Proyecto Cliente-Servidor en Java

Integrantes

- José Ramirez Dure
- Javier Toshifumi Goto Dominguez
- Kenya Takeuchi

Introducción

Este documento ofrece una visión general del proyecto de comunicación cliente-servidor en Java y la utilización de base de datos no relacional Redis. Se explican las funcionalidades de cada clase involucrada, los métodos principales y cómo interactúan entre sí. También encontrará una sección de un pequeño manual para la prueba del mismo.

Clase TCPMultiServer

La clase TCPMultiServer se encarga de gestionar las conexiones de los clientes. Su función principal es aceptar las conexiones y delegar el manejo de cada cliente a un hilo independiente a través de la clase TCPServerHilo. Además, esta clase mantiene una lista de usuarios en línea y utiliza la base de datos Redis para almacenar y recuperar información de los usuarios.

Métodos principales

• ejecutar(): Abre un puerto en el servidor (puerto 4444) y acepta conexiones de clientes. Cada conexión es manejada por un nuevo hilo TCPServerHilo.
• mostrarUsuariosConectados(): Imprime una lista de los usuarios actualmente conectados.
• mostrarUsuarios(): Imprime los datos de los usuarios almacenados en la base de datos Redis.
• desconectarUsuario(): Elimina un usuario de la lista de usuarios en línea.

Clase TCPServerHilo

TCPServerHilo maneja la comunicación con un cliente de manera independiente. Esta clase se ejecuta como un hilo, lo que permite manejar múltiples clientes de manera simultánea. Implementa funcionalidades como el inicio de sesión, la creación de nuevas cuentas, y un menú para mostrar usuarios en línea o almacenados en la base de datos.

Métodos principales

• run(): Se encarga del ciclo principal de interacción con el cliente, mostrando un menú para elegir opciones como crear una cuenta, iniciar sesión, o cerrar la conexión.
• crearCuenta(): Permite al cliente registrar un nuevo usuario, que luego es almacenado en

la base de datos Redis.
• iniciarSesion(): Verifica las credenciales de un cliente para iniciar sesión. Si es exitoso, agrega al cliente a la lista de usuarios en línea.

Clase RedisDB

RedisDB se encarga de la conexión con la base de datos Redis. Proporciona métodos para almacenar y recuperar objetos Persona, así como obtener una lista de todos los IDs de personas almacenadas.

Métodos principales

• almacenarPersona(): Almacena un objeto Persona en la base de datos Redis utilizando un ID único basado en la cédula del usuario.
• recuperarPersona(): Recupera los datos de una Persona a partir de su cédula.
• obtenerTodosLosCisDePersonas(): Devuelve una lista con todas las cédulas de los usuarios almacenados en la base de datos.

Clase Persona

La clase Persona representa un usuario del sistema. Contiene atributos básicos como la cédula, el nombre, el apellido, y el password. Esta clase es utilizada tanto por el servidor como por la base de datos Redis para almacenar y recuperar información de los usuarios.

Métodos principales

• toString(): Devuelve una representación en cadena de texto de un objeto Persona, mostrando el nombre, apellido y cédula.

Clase TCPClient

TCPClient implementa la lógica del cliente que se conecta al servidor. Envía comandos al servidor y recibe respuestas, permitiendo al usuario interactuar con el sistema a través de un menú. Utiliza sockets para la comunicación con el servidor.

Métodos principales

• main(): Establece la conexión con el servidor en el puerto 4444 y maneja la interacción del cliente con el servidor a través de un menú.
• leerEntrada(): Lee las respuestas del servidor y las muestra por pantalla.

Relación entre las clases

• TCPMultiServer es la clase principal que gestiona las conexiones de los clientes y los delega a un nuevo hilo TCPServerHilo.
• TCPServerHilo maneja la comunicación de cada cliente de manera independiente y se conecta a la base de datos Redis a través de RedisDB para almacenar o recuperar

información de usuarios.
• Persona es la clase que representa a los usuarios del sistema, cuyas instancias son almacenadas y recuperadas por RedisDB.
• TCPClient es el cliente que interactúa con el servidor, enviando solicitudes y recibiendo respuestas del servidor.

Manual de uso

Requisitos previos

1. **Instalar Docker Desktop:**
    - Descarga e instala Docker Desktop desde Docker Desktop.
    - Asegúrate de que Docker Desktop esté ejecutándose correctamente.

Pasos para ejecutar el servidor

1. **Abrir la terminal en el directorio del servidor:**
    - Navega hasta el directorio del proyecto `/Servidor` utilizando la terminal.
2. **Construir y levantar la base de datos con Docker:**
    - Ejecuta el siguiente comando para construir las imágenes:
        i. **docker-compose build**
    - Inicia los contenedores en segundo plano:
        i. **docker-compose up -d**
3. **Ejecutar el servidor java:**
    - Iniciar el servidor ejecutando el archivo .jar:
        i. java -jar target/Servidor-1.0-SNAPSHOT.jar
4. Con esto ya estaría corriendo el servidor el el puerto 4444.

Pasos para ejecutar el cliente

1. **Abrir otra terminal en el directorio del cliente:**
    - Navega hasta el directorio `/Cliente` en una nueva terminal.
2. **Ejecutar el cliente Java:**
    - Inicia el cliente ejecutando el siguiente comando:
        i. java -jar target/Cliente-1.0-SNAPSHOT.jar
3. Con esto ya podremos interactuar con el servidor, para agregar otro cliente se repite el mismo proceso.