“Relazione Chat Client-Server”

per il corso Programmazione di Reti
A.A. 2023/2024

Elisa Yan - 0001070882
elisa.yan@studio.unibo.it

16 maggio 2024

1 Introduzione

Per il progetto di Programmazione di Reti è stato richiesto di sviluppare un sistema che implementa una chat client-server in Python utilizzando socket programming.

Il server deve essere in grado di gestire più client contemporaneamente e di consentire agli utenti di inviare e ricevere messaggi in una chatroom condivisa.

2 Design ed implementazione

Il sistema è composto da due componenti principali: il server e il client.

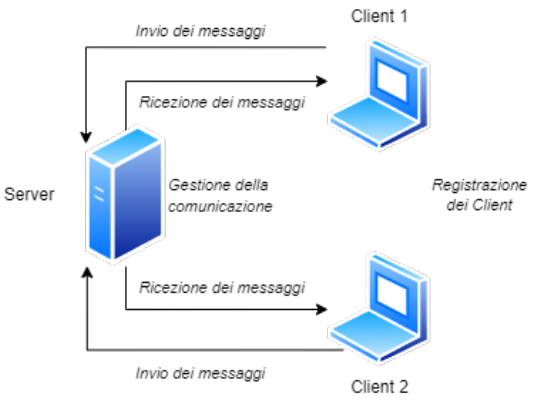

Figura 1: Componenti principali e funzionamento del sistema

2.1 Server

Il server, progettato per gestire una chat room, si mette in ascolto su una socket TCP per accettare le connessioni dei client. Quando un client è connesso, gli viene richiesto di

inserire un nickname, che sarà associato alla connessione. Ogni client è gestito tramite un thread individuale dal server, consentendo una comunicazione simultanea con più client.

Nel caso in cui un client si disconnette, il server lo rimuove dalla lista dei client attivi, informando altri partecipanti della chat.

Il server può essere chiuso dall'amministratore tramite i comandi "quit" o "shutdown" inseriti nella console. Successivamente il server invia un messaggio di chiusura a tutti i client connessi, terminando tutte le connessioni attive e chiudendo la socket.

2.2 Client

Il client è progettato utilizzando l'interfaccia grafica Tkinter per fornire una finestra di chat intuitiva e interattiva. All'avvio viene richiesto all'utente di inserire l'indirizzo del server e ne viene verificata la validità. In caso di errore, viene chiesto all'utente di reinserirlo. Dopo aver ottenuto un indirizzo valido, viene chiesto di inserire il numero di porta e il client tenta di stabilire una connessione con il server. Se la connessione fallisce, viene mostrato un messaggio all'utente e gli verrà chiesto di riprovare.

Una volta stabilita la connessione, il client avvia la GUI.i messaggi inviati dall'utente vengono transmessi al server. Il server li distribuisce a tutti gli altri client connessi, per poi visualizzati nell'area di testo della chat.

Il client gestisce la connessione al server in modo asincrono utilizzando due thread separati: un thread per la GUI, che gestisce l'interazione dell'utente, e un thread per la ricezione dei messaggi dal server. In caso di disconnessione o errore di rete, il client gestisce la chiusura della connessione in modo ordinato e informa l'utente.

L'utente può chiudere la finestra di chat, provocando la chiusura della connessione al server e l'uscita dall'applicazione.

3 Esempio di utilizzo

```
PS C:\Users\jiaax\Desktop\Chat-ClientServer-TCP> python .\server.py
Server in ascolto...
127.0.0.1:38176 si è collegato
Il nickname del client è user1
127.0.0.1:30637 si è collegato
Il nickname del client è user2
```

```
PS C:\Users\jiaax\Desktop\Chat-ClientServer-TCP> python .\client.py
Inserisci il server host: local
Host non valido, riprova
Inserisci il server host: localhost
Inserisci il server port:
Connessione riuscita!
```

Figura 2: Collegamento al server: gestione del caso in cui viene inserito un host errato

