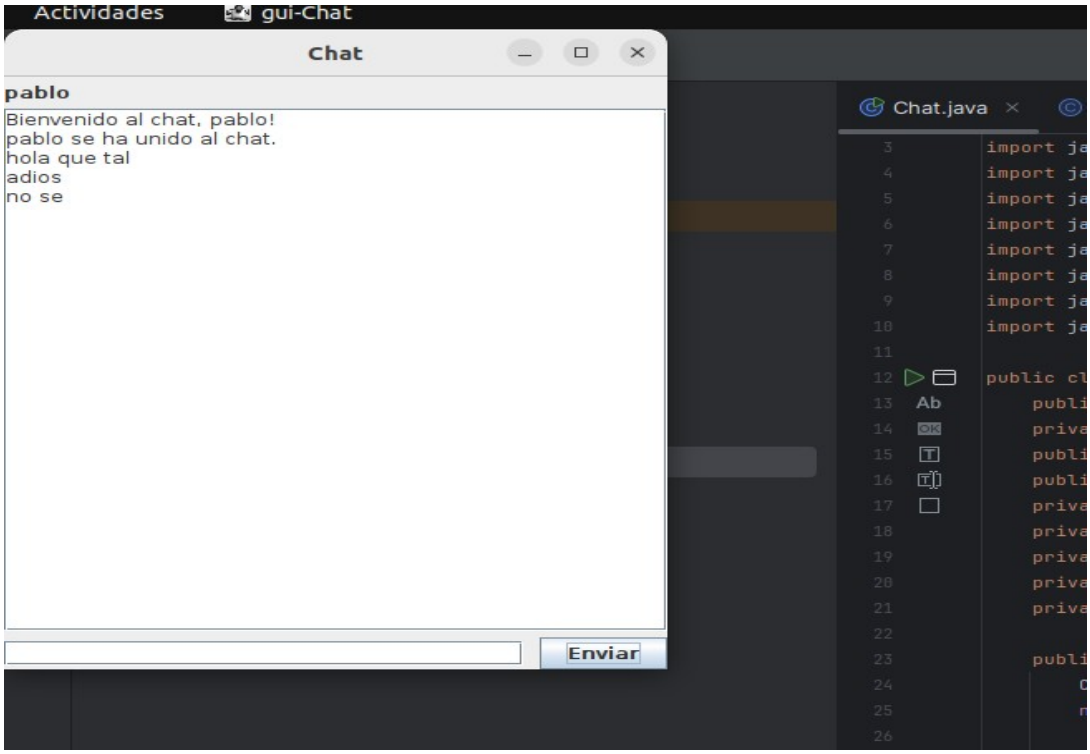
Este código consiste en un sistema básico de chat que consta de dos partes: el cliente (GUI) y el servidor.

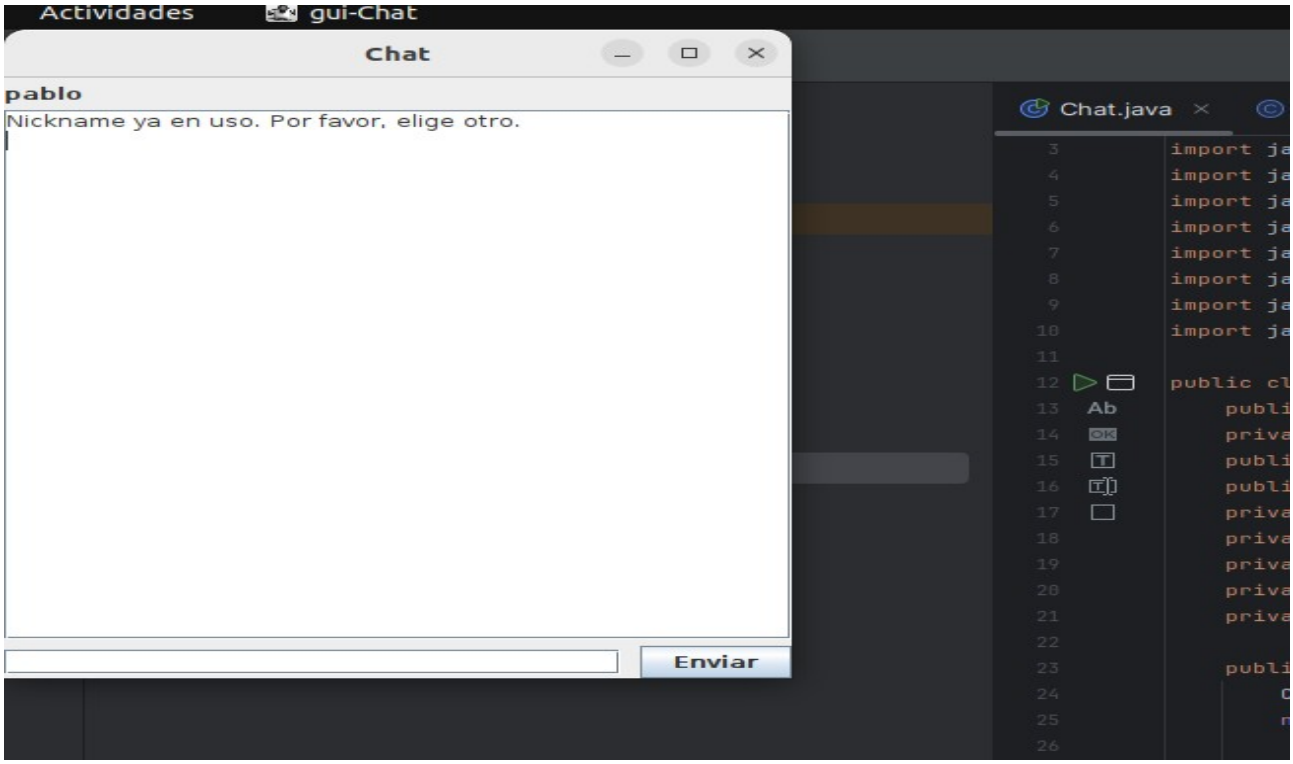
En el lado del cliente (GUI), se proporciona una interfaz de usuario que permite a un usuario ingresar un apodo, enviar mensajes y ver los mensajes recibidos. Utiliza sockets para comunicarse con el servidor. Cuando se envía un mensaje, se envía al servidor a través del socket, y cuando se recibe un mensaje del servidor, se muestra en la interfaz de usuario.

En el lado del servidor, se crea un servidor socket que acepta conexiones de clientes. Cuando un cliente se conecta, se crea un nuevo hilo (Handler) para manejar esa conexión. El servidor recibe el apodo del cliente, lo muestra a todos los clientes conectados y retransmite los mensajes enviados por un cliente a todos los demás clientes conectados.

Ambas partes del código están diseñadas para funcionar en conjunto para proporcionar una experiencia de chat básica entre múltiples clientes y un servidor central.

Pruebas





PROBLEMAS

no he conseguido que al crear el segundo usuario con nombre diferente se vean los mensajes solo puedo con el primero aun asi si repito el usuario si sale bien