Relazione Progetto Programmazione di Reti

Irene Sofia Lotti 0001071300

May 2024

1 Introduzione

Questo documento descrive il funzionamento di un sistema di chat client-server implementato in Python utilizzando la programmazione con socket. Il sistema consente a più client di connettersi a un server e partecipare a una chatroom condivisa. Gli utenti possono inviare e ricevere messaggi all'interno di questa chatroom. Il client ha una semplice interfaccia grafica (GUI) basata su Tkinter. Chiaramente, i client non solo possono vedere i messaggi degli altri utenti, ma anche i propri messaggi.

2 Requisiti

I requisiti per eseguire il codice sono i seguenti:

- Python 3.x = per eseguire il codice è necessario avere installata una versione recente di Python 3.
- Tkinter = si tratta di una libreria GUI per Python che è generalmente inclusa con le distribuzioni standard di Python.
- Una rete locale o Internet = è necessaria per consentire la comunicazione tra server e client.

3 Descrizione del Server e caratteristiche

Il server di questo sistema è responsabile della gestione delle connessioni dei client e della condivisione dei messaggi all'interno della chatroom condivisa.

All'inizio, il server configura un logger per tenere traccia degli eventi significativi e degli errori. Questo aiuta a monitorare l'attività del server e a diagnosticare eventuali problemi.

Il server crea poi una socket per ascoltare le connessioni in arrivo su una porta specifica. Usando la funzione `accetta_connessioni_in_entrata`, accetta continuamente nuove connessioni da parte dei client. Infatti, ogni volta che un client si connette, il server crea un nuovo thread per gestire la comunicazione

con quel client, permettendo così la gestione di più client contemporaneamente senza bloccare il server.

Una volta stabilita la connessione, il client viene invitato a fornire un nome utente. Il thread dedicato al client esegue la funzione `amministra_client`, che gestisce la comunicazione continua con il client. Il server tiene traccia dei client attivi in un dizionario (clients)

La funzione `broadcast` invia messaggi a tutti i client connessi. Se un tentativo di invio a un client fallisce, il client viene rimosso dalla lista dei client attivi.

!off è il messaggio che viene mandato dal client quando vuole terminare la connessione. Quindi il server chiude la connessione con quel client e notifica agli altri client la sua disconnessione. La funzione `amministra_client` si occupa anche di gestire eventuali errori o eccezioni che possono verificarsi durante la comunicazione, garantendo che il server rimanga operativo anche se un singolo client causa un problema.

Il server può essere terminato manualmente, e in tal caso, tutte le connessioni vengono chiuse in modo ordinato. Questo è gestito tramite un'eccezione `KeyboardInterrupt` che consente di chiudere il server in modo sicuro quando l'utente termina il processo.

4 Descrizione del Client e caratteristiche

Per quanto riguarda il client, esso si connette al server usando una GUI basata su Tkinter per inviare e ricevere messaggi.

All'inizio, il client configura un logger per registrare eventi importanti e errori. Questo aiuta a monitorare l'attività del client e a diagnosticare eventuali problemi.

Il client viene avviato creando la GUI e stabilendo una connessione al server. Viene creato un thread per ricevere i messaggi dal server e viene avviato il loop principale di Tkinter per gestire l'interfaccia utente. Abbiamo un campo per visualizzare i messaggi della chat, un'area per scrivere e inviare nuovi messaggi, un pulsante SEND e una scrollbar per scorrere i messaggi visualizzati.

L'utente inserisce l'indirizzo del server e la porta su cui il server è in ascolto. Il client utilizza queste informazioni per stabilire una connessione al server tramite una socket. Dopo la connessione, il client crea un thread separato per gestire la ricezione dei messaggi dal server. Questo thread garantisce che i messaggi in arrivo possano essere visualizzati nella GUI senza bloccare l'interfaccia utente.

La funzione `ricezione_messaggi` viene eseguita in un thread separato e si occupa di ricevere continuamente i messaggi dal server. Invece, la funzione `invia_messaggi` recupera il testo dall'area di input, lo codifica e lo invia al server. Se l'utente scrive "!off", il client chiude la connessione con il server e termina l'applicazione GUI.

La funzione `on_closing` viene chiamata quando l'utente chiude la finestra della GUI. Questa funzione invia il comando "!off" al server per chiudere la

connessione in modo ordinato e poi termina l'applicazione.

5 Esecuzione

Ecco come deve avvenire l'esecuzione:

1 Si avvii il server usando il codice fornito

2 Si avviino una o più istanze di client usando i codici forniti

3 Si testi l'implementazione scrivendo nella text area e premendo il pulsante invio.