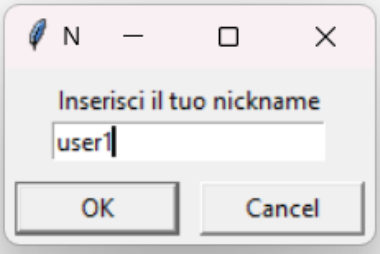

Figura 3: Pagina dell'inserimento del nickname

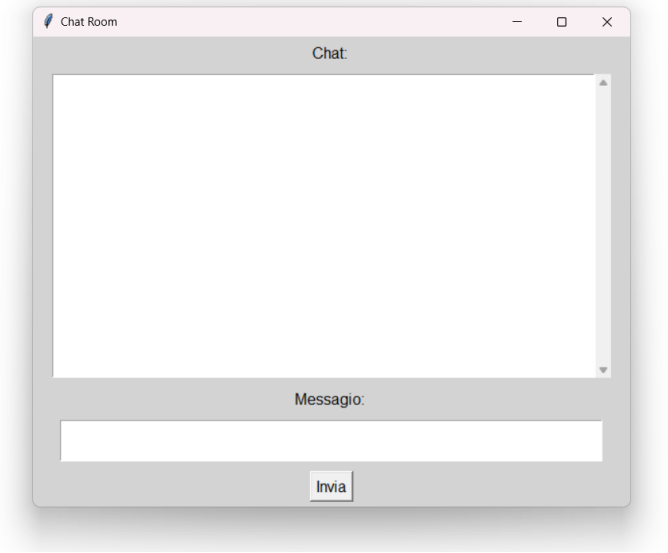

Figura 4: Pagina iniziale della chat, una volta che l'utente è entranto

L'immagine a destra mostra i messaggi ricevuti e inviati da *user1*, mentre quella a sinistra mostra quelli di *user2*.

