Relazione elaborato di progetto corso di Programmazione di Reti 2021

Salomone Marco

04/06/2021

Indice

1 Analisi **2**
1.1 Requisiti 2
1.1.1 Requisiti funzionali 2
1.1.2 Requisiti aggintivi 2

2 Design **3**
2.1 Client 3
2.2 GUI 3
2.3 Server 4

3 Organizzazione della codebase **6**

4 Guida all'utilizzo **7**
4.1 Avvio 7
4.2 Esecuzione 7

Capitolo 1

Analisi

Il progetto è stato sviluppato seguendo la traccia numero 3, che richiede di sviluppare un minigame testuale in cui si deve rispondere a una di 3 domande, e grazie ad un sistema a punti, viene decretato un vincitore.

1.1 Requisiti

1.1.1 Requisiti funzionali

- Partecipazione alla chat con un nome scelto dall'utente;
- Possibilità di vincere rispondendo correttamente alle domande;
- Possibilità di avere uno o più vincitori;
- Creazione di domande a trabocchetto che fanno perdere automaticamente il gioco;
- Timeout per la partita.

1.1.2 Requisiti aggiuntivi

- Creazione di una leaderboard aggiornata in tempo reale
- Divisione tra messaggi ricevuti da parte del server a tutti gli utenti e messaggi ricevuti solo dall'utente specifico

Capitolo 2

Design

Il design completo è mostrato nella figura Figura 2.2

2.1 Client

Il client inizializza 3 diversi socket: uno per la comunicazione riguardante il gioco, uno per la gestione di messaggi di broadcast da parte del server e uno per la gestione della leaderboard. Ho scelto di dividere queste funzionalità in 3 diversi socket in modo da gestire concorrentemente le diverse funzionalità dell'applicazione; se fosse stato usato un solo socket per tutta la comunicazione sarebbe stato necessario inviare i messaggi di broadcast e leaderboard alla conclusione di un turno di gioco. Ogni socket è quindi gestito da un proprio thread proprio per garantire la concorrenza. Si possono impostare due flag al lancio dell'applicazione che descrivono *hostname* e *porta* del server a cui collegarsi; per ulteriori informazioni è possibile utilizzare il flag *-h* dopo *chat-client.py*; i valori di default impostati sono *localhost* come 'hostname' e *53000* come 'porta'.

2.2 GUI

La GUI è suddivisa in diversi pannelli, ognuno indipendente dall'altro. In alto vengono mostrati il ruolo e l'attuale punteggio; al centro invece vi sono, a sinistra la schermata di gioco interattiva in cui vi è un campo di testo per inserire scelte e/o risposte ed un bottone per inviare le proprie risposte al server mentre a destra vengono mostrati i messaggi di broadcast. Il pannello in basso è dedicato alla leaderboard. L'architettura della GUI è mostrata nella figura Figura 2.1

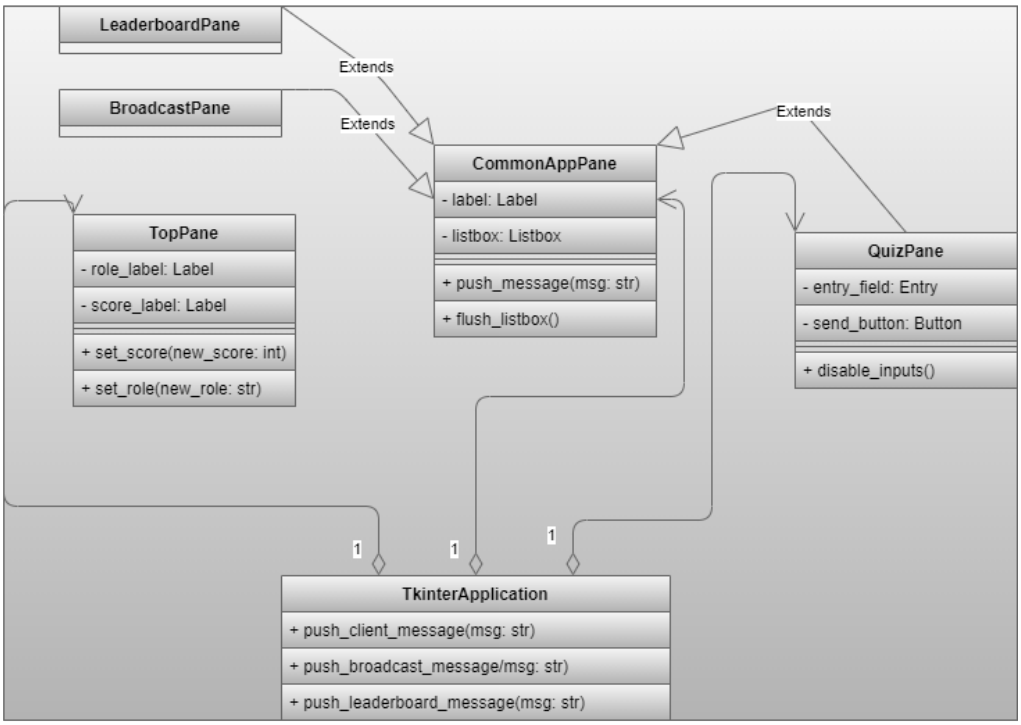

Figura 2.1: GUI

2.3 Server

Il server gestisce la comunicazione attraverso un thread per ricevere connessioni da cui poi viene avviato un thread per ogni socket che entra in comunicazione. Successivamente si attende la ricezione del nome dell'utente che può anche essere espresso dalle parole chiave 'BROADCAST' e 'LEADERBOARD' che rappresentano rispettivamente socket di broadcast e di leaderboard, i quali chiaramente non parteciperanno al gioco. Successivamente viene scelto un ruolo casuale da assegnare al player e successivamente inizia il game loop che coinvolge il client dove vengono 3 domande tra 50 riguardanti l'*Informatica*; una di queste 3 viene designata come domanda trabocchetto che, se scelta, elimina in automatico il giocatore. Viene istanziata una leaderboard che decreterà, dopo 2 minuti dalla creazione del server, il o i vincitori del gioco.

