Università degli Studi di Pisa

Dipartimento di Informatica

Word Quizzle

gioco di traduzione parole dall'italino all'inglese

Autore:

Luca Canessa (516639)

Laboratorio di Reti

anno accademico 2019-2020

Indice

1 Introduzione

Il progetto ha come scopo l'implementazione di un gioco dal nome «**Word Quizzle**», il cui obiettivo è la traduzione di parole dall'italiano all'inglesi tramite sfide tra utenti registrati al sistema.

Come da richiesta, per la realizzazione di tale lavoro è stato utilizzato il linguaggio di programmazione *Java*, nello specifico in versione *11*. La richiesta iniziale, inoltre, prevedeva la realizzazione di database in formato *JSON* che sono stati manipolati attraverso le API della libreria ''*JSON-Simple*''.

Per la realizzazione dell'interfaccia grafica riguardante il codice del client, è stata utilizzata la libreria ''*forms_rt*'' presente nel realizzatore grafico del IDE ''*IntelliJ*'' prodotto da JetBrains.

Il progetto è corredato da una documentazione tecnica, trascritta con l'aiuto del tool ''*Doxygen*'', visionabile nel file ''*Technical-Documentation*'' all'interno della cartella ''*doc*'' nella root del progetto.

Per facilitare l'avvio del sistema sono stati creati due script in linguaggio Bash, uno per il server e l'altro per i clients

1.1 Struttura del progetto

Il progetto è stato organizzato in questo modo:

Fig.1: Struttura gerarchica del progetto

Nella cartella *bin* sono presenti i files binari generati a tempo di compilazione. La cartella *config* contiene i files di configurazione del server, dei clients e i rispettivi esempi di impostazioni, per aiutare l'utenza a generare facilmente, evitando errori, la propria configurazione personale. All'interno della cartella *data* vengono inseriti e mantenuti i database utili al server, in particolare il database degli utenti e quello delle amicizie tra essi. La cartella *doc* include i file della documentazione tecnica presenti in vari formati: formato *html* eseguibile attraverso il file "*index.html*" contenuto nella cartella all'interno dell'archivio *'tar.gz'* e il formato *pdf. Nella cartella lib* vi sono librerie utilizzate all'interno dei codici sorgenti. La cartella *out* contiene due archivi dei files eseguibili in formato "*JAR*", uno per l'esecuzione del server e l'altro per l'esecuzione dei clients. La cartella *resources* racchiude risorse, come ad esempio il dizionario delle parole italiane. Nella cartella *src* sono presenti i codici sorgenti dei due *package*: client e server, rispettivamente nelle loro cartelle.

2 Struttura dei *packages*

I *packages* sono stati suddivisi come precedentemente detto in *server* e *client*. Il server mette a disposizione un certo numero di operazioni che i *clients* possono richiedergli, ottenendo una risposta adeguata al compito richiesto.

2.1 Server

Il server implementa i metodi necessari per soddisfare le richieste dei clients. Mette a disposizione la funzione di registrazione dell'utente, andando ad inserire nel database le sue informazioni, come *nickname*, *password*, *nome, cognome*; *nickname* e *password* sono dati indispensabili per portare a termine l'operazione. Come richiesto, questa, viene fornita attraverso la tecnologia RMI (RemoteMethodInvocation), che rimane attiva per tutta la durata di esecuzione del server. Un'altra richiesta che il server può soddisfare è il login dell'utente che lo richiede, il quale fornisce il suo *nickname* e la sua *password,* attività completabile dopo aver effettuato obbligatoria registrazione. Inoltre, il login, è necessario per svolgere successivamente tutte le altre richieste possibili. L'utente può inoltre richiedere l'inserimento di un altro utente registrato, tra le sue amicizie, oppure la rimozione di uno di essi da tale lista, se già presente. Tra le attività che l'utente può esigere, vi sono la richiesta di invio della propria lista di amicizie e della propria classifica di gioco. L'utente ha inoltre la possibilità di aggiornare il proprio profilo interno al sistema, richiedento la modifica del personale nome e cognome, e dalla password di accesso; oppure la sua cancellazione.

