

Progetto Laboratorio Programmazione di Reti

Traccia 1 - Sistema di Chat Client - Server

Marilia Merendi - matricola 0001071190
marilia.merendi@studio.unibo.it

Funzionamento del sistema

Il sistema opera tramite la comunicazione via **socket TCP** tra un server e molteplici client. Ogni client è rappresentato da un **thread** indipendente, pertanto si tratta di un sistema multi-threading.

La porta del server è la well-known port 53000.

L'indirizzo del server è 127.0.0.1 (localhost). Il sistema, dunque, funziona in locale.

La **codifica** utilizzata per il trasferimento dei messaggi è UTF8.

Lato Server

Il server dispone di due dizionari:

- `clients[]` : mantiene l'elenco di tutti i client connessi.
- `addresses[]` : mantiene l'elenco degli indirizzi dei client connessi.

Il server opera tramite tre funzioni principali:

- **`accept_incoming_connections()`**
si occupa di gestire le connessioni in entrata e di dare un messaggio di benvenuto allo user. Aggiunge il client al dizionario `clients[]`.
- **`manage_clients()`**
recepisce il nome dello user e lo informa sui comandi da utilizzare per uscire dalla chat. Informa inoltre tutti gli altri user del suo ingresso. Tramite un loop, gestisce i messaggi inviati dallo user e, qualora desiderasse uscire dalla chat, si occupa di chiudere il suo thread e informare gli altri user della sua uscita.
- **`broadcast()`**
è la funzione che permette al server di inviare messaggi a tutti gli user connessi, sfruttando `clients[]`.

Lato Client

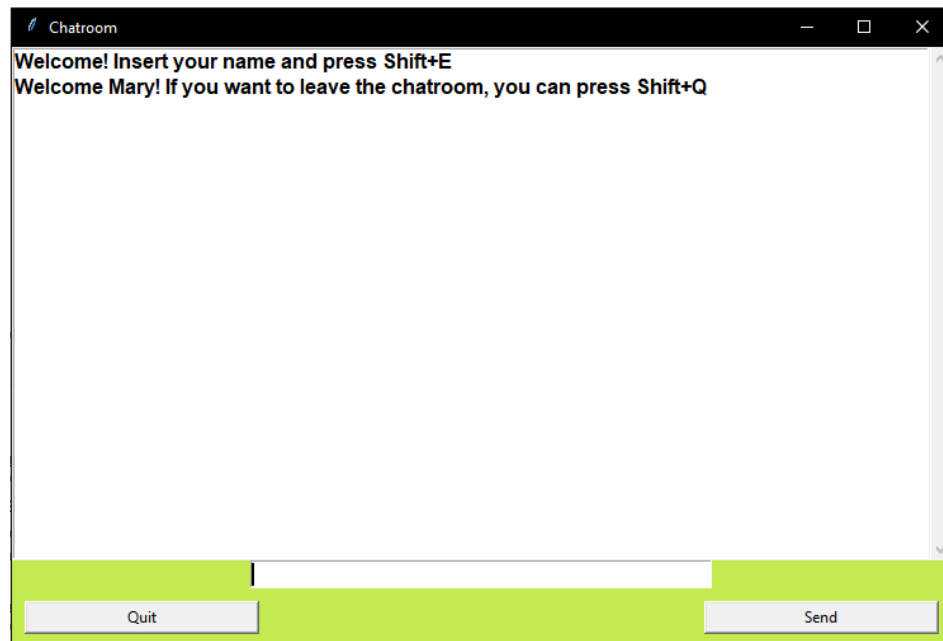


Fig 1: GUI della chatroom

La **GUI** è sviluppata tramite la libreria tkinter. Si presenta come un'unica finestra minimale.

Elementi della GUI:

- una **barra per l'input** (StringVar di tkinter), in basso al centro
- una **barra di scorrimento** (Scrollbar), sulla destra
- una **listbox** contenente i messaggi della chat
- un pulsante per uscire dalla chatroom (**Quit**)
- un pulsante per inviare il messaggio (**Send**)

Il client opera tramite tre funzioni principali:

- **receive_message()**
una loop function che si mette in ascolto in attesa di messaggi in arrivo sul socket da parte del server. Una volta recepito e decodificato il messaggio, esso è aggiunto alla message list visibile a schermo sulla finestra.
In caso di errori di sistema, il ciclo si interrompe.
- **send_message()**
Il messaggio è estratto dalla stringa di input dell'interfaccia (StringVar) e inviato tramite il socket al server.

Qualora il messaggio contenesse la keyword per uscire dalla chat (qui impostata a Q), chiude il socket del client e chiude la finestra.

- **on_closing():**

funzione di utility per il binding del pulsante di Quit e della chiusura della finestra (pulsante X sulla GUI) con l'uscita dalla chatroom.