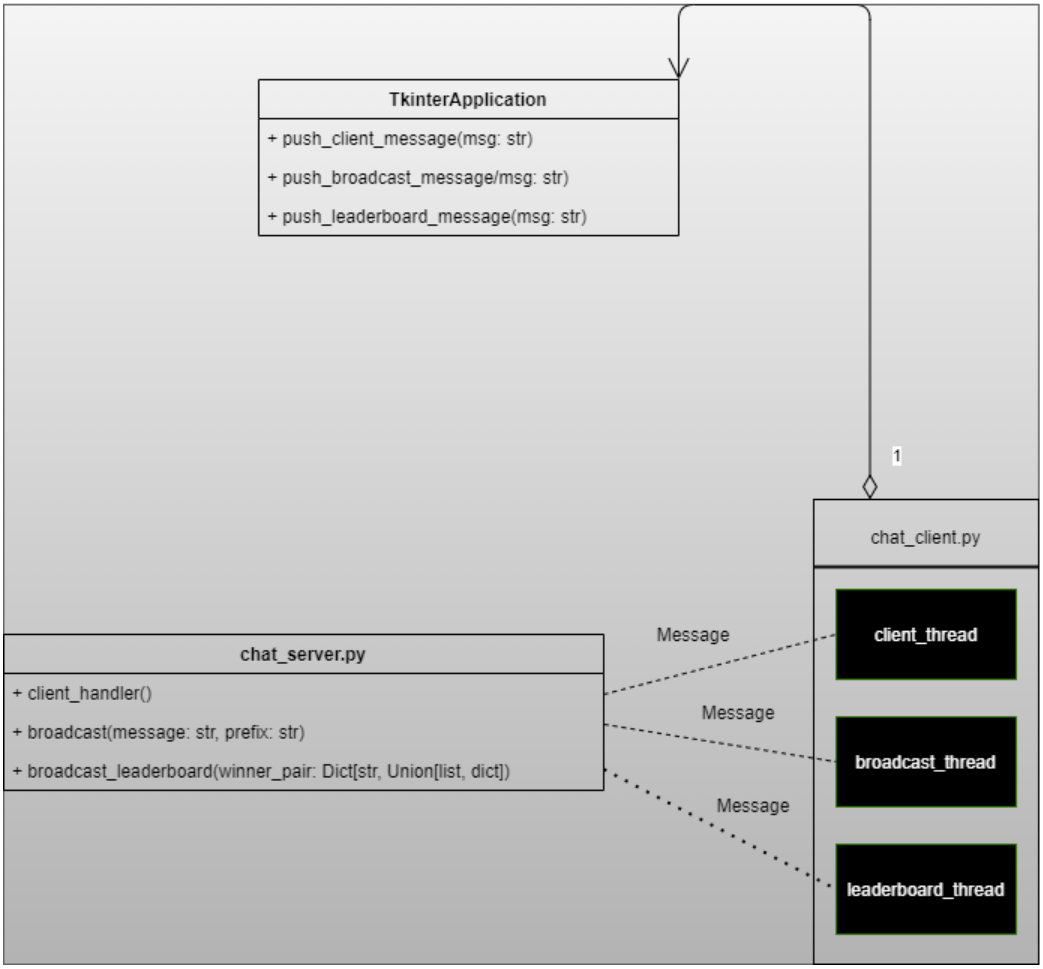

Figura 2.2: Architettura

Capitolo 3

Organizzazione della codebase

La codebase è stata suddivisa per funzionalità in modo da distinguere i tre diversi componenti dell'architettura: Client, Server e GUI. Il file *GUI.py* include tutti i pannelli per la gestione della GUI mostrata all'utente; ci sono utility incluse in *tkinterutils.py* che consentono di creare dei pane contententi delle *Listbox* e anche un metodo per configurare una griglia in Tkinter. Il file *chat_client.py* invece rappresenta il vero e proprio client in cui coestitono tutti i socket che risiedono in diversi thread; esiste inoltre un file denominato *client_utils.py* nel quale è definito un metodo per creare un socket, connetterlo ad un server e creare un thread che ne gestisce il funzionamento ed inoltre un metodo per leggere i messaggi riservati al client da parte del server. Il file *chat_server.py* invece gestisce tutta la parte dedicata al server, in cui è presente un thread per accettare le connessioni in entrata e un thread per ogni client.

Capitolo 4

Guida all'utilizzo

4.1 Avvio

- Avvia il server con *python chat_server.py*
- Avvia il client con *python chat_client.py [-host HOST] [-port PORT]*
  - Per ottenere un'aiuto *python chat_client.py -h*

4.2 Esecuzione

Premere il tasto *Send* o premere il tasto *Invio* nel campo di testo è equivalente

- Scrivi il tuo nome nel campo di testo
- Segui le istruzioni
- Se scegli una domanda trabocchetto perdi automaticamente
- Dopo 2 minuti dal lancio del server, il campo di testo e il bottono si disabilitano e il vincitore o i vincitori vengono mostrati con un popup
- A questo punto è possibile chiudere la finestra e/o il server