Un'altra funzione, che l'utenza può domandare al server, è l'invio di una richiesta di sfida ad un utente registrato al sistema, presente nella lista delle amicizie del richiedente e con stato 'ONLINE'. Tale richiesta di gioco viene recapitata al destinatario attraverso un pacchetto contenente il nome dello sfidante, utilizzando il protocollo di rete UDP di quarto livello (trasporto), dello stack ISO/OSI, come richiesto nei requisiti. Il server, per poter comunicare con i clients su protocollo UDP, ha bisogno di generare un numero di porta univoco per ogni client. Per fare ciò è stato implementato il

calcolo estrapolando il codice HASH del nickname dell'utente che riferisce quel client, tale procedimento minimizza il rischio di collisione. L'utente ha a disposizione un tempo, preimpostato nel file di configurazione del server, per rispondere alla richiesta in modo positivo o negativo. Allo scadere di tale tempo, la risposta calcolata è il rifiuto. In caso di accettazione, il server prepara un set di parole italiane, di numero prestabilito nel file di configurazione, e un set con la loro rispettiva traduzione, da inviare, step-by-step, ai due sfidanti. Per ogni parola che l'utente traduce, il sever la confronta con la sua versione e ne verifica la correttezza; in caso di corretta trasposizione vengono assegnati *N* punti, stabilti nel file di configurazione, in caso di errore ne vengono decurtati un altro numero, anch'esso estratto dal file di configurazione. In mancanza di risposta, non vi sono modifiche nel calcolo del punteggio. Al termine delle parole o allo scadere del tempo che gli utenti hanno a disposizione per completare la sfida, il server decreta il risultato, e nel caso in cui vi sia un vincitore, a questo vengono assegnati punti extra, precedentemente statuiti nella configurazione. Il server estrae le parole da un dizionario di 2110 termini di lingua italiana; per la loro traduzione si affida, come richiesto, al servizio esterno consigliato, accedendovi tramire API proprie del servizio e ricevendo in risposta un JSON, dal quale viene ricavata la traduzione. Altre due richieste che il server può soddissfare sono, il logout dell'utente o la sua rimozione dal sistema. Il logout consente all'utente di uscire dal gioco salvando i propri risultati nel database. La rimozione, invece, cancella definitivamente l'utente dal database con tutti i suoi dati.

Tutte le comunicazioni tra client e server sono gestite attraverso il protocollo di trasporto TCP, che permette di mantenere la connessione tra essi, fino a che il client non chiuda la comunicazione.

Il database utenti è implementato come oggetto JSON e gestito attraverso la libreria *JSON-Simple*, dentro cui abbiamo un JSONObject che rappresenta l'intero database, il quale a sua volta contiene chiavi corrispondenti al *nickname* degli utenti, che sono univoci. Per ogni chiave, quindi *nickname*, esiste un altro JSONObject al cui interno sono salvati i dati dell'utente che sono:

- password
- nome
- cognome
- stato dell'utente

    - online
    - offline
    - in partita
- punti totali guadagnati
- punti totalizzati nell'ultima partita

Tali chiavi, compreso il nickname, sono la rappresentazione dell'oggetto '*User*' all'interno del server, corrispondente all'utente iscritto.

Il database delle amicizie è anch'esso rappresentato come JSON, ma strutturato in questo modo: all'interno esiste un JSONObject ragruppante i nickname degli utenti, ognuno dei quali ha un altro JSONObject contenente un JSONArray con i nickname legati a quell'utente.

I valori di configurazione sono analizzati e acquisiti dal file apposito, strutturato come oggetto JSON. All'interno vi sono le chiavi, che corrispondo ai vari settaggi da impostare, aventi come valore quello conforme ed adeguato a quella chiave, contenute in altri JSONObject corrispondenti al settore da impostare.

Ogni operazione possibile è enumerata all'interno di una classe in modo che, sia server che client, abbiano il codice di ogni operazione. Ad ogni operazione il server rispode con un messaggio di acknowledge, rappresentato anch'esso come un enumeratore e condiviso con il client.

