RELAZIONE TECNICA

OGETTO

socket TCP

ALUNNO

Topcii Daniil

DOCENTI

Crafa Nicola

Conte Alberto

SOCKET TCP

Per la creazione di un socket TCP è stato proposto come esercizio la creazione di una chat che avesse un server che gestisce le connessioni e conta il numero di iterazioni per utente, quando il client si disconnette stampa a video il numero di iterazioni, il lato client invece deve permettere l'autenticazione tramite nickname, per svolgere l'esercitazione si sono creati due file:

SERVER e CLIENT.

SERVER:

- Il metodo **run**() viene eseguito quando viene avviato il thread del server. Crea un oggetto ServerSocket sulla porta 8000 e accetta le connessioni dei client in modo iterativo. Per ogni connessione, viene avviato un nuovo thread ClientThread che gestisce le interazioni del client.

CODICE:

```
public void run() {
        try (ServerSocket serverSocket = new ServerSocket(8000)) {
            // Accetta le connessioni dei client in modo iterativo
             // Crea un oggetto della classe ChatClient
            while (true) {
                Socket clientSocket = serverSocket.accept();
                // Avvia un nuovo thread per gestire le interazioni del client
                new ClientThread(clientSocket).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- La classe **ClientThread** rappresenta un thread che gestisce le interazioni di un client con il server. Il metodo **run**() viene eseguito quando viene avviato il thread del client. Riceve il nickname del client e lo aggiunge alla lista dei nickname utilizzati. Se il nickname è già stato

utilizzato, invia un messaggio di errore al client e interrompe il thread. Altrimenti, aggiunge il nickname del client alla mappa che associa ad ogni client connesso il suo nickname e invia a tutti i client connessi il messaggio di connessione del client. Riceve e invia i messaggi in modo iterativo finché il client è connesso. Alla disconnessione del client, rimuove il nickname dalla lista dei nickname utilizzati e dalla mappa dei nicknames, invia a tutti i client connessi il messaggio di disconnessione e stampa a video le iterazioni del client.

- Il metodo **broadcast**(String message) invia il messaggio a tutti i client connessi. Per fare ciò, scorre la mappa dei nicknames e invia il messaggio utilizzando l'output stream del socket di ogni client.

CODICE:

```
private void broadcast(String message) {
        for (Map.Entry<Socket, String> entry : clientNicknames.entrySet()) {
            Socket clientSocket = entry.getKey();
            try {
                PrintStream out = new PrintStream(clientSocket.getOutputStream());
                out.println(message);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

- Il metodo **main**(String[] args) crea un oggetto Server e avvia il suo thread.

CLIENT:

- La classe Client estende JFrame e definisce alcune variabili d'istanza per la finestra, il pannello dei messaggi, il campo di testo per l'invio dei messaggi e il pulsante per inviare i messaggi.
- Nel costruttore della classe Client viene creata la finestra della chat e vengono aggiunti i pannelli per i messaggi e l'invio dei messaggi.
- Viene inoltre mostrata una finestra di dialogo dove l'utente può inserire il suo nickname.

- Viene poi avviato un nuovo thread che si occupa di connettere il client al server e di gestire la ricezione e l'invio dei messaggi.
- Quando il pulsante "Invia" viene premuto, il codice all'interno dell'ActionListener invia il testo del campo di testo al server.