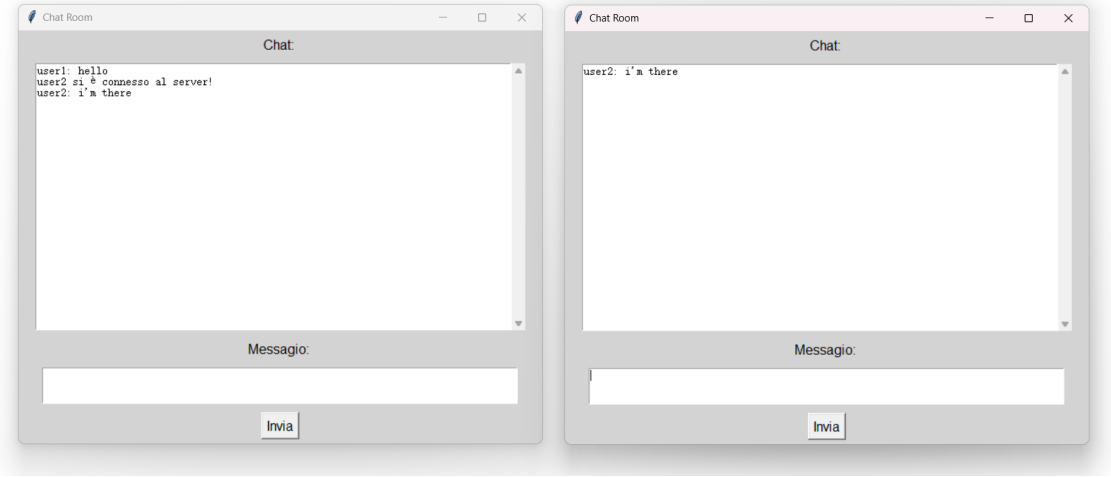

Figura 5: Pagina che mostra l'entrata di un nuovo utente

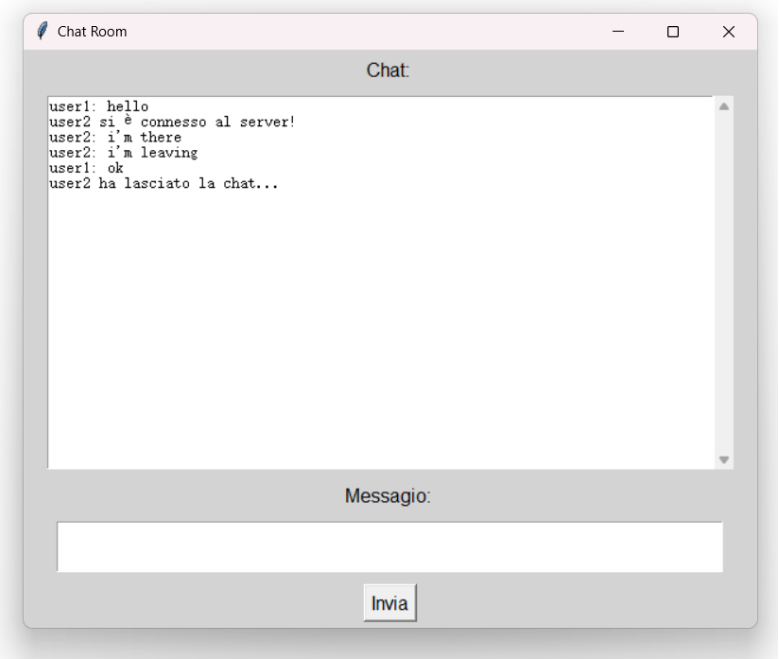

Figura 6: Pagina che mostra l'uscita di un utente

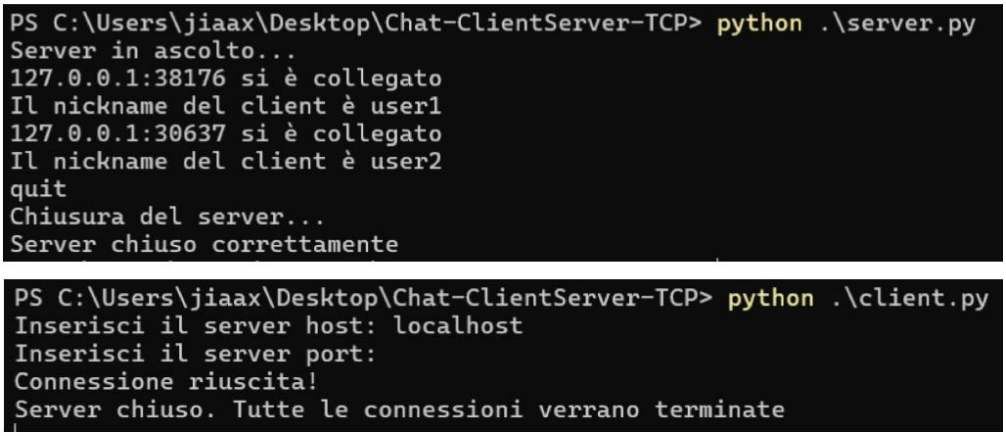

Figura 7: Chiusura del server dal terminale

4 Esecuzione dell'applicazione

Per eseguire l'applicazione del sistema di chat client-server, è necessario avere installato Python sul proprio sistema.

È necessario avviare prima il server eseguendo il seguente comando sul terminale:

```
python .\server
```

Questo comando avvia il server, mettendolo in ascolto dei client che tentano di connettersi. Successivamente, il client può essere eseguito digitando il seguente comando nel terminale:

```
python .\client
```

Durante l'esecuzione del client, all'utente viene richiesto di specificare l'indirizzo IP e la porta del server a cui desidera connettersi. È possibile avviare più client in contemporanea ripetendo la stessa operazione.

Una volta che il client è connesso con successo al server, può inviare messaggi nella chat e ricevere messaggi dagli altri client connessi.

5 Commenti finali

Il sistema attuale fornisce solo funzionalità di base per la trasmissione di messaggi, ma può essere esteso ulteriormente con nuove funzionalità e miglioramenti.