Da notare che, per ogni client che si connette al server, viene aggiunto un thread nel threadpool, avente dimensione variabile. Inoltre, per ogni client in sfida, viene attivato un thread che esegue l'handler del match. Ad ogni utente, che effettua il login, vengono copiati, nelle proprie variabili locali, i dati essenziali. Al termine di ogni compito, che richiede l'aggiornamento di tali dati, viene invocato un metodo sincronizzato, che copia tutti questi all'interno del database. Alla fine di ogni partita, per dedurre l'esito, il thread di ogni client aspetta che l'utente amico, dopo aver aggioranato i propri punti, cambi lo stato da '*INCHALLENGE*' a *'ONLINE',* in modo che i dati che verranno letti saranno consistenti ed aggiornati. Tutti i metodi, che riguardano i database, sono sincronizzati sull'oggetto, inoltre, tali database sono rappresentati nel server come **singoletti**, questo evita ridondanze di dati e problemi di consistenza.

Nel server è stato implementato anche un metodo che permette di stampare su un foglio di ‘‘*Log*’’ tutte le operazioni che compie, i possibili errori riscontrati ed altre informazioni, tenendone una traccia temporale.

2.1.1 Dettagli tecnici del server

Per dettagli specifici riguardanti ogni classe e ogni metodo, le chiamate ad essi ed il loro specifico utilizzo, fare riferimento alla documentazione tecnica.

2.2 Client

Il codice sorgente del client mette a disposizione dell'utenza funzioni necessarie per comunicare con il server. Tali funzioni sono facilmente raggiungibili grazie ad un'interfaccia grafica basilare, creata con l'aiuto della libreria "*Swing*" di Java, nella quale si trovano tutti gli elementi necessari per lo sviluppo di una *GUI,* e della libreria "*forms_rt*" di JetBrains, che aiuta a configurare tale *GUI*.

All'avvio del client, il primo pannello visibile, è un menù a doppia scelta, in cui è possibile decidere o l'*iscrizione* al sistema di un nuovo utente, oppure effettuare il *login* di un utente già iscritto.

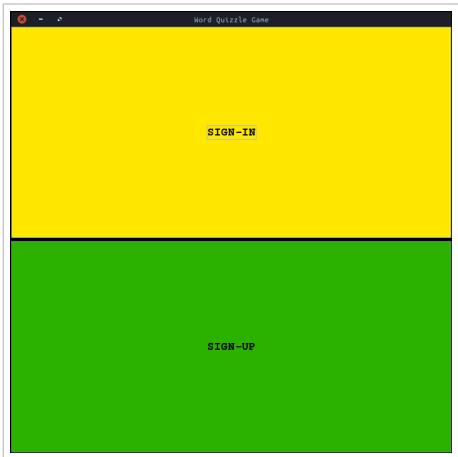

Fig.2: Menù iniziale

Tale pannello viene esposto solo al termine della richiesta di connessione al server, utile per le successive comunicazioni, e l'impostazione della "*RMI*" che consente la registrazione di un utente.

Se l'utente richiede la sua registrazione, premendo sul "*Button*" verde di "*Sign-up*", compare un nuovo pannello, in cui inserire le informazioni utili a tale attività.

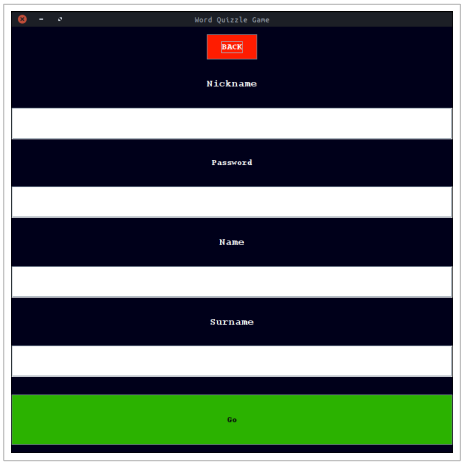

Fig.3: Pannello di registrazione

Per eseguire con successo la registrazione è obbligatorio inserire ‘‘*Nickname*’’ e ‘‘*Password*’’, mentre ‘‘*Nome*’’ e ‘‘*Cognome*’’ sono informazioni facoltative, il server, in mancanza di questi dati, si preoccuperà di gestire adeguatamente l’assenza, sostituendo al loro posto il valore speciale ‘‘*null*’’. I dati inseriti verranno recepiti dal server tramite l’invocazione del metodo remoto, da lui messo a disposizione attraverso la ‘‘*RMI*’’. Al termine dell’attività il client riceve un messaggio di risposta, stampato a video tramite *pop-up,* contenente il risultato dell’operazione.

