Relazione Progetto
di
Programmazione di Reti

Luca Rengo
Matricola: 0000936177

Giugno | Luglio 2021

Indice

Descrizione del Progetto

Introduzione

Il progetto consiste nello sviluppo di un programma che permetta la connessione TCP client-server, per svolgere un mini-quiz tra più utenti. [Traccia 3]

Descrizione

Server: è il tramite che instaura e permette la connessione tra gli utenti, giocatori. Si occupa di tutte le fasi di gioco.
Clients: Sono gli utenti giocatori. Interagiscono col server per poter giocare.

Il gioco comprende due fasi:

- Scelta tra 3 domande, di cui 1 a trabocchetto.
- Risposta alla domanda scelta.

Se l'utente avrà scelto la risposta giusta, otterrà un punto, altrimenti ne perderà uno. Invece se avrà scelto la domanda trabocchetto verrà eliminato.

Scopo del gioco: Totalizzare più punti dei propri avversari.

Requisiti

Requisiti funzionali

- Possibilità di connettersi al server.
- Possibilità di comunicare col server.
- Possibilità di rispondere alle domande.
- Avere un timer di gioco.
- Avere delle domande trabocchetto che fanno eliminare automaticamente il giocatore.

Requisiti aggiuntivi | opzionali

- Avere una classifica finale coi punteggi dei giocatori.
  - Incluso il numero di risposte corrette e sbagliate per ciascun giocatore.
- Possibilità di rigiocare una volta conclusa la partita.
- Possibilità di poter uscire senza bloccare il server e comunque lasciando la possibilità agli altri utenti di poter rigiocare.

Architettura e Design dell'Applicazione

Comunicazione Client-Server

- Gli utenti si connettono al server e scelgono un nome.
- Il server invia loro 3 domande (di cui 1 trabocchetto)
- I clienti scelgono una tra le tre domande.
  - Se la domanda scelta è quella a trabocchetto, allora verranno eliminati e disconnessi dal server.
  - Altrimenti proseguono col gioco.
- Ricevono dal server 2 possibili risposte alla domanda da loro scelta.
  - Se rispondono correttamente , il loro punteggio (inizialmente a zero) salirà di 1.
  - Se sbagliano il loro punteggio verrà decrementato di 1.
- Una volta risposto alla prima domanda (sia erratamente che correttamente) verrà fatto partire il timer.
  - Il timer durerà di default 30 secondi.
- Gli utenti continueranno, in questo modo, a ricevere domande e a dare risposte finchè il timer non terminerà.
- Una volta che il timer avrà terminato, verrà segnalato al server di smettere di inviare domande.
- Gli utenti , a questo punto, invieranno il loro punteggio al server.
- Il server così decreterà il vincitore e i perdenti oppure il pareggio nel caso che tutti abbiano ottenuto lo stesso punteggio.
- Dopodichè il Server invierà ai clients la classifica con i loro punteggi e quante risposte corrette e sbagliate hanno dato.
- Conclusa la partita, il server resterà in attesa di notizie dai clients, per sapere se vogliono rigiocare oppure se si sono disconnessi.

