Manual Técnico

Chat cliente-servidor

PROGRAMACIÓN

DE SERVICIOS

Y PROCESOS

IES ARMANDO COTARELO VALLEDOR

2º DAM

GRUPO CHAT-PTG

MANUEL ABALO RIETZ | ADRIAN CES LOPEZ | PABLO DOPAZO SUAREZ

Indice

Introducción

Este manual técnico describe el sistema de chat cliente-servidor desarrollado en Java. El sistema está compuesto por varias clases clave que trabajan en conjunto para proporcionar una comunicación eficiente y confiable entre los usuarios. Utiliza sockets para la transferencia de datos, una arquitectura basada en hilos para manejar múltiples conexiones y una interfaz gráfica que facilita la interacción del usuario con el sistema.

El sistema se organiza en los siguientes componentes principales:

1. **Servidor:** Maneja las conexiones de los clientes, coordina la comunicación entre ellos y mantiene un registro de los usuarios conectados.
2. **Cliente:** Representa a un usuario individual en el sistema, permitiendo enviar y recibir mensajes a través del servidor.
3. **HiloCliente:** Implementa un hilo independiente para cada cliente conectado al servidor, gestionando la comunicación bidireccional.
4. **VentanaS:** Proporciona una interfaz gráfica para el servidor, permitiendo visualizar el estado y los eventos del sistema.
5. **VentanaC:** Proporciona una interfaz gráfica para los clientes, permitiendo enviar y recibir mensajes de manera intuitiva.

Cada componente está diseñado para cumplir un rol específico en el sistema, utilizando principios de modularidad y reusabilidad. Este manual detalla los atributos, métodos y ejemplos de uso de cada clase, proporcionando una guía completa para la comprensión e implementación del sistema.

Package chat_servidor.cliente

Cliente.java

La clase Cliente representa un cliente en un sistema de chat desarrollado en Java. Su objetivo principal es gestionar la comunicación entre un cliente y el servidor a través de sockets. Además, provee funcionalidades para enviar y recibir mensajes, así como para gestionar la sesión del cliente.

La clase Cliente extiende Thread para manejar la comunicación con el servidor en un hilo separado.

Atributos

private Socket socket
Socket utilizado para la comunicación con el servidor.

private ObjectOutputStream objectOutputStream
Flujo de salida para enviar datos al servidor.

private ObjectInputStream objectInputStream
Flujo de entrada para recibir datos del servidor.

private final VentanaC ventanaCliente
Referencia a la ventana gráfica del cliente.

private String idCliente
Identificador único del cliente.

private boolean listening
Indicador de si el cliente debe seguir escuchando mensajes del servidor.

private final String host
Dirección IP del servidor.

private final int puerto
Puerto del servidor.

Métodos

public Cliente(VentanaC ventana, String host, Integer puerto, String nombre)
Constructor. Inicializa un nuevo cliente, configurando la conexión con el servidor y asociando una ventana gráfica.

public void run()
Establece la conexión con el servidor y gestiona la comunicación principal.

public void desconectar()
Cierra los flujos de comunicación y el socket para desconectar al cliente.

public void mensajear(String cliente_receptor, String mensaje)
Envía un mensaje a otro cliente a través del servidor

private void escuchar()
Escucha mensajes provenientes del servidor y los procesa.

public void ejecutar(LinkedList<String> lista)
Ejecuta acciones según el tipo de mensaje recibido.

private void conexionCliente(String identificador)
Solicita la conexión del cliente al servidor.

void desconexionCliente()
Solicita la desconexión del cliente al servidor.

String getIdCliente()
Obtiene el identificador único del cliente.

VentanaC.java

La clase VentanaC constituye la interfaz gráfica de usuario (GUI) del cliente en un sistema de chat desarrollado en Java. Esta clase proporciona la funcionalidad necesaria para interactuar con el sistema, permitiendo el envío de mensajes y la gestión de contactos.

La clase VentanaC extiende JFrame y constituye la ventana principal de la GUI

Atributos

private final String DEFAULT_PORT = "10101"
Puerto por defecto para conectarse al servidor.

private final String DEFAULT_IP = "127.0.0.1"
Dirección IP por defecto del servidor.

private final Cliente cliente
Cliente que maneja la comunicación con el servidor.

private javax.swing.JButton btnEnviar
Botón para enviar mensajes.

private javax.swing.JComboBox contactos
Lista desplegable con los contactos disponibles.

private javax.swing.JLabel jLabel1
Etiqueta de texto para indicar destinatarios.

private javax.swing.JScrollPane jScrollPane1
Panel de desplazamiento para el área de texto del historial de mensajes.

private javax.swing.JTextArea historial
Área de texto para mostrar el historial de mensajes.

private javax.swing.JTextField txtMensaje
Campo de texto para ingresar el mensaje a enviar.

Métodos

public VentanaC()
Constructor. Inicializa la interfaz gráfica y configura la conexión del cliente.

private void initVentanaC()
Inicializa los componentes de la ventana.

private void btnEnviarActionPerformed(java.awt.event.ActionEvent evt)
Acción del botón Enviar. Envía un mensaje al destinatario seleccionado.

private void formWindowClosing(java.awt.event.WindowEvent evt)
Cierra la sesión del cliente al cerrar la ventana.

private void formWindowClosed(java.awt.event.WindowEvent evt)
Acción al cerrar la ventana.

private String[] getIPPuerto()
Muestra un diálogo para configurar la IP, puerto y nombre de usuario.

void addContacto(String contacto)
Añade un nuevo contacto a la lista de destinatarios.

Parámetros:

- contacto (String): Nombre del contacto.

void addMensaje(String emisor, String mensaje)
Añade un mensaje al historial.

Parámetros:

- emisor (String): Nombre del remitente.
- mensaje (String): Contenido del mensaje.

void sesionIniciada(String identificador)
Configura el título de la ventana al iniciar sesión.

Parámetros:

- identificador (String): Identificador del cliente.

void eliminarContacto(String identificador)
Elimina un contacto de la lista de destinatarios.

Parámetros:

- identificador (String): Identificador del contacto.

public static void main(String args[])
Método principal. Inicia la interfaz gráfica.

Package chat_servidor.servidor

HiloCliente.java

La clase HiloCliente es un componente clave del sistema de chat en el servidor. Esta clase representa un hilo que maneja la comunicación con un cliente a través de sockets. Permite recibir y procesar mensajes, así como enviar respuestas desde el servidor al cliente.

La clase HiloCliente extiende Thread para ejecutar la comunicación con el cliente en un hilo separado.

Atributos

private final Socket socket
Socket de comunicación con el cliente.

private ObjectOutputStream oos
Flujo de salida para enviar objetos al cliente.

private ObjectInputStream ois
Flujo de entrada para recibir objetos del cliente.

private final Servidor server
Instancia del servidor al que pertenece este hilo.

private String id
Identificador único del cliente.

private boolean listening
Indicador para controlar si el hilo sigue escuchando mensajes del cliente.

Métodos

public HiloCliente(Socket socket, Servidor server)
Constructor. Inicializa el hilo para manejar la comunicación con un cliente.

Parámetros:

- socket (Socket): Socket de comunicación con el cliente.
- server (Servidor): Instancia del servidor que administra este hilo.

public void run()
Método principal del hilo. Maneja la escucha y la comunicación con el cliente.

public void desconectar()
Desconecta al cliente cerrando el socket y deteniendo la escucha.

public void listening()
Inicia el bucle de escucha para recibir mensajes del cliente.

public void ejecutar(LinkedList<String> lista)
Ejecuta acciones basadas en los mensajes recibidos del cliente.

Parámetros:

- lista (LinkedList<String>): Lista de cadenas que representa el mensaje recibido.

private void enviarMensaje(LinkedList<String> lista)
Envía un mensaje al cliente asociado a este hilo.

Parámetros:

- lista (LinkedList<String>): Lista de cadenas que representa el mensaje a enviar.

private void confirmarConexion(String id)
Confirma la conexión del cliente asignándole un identificador único.

Parámetros:

- id (String): Identificador proporcionado por el cliente.

public String getIdent()
Obtiene el identificador único del cliente asociado a este hilo.

private void confirmarDesConexion()
Confirma la desconexión del cliente y notifica a los demás.

Servidor.java

La clase Servidor representa el servidor principal en un sistema de chat. Su función es gestionar las conexiones de los clientes, manejar la comunicación entre ellos y mantener un registro de los usuarios conectados. Extiende Thread para ejecutar la escucha de conexiones en un hilo separado.

La clase Servidor extiende Thread y maneja las conexiones de los clientes al servidor.

Atributos

private ServerSocket socketServer
Socket del servidor que escucha conexiones entrantes.

LinkedList<HiloCliente> clientes
Lista de hilos de cliente conectados al servidor.

private final VentanaS ventana;
Ventana gráfica asociada al servidor.

private final String puerto;
Puerto en el que se inicia el servidor.

static int diferenciador;
Diferenciador para generar identificadores únicos para los clientes.

Métodos

public Servidor(String puerto, VentanaS ventana)
Constructor. Inicializa el servidor y lo configura para escuchar conexiones en un puerto dado.

Parámetros:

- puerto (String): Puerto en el que se iniciará el servidor.
- ventana (VentanaS): Interfaz gráfica del servidor.

public void run()
Método principal del hilo. Maneja la recepción de conexiones entrantes.

LinkedList<String> getUsuariosConectados()
Obtiene la lista de identificadores de los usuarios conectados al servidor.

void log(String texto)
Agrega una entrada al log en la interfaz gráfica del servidor.

Parámetros:

- texto (String): Texto a agregar al log.

VentanaS.java

La clase VentanaS representa la interfaz gráfica del servidor en un sistema de chat. Proporciona un entorno visual para configurar y monitorizar la actividad del servidor, incluyendo la conexión y la visualización de eventos en tiempo real.

La clase VentanaS extiende JFrame y representa la ventana principal del servidor.

Atributos

private final String DEFAULT_PORT = "10101"
Puerto por defecto para la conexión del servidor.

private final Servidor server
Instancia del servidor asociado a esta ventana.

JTextArea txtClientes = new JTextArea()
Área de texto para mostrar el log del servidor.

Métodos

public VentanaS()
Constructor. Inicializa la ventana gráfica del servidor y su configuración.

private void initComponents()
Configura los componentes gráficos de la ventana

public static void main(String[] args)
Método principal que inicia la aplicación del servidor.

void log(String texto)
Agrega texto al log en la interfaz gráfica del servidor.

Parámetros:

- texto (String): Texto a agregar al log.

private String getPuerto()
Obtiene el puerto de conexión ingresado por el usuario.

void addServidorIniciado()
Muestra un mensaje indicando que el servidor se ha iniciado