Sistema di Chat Client-Server in Python

Relazione progetto

Introduzione

Il progetto ha come scopo quello di implementare un sistema di chat client-server in Python utilizzando socket programming. Il server deve essere in grado di gestire più client contemporaneamente e deve consentire agli utenti di inviare e ricevere messaggi in una chatroom condivisa. Il client deve consentire agli utenti di connettersi al server, inviare messaggi alla chatroom e ricevere messaggi dagli altri utenti.

Obiettivi del progetto

- Implementare il lato server che accetta connessioni multiple e gestisce la trasmissione dei messaggi nella chatroom.
- Implementare il lato client che consente agli utenti di connettersi al server e inviare messaggi.
- Gestire gli errori e delle eccezioni in modo appropriato

Descrizione Server

1. Importazioni e Inizializzazioni:

- Le librerie '**socket**' e '**threading**' vengono importate per gestire le comunicazioni di rete e il multi-threading.
- I dizionari '**clients**' e '**indirizzi**' sono utilizzati per tenere traccia dei client connessi e dei loro indirizzi IP.

2. Funzioni Principali:

- **Accetta_connessioni_in_entrata**: Questa funzione accetta le nuove connessioni in entrata. Per ogni connessione, crea un nuovo thread che gestisce la comunicazione con quel client.
- **Gestisce_client**: Gestisce la comunicazione con un singolo client. Riceve il nome del client, invia messaggi di benvenuto e notifica a tutti i client connessi l'entrata e l'uscita di un utente. Inoltre, ascolta i messaggi inviati dal client e li inoltra a tutti gli altri client.
- **Broadcast**: Invia un messaggio a tutti i client connessi, aggiungendo un prefisso per l'identificazione del mittente.

3. Configurazione e Avvio del Server:

Il server viene configurato per ascoltare su una specifica porta e avvia un thread per accettare le connessioni in entrata.

Flusso di esecuzione

- Il server si mette in ascolto per nuove connessioni.
- Quando un client si connette, viene creato un nuovo thread per gestire la comunicazione con quel client.
- Il client invia il proprio nome, che viene registrato dal server.
- I messaggi inviati dal client vengono ricevuti dal server e inviati a tutti gli altri client connessi.
- Se un client si disconnette, il server notifica a tutti gli altri client l'abbandono alla chat.

Descrizione Client

1. Importazioni e Inizializzazioni:

- Le librerie ‘**socket**’ e ‘**threading**’ vengono importate per gestire le comunicazioni di rete e il multi-threading.
- Le variabili globali vengono definite per gestire il socket del client e il buffer di dimensione.

2. Funzioni Principali:

- **Receive**: Questa funzione viene eseguita in un thread separato e si occupa di ricevere i messaggi dal server e stamparli sulla console del client.
- **Send**: Invia i messaggi inseriti dall'utente al server. Se l'utente digita {quit}, il client si disconnette dal server.

3. Configurazione e Avvio del Server:

Il client Inizializza la connessione al server e avvia i thread per ricevere e inviare messaggi.

Flusso di esecuzione

- Il client si connette al server specificando l'indirizzo IP e la porta del server.
- Vengono creati due thread: uno per ricevere messaggi dal server e uno per inviare messaggi al server.
- Il client riceve i messaggi dal server e li visualizza nella casella di testo della GUI.
- L'utente può ricevere e inviare messaggi dalla casella di input al server, che li inoltra a tutti gli altri client connessi.
- Se l'utente digita {quit}, il client si disconnette dal server e la GUI viene chiusa.

Esempio di Comunicazione

- Il client A si connette al server e invia il proprio nome.
- Il server registra il nome e l'indirizzo del client A e notifica tutti gli altri client che A si è unito alla chat.
- Il client B si connette e segue lo stesso processo.
- Il client A invia un messaggio che viene ricevuto dal server e inoltrato al client B.
- Il client B riceve il messaggio e lo visualizza sulla sua console.
- Se il client A si disconnette, il server notifica tutti i client rimanenti.

Libreria Tkinter

Tkinter è una libreria standard di Python per la creazione di interfacce grafiche (GUI). Il client utilizza questa libreria per migliorare l'esperienza di utilizzo, infatti5 fornisce una vasta gamma di widget per costruire GUI, come pulsanti, caselle di testo ed altri.

Gestione degli errori

Sia il server che il client includono meccanismi per gestire errori ed eccezioni, come la gestione delle connessioni perse e gli errori durante l'invio e la ricezione dei messaggi. Questo garantisce la robustezza del sistema e una migliore esperienza utente.

Conclusioni Finali

Il sistema di chat implementato è un esempio semplice ma efficace di comunicazione di rete. Utilizza il multi-threading per gestire più connessioni contemporaneamente e permette una comunicazione in tempo reale tra più client. L'integrazione di Tkinter offre una GUI semplice e intuitiva che facilita l'interazione fra gli utenti, e organizza le informazioni, separando i messaggi ricevuti e inviati. Il codice è strutturato in modo tale da separare chiaramente le responsabilità tra il server e il client. Infine le eccezioni sono gestite per garantire robustezza al sistema, e i messaggi di log aiutano nel monitoraggio delle attività e nel debug.

Esecuzione del progetto

- Avviare il codice del server
- Avviare le istanze del client