Documentazione Elaborato

Programmazione Di Reti

CHAT CLIENT-SERVER

Alessandro Ricci

matricola: 0001043909

mail: alessandro.ricci47@studio.unibo.it

Sommario

Funzionamento del sistema

Il sistema consente di effettuare comunicazioni Client-Server in tempo reale e in parallelo, con capacità di gestire molteplici Client in contemporanea, per realizzare una chat di gruppo tra più utenti. Per permettere questo, entrambe le entità (sia Server che Client) utilizzano i Socket, per i quali si è deciso di utilizzare il protocollo TCP.

Gli utenti possono connettersi al Server tramite il suo indirizzo (rappresentato dal nome e dalla porta), entrando così nella relativa stanza.

Una volta all'interno, ogni Client può inviare dei messaggi al Server, il quale, di conseguenza, si occupa di inviarlo a sua volta a tutti i Client connessi, in modo che tutti siano costantemente aggiornati e allineati allo stesso modo.

I Client possono poi disconnettersi in qualsiasi momento, lasciando quindi la chat di gruppo. In tal caso, il Server provvede ad aggiornare i Client rimanenti, notificando l'avvenuta disconnessione di quel determinato utente.

Di seguito sono elencate nel dettaglio le specifiche relative al Server e al Client.

Server

Il Server è implementato nel file "`ChatRoom_Server.py`".

Come prima operazione, il Server crea un singolo Socket che si mette in attesa di connessioni esterne, attraverso un indirizzo composto da un nome (localhost) e una specifica porta (53000). Viene poi avviato un Thread a cui viene associata la funzione che fisicamente esamina le richieste di connessioni all'interno di un loop infinito.

Quando un Client tenta di connettersi, attraverso il medesimo indirizzo precedentemente indicato, il Thread accoglie la richiesta e crea l'istanza locale rappresentativa del Client. Dopodiché, procede ad avviare un altro Thread per ogni Client, ognuno dei quali si occuperà di gestire i singoli Client.

Appena un Client si connette con successo, il Server gli invia un messaggio privato di benvenuto, invitando il Client a fornire, come primo messaggio, il proprio nome utente. Ricevuto questo dato, il Server aggiunge il Client ad un dizionario locale, che ha il duplice obbiettivo di tenere traccia di tutti i Client connessi e di consentire l'accesso in qualsiasi momento ad uno specifico Client.

Il Client, dopo aver fornito il proprio nome, può quindi iniziare ad inviare messaggi nella ChatRoom. Quando un messaggio viene inviato al Server da un singolo Client, il Server si occupa di inviarlo a tutti i Client connessi in quel momento (broadcast), compreso lo stesso da cui origina il messaggio, creando quindi un'effettiva chat globale, in cui tutti possono vedere i messaggi di tutti, ciascuno con il nome utente di chi l'ha inviato come prefisso.

Se la connessione con un Client viene persa per qualsiasi motivo, volontariamente (l'utente ha deciso di disconnettersi) o involontariamente (la connessione con l'utente è caduta per un imprevisto), il Server provvede a notificare l'avvenuta disconnessione di quel determinato utente a tutti gli altri Client (gestendo dunque anche le relative eccezioni). Inoltre, subito dopo il Server cancella quel Client dal proprio dizionario locale, per evitare di inviare i messaggi successivi ai Client che non sono più connessi.

Client

Il Client è implementato nel file "`ChatRoom_Client.py`".

All'avvio del programma del Client, viene visualizzata una finestra realizzata mediante la libreria GUI "TKinter" di Python. All'utente che utilizza il Client, vengono prima mostrati due campi che contengono rispettivamente il nome del Server e la porta a cui si desidera collegarsi.

Premendo poi il pulsante "Connetti", è possibile utilizzare i valori impostati nei precedenti campi per creare un indirizzo e tentare la connessione verso il Server. Se la connessione va a buon fine, la pagina si chiude e lascia spazio alla ChatRoom effettiva. Altrimenti, viene visualizzato un messaggio di errore, sotto al pulsante, che spiega la natura del problema avvenuto.

Nella pagina della ChatRoom, è presente una lista che conterrà, in cascata, i messaggi che tutti i Client invieranno dopo che l'utente corrente si è connesso. È inoltre presente una casella di testo per scrivere i propri messaggi, un pulsante "Invia Messaggio" per mandarli al Server e un altro pulsante "Disconnetti" per uscire dalla stanza e quindi dall'applicazione (quest'ultimo ha lo stesso effetto di chiudere la finestra tramite click sulla croce in alto a destra).

Dopo aver inserito il proprio nome utente, come richiesto dal Server, si può procedere ad inviare i propri messaggi, mentre si visualizzeranno in tempo reale quelli inviati da tutti gli altri utenti connessi.

Requisiti per eseguire il codice

Entrambi gli script, sia il Server che il Client, possono essere eseguiti dalla riga di comando, avvalendosi dell'interprete Python.

È sufficiente attenersi ai seguenti passaggi:

1. Verificare di aver installato Python sulla propria macchina
2. Posizionarsi sulla cartella principale del progetto
3. Aprire un terminale in questa cartella, quindi eseguire il comando "`py ChatRoom_Server.py`" per avviare il Server
4. Aprire un altro terminale, quindi eseguire il comando "`py ChatRoom_Client.py`" per avviare un'istanza di Client, che apre la finestra GUI con la possibilità di connettersi al Server
5. A questo punto, per simulare la presenza di più Client, è possibile aprire un qualsiasi altro numero di terminali, e in ognuno digitare il medesimo comando "`py ChatRoom_Client.py`", che consente di avviare altrettante istanze di Client assestanti
6. Utilizzare le varie finestre aperte dei Client per testare il sistema, connettendosi prima al Server e andando poi ad indicare un nome utente specifico per ogni Client
7. Iniziare la Chat di gruppo e verificare il corretto funzionamento del sistema
8. Provare a disconnettersi con qualche Client per verificare che tutti gli altri Client connessi vengano notificati di tale evento

Considerazioni aggiuntive

Gestione eccezioni

Sia il Server che il Client implementano la gestione delle eccezioni, soprattutto riguardanti i possibili problemi di connessione, in ogni punto in cui si è ritenuto fosse necessario.

Tuttavia, poiché non di fondamentale importanza, la messaggistica dettagliata associata alle eccezioni non è impostata in ogni occorrenza delle stesse. Inoltre, per semplicità, sono state spesso utilizzate eccezioni più generiche invece di un elenco di più specifiche, ad esempio usando la classe “Exception” anziché gestire i singoli casi di “OSError” o “ConnectionError”.

In ogni caso, questo garantisce comunque la corretta gestione delle possibili eccezioni, soprattutto riguardo alla possibilità del Server di non andare in errore in caso di disconnessioni improvvise da parte dei Client, assicurando quindi il funzionamento del servizio anche in questi casi.