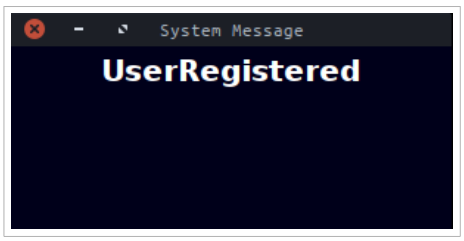

Fig.4: Esempio di pop-up

Se l’utente richiede di effettuare il ‘‘*login*’’ al sistema, utilizzando il ‘‘*Button*’’ giallo di *‘‘Sign-in’’*, compare a video un pannello in cui inserire *nickname* e *password* personali. Tutte le operazioni successive alla registrazione, compreso il *login* sono notificate al server tramite l’invio di un messaggio nominale ‘‘*ClientMSG*’’, specifico per ogni azione. Se il server durante *login* riscontra errori, questi vengono notificati, tramite messaggio apposito *‘‘ACK’’* con valore specifico, al client che provvede a stamparli a video, usando finestra *pop-up*. In caso di accesso avvenuto, il server comunica al client il successo dell’operazione, tramite appostito *‘‘ACK’’*, e quest’ultimo mostra a video il pannello personale del client.

Su questo pannello è possibile visulizzare, a sinistra, lista e numero delle amicizie utente, mentre in alto a destra compaiono un messaggio di benvenuto e il totale punti dell'utente; sotto a quest'ultimi, sono visibili i bottoni necessari all'avvio delle operazioni possibili:

- Aggiunta di un amico alla propria lista amicizie
- Rimozione di un amico già presente in tale lista
- Aggiornamento di questa lista
- Invio di una sfida ad un proprio amico online
- Aggioramento dati personali dell'utente
- Richiesta classifica personale di gioco
- Rimozione dell'utente dal sistema
- Uscita dal sistema

Fig.5:Esempio schermata ad accesso avvenuto

Premuto il bottone ‘‘*Aggiungi Amico*’’, compare una finestra in cui è possibile inserire il *nickname* desiderato ed inviare, tramite il bottone sottostante, la richiesta al server di aggiungere il *nickname alla lista amicizie.* Il server risponde con messaggio ‘‘*ACK*’’ contenente il risultato dell’operazione, che il client visualizza sotto tale bottone.

Fig.6: Esempio finestra per aggiunta di un amico alla propria lista

Premuto il bottone ‘‘Rimuovi *Amico*’’, compare una finestra in cui è possibile inserire il *nickname* desiderato ed inviare, tramite il bottone sottostante, la richiesta al server di rimovere il *nickname dalla lista amicizie.* Il server risponde con messaggio ‘‘*ACK*’’ contenente il risultato dell’operazione, che il client visualizza sotto tale bottone.

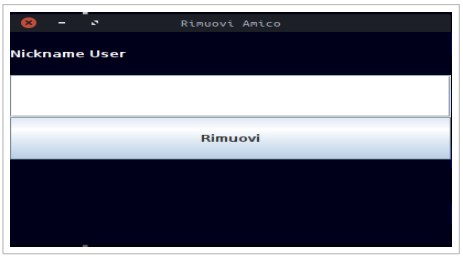

Fig.6: Esempio finestra per rimozione di un amico alla propria lista

Al termine di queste operazioni, automaticamente, il client invia al server la richiesta di invio lista amicizie per visionarla aggiornata sulla GUI. L’operazione di aggioramento può essere anche richiesta manualmente premendo il bottone ‘‘*Aggiorna Lista Amicizie*’’, ottenendo il medesimo risultato.