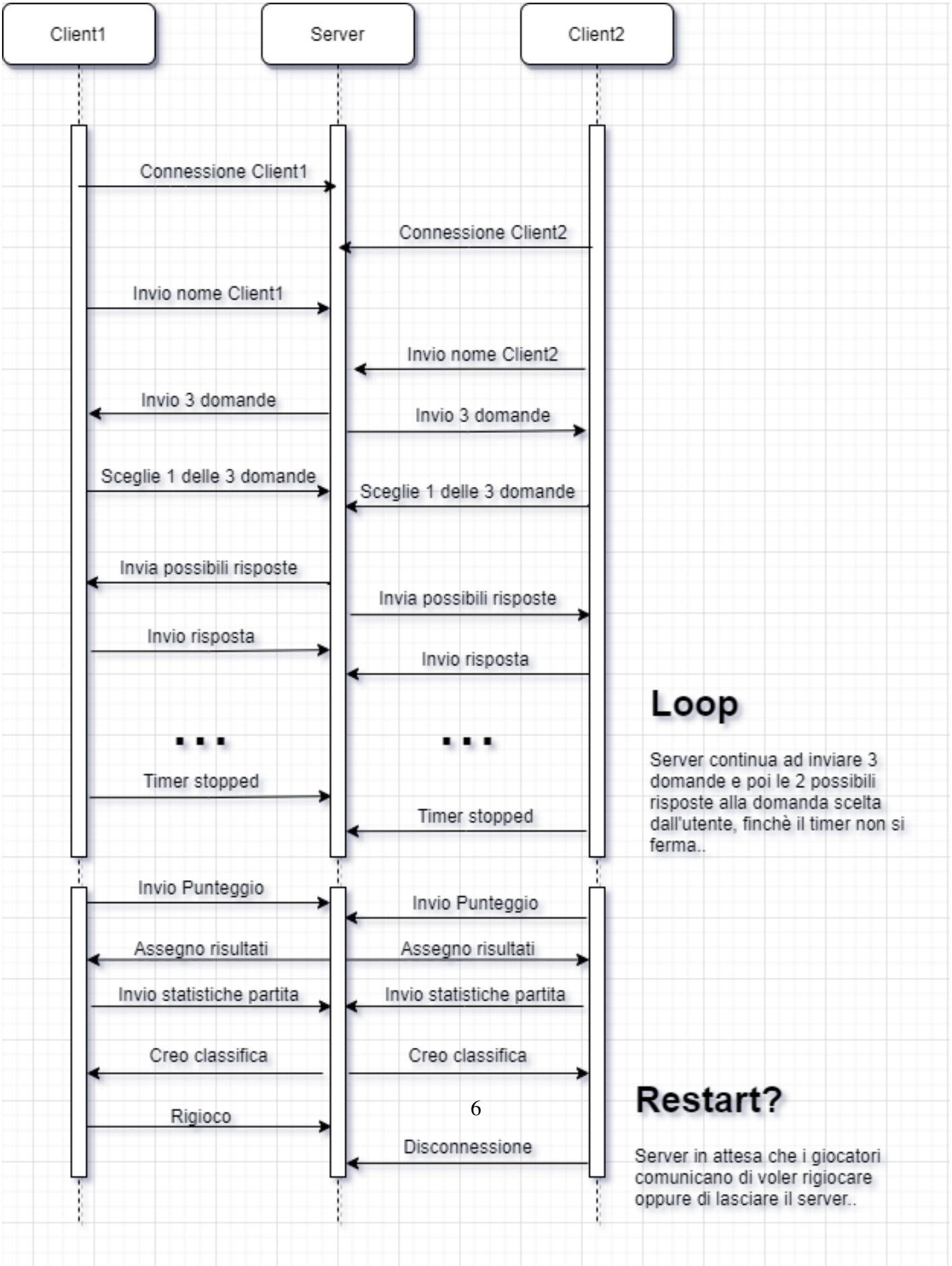


Figura 1: Communicazione Client-Server Sequence Diagram

Il Client

Il client si occupa di inizializzare un socket e di tutti gli aspetti che riguardano l'interfaccia grafica, in base alle direttive del server. Inoltre ogni client ha un proprio timer locale che parte in autonomia dal server, ma lo informa quando ha terminato.

UML Client

Funzioni del cliente

- **connect():**
  - Questa funzione che viene chiamata dal cliente, premendo il pulsante connect, permette la connessione con il server.
- **receive_channel(client):**
  - Questa è la funzione principale che prende come parametro un client (socket) e chiama la funzione recv_initial_questions(client)
- **recv_initial_questions(client):**
  - Questa funzione permette al cliente di ricevere le 3 domande (con 1 trabocchetto) dal server.
- **receive_questions(client):**
  - Questa funzione permette al client di ricevere le possibili risposte alla domanda da lui/lei scelta.
- **scelta(arg, client):**
  - Questa funzione che viene chiamata dal client dopo aver premuto uno dei due pulsanti di risposta alla domanda; aggiorna il punteggio del giocatore in base all'arg che corrisponde ad una risposta alla domanda e richiama recv_inital_questions(client) per poter farsi inviare dal server altre domande.
- **timer(counter_started, client):**
  - Questa funzione viene chiamata la prima volta che viene chiamata la funzione scelta(arg, client). Questa fa partire un timer di gioco di 30 secondi (di default) se il contatore (counter_started) non era già partito.
- **end_game(client):**

