"Relazione Chat Client-Server"

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

Elisa Yan
elisa.yan@studio.unibo.it

11 maggio 2024

1 Introduzione

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

Il server è in grado di gestire più client contemporaneamente e consente agli utenti di inviare e ricevere messaggi in una chatroom condivisa.

Il client fornisce un'interfaccia grafica utente (GUI) tramite Tkinter, consentendo agli utenti di connettersi al server, inviare messaggi alla chatroom e ricevere messaggi dagli altri utenti.

2 Design ed implementazione

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

2.1 Server

Il server si mette in ascolto su un socket TCP per accettare le connessioni dei client. Una volta connesso un client, il server richiede al client di inserire un nickname. Questo nickname viene quindi associato al client e la lista degli utenti attivi viene inviata a tutti i client connessi. Il server gestisce i messaggi inviati dai client e li inoltra a tutti gli altri client connessi. In caso di disconnessione di un client, il server rimuove il client dalla lista degli utenti attivi e aggiorna la lista degli utenti connessi per tutti i client rimanenti.

2.2 Client

Il client si connette al server specificando l'indirizzo IP e la porta del server. Una volta connesso, il client richiede all'utente di inserire un nickname tramite una finestra di dialogo. Successivamente, viene avviata la GUI della chatroom. L'utente può digitare i messaggi nella casella di input e inviarli premendo il pulsante "Invia". I messaggi inviati

vengono quindi codificati e inviati al server tramite il socket. Il client riceve i messaggi dal server e li visualizza nella finestra di chat.