Requisiti per eseguire il codice

Per il corretto funzionamento del programma, va innanzitutto lanciato il server.

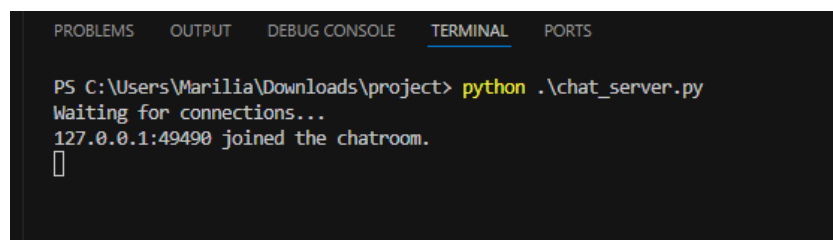
Da terminale: **python .\chat_server.py**

Successivamente, si possono lanciare i client.

Da terminale: **python .\chat_client.py**

Una volta lanciato un client è necessario inserire tramite terminale l'indirizzo del server (**127.0.0.1**) e, opzionalmente, il suo numero di porta (**53000**, il default qualora non venisse inserito nulla).

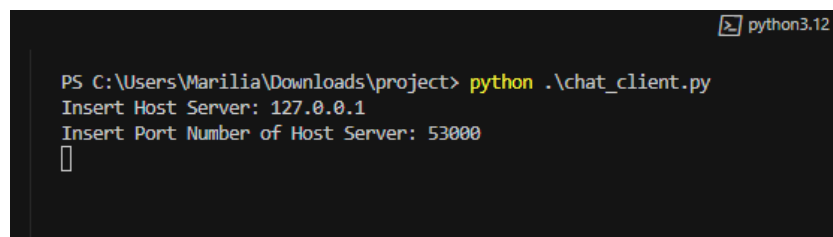
Successivamente bisogna comunicare il proprio **nickname** al server tramite la chat. Una volta fatto ciò, sarà possibile inviare messaggi e uscire dalla chatroom.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Marilia\Downloads\project> python .\chat_server.py
Waiting for connections...
127.0.0.1:49490 joined the chatroom.
█
```

Fig 2: esempio di lancio del server
(si vede già la connessione di un client)



```
python3.12

PS C:\Users\Marilia\Downloads\project> python .\chat_client.py
Insert Host Server: 127.0.0.1
Insert Port Number of Host Server: 53000
█
```

Fig 3: esempio di lancio di un client e connessione al server

Comandi:

- **Enter** (o pulsante **Send**) per inviare il messaggio
- **Shift+Q** (o pulsante **Quit**) per uscire dalla chatroom

Considerazioni aggiuntive

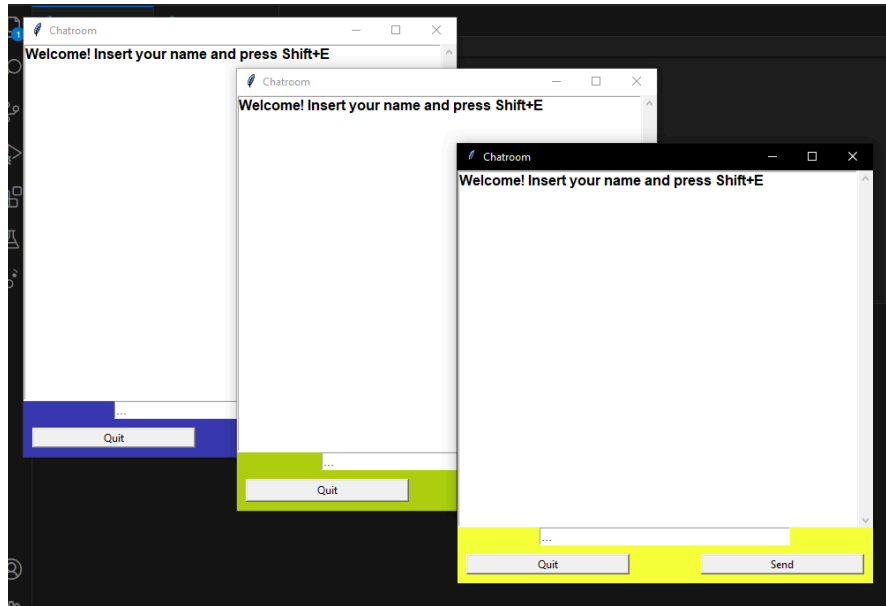


Fig 4: esempi di finestre di più client

Ogni finestra ha un colore di sfondo differente, associato a un numero in esadecimale generato in modo randomico. Serve puramente a rendere più semplici da distinguere i vari user.