Relazione: Sistema di Chat

Introduzione

Progetto sviluppato nel corso di Programmazione di Reti dell'Università di Cesena.
Consiste nella realizzazione di un sistema di chat con architettura Client-Server.
Questo documento descrive dettagliatamente le funzionalità implementate sia sul lato server che sul lato client.

Architettura

Il sistema è composto da due componenti principali:

1. **Server Chat:** Gestisce le connessioni dei client, riceve i messaggi e li distribuisce a tutti i client connessi.
2. **Client Chat con GUI:** Permette agli utenti di connettersi al server, inviare e ricevere messaggi attraverso una interfaccia grafica.

Obiettivi

- **Creare un server di chat** capace di gestire connessioni multiple in modo concorrente.
- **Sviluppare un client con un'interfaccia grafica intuitiva** che permetta agli utenti di inviare e ricevere messaggi facilmente.
- **Implementare meccanismi efficaci** per la gestione delle connessioni e garantire una comunicazione sicura tra client e server.

Istruzioni per l'Esecuzione del Codice

Server

Esecuzione del Server:

Eseguire il file server.py utilizzando il comando seguente nel terminale: python chat_server.py. Il server si metterà in ascolto sulla porta predefinita 53000. Non è necessario specificare manualmente il numero massimo di connessioni simultanee, in quanto il server è configurato per accettare fino a 5 connessioni simultanee.

Client

Esecuzione Client:

Eseguire il file client.py utilizzando il comando seguente nel terminale: python chat_client.py
Verrà visualizzata una finestra GUI per la chat. Nella parte superiore della finestra, inserire l'host del server (ad esempio, localhost se si esegue il server sulla stessa macchina) e la porta (predefinita 53000). Se si preme "Invio" senza inserire nulla, verranno utilizzati i valori predefiniti.
Premere il pulsante "Connetti" per connettersi al server.

Utilizzo Chat:

Dopo la connessione, si può iniziare a inviare e ricevere messaggi nella chatroom.
Digitare i messaggi nel campo di input in basso e premere "Invia" o il tasto "Invio" per mandarli.
Per uscire dalla chat, digitare {quit} e inviare il messaggio. Questo chiuderà la connessione e uscirà dalla chat.

Implementazione

Metodi e variabili Server

Le variabili globali vengono definite all'inizio dello script per configurare il comportamento del server:

-HOST: L'indirizzo IP sul quale il server ascolta le richieste in arrivo.: L'indirizzo IP sul quale il server ascolta le richieste in arrivo.

-BUF_SIZE: La dimensione del buffer utilizzato per la ricezione dei dati.

-**ADDR:** Una tupla che combina l'indirizzo IP e la porta per identificare in modo univoco il punto di connessione del server.

-PORT: La porta specifica utilizzata dal server per accettare connessioni.: La porta specifica utilizzata dal server per accettare connessioni.

La variabile __name__ viene utilizzata per determinare se uno script Python è eseguito come programma principale o se è importato come modulo in un altro script. Quando uno script viene eseguito, Python assegna automaticamente il valore "__main__" alla variabile__name__. Pertanto, possiamo utilizzare questa variabile per condizionare l'esecuzione di determinate parti dello script.

- Metodo: accetta_connessioni_client (Gestisce l'ascolto delle connessioni in ingresso dai client.)

Questo metodo è responsabile di accettare nuove connessioni dai client. Il server ascolta costantemente su una porta specifica per le richieste di connessione in ingresso e crea un nuovo thread per gestire ogni client connesso. Questo permette al server di gestire più connessioni contemporaneamente.

- Metodo: gestore_client (Gestisce i messaggi provenienti da un singolo client.)

Questo metodo gestisce la comunicazione con un singolo client. Una volta che il client si è connesso, riceve il nome del client e invia un messaggio di benvenuto. Successivamente, il metodo ascolta costantemente i messaggi inviati dal client e li inoltra agli altri client connessi. Se il client decide di lasciare la chat, il metodo gestisce la chiusura della connessione e notifica agli altri client la disconnessione.

- Metodo: notifica_tutti (Invia un messaggio a tutti i client connessi.)

Questo metodo invia un messaggio a tutti i client connessi. Viene chiamato ogni volta che un client invia un messaggio, o si verifica un evento di sistema come l'ingresso o l'uscita di un utente. È essenziale per garantire che tutti i client siano informati sugli aggiornamenti in tempo reale.

Metodi Client

- Metodo: ricevi_messaggi (Gestisce la ricezione dei messaggi dal server.)

Questo metodo è eseguito in un thread separato e gestisce la ricezione dei messaggi dal server. Quando il client riceve un messaggio, lo visualizza nell'area della chat. Se si verifica un errore o il server chiude la connessione, il metodo gestisce questi eventi e interrompe il loop di ricezione.

- Metodo: invia_messaggio (Gestisce l'invio dei messaggi al server.)

Questo metodo è responsabile dell'invio dei messaggi al server. Quando l'utente inserisce un messaggio e preme il tasto "Invia" (o il tasto Invio), il messaggio viene inviato al server. Se l'utente inserisce il comando per uscire dalla chat, il metodo chiude la connessione e termina l'applicazione client.

- Metodo: on_close (Gestisce la chiusura della finestra di chat.)

Questo metodo gestisce la chiusura della finestra della chat. Se l'utente chiude la finestra dell'applicazione, il metodo invia un messaggio al server indicando che il client sta uscendo dalla chat e chiude la connessione.

- Metodo: connetti_al_server (Gestisce la connessione al server di chat.)

Questo metodo gestisce la connessione al server della chat. L'utente inserisce l'indirizzo del server e la porta nella GUI. Il metodo tenta di connettersi al server utilizzando queste informazioni. Se la connessione ha successo, viene avviato il thread per ricevere i messaggi. In caso di errore, viene mostrato un messaggio di errore all'utente.

Conclusioni

Il progetto ha realizzato un sistema di chat funzionale. L'implementazione del server con gestione concorrente delle connessioni e del client con interfaccia grafica offre un esempio pratico di applicazione di tecniche di programmazione di reti e sviluppo di GUI in Python. Il sistema può essere ulteriormente migliorato.

Nome: Gjovalin Cognome:Mujaj Matricola: 0001098294