PROGETTO PROGRAMMAZIONE DI RETI 2024

E-mail dei componenti:
- giacomo.arienti2@studio.unibo.it
- gianmarco.fabbri3@studio.unibo.it
- kevin.shimaj@studio.unibo.it

Obiettivo del progetto:

L'obiettivo del progetto è quello di realizzare un sistema di chat client-server in linguaggio Python, utilizzando socket-programming.
Il gruppo si impegna nella realizzazione di un codice efficiente e ben commentato che garantisca la corretta gestione degli errori.

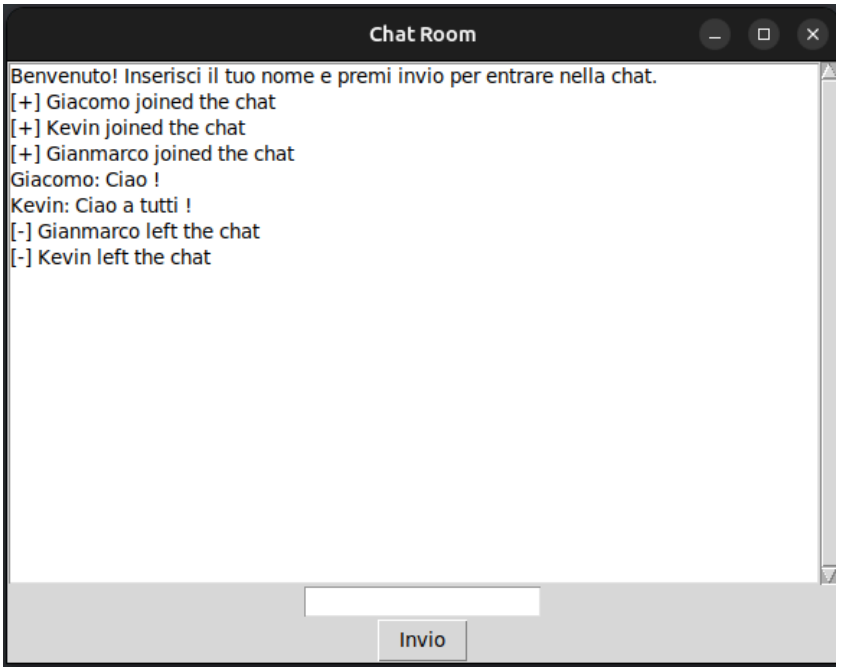
Funzionamento del sistema:

Il protocollo su cui sono basate le connessioni è quello TCP.
Il codice si compone di due script rappresentanti gli enti che operano all'interno dell'architettura client-server:

1. client.py, il client si interfaccia al sistema tramite una apposita interfaccia grafica che permette al soggetto di scegliere il proprio nickname all'apertura del client, inviare e ricevere messaggi testuali e di visualizzarli nell'apposita chat room insieme ai "log"(ingresso e uscita dalla chat-room dei vari utenti).
2. server.py, il compito del server è quello di accettare nuove connessioni, gestire le connessioni instaurate, eseguire la trasmissione broadcast dei messaggi a tutti i client connessi.

Implementazione dettagliata client:

Il sistema si compone di due thread, il principale dove viene eseguita la GUI, uno secondario che contiene il thread "daemon" che si occupa della ricezione dei messaggi da parte del server.



Implementazione dettagliata server:

Il server è stato realizzato tramite i socket non-blocking, i quali permettono di gestire molteplici connessioni contemporanee all'interno dello stesso thread. Le chiamate bloccanti come "receive" e "sendall" vengono effettuate in maniera asincrona solamente quando lo stato dei file descriptor lo permette.

Per ottenere lo stato delle connessioni attive abbiamo utilizzato la system call messa a disposizione dal sistema operativo select. Fonte: https://man7.org/linux/man-pages/man2/select.2.html

```
# array containing all the active connections and the
# binded socket
connections = list(self.connections.keys()) + [self.socket]
readable, _, _ = select.select(connections, [], [])

# loop trough readable sockets
for sock in readable:
    # if socket is the binded socket, accept incoming conn
    if sock == self.socket:
        self.accept_connection()
    # receive data from existing connection
    else:
        self.receive_data(sock)
```

Requisiti per poter eseguire il codice:

Insieme alle librerie messe a disposizione da python, è necessario installare la libreria tkinter, la quale viene utilizzata per la GUI del client.

Il client può essere eseguito da terminale tramite il seguente comando:
python3 client.py [SERVER_PORT]
In maniera analoga può essere eseguito il server:
python3 server.py [SERVER_PORT]
L'ultimo argomento è opzionale per cui se non specificato verrà usata la porta di default (49152).