Relazione Dettagliata sul Progetto: Sistema di Chat

Introduzione

Il progetto, sviluppato nell'ambito del corso di Programmazione di Reti dell'Università di Bologna, 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, concentrandosi sui metodi e le tecniche utilizzate.

Obiettivi del Progetto

- **Realizzare un server di chat** in grado di gestire connessioni concorrenti da parte di più client.
- **Sviluppare un client con interfaccia grafica** per permettere agli utenti di inviare e ricevere messaggi in modo intuitivo.
- **Implementare meccanismi di gestione delle connessioni** e di comunicazione sicura tra client e server.

Architettura del Sistema

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.

Dettagli Implementativi

Server

Variabili Globali e Inizializzazione del 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.
- **PORT:** La porta specifica utilizzata dal server per accettare connessioni.
- **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.
- 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`

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`

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`

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.

Client

Metodo: `ricevi_messaggi`

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`

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`

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`

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 raggiunto gli obiettivi prefissati, realizzando un sistema di chat funzionale e user-friendly. 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 con funzionalità aggiuntive come la crittografia dei messaggi e la gestione avanzata degli utenti.

Nome: Giuseppe. Cognome:Marini. Matricola: 0001091174.