Documentazione applicazione "*Trivia Quiz Multiplayer*"

Corso di Reti Informatiche a.a.2024/2025

Studente Guidi Dario

Funzionalità implementate

Nell'applicazione distribuita *Trivia Quiz Multiplayer* ogni client collegato al server deve rispondere a *5* domande ad un tema disponibile scelto. Si può partecipare solo una volta ad un tema di un quiz completato. Un giocatore che ha completato tutti i temi non può più collegarsi al gioco fino a quando non è presente un nuovo tema che non ha ancora completato. È possibile giocare di nuovo ad un tema solo se non è stato ancora completato.

Per partecipare, il giocatore deve prima scegliere un *nickname univoco* in modo da garantire la gestione corretta dei client che sono connessi al server contemporaneamente e per poter gestire correttamente questi i vincoli sui giocatori discussi precedentemente. Sono accettati un *numero massimo di 10 giocatori* che possono collegarsi al server contemporaneamente. La scelta di questo numero basso è motivata dal fatto che il server gestisce i vari client connessi con il meccanismo di *I/O Multiplexing*. Un aumento significativo del numero di giocatori collegati porterebbe il client collegato a sperimentare un ritardo nella risposta durante una sessione di gioco. Tuttavia, è stato scelto questo meccanismo e non altri, come ad esempio realizzare un server concorrente per la gestione multiprocesso delle richieste, perché è stato ritenuto comunque valido realizzare un videogioco molto semplice.

I temi già presenti sono "*Curiosità sulla tecnologia*" e "*Cultura Generale*". È possibile comunque aggiungere nuovi temi modificando il file "*quiz_files/quiz.txt*" e la macro MAX_QUIZ nel codice del server. Ogni tema è codificato nel file con un formato del tipo "<*id_tema*>**:**<*nome_tema*>". Per aggiungere nuove domande di un tema con le relative risposte è necessario modificare altri due file distinti: "*quiz_files/questions.txt*" e "*quiz_files/answers.txt*". Sia le domande che le risposte di un tema sono codificate nel file nel formato "<*id_tema*>**:**<*id_domanda*>**:**<*domanda*>" e "<*id_tema*>**:**<*id_domanda*>**:**<*risposta*>".

Nel file "*quiz_files/sessions.txt*" sono memorizzate tutte le sessioni di gioco iniziate e poi chiuse correttamente (abbandono partita o completamento di un tema) dei vari giocatori. Le sessioni sono codificate in questo file con il formato "<*id_tema*>**:**<*nome_giocatore*>**:**<*id_domanda*>**:**<*domanda*>". Questo file è utilizzato per tenere traccia dei temi completati e da completare per ogni giocatore che ha acceduto con un certo nickname specifico.

Il *client* non gestisce nessuna particolare *struttura di dati*, se non tenere memorizzato in delle variabili il *numero dei temi disponibili*, *l'identificatore del tema a cui sta giocando* ed il *numero dei punteggi presenti nel server*. Il *server*, invece, ha delle *strutture dati* fondamentali per caricare i dati presenti nei file per gestire i *client collegati*, i *temi presenti* e le *sessioni di gioco*. Per i *client collegati* si utilizza un vettore di *struct Giocatori* di *10* elementi. Per ogni giocatore di tipo *struct Giocatore* si memorizza: il *socket* associato alla connessione, il *nickname* e l'*id del tema* a cui sta giocando. Per i temi presenti si utilizza un vettore di stringhe di caratteri *lista_temi* con MAX_QUIZ elementi. Ogni elemento specifica il titolo del tema. Per le *sessioni di gioco* si utilizza una lista puntata *sessioni* con nodi del tipo *struct SessioneQuiz* nel quale si memorizza: l'*id del tema*, il *nickname del giocatore*, il *punteggio*, la *domanda* e il *puntatore ad un altro nodo*. È stata scelta una lista in modo che ogni volta che viene

inserita una nuova sessione viene posizionata efficientemente in ordine di punteggio per ogni tema senza dover ogni volta riordinarla per la stampa dei punteggi.

Durante la sessione di gioco, il client può dare dei *comandi* al server tra una domanda e l'altra. Sono stati implementati tre tipi di comandi: "*next*", "*show score*" e "*endquiz*". Con il primo comando richiede semplicemente di proseguire con una nuova domanda; con il secondo comando richiede la visualizzazione della lista dei punteggi ottenuti da tutti i giocatori in tutti i temi; con il terzo comando richiede di chiudere la sessione di gioco e di disconnetterlo.

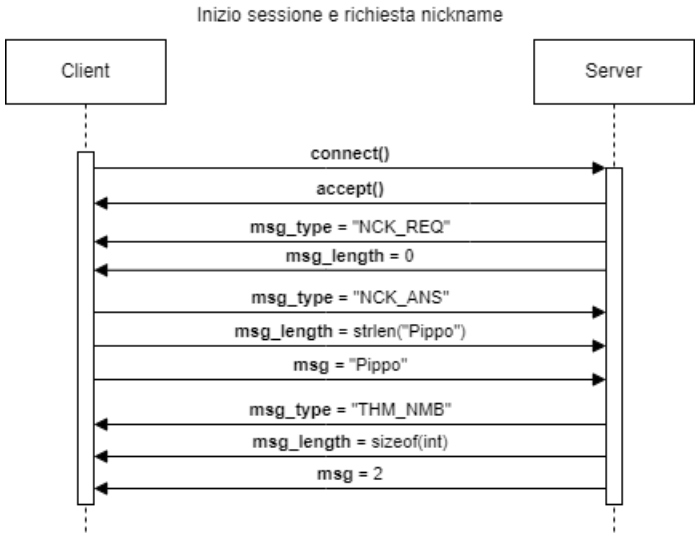
Schema dello scambio dei messaggi

Ogni messaggio è caratterizzato da tre informazioni: dal *tipo* del messaggio, dalla *dimensione* del contenuto del messaggio e dal *contenuto* del messaggio. Per alcuni tipi di messaggi si scambiano solamente il *tipo* e la *dimensione* del contenuto del messaggio pari a *0*, solo per scopo di trasmettere un segnale al destinatario per fare avanzare lo stato del gioco.

Il *tipo* del messaggio è una stringa di caratteri con una lunghezza fissa di *8* incluso anche il carattere di fine stringa. Ogni *tipo* del messaggio *è noto* sia al server che al client.

Il *contenuto* del messaggio può essere una stringa di caratteri (*TextProtocol*), un intero (*BinaryProtocol*) oppure una struttura dati serializzata (*BinaryProtocol*). Le strutture dati serializzate scambiate sono la *lista dei temi* e la *lista dei punteggi*. Il client deserializza queste liste solo per stamparle a video, senza doverle memorizzare. Tuttavia, deve memorizzare prima dell'arrivo di questi dati il numero di elementi che si aspetta dal server con un'altra fase di scambio. Quindi, questo tipo di scambio costa di più rispetto ad altri in quanto prevede due fasi: *scambio del numero di elementi* e *scambio dei dati*. Tutti gli altri scambi si basano su *TextProtocol*: domande/risposte, opzioni dei menu, inserimento nickname.

Un esempio dello schema di scambio dei messaggi *client* e *server*:



La scelta di questo tipo di scambio dà l'apparenza di poter scambiare un messaggio con il contenuto sia con *TextProtocol* o sia con *BinaryProtocol*. Il client ed il server si ricevono/inviano sempre due o tre informazioni in base alla *dimensione* ricevuta del contenuto del messaggio. Lo svantaggio è che per ogni scambio di messaggio servono alle peggio tre send() e tre recv().