Premendo il pulsante digitale ‘‘*Sfida un Amico*’’, compare una finestra nella quale inserire il *nickname* dell’amico che l’utente ha deciso di sfidare. Inviando tale richiesta al server, se l’amico è *‘‘ONLINE’’,* l’utente dovrà attendere la risposta di tale amico, altrimenti riceverà, senza nessuna attesa, il messaggio del server contenente lo stato dell’amico che non può ricevere la notifica di sfida. Nel caso in cui l’amico sia ‘‘ONLINE’’ ed abbia accettato la sfida, la finestra principale viene sostituita da quella di gioco, nella quale compaiono, una alla volta le parole italiane da tradurre ad ogni invio della

risposta. Allo scadere del tempo, che il server ha impostato per la terminazione della sfida, o al termine delle parole recapitate, questa finestra viene chiusa, facendo ricomparire quella principale. Se l'amico è ancora in gioco si attende la sua terminazione prima di ricevere il messaggio dal server contenente l'esito della partita. Se necessario, verrà aggiornato automaticamente il numero di punti totali. In caso di rifiuto della sfida viene visualizzato il messaggio apposito, inviato dal server, visualizzato sotto il bottone di invio della richiesta.

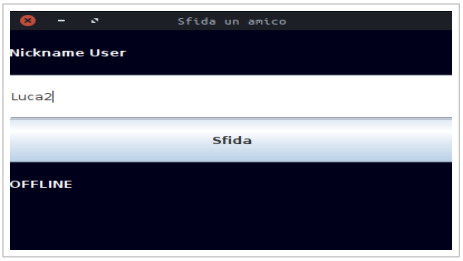

Fig.7: Finestra per richiesta di sfida

Fig.8: Esempio di finestra di gioco

Il client, che deve ricevere la sfida, comunica con il server tramite il protocollo UDP, attraverso la porta decifrata usando un particolare calcolo ricavato dal codice hash del suo *nickname*. Tale client, ricevuta la richiesta di sfida, visualizza a schermo una finestra contenente il *nickname* dello sfidante e i bottoni di accettazione e rifiuto; allo scadere del tempo, che il server concede per valutare la risposta, viene inviato al server, come risposta di default, il rifiuto della sfida.

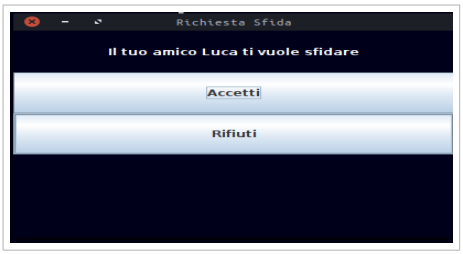

Fig.9: Finestra di ricezione di una sfida

Premendo il pulsante "*Aggiorna Profilo*" viene visualizzata una nuova schermata in cui è possibile inserire alcuni campi riferiti all'utente: *nome,*

cognome, password, che spediti al server verranno aggiornati nel database. Tali campi sono aggiornabili solo se contengono almeno 1 carattere e senza nessun altro vincolo.

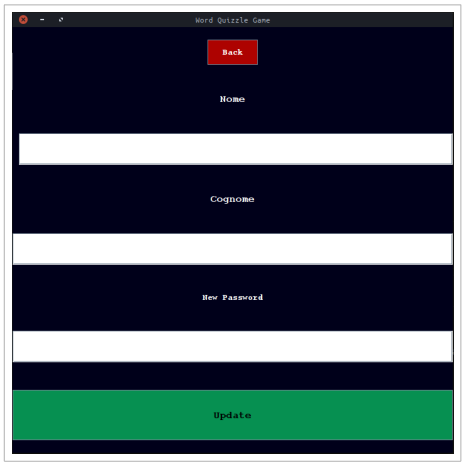

Fig.10: Finestra di aggiornamento profilo

Premendo il bottone "*Classifica*" viene richiesto al server l'invio della ranking di gioco, ordinata in modo decrescente sul numero di punti guadagnati dagli amici dell'utente che ha fatto tale richiesta, e dall'utente stesso. Ricevuta tale classifica, il client la modella per visualizzarla al posto della lista amici.

Fig.11: Esempio di ranking list (visualizzabile nella colonna di sinistra)

Premendo i "*Button*" rossi, si eseguono operazioni delicate come la rimozione dell'utente dal sistema, che comporta anche la cancellazione di tutti

i suoi dati e l'eliminazione dell'utente dalla lista di tutti i suoi amici, e il logout che comporta l'uscita dal gioco rendendo lo stato dell'utente ''*OFFLINE*''. In entrambi i casi, se le operazioni hanno successo, viene visualizzato su finestra *pop-up* il messaggio ricevuto dal server con l'esito positivo dell'operazione e chiuso il profilo utente visualizzando la schermata di menù principale, in caso di fallimento delle operazioni viene solamente visualizzato il messaggio, ricevuto dal server, di error sulla finestra di *pop-up,* senza alcuna conseguenza per lo stato dell'utente.

Solo alla chiusura della finestra il client chiude la connessione con il server.

Il client estrapola i valori delle sue impostazioni dal file di configurazione, tranne il tempo massimo per la terminazione del gioco, che gli viene fornito dal server nel momento esatto in cui la connessione risulta stabilita, il client comunque provvede a salvarlo nel suo file di configurazione.

Per ogni client che risulta aver effettuato il login, viene creato un thread in cui è eseguito il gestore delle comunicazioni su protocollo UDP. Un altro thread per client viene generato nel momento di avvio di una sfida, questo thread ha il compito di eseguire l'handler di tale sfida.

2.2.1 Dettagli tecnici del client

Per dettagli specifici riguardanti ogni classe e ogni metodo, chiamate ad essi ed il loro specifico utilizzo, fare riferimento alla documentazione tecnica.

3 Avvio del Sistema

Per facilitare l'utenza sono stati creati due script in linguaggio *Bash,* che avviano il server ed i clients. Entrambi fanno riferimento agli eseguibili archiviati in formato *JAR* all'interno della cartella *out* nella *root* del progetto ed hanno impostato come file di configurazione quello di default all'interno della cartella *config.* Tali files possono essere editati o ricreati secondo le proprie esigenze, ricordandosi però di mantenere i files creati all'interno della solita cartella *config,* e di modificare il nome dei files, che si voglio usare, nei rispettivi script *Bash*.

3.1 Avvio del server

Per avviare il server è sufficiente lanciare da riga di comando lo script ''*server.sh*'', con l'opzione ''*-s*'' presente nella root del progetto. Lanciato suddetto comando lo script si preoccupa di controllare che non vi siano altre istanze del medesimo processo. Nel caso in cui si stia per lanciare la prima istanza, lo script manda in esecuzione tale processo in background e termina il suo compito. Se invece, tale istanza non è la prima, lo script controlla che effettivamente esista nell'insieme di tutti i processi attivi della macchina, e ritorna un messaggio di errore, evitando problemi di conflittualità.

Nel caso in cui si voglia fermare il processo server in esecuzione in modo gentile, salvando tutti i dati e terminando tutti i threads mantenendo consistenza, è necessario lanciare lo script ''*server.sh*'' con l'opzione ''*-S*'', il quale provvederà a sollevare un segnale di terminazione, che il server gestirà nel modo più opportuno.

Prima di inviare un qualsiasi comando è possibile consultare la guida sull'uso dello script lanciando il comando ''server.sh -h''.

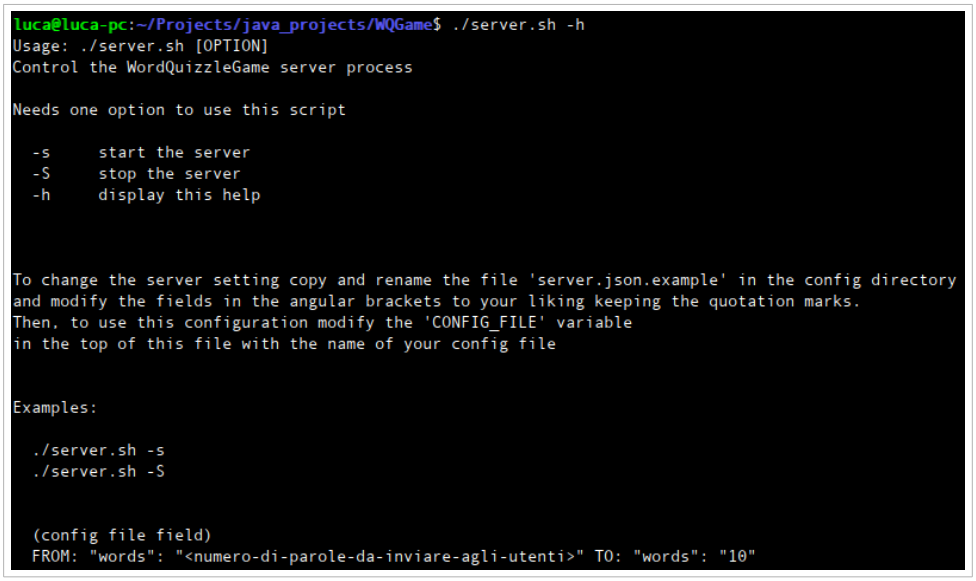

Fig.12: Esecuzione dello script con opzione *-h*

3.2 Avvio del client

Per l'avvio di ogni client, che si desidera eseguire, è sufficiente lanciare da terminale lo script "*client.sh*" con opzione "*-s*", presente nella root del progetto. Questo avvierà un processo client ad ogni lancio.

Per terminare l'attività del client è sufficiente chiudere la finestra *GUI* che si sta utilizzando per il controllo di esso.

Prima di avviare un client è possibile visualizzare una guida sull'uso e sulla configurazione eseguendo lo script con l'opzione "*-h*".

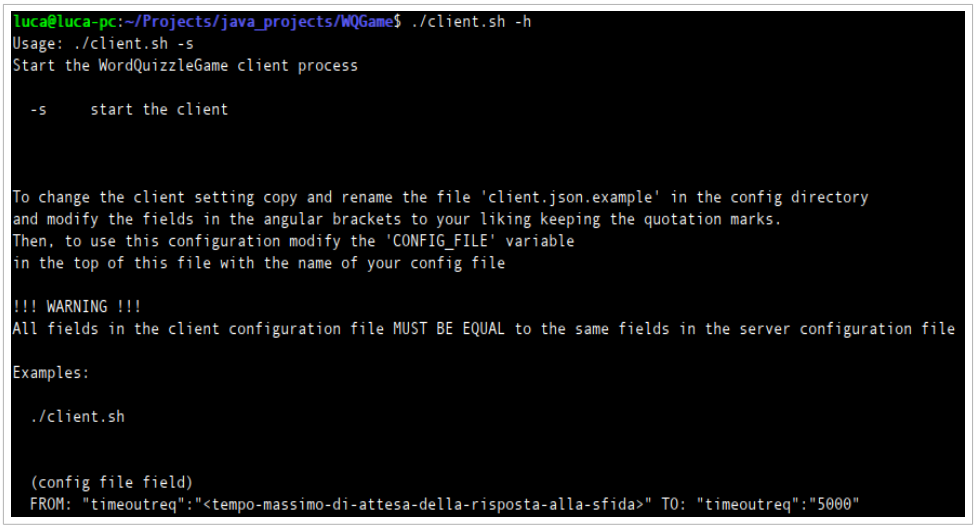

Fig.13: Esecuzione dello script client.sh con opzione -h

4 Overview della documentazione tecnica

Per completezza è stata creata una documentazione specifica di ogni classe, in cui sono descritti, in modo dettagliato, i metodi e le variabili utilizzati all'interno di questa, sia pubblici che privati. Tale dossier è stato ottenuto commentando, secondo la sintassi doxygen, tutto il codice sorgente del progetto e fatto analizzare successivamente dal tool, ''*Doxygen*'', il quale ha estratto e ordinato tale documentazione. Inoltre, settando certi suoi parametri è stato possibile estrarre anche i diagrammi di: astrazione dei metodi, chiamate ai metodi ed il loro utizzo nel resto del codice, in sintassi UML.