Relazione di Progetto: Sistema di Chat Client-Server

Marco Massa

14/05/24

1 Introduzione

La comunicazione tra dispositivi tramite reti è diventata sempre più diffusa e cruciale nelle moderne applicazioni informatiche. In questo progetto, ho implementato un sistema di chat client-server in Python utilizzando la programmazione a socket.

2 Obiettivo

L'obiettivo principale del progetto è stato sviluppare un'applicazione di chat client-server che permetta a più utenti di comunicare tra loro all'interno di una chatroom condivisa.

3 Architettura del Sistema

3.1 Server

Il server è responsabile per l'accettazione delle connessioni dei client e la gestione della comunicazione tra di essi. Il server utilizza un socket TCP per ascoltare le richieste di connessione dai client. Una volta accettata una connessione, il server avvia un thread separato per gestire la comunicazione con quel client.

3.2 Client

Il client è l'interfaccia utente attraverso la quale gli utenti possono interagire con la chat. Utilizza anche un socket TCP per connettersi al server. Il client offre una finestra di chat in cui gli utenti possono visualizzare i messaggi inviati dagli altri utenti e inviare i propri messaggi.

4 Implementazione Tecnica

4.1 Server-Side

Il server è implementato utilizzando la libreria `socket` di Python. Utilizzo il modulo `threading` per gestire più connessioni client contemporaneamente. Quando un nuovo client si connette, il server avvia un thread separato per gestire la comunicazione con quel client.

4.2 Client-Side

Il client è implementato utilizzando la libreria `tkinter` per l'interfaccia grafica e la libreria `socket` per la comunicazione con il server. La finestra di chat offre un campo di inserimento per l'utente per inviare i propri messaggi e una lista di messaggi ricevuti dagli altri utenti.

5 Gestione degli Errori

Durante lo sviluppo del progetto, ho prestato attenzione alla gestione degli errori al fine di garantire un'esperienza utente stabile e priva di problemi. Ho implementato la gestione degli errori attraverso l'utilizzo di blocchi try-except sia lato client che lato server per affrontare eventuali situazioni impreviste che potrebbero verificarsi durante l'esecuzione del sistema, lato server visualizzando i messaggi di errore nel terminale ma continuando ad ascoltare nuove connessioni e lato client visualizzando il messaggio di errore e chiudendo la connessione.

6 Utilizzo del sistema

Per utilizzare il codice, aprire un'istanza del terminale e avviare il server tramite il comando 'python chat_server.py'. Ora il server sarà in ascolto in attesa di connessioni. Successivamente, aprire altre istanze del terminale ed eseguire il client tramite il comando 'python chat_client.py'. Verrà richiesto di inserire il Server host, inserire "127.0.0.1", e il numero di porta, 1918. Una volta eseguiti questi passaggi, saremo connessi al server. Sarà richiesto di inserire un nome, quindi i messaggi inseriti verranno inviati al server per la comunicazione nella chat.

7 Conclusioni

In conclusione, ho sviluppato un sistema di chat client-server in Python utilizzando la programmazione a socket. Il sistema permette a più utenti di comunicare tra di loro in tempo reale all'interno di una chatroom condivisa.