3 Esempio di utilizzo

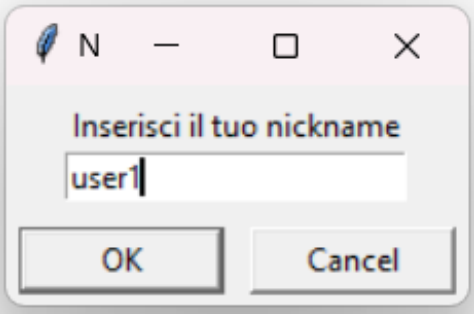

Figura 1: Registrazione del nickname

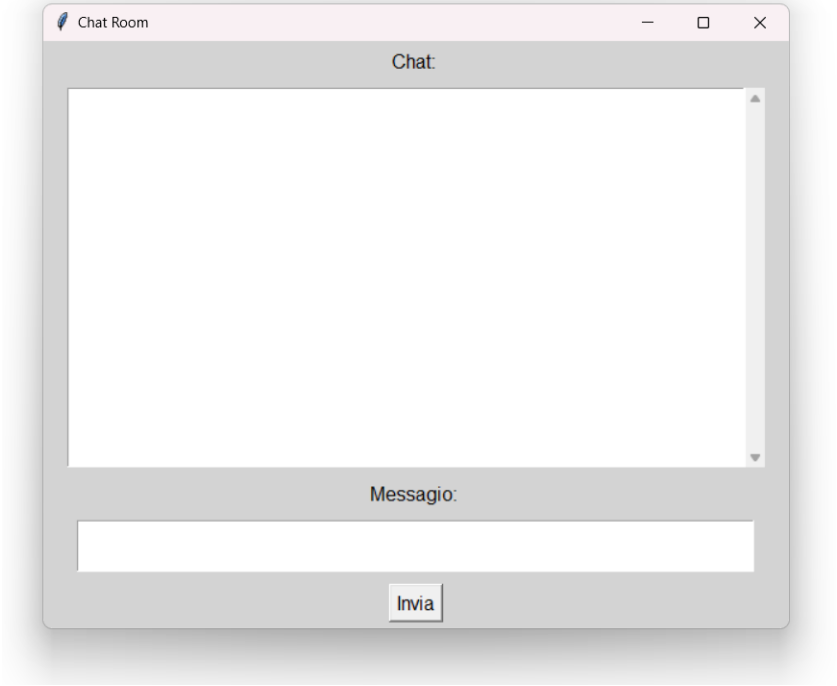

Figura 2: Entrata chat

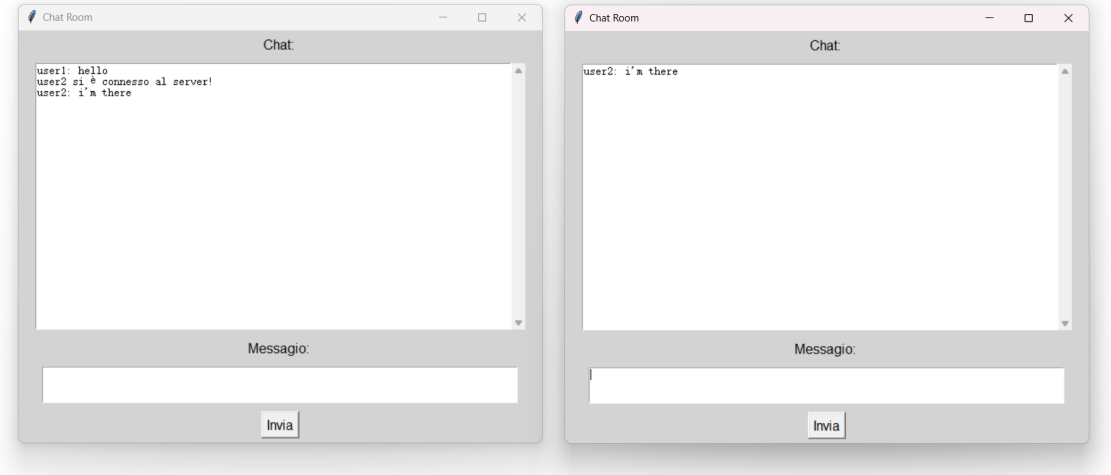

Figura 3: Entrata di un altro utente

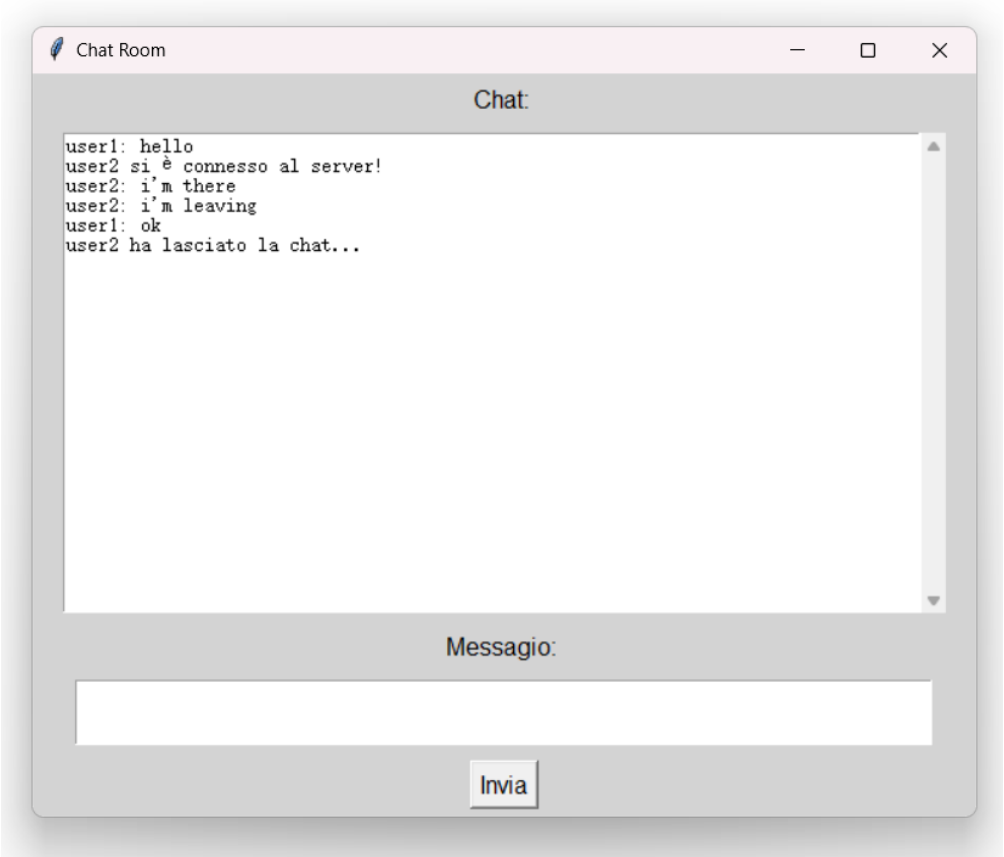

Figura 4: Uscita di un utente

4 Esecuzione del codice

4.1 Server

Il server è avviato utilizzando il comando:

```
python .\server
```

Questo comando avvia il file del server Python, che si mette in ascolto dei client che cercano di connettersi. Il server utilizza il socket con il protocollo TCP/IP per la comunicazione. Il server accetta le connessioni dei client, gestisce i loro nickname e gestisce la trasmissione dei messaggi tra i client connessi.

4.2 Client

Il client è avviato dopo il server utilizzando il comando:

```
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. L'utente inserisce queste informazioni tramite l'input da tastiera.

Una volta che il client si connette con successo al server, può inviare messaggi nella chat e ricevere messaggi dagli altri client connessi. La comunicazione tra client e server avviene tramite socket TCP/IP.

Il client fornisce un'interfaccia grafica utilizzando il modulo Tkinter di Python, che include una casella di testo per l'inserimento dei messaggi e un'area per la visualizzazione dei messaggi della chat.

5 Commenti finali

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