Relazione Tecnica sul Funzionamento del Codice

Nome Cognome

24 maggio 2024

Indice

1 Introduzione

Il codice implementa una semplice chat room utilizzando il linguaggio Python e la libreria Tkinter per l'interfaccia grafica del client.

2 Funzionamento del Codice

2.1 Client (`client.py`)

Il file `client.py` definisce una classe `ChatClient` che estende la classe `Tk` di Tkinter per creare un'applicazione grafica per il client della chat. Di seguito sono descritti i principali componenti del file:

2.1.1 Importazione delle Librerie

```
import tkinter as tk
from tkinter import scrolledtext
import socket
import threading
import sys
```

2.1.2 Classe `ChatClient`

`ChatClient` è una classe che rappresenta il client della chat. Questa classe eredita da `tk.Tk` e gestisce l'interfaccia grafica, l'invio e la ricezione dei messaggi tramite socket.

2.1.3 Metodi Principali della Classe `ChatClient`

- `send_message`: Metodo che permette di inviare un messaggio al server, invocato quando viene premuto il pulsante `send` oppure invio dovo aver scritto qualcosa nel campo di inserimento, il messaggio viene inviato codificato tramite l'utilizzo del metodo `encode`, viene gestito anche un eventuale errore della ricezione da parte del server.

- `handle_close`: Un metodo che gestisce la chiusura della socket tamite `client_socket.close()` e successivamente dopo aver atteso per mezzo secondo chiude la finestra della GUI.

- `receive_messages`: Metodo per la ricezione dei messaggi da parte del server, gestito da un thread dedicato, nel caso in cui viene ricevuto un

messaggio non vuoto viene inserito nella chat, nel caso in cui viene ricevuto un messaggio vuoto vuol dire che il server ha richiesto la chiusura della socket e quindi si esce dal ciclo e si avvia la procedura di chiusura, viene gestito anche l'errore in caso di server non raggiungibile.

2.1.4 Configurazione iniziale

Nella parte di codice subito dopo la classe avviene la configurazione della connessione e tutto ciò che è necessario per il funzionamento del client. Viene chiesto l'indirizzo ip del server a cui ci si vuole connettere, la porta su cui il server è in ascolto ed infine il nome che si vuole avere all'interno della chat room. Successivamente viene effettuata la connessione al server tramite la socket precedentemente creata, viene gestito anche un eventuale errore che indica che il server è irraggiungibile. Infine viene lanciata la `GUI`.

2.2 Server (`server.py`)

Il file `server.py` definisce il server della chat che gestisce le connessioni dei vari client, la ricezione e l'inoltro dei messaggi.

2.2.1 Importazione delle Librerie

```
import socket
import threading
```

2.2.2 Variabili e Funzioni Principali

- `clients`: Lista dei client connessi al server
- `BUFFER_SIZE`: LA dimensione del buffer di ricezione

2.2.3 Metodi principali

- `client_remove`: Metodo per la rimozione di client dalla lista e comunicazione a tutti i client connessi alla chat tramite boradcast e stampa nel terminale del server dell'avvenuta disconnessione.
- `handle_client`: Metodo per la gestione dei singoli client, gestito da un thread dedicato, implementa anche la ricezione dei messaggi dai singoli client, valuta se si tratta di messaggio vuoto (in questo caso si tratta di una comunicazione da parte del server di chiusura del socket) oppure

di un messaggio che va inoltrato a tutta la chat room e in questo caso viene fatta una broadcast del messaggio.

- `broadcast`: Invia a tutti i client connessi il messaggio specificato come argomento che gli viene passato.
- `serverStart`: È il metodo principale che serve per avviare il server, al suo interno vengono inizializzate le variabili che contengono l'indirizzo, la porta del server, successivamente viene messo in ascolto di richieste di connessione, quando un nuovo client vuole connettersi viene accettato vengono salvati nickname, socket e indirizzo, inoltre viene avviato un thread dedicato al client e viene comunicato alla chat room che un nuovo utente si è unito, infine viene gestita la chiusura del server tramite interrupt (`CTRL+C`) e quindi disconnettere tutti gli eventuali client connessi.

3 Requisiti per l'Esecuzione

Per eseguire l'applicazione è necessario aver installato `Python 3.x`, avere installata la libreria per la `GUI Tkinter` e quelle per la gestione dei thread e delle socket.

3.1 Istruzioni per l'Esecuzione

1. Avviare il server eseguendo il comando `python server.py`.
2. Avviare uno o più client eseguendo il comando `python client.py` su diverse istanze di terminale.
3. Chiudere il client chiudendo la finestra della chat room oppure tramite interrupt da terminale.
4. Chiudere il server tramite interrupt da terminale.

4 Considerazioni Finali

Il sistema di chat sviluppato permette una comunicazione semplice tra più client attraverso un server centrale. Il codice può essere migliorato aggiungendo funzionalità come l'autenticazione degli utenti, una migliore gestione delle disconnessioni e una gestione più efficace degli errori. Inoltre, l'interfaccia utente può essere migliorata per una migliore esperienza d'uso.