Sbrighi Patrick 0001071460

Documentazione
Sistema di chat Client-Server

Elaborato traccia 1

Descrizione del Sistema

Il sistema implementa un'applicazione client-server per la gestione di una chatroom condivisa tra più client, i quali connettendosi allo stesso server hanno la possibilità di inviare e ricevere messaggi.
L'applicazione è realizzata attraverso l'utilizzo di socket e thread in Python, infatti ogni per ogni client che si connette al server viene avviato un thread dedicato per la gestione della sua connessione, permettendo quindi al server di gestire più client contemporaneamente.

Funzionamento del Sistema

Server

1. **Configurazione iniziale**
   Il server in questione è configurato per stare in ascolto su HOST = '127.0.0.1' e PORT = 54321, utilizzando questo indirizzo e questa porta i vari client potranno connettersi al server instaurando una nuova connessione ed inviando i propri messaggi. La socket del server viene creata utilizzando il seguente comando:
   socket.socket(socket.AF_INET, socket.SOCK_STREAM).

2. **Accettazione delle connessioni**
   Attraverso il comando server.listen(5) il server viene messo in ascolto e potrà accettare nuove connessioni utilizzando server.accept(). Dopo che il server ha accettato la connessione di un client si occupa di creare ed avviare un nuovo thread per poter gestire la connessione appena instaurata.

3. **Gestione dei Client**
   Il server memorizza i client che sono connessi salvandoli all'interno di

una lista (clients), quando viene accettata la connessione di un nuovo client questo verrà inserito nella lista, mentre quando un client decide di terminare la sua connessione verrà immediatamente rimosso da essa. Ogni client è gestito tramite la funzione clientHandler(clientSocket, addres), la quale si occupa di ricevere gli eventuali messaggi che il client vuole mandare e di inoltrarli a tutti gli altri client in quel momento connessi.

4. **Invio dei Messaggi**
   L'invio dei messaggi è gestito dalla funzione broadcast(mess, senderSocket), questa riceve come parametro il messaggio da inviare (mess) e si occupa di mandarlo a tutti tranne che al client mittente, quest'ultimo viene identificato tramite il parametro senderSocket. Inoltre questa funzione si occupa anche di inserire nel messaggio le informazioni del mittente utilizzando senderSocket.getpeername().

Client

1. **Configurazione iniziale**
   Il client per potersi connettere al server dovrà utilizzare HOST='127.0.0.1' e PORT=54321. La socket viene creata utilizzando il comando socket.socket(socket.AF_INET, socket.SOCK_STREAM).

2. **Ricezione dei Messaggi**
   Se il tentativo di connessione con il server va a buon fine il client avvierà un thread che utilizzerà la funzione che si occupa della ricezione dei messaggi che verranno inviati dal server e di stamparli a video. La funzione che si occupa di questo è denominata receive().

3. **Invio dei Messaggi**
   Il client invia i messaggi scritti dall'utente in un ciclo continuo fino a quando non riceve il comando di uscita, ossia la stringa "exit". Quando un utente digita il comando di uscita il client si disattiva ed invia il messaggio al server che ne chiude la connessione.

Requisiti per Eseguire il Codice

Per poter eseguire questo codice è necessario avere Python 3.x installato sul proprio pc ed è necessario anche avere le librerie standard di Python, dato che vengono utilizzate le librerie per le socket e per i thread.

Per poter avviare l’applicazione è necessario prima far partire il server utilizzando il file chiamato server.py e poi aprire tutti i client che si vuole attraverso il file client.py.

Considerazioni aggiuntive

- **Chiusura delle Connessioni**
  Un utente può scegliere di terminare il proprio collegamento digitando il comando “exit”. Ricevendo questo comando il client si occuperà di interrompere il ciclo di invio dei messaggi e di chiudere correttamente la socket, mentre sarà compito del server occuparsi della chiusura della connessione rimuovendo anche il client in questione dalla lista dei client in ascolto.

- **Gestione degli Errori**
  I codici dell’applicazione si occupano anche di gestire eventuali errori, infatti il codice dal lato client gestisce le eventuali eccezioni durante la connessione con il server, l’invio dei propri messaggi e la ricezione dei messaggi dal server, mentre quello del server si occupa della gestione delle eccezioni che possono essere generate durante la ricezione dei messaggi da parte dei client e si preoccupa di gestire correttamente le disconnessioni.