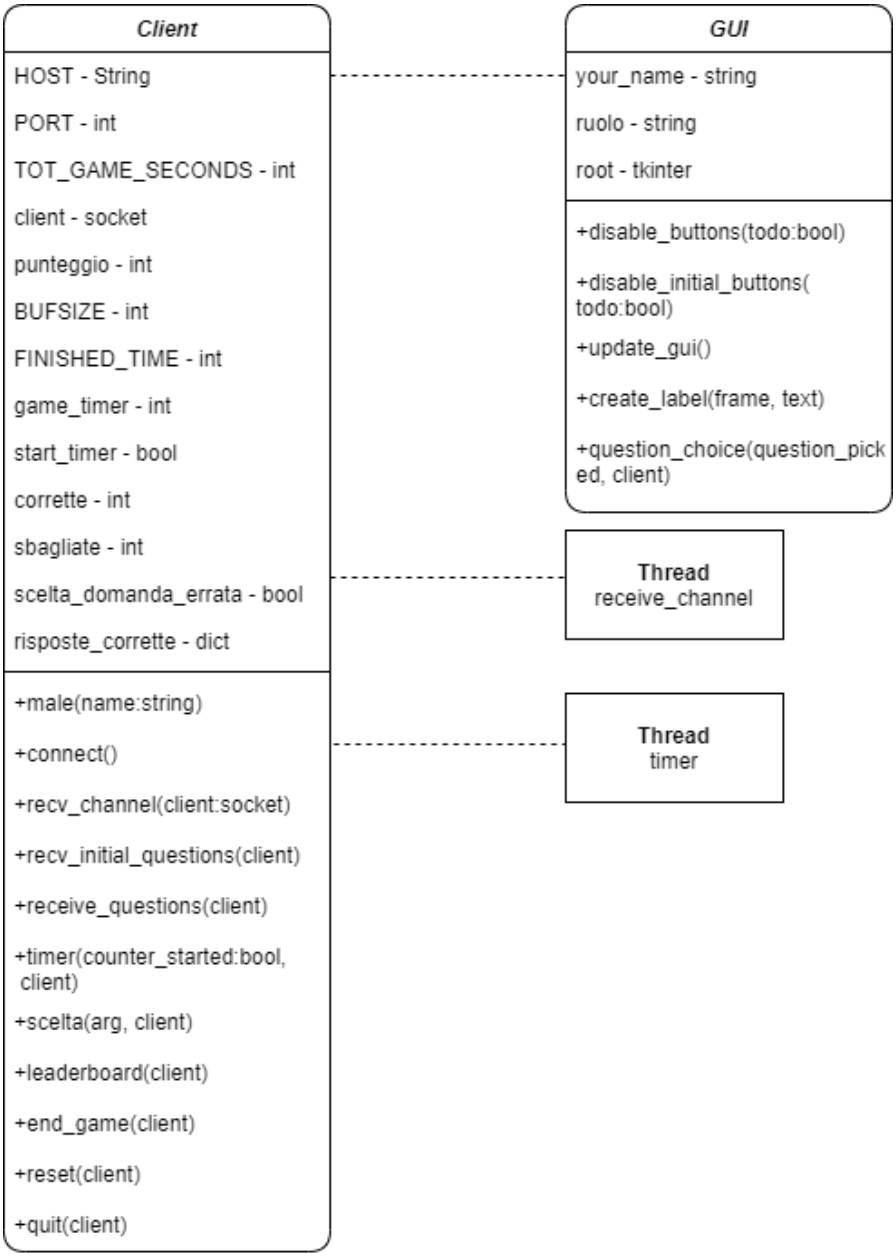


Figura 2: Client UML

  - Questa funzione invia al server le statistiche di gioco e riceve da esso la classifica con tutti i giocatori.

- **leaderboard(client):**
  - Questa funzione viene chiamata una volta che il timer si è fermato. Riceve dall'utente le statistiche riguardo alle risposte date corrette e sbagliate e crea la classifica di tutti i giocatori e la invia al client.

- **reset(client):**
  - Questa funzione resetta alcune variabili e liste per poter far ripartire il gioco e chiama receive_channel(client) per ricevere nuove domande dal server.

- **male(name):**
  - Questa funzione prende un name, ovvero una stringa e restituisce **True** se il nome passato come argomento è maschile altrimenti **False** se è femminile.

- **quit(client):**
  - Questa funzione notifica il server che il client sta per uscire e poi termina il programma.

Funzioni del cliente che riguardano la GUI (Interfaccia grafica)

- **disable_initial_buttons(todo)**
  - Questa funzione abilita/disabilita i 3 pulsanti delle 3 domande in base al valore di todo (bool).

- **disable_buttons(todo)**
  - Questa funziona, come quella sopra, abilita/disabilita i 2 pulsanti delle 2 possibili risposte alla domanda scelta dall'utente in base al valore di todo (bool).

