RELAZIONE PROGETTO DI RETI

a.a 2019-2020

Tumino Nicolò 544094

Indice

Introduzione

Il progetto consiste in un servizio basato su architettura client-server, che realizza un sistema di sfide di traduzione italiano-inglese. Il sistema offre anche una rete sociale agli utenti registrati al servizio, che permette, ad esempio, di controllare la classifica dei punteggi.
Il client comunica al server le proprie richieste mediante l'utilizzo del protocollo TCP, il server assegna al client un thread che gestirà le richieste di quest'ultimo.
L'utente interagisce con il sistema mediante un'interfaccia grafica basata sulle librerie "javax.swing" e "java.awt".

E' presente, lato "front-end", un ulteriore thread, oltre al principale, che si occupa di comunicare e ricevere notifiche (es. Richiesta di sfida, risposta ad una richiesta di sfida), mediante l'utilizzo del protocollo UDP.

Il sistema utilizza due file "JSON" per implementare:

- La persistenza delle informazioni
- Il dizionario per poter svolgere la sfida

Architettura Server

Server

Il lato “back-end” del sistema è implementato mediante un server multi-thread, il quale implementa il servizio di registrazione mediante “RMI” e sfrutta un pool di oggetti “Runnable” per soddisfare tutte le ulteriori richieste di ogni client.

Il server, subito dopo l’avvio, carica on-line il servizio basato su “RMI”, mettendosi successivamente in ascolto di nuove connessioni su una socket TCP. Ogni nuova connessione viene affidata ad un thread prelevato dal thread-pool.

Il server inoltre sfrutta un database che, associato al corrispettivo file “JSON”, permette di memorizzare in maniera persistente le informazioni dei vari utenti (nome, password, punteggio, amici).

Comunicazione ed Errori

Protocollo

Il seguente protocollo di comunicazione specifica i comandi e i vari parametri utilizzati per implementare il servizio e le comunicazioni client-server:

Login	Username	Password	Hostaddress	UDPport
Logout	Username			
Add	Username	Friendname		
List	Username			
Score	Username			
Rank	Username			
Challenge	Username	Friendname		

Il comando per la registrazione è realizzato tramite “RMI”, quindi non

incluso nella tabella di cui sopra.

Errori

I codici di errore sono specificati nella classe “ReturnCodes”, dove sono presenti due metodi per la traduzione da codice a stringa e viceversa. Possibili codici:

- *EMPTY_NICK_OR_PASS* : username o password vuoti
- *ALREADY_REGISTERED* : utente già registrato
- *USER_NOT_FOUND* : username non esistente
- *WRONG_PASSWORD* : password errata
- *ALREADY_LOGGED_IN* : utente già online
- *ALREADY_LOGGED_OUT* : utente già offline
- *ALREADY_FRIENDS* : l’utente è già tra gli amici
- *NOT_A_FRIEND* : impossibilità di sfidare l’utente perchè non presente nella lista degli amici
- *USER_NOT_ONLINE* : utente non online
- *SUCCESS* : operazione eseguita con successo
- *COMMAND_NOT_FOUND* : comando inesistente
- *UNKNOWN_CODE* : codice di errore sconosciuto

Database

Componente del sistema che si occupa di memorizzare, e rendere persistenti, le informazione durante l’intera esecuzione del server.

L’utente viene rappresentato, all’interno del database, con la classe “DataObject” che, oltre password e lista di amici, permette di identificare anche la connessione relativa a quello specifico utente, memorizzando “InetAddress” e “Porta UDP” (quest’ultima relativa alla ricezione delle notifiche).

Viene memorizzato anche lo stato dell'utente (On-line, Off-line) che per ovvie ragioni, insieme all'indirizzo e alla porta UDP, non è un'informazione persistente.

La struttura portante del database è una "HashMap", dove la chiave rappresenta il nome dell'utente, che ci permette l'accesso al corrispettivo "DataObject".

Il database sfrutta la classe "ReturnCodes" per mappare i vari codici di errore o di successo, che verranno restituiti per ogni azione richiesta dall'utente.
Le informazioni vengono rese persistenti ad ogni modifica del "DataObject" corrispettivo o per ogni sfida terminata con successo.

Client Handler

La classe "ClientHandler" definisce l'oggetto "Runnable" che permette di gestire tutte le richieste da parte dell'utente.

Nel caso particolare della richiesta di "Challenge" il "Client Handler" per prima cosa avvia il thread che successivamente gestirà la sfida, cosi da essere già pronto nel caso in cui la richiesta di sfida venga accettata dallo sfidato.
La richiesta viene inviata sulla porta UDP dello sfidato, la quale sarà in ascolto per notifiche. Il "Client Handler" setta un timer di attesa, subito dopo aver mandato la richiesta, il quale, se scade prima che si riceva una risposta, determinerà una risposta negativa che verrà inviata allo sfidante.
Una volta ricevuta la risposta dall'utente sfidato, il "Client Handler" la comunicherà allo sfidante, inoltre nel caso di risposta negativa o timeout si occuperà di interrompere il "Challenge Handler" precedentemente attivato.

Challenge Handler

Il "Challenge Handler" è la componente "back-end" del sistema che si occupa di gestire la sfida.
Appena avviato sceglie 8 parole, dal dizionario, che saranno inviate successivamente ai due utenti partecipanti alla sfida.
Una volta selezionate le parole, apre una socket "TCP" sulla quale si mette in

ascolto dei due utenti.

Quando i due utenti sono collegati con il “Challenge Handler”, quest’ultimo, mediante il multiplexing dei canali con “NIO”, si occupa effettivamente di dare inizio alla sfida.
Il thread controlla:

- Se i canali sono pronti ad accettare una nuova connessione. In questo caso vengono registrate le chiavi rappresentanti i canali aperti, viene richiesta la traduzione delle parole scelte mediante richieste “HTTP” e viene avviato il timer per la sfida.

- Se i canali sono pronti per la scrittura. In questo caso viene mandata una parola da tradurre ai due client o, nel caso si sia finito il tempo o si siano finite le parole, verrà inviato “Timeout” o “End”.

- Se i canali sono pronti per la lettura. In questo caso viene letta la parola inviata dall’utente e se ne controlla la correttezza della traduzione assegnando il corrispettivo punteggio. Nel caso in cui la sfida sia terminata la parola inviata dal client sarà “Exit”, in tal caso si chiude il canale.

Una volta chiusi tutti i canali si controlla che, la sfida sia effettivamente terminata con successo, quindi che sia scaduto il tempo o che entrambi gli utenti abbiano terminato le parole da tradurre, quindi si procede a rendere il risultato della sfida persistente aggiornando il database e il corrispettivo file “JSON”.

Architettura Client

Client

Il client appena avviato ricerca il servizio “RMI” offerto dal server per

potersi registrare, successivamente avvia la prima interfaccia grafica “StartView” che permette l’accesso al servizio, avviando l’interfaccia grafica di accesso “AccessView” dove sarà possibile effettuare l’operazione “Register” e “Login”.

Dopo aver effetuato l’operazione di “Login”, il client avvia la schermata principale del servizio “MainView” dove sarà possibile effettuare tutte le operazione messe a disposizione dal sistema, e dove sarà possibile vedere le notifiche e rispondere alle richieste di sfida.

Quando l’utente accetta una richiesta di sfida verrà avviata l’interfaccia grafica “ChallengeView” che permetterà all’utente di effettuare la sfida.

All’interno di questa classe sono implementate tutte le funzioni, richiamate dalle GUI, che permettono la comunicazione con il server.
In particolare durante la richiesta di “Login” verrà creato anche il thread “Notify” di cui sotto.

Notify

Il thread “Notify” è la componente del sistema che permette all’utente di ricevere notifiche e rispondere ad esse.
Esso controlla, ciclicamente, se ci sono notifiche sulla porta UDP creata in fase di “Login”. Per ogni notifica ricevuta controlla se si tratta di:

- Richiesta di sfida. In questo caso provvede ad aggiornare l’interfaccia grafica in modo da mostrare la nuova notifica.
- Messaggio di timeout. In questo caso provvede semplicemente a rimuovere la notifica dall’interfaccia grafica.
- Risposta ad una precedente richiesta di sfida. In questo caso, se la risposta è negativa provvede ad avvisare l’utente mediante un messaggio, altrimenti, in caso di risposta affermativa viene avviata l’interfaccia di gioco.

