PROGETTO PROGRAMMAZIONE DI RETI: CHAT CLIENT-SERVER MULTITHREADED

Come traccia ho scelto la traccia 1, il progetto consiste quindi in due script Python.
Durante lo sviluppo degli script è stata utilizzata la versione 3.11.7 di Python
IMPORTANTE: Il server una volta avviato è raggiungibile all'indirizzo 127.0.0.1:1100 e il client, di conseguenza, esegue la richiesta direttamente a quell'indirizzo.

LATO SERVER

Lo script "CHAT_SERVER.py" serve per avviare il server e per metterlo in ascolto in attesa di possibili richieste di connessione da parte dei client.
Per ogni richiesta di connessione il server avvia un thread che si occuperà di gestire tutto ciò che riguarda il client che gli viene assegnato in termini di connessione e gestione dei messaggi.
Il server inoltre tiene traccia a terminale di tutte le interazioni in termini di connessione con i client creando man mano così un real-time log a terminale.

Come avviare il server:
Per avviare il server basterà lanciare lo script in questo modo:
C:\Users\Lorenzo\Desktop\PROGETTORETI>python CHAT_SERVER.py
Una volta lanciato lo script, se il server si avvierà correttamente la console apparirà così:

```
C:\Users\Lorenzo\Desktop\PROGETTORETI>python CHAT_SERVER.py
Ready to serve on 127.0.0.1:1100.
Waiting for connections...
```

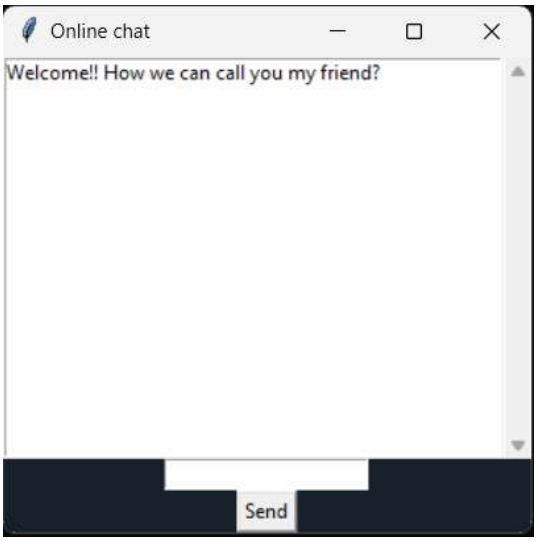
Per ogni client che tenterà di aprire una connessione con il server, se la richiesta va a buon fine, a terminale potremo vedere questi messaggi:

```
Accepted connection request from: 127.0.0.1:49982.
127.0.0.1:49982 connection complete.
```

In questo caso il client 127.0.0.1:49982 ha richiesto al server di connettersi e ci è riuscito.

LATO CLIENT

Lo script "CHAT_CLIENT.py" in primis si occupa di inviare una richiesta di connessione all'indirizzo del server e una volta connesso avvia una GUI in cui l'utente può inviare messaggi sulla chat gestita dal server.
Come avviare il client:
Per avviare i client basterà lanciare lo script in questo modo:
C:\Users\Lorenzo\Desktop\PROGETTORETI>python CHAT_CLIENT.py
Una volta lanciato lo script il client eseguirà una richiesta di connessione al server.
Se la richiesta andrà a buon fine allora si aprirà la GUI per chattare.



Cosa fare una volta aperta la GUI:
Il primo passo, una volta ricevuto il saluto da parte del server, sarà inviare come primo messaggio il nickname (scrivendolo nell'apposita casella di testo) con cui vogliamo essere identificati all'interno della chat.

Una volta inviato il nostro nickname possiamo iniziare a chattare liberamente con gli altri client.

Arrivati a questa fase l'utente finale può anche inviare dei comandi speciali al server.

I comandi sono:	
!end	Utilizzabile dall'utente per terminare correttamente la connessione con il server.
!close	Corrisponde ad una alternativa di !end. Può essere utile nel caso il server diventi irraggiungibile per chiudere la connessione lato client e chiudere anche l'applicazione.

GESTIONE DELLE ECCEZIONI LATO SERVER

1) Il server non riesce ad avviarsi:

```
C:\Users\Lorenzo\Desktop\PROGETTORETI>python CHAT_SERVER.py
ERROR: Failed to start up the server: OSError(10048, 'Di norma è consentito un solo utilizzo di ogni indirizzo di socket
 (protocollo/indirizzo di rete/parta)', None, 10048, None)
```

In questo caso viene semplicemente catturata l'eccezione e mostrato a video il messaggio di errore.

2) Il client chiude la connessione forzatamente dopo aver ricevuto il nickname.
Ad esempio utilizzando !close, chiudendo la GUI oppure per un eventuale crash.

```
127.0.0.1:49999 may have crashed or closed the connection prematurely.
ConnectionResetError(10054, "Connessione in corso interrotta forzatamente dall'host remoto", None, 10054, None)
Removing 127.0.0.1:49999 from active clients registers...
Closing death client socket...
Connection closed successfully.
```

In questo caso viene scritto a terminale l'avviso che un client ha terminato in maniera inaspettata la connessione.
Il server di conseguenza rimuove le informazioni del client dai suoi registri interni, chiude il socket inerente a quel client e avvisa di aver chiuso la connessione correttamente.

3) Il client chiude la connessione forzatamente prima di aver inviato al server il proprio nickname.
Cioè, quando la GUI viene chiusa prima aver inviato il messaggio di inerente al nickname (completando di conseguenza la registrazione del client nei registri interni del server).

```
127.0.0.1:50147 may have crashed or closed the connection prematurely.
ConnectionResetError(10054, "Connessione in corso interrotta forzatamente dall'host remoto", None, 10054, None)
Removing 127.0.0.1:50147 from active clients registers...
The connection was dropped before the client was registered in the active clients registers.
Closing death client socket...
Connection closed successfully.
```

In questo caso il server tenta di rimuovere le informazioni del client dai suoi registri interni ma essendo che la connessione è stata interrotta prima di riuscire a compilarli questo genererebbe una eccezione.
Di conseguenza il server si accorge di questa cosa e semplicemente gestisce l'eccezione chiudendo il socket rimasto in sospeso generando comunque il log corrispondente a ciò che è successo.

4) Chiusura della connessione in maniera corretta da parte del client (utilizzando !end):

```
Received connection closing request from 127.0.0.1:50506
Sending connection closure confirmation to 127.0.0.1:50506
Removing 127.0.0.1:50506 from active clients registers...
Closing 127.0.0.1:50506 reserved socket
Connection with 127.0.0.1:50506 closed successfully.
```

Il server riceve la richiesta da parte del client di chiudere la connessione, quindi invia al client la conferma della chiusura (inviandogli un !end in risposta) per poi chiudere il socket riservato a quel client.

Eccezioni più generiche lato server:

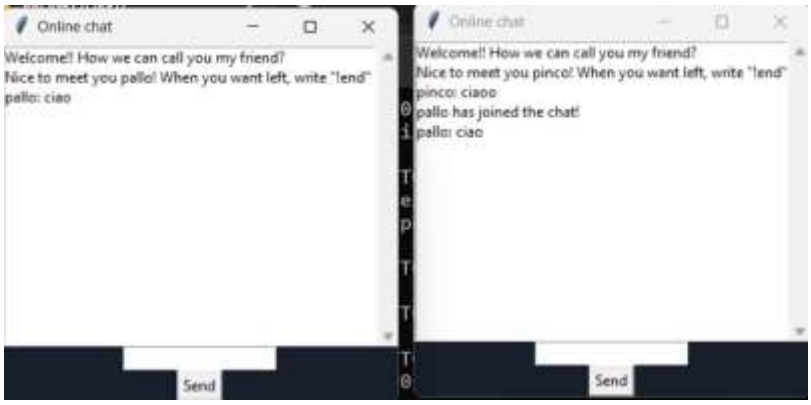
- Nel caso che la funzione di broadcast non riesca a notificare il messaggio a tutti i client connessi viene scritto dal server sul terminale l'errore avvenuto.
- Nel caso il server riceva una richiesta di connessione correttamente ma per qualche motivo poi non riesca a soddisfarla scrive a terminale l'errore avvenuto.

ECCEZIONI LATO CLIENT

- Nel caso che il client non riesca a connettersi al server viene scritto a terminale l'errore avvenuto.
- Nel caso il server vada offline e venga generata una eccezione sul thread incaricato di ricevere i messaggi da parte del server l'eccezione viene ignorata e il thread ricevitore viene chiuso.
- Nel caso durante la connessione già attiva tra client e server il server vada offline, per ogni messaggio che viene inviato dal client in chat apparirà in locale "**Server not reachable...**". (In questo caso la cosa ottimale sarebbe utilizzare il comando **!close**).

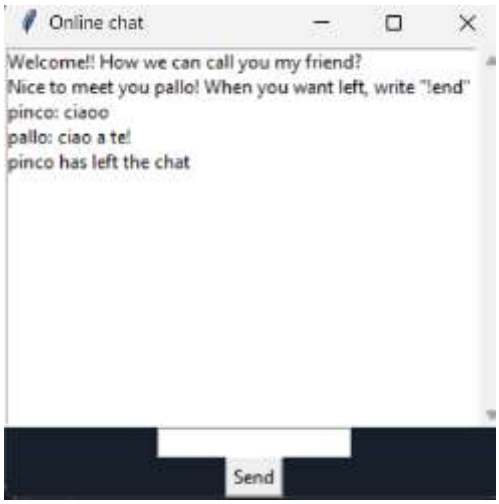
FUNZIONAMENTO NORMALE DELLA CHAT

CHAT TRA CLIENT CONNESSI CONTEMPORANEAMENTE



In uno scenario di funzionamento normale senza incidenti o eccezioni, più client possono connettersi al server e chattare tra loro senza alcun problema.
Il server avvierà un thread di gestione che si occuperà di ogni nuovo client connesso.
Quando un nuovo client si connette alla chat il server lo notificherà agli altri client connessi via chat.

ABBANDONO DELLA CHAT DA PARTE DI UN CLIENT



Quando uno dei client connessi alla chat chiude la connessione il server lo notifica agli altri.
In questo caso l'utente pinco ha chiuso la connessione al server, abbandonando la chat.