- **update_gui()**
  - Questa funzione crea alcuni frames e aggiorna il testo di alcuni labels (quello del nome e quello del ruolo).

- **create_label(frame, text)**

  - Questa funzione crea un label e viene usata per la creazione della leaderboard (classifica) visto che il numero di giocatori non è conosciuto a priori.

- **question_choice(question_picked, client)**
  - Questa funzione valuta quale domanda abbia scelto l'utente e se era la domanda trabocchetto allora lo elimina dal gioco.

Ordine delle Chiamate alle Funzioni del Cliente

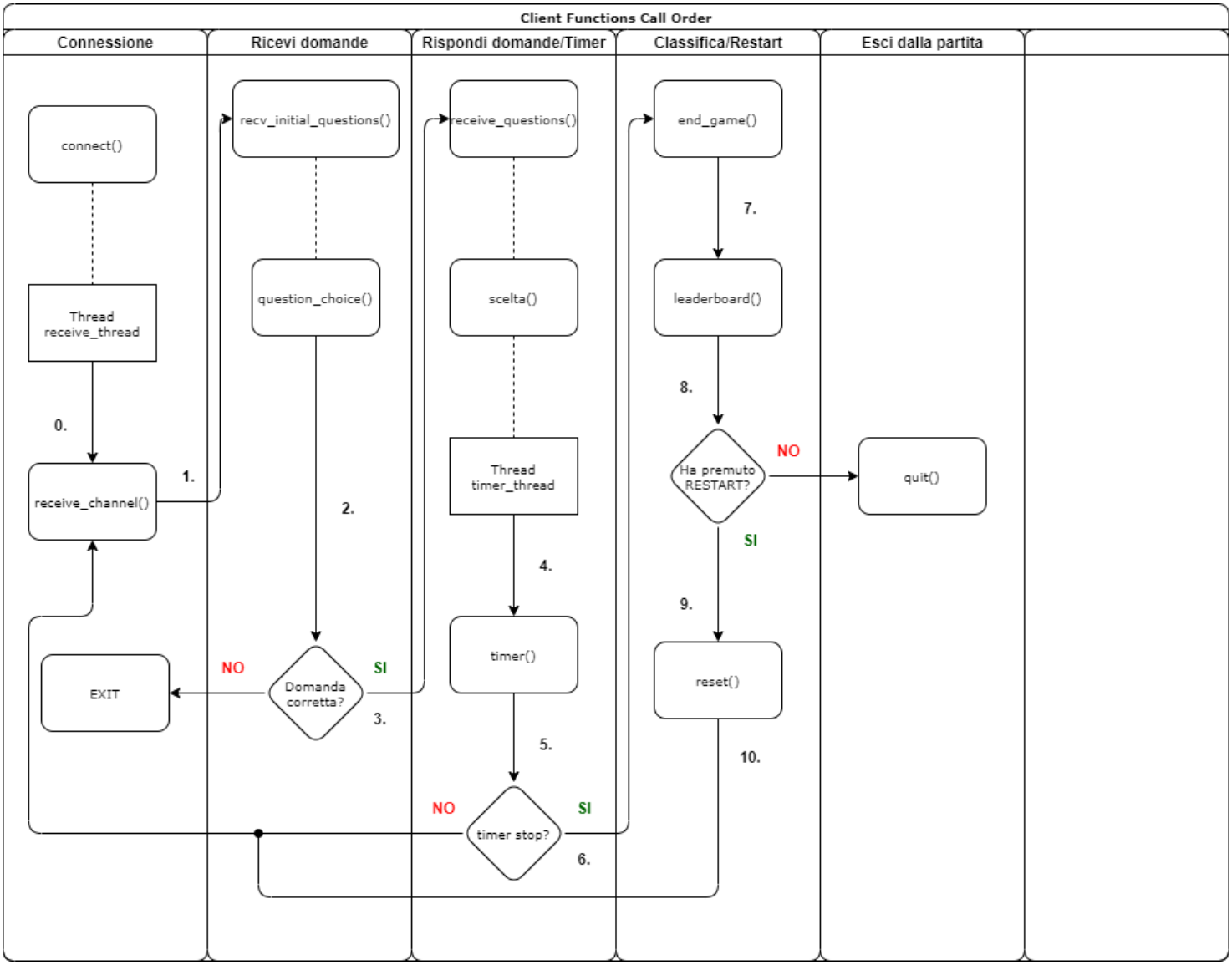


Figura 3: Diagramma di flusso: Ordine Chiamate Funzioni

Il Server

Il server è il cuore portante dell'applicazione, fa partire il gioco, coordina i giocatori, stila una classifica ed informa i giocatori del risultato della partita.

Server UML

Funzioni del Server

- **accept_connections()**
  - Questa funzione viene chiamata da una thread ogni qual volta che un client si connette al server.
- **pick_role(nome)**
  - Questa funzione restituisce un ruolo maschile o femminile in base al nome passato come parametro.
- **pick_questions(client_connection)**
  - Questa funzione sceglie delle domande casuali (tra cui una trabocchetto) li invia al client e poi aspetta di sapere dal client quale ha scelto per poter chiamare evaluate_question(client_connection) per poterla valutare.
- **evaluate_question(client_connection)**
  - Questa funzione valuta la domanda scelta dall'utente, se è la domanda trabocchetto informa l'utente che ha perso ed è stato eliminato, altrimenti prosegue con send_question(client).
- **send_question(client)**
  - Questa funzione invia all'utente la domanda che ha scelto più le 2 possibili risposte ad essa.
- **gestisci_utente(utente_connection)**
  - Questa è la funzione principale che viene chiamata una volta che l'utente ha acceduto al server. Si occupa di inviare/ricevere comunicazioni col client e creare la classifica e di terminare la partita.
- **send_leaderboard(utente_connection)**
  - Questa funzione si occupa di raggruppare le varie statistiche dai giocatori e creare la classifica ed inviarla ai clients.

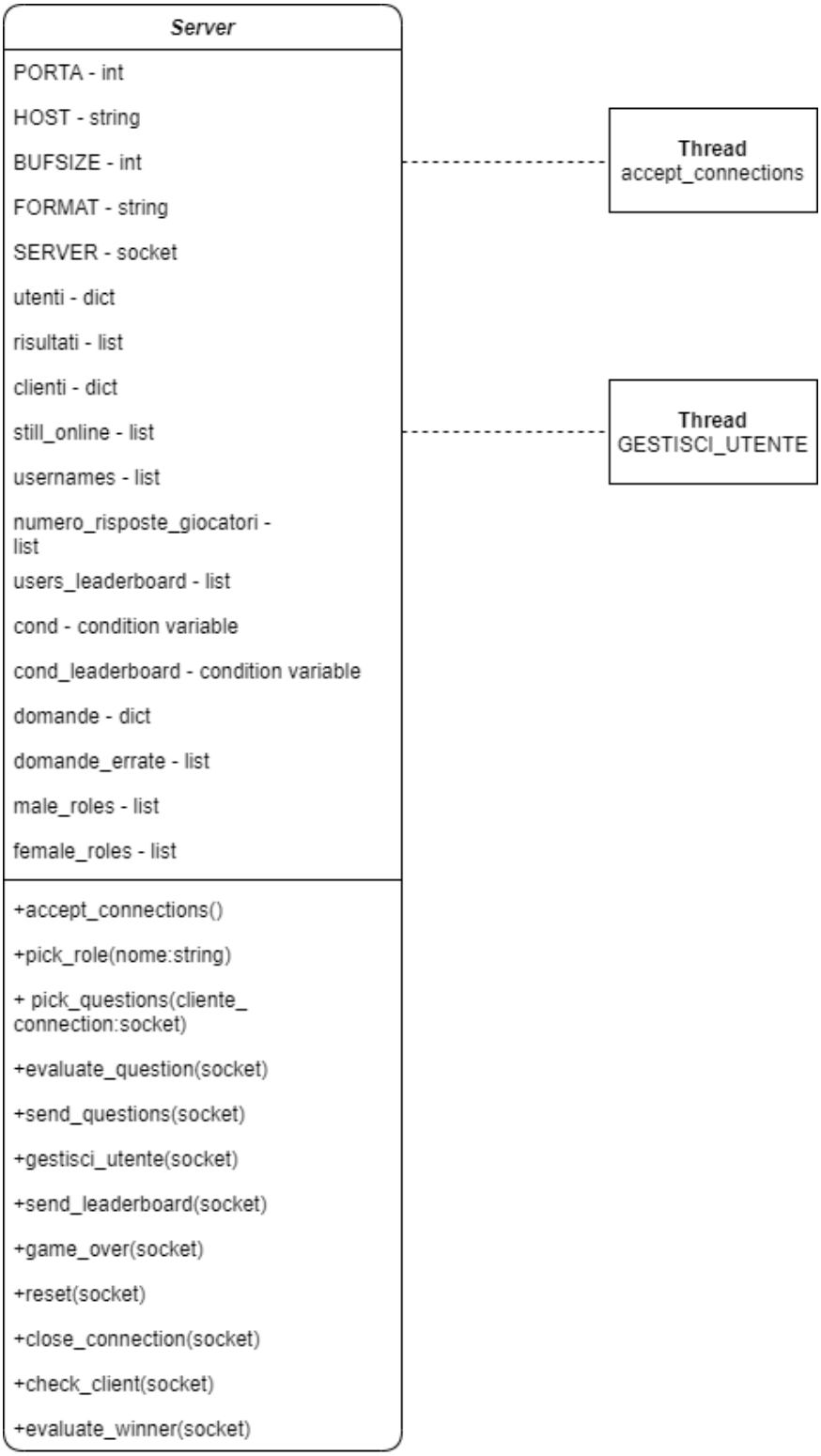


Figura 4: Server UML

- **game_over(utente_connection)**
  - Questa funzione chiama check_client(client) per controllare se gli utenti sono ancora connessi e poi chiama reset(utente) per far ripartire il gioco.
- **reset(utente)**
  - Questa funzione si occupa di azzerare alcune variabili e liste per poter far ripartire il gioco da capo.
- **close_connection(utente_connection)**
  - Questa funzione si occupa di chiudere la connessione con un determinato client e di cancellarlo dalle liste in cui era presente.
- **check_client(client_connection)**
  - Questa funzione riceve un messaggio dal client e in base a ciò valuta se esso resterà connesso o no. Se intende interrompere la connessione chiama close_connection(utente_connection)
- **evaluate_winner(utente_connection)**
  - Questa funzione riceve dai clients i loro risultati e li riordina e decreta il vincitore/perdente/pareggio ed invia ad ogni client il proprio risultato.

Ordine delle Chiamate alle Funzioni del Server

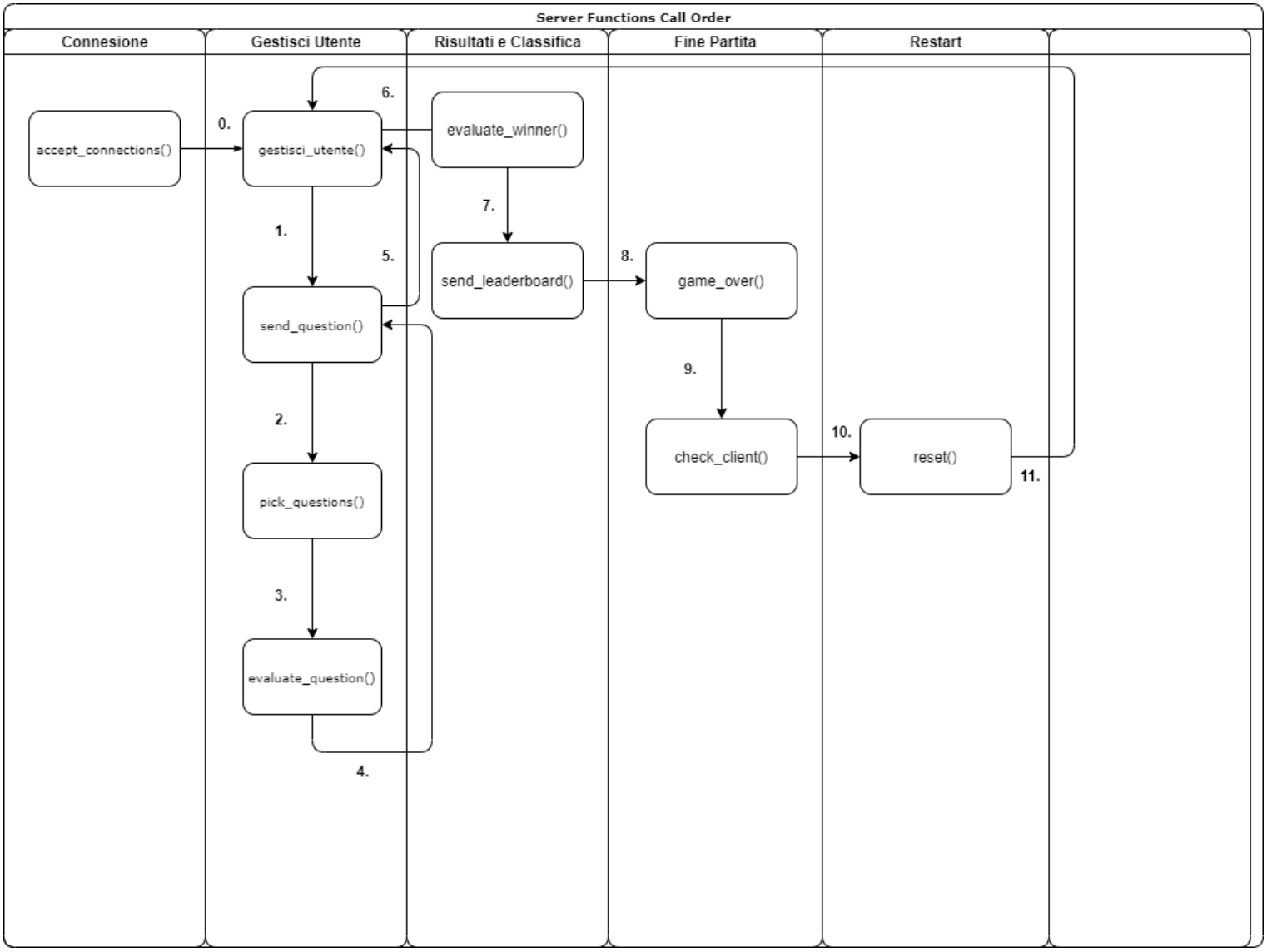


Figura 5: Diagramma di flusso: Ordine Chiamate Funzioni

Configurazione e Guida all'Utilizzo

Configurazione

E' possibile modificare alcune configurazioni del programma modificando alcune variabili:
Nel file **game_server**:

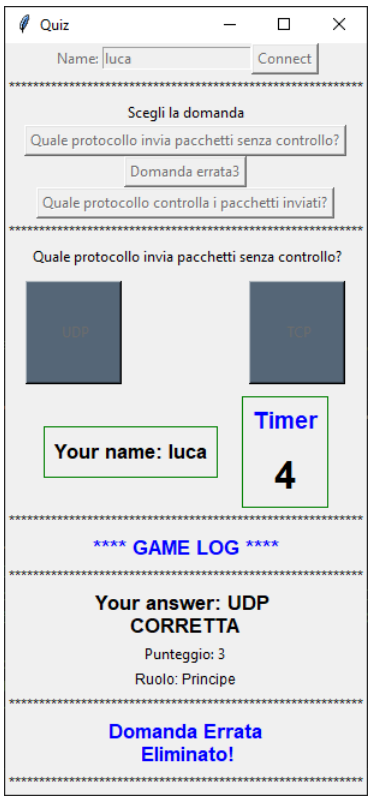
- **HOST**: L'indirizzo del server.
- **PORT**: La porta del server.
- **BUFSIZE**: La dimensione del buffer di ricezione dei messaggi dal cliente.
- **FORMAT**: E' possibile cambiare il FORMAT dei messaggi che è di default **utf-8**, ma andrebbe così cambiato anche nel client.
- **domande**: E' possibile aggiungere/rimuovere delle domande con le loro possibili risposte. (max 2 possibili risposte)
- **domande_errate**: E' possibile aggiungere/rimuovere delle domande trabocchetto.
- **male_roles**: E' possibile aggiungere/rimuovere ruoli maschili.
- **female_roles**: E' possibile aggiungere/rimuovere ruoli femminili.

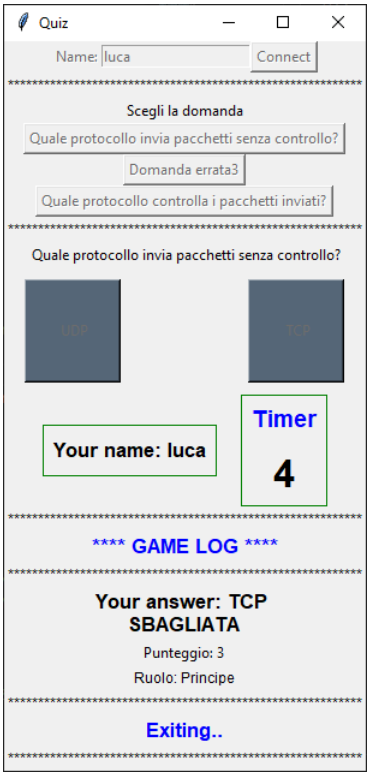
Nel file **game_client**:

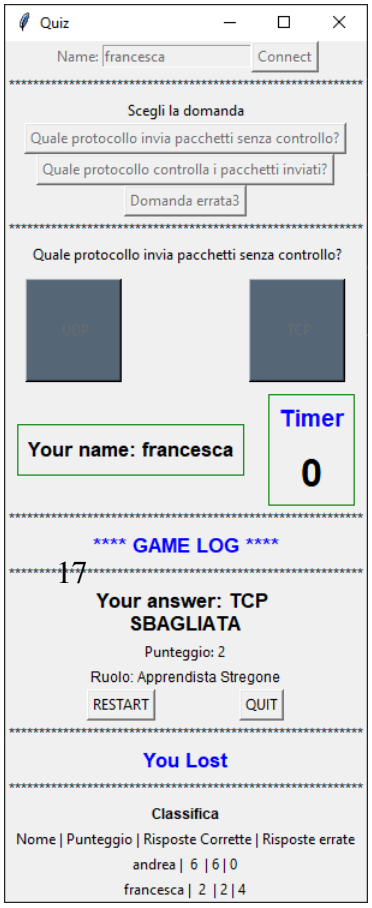
- **HOST**: L'indirizzo del server a cui ci si vuole connettere.
- **PORT**: La porta con cui ci si vuole connettere al server.
- **BUFSIZE**: La dimensione del buffer di ricezione dei messaggi dal server.
- **FORMAT**: E' possibile cambiare il FORMAT dei messaggi che è di default **utf-8**, ma andrebbe così cambiato anche nel server.
- **TOT_GAME_SECS**: Il numero di secondi che durerà la partita.
- **risposte_corrette**: E' possibile modificare questo dizionario che contiene le stesse identiche domande del server, ma con solo la risposta giusta, casomai si volessero aggiungere/rimuovere domande.

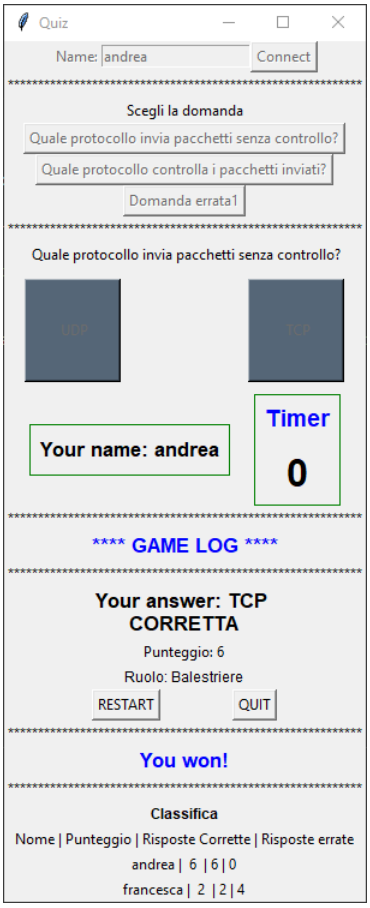
Guida all'utilizzo

- Aprire un terminale per il server ed eseguire il comando: **python game_server**.
- Dopodichè aprire almeno un altro terminale per almeno un client e digitare: **python game_client**.
- Con il client si aprirà un'interfaccia grafica che chiederà l'utente di inserire un nome di almeno un carattere.
- Inserito il nome si prema il pulsante **"Connect".**
- Dunque si vedranno, sotto il nome, 3 pulsanti con 3 domande, l'utente ne scelga uno.
- Se ha scelto la domanda trabocchetto, la GUI mostrerà *"Sei stato/a eliminato/a!"*, dopodichè si aggiornerà di nuovo e mostrerà *"Exiting.."* e dopo tot secondi uscirà dall'applicazione.
- Se, invece, si è scelta una domanda corretta allora, sotto quei 3 pulsanti delle domande, si vedrà un label con la domanda scelta e 2 altri pulsanti di cui l'utente dovrà sceglierne uno.
- Se l'utente ha scelto la risposta esatta, il suo contatore *"Punteggio"* incrementerà di 1 altrimenti resterà al valore che aveva.









- Dopo aver risposto alla prima domanda, partirà un timer e il giocatore continuerà a ricevere domande e a rispondere finchè quel timer non si fermerà.
- Una volta che il timer avrà concluso, l'utente resterà in attesa dal server del punteggio finale e della classifica.
- Terminata la partita l'utente potrà cliccare su due pulsanti: **"RESTART"** per poter rigiocare o **"QUIT"** per poter uscire.
- C'è anche la possibilità di pareggio.