Taller de aplicación: comunicación TCP

Sala de chat

Problema: Implementación de una sala de chat en Java utilizando sockets TCP

Descripción:

Parte 1: El objetivo de este proyecto es implementar un aplicativo de sala de chat en Java utilizando sockets TCP. El aplicativo constará de un servidor que permitirá a múltiples clientes comunicarse entre sí en tiempo real. Cada cliente podrá seleccionar un nombre de usuario y enviar mensajes a la sala de chat, que serán visibles para todos los demás clientes conectados.

Requisitos del Aplicativo:

1. Servidor:

- Debe ser capaz de aceptar conexiones de múltiples clientes utilizando sockets TCP.

- Mantener una lista de usuarios conectados a la sala de chat.

- Permitir a los clientes enviar mensajes a la sala de chat, que serán distribuidos a todos los clientes conectados.

- Implementar una estrategia para manejar nombres de usuario únicos y evitar duplicados.

- Manejar posibles problemas de conexión, como conexiones perdidas o interrupciones en la red.

2. Cliente:

- Debe permitir al usuario establecer una conexión con el servidor utilizando un socket TCP.

- Solicitar al usuario que ingrese un nombre de usuario antes de continuar.

- Enviar el nombre de usuario al servidor y esperar la confirmación.

- Después de la confirmación, permitir al usuario enviar y recibir mensajes en la sala de chat.

- Implementar un mecanismo para manejar mensajes entrantes de forma asíncrona mientras el cliente sigue siendo capaz de enviar mensajes.

3. Lógica de Comunicación:

- Implementar una lógica para enviar y recibir mensajes entre el servidor y los clientes utilizando sockets TCP.

Los estudiantes deben trabajar en equipos para diseñar e implementar los diferentes componentes del aplicativo de sala de chat en Java.

Deben utilizar los conceptos de programación orientada a objetos y sockets TCP para desarrollar el aplicativo.

Se debe prestar especial atención a la lógica de comunicación entre el servidor y los clientes, así como a la gestión de usuarios y mensajes en la sala de chat.

Definir las clases y sus atributos

Del lado del Servidor

1. Server:

- Define una clase **Server** con un método **main**.
- El servidor debe estar atento a nuevos clientes que se conecten y para cada uno lanzar un hilo usando una clase llamada ClientHandler para manejar a cada cliente individualmente.
- Debe iniciar un `ServerSocket` en un puerto específico para escuchar las conexiones de los clientes y pasárselo a ClientHandler para que gestione la comunicación con ese cliente.

2. ClientHandler:

- Define una clase **ClientHandler** que implementa la interfaz **Runnable**.
- Cada ClientHandler manejará a un cliente individual.
- Debe tener un Socket para la comunicación con el cliente y un nombre de usuario asociado.
- Se encargará de recibir y enviar mensajes entre el cliente y el servidor.
- Cuando un nuevo cliente se conecta, agrega su nombre de usuario a la lista de Chatters, junto con el canal de salida para poder enviar mensajes a ese usuario.
- Cuando el cliente envía un mensaje, desde este hilo se reenvía el mensaje a todos los clientes conectados a través de la lista de Chatters usando un método llamado **broadCastMessage().**

3. Chatters:

- Define una clase **Chatters** para mantener la lista de usuarios en la sala de chat a la cual tendrá acceso la clase ClienHandler.
- Debe permitir agregar y eliminar usuarios de la lista.
- Debe proporcionar un método para enviar mensajes a todos los usuarios en la lista.

4. Person:

- Define una clase **Person o User (como usted prefiera)** para representar a cada usuario.
- Debe tener atributos para el nombre de usuario y un **PrintWriter** para enviar mensajes a ese usuario.

Del lado del cliente

5. Client:

- Define una clase **Client** con un método **main**.
- El cliente debe establecer una conexión con el servidor mediante un Socket.
- Debe permitir al usuario elegir un nombre de usuario antes de continuar.
- Después de que se acepte el nombre de usuario, debe lanzar un hilo para leer mensajes del servidor y mostrarlos al usuario.
- Debe permitir al usuario ingresar mensajes desde el teclado y enviarlos al servidor.

Parte 2: Modificar el diseño para permitir mensajes privados entre usuarios. ¿Que estrategia usaría?

Se les entrega una plantilla de trabajo que permitirá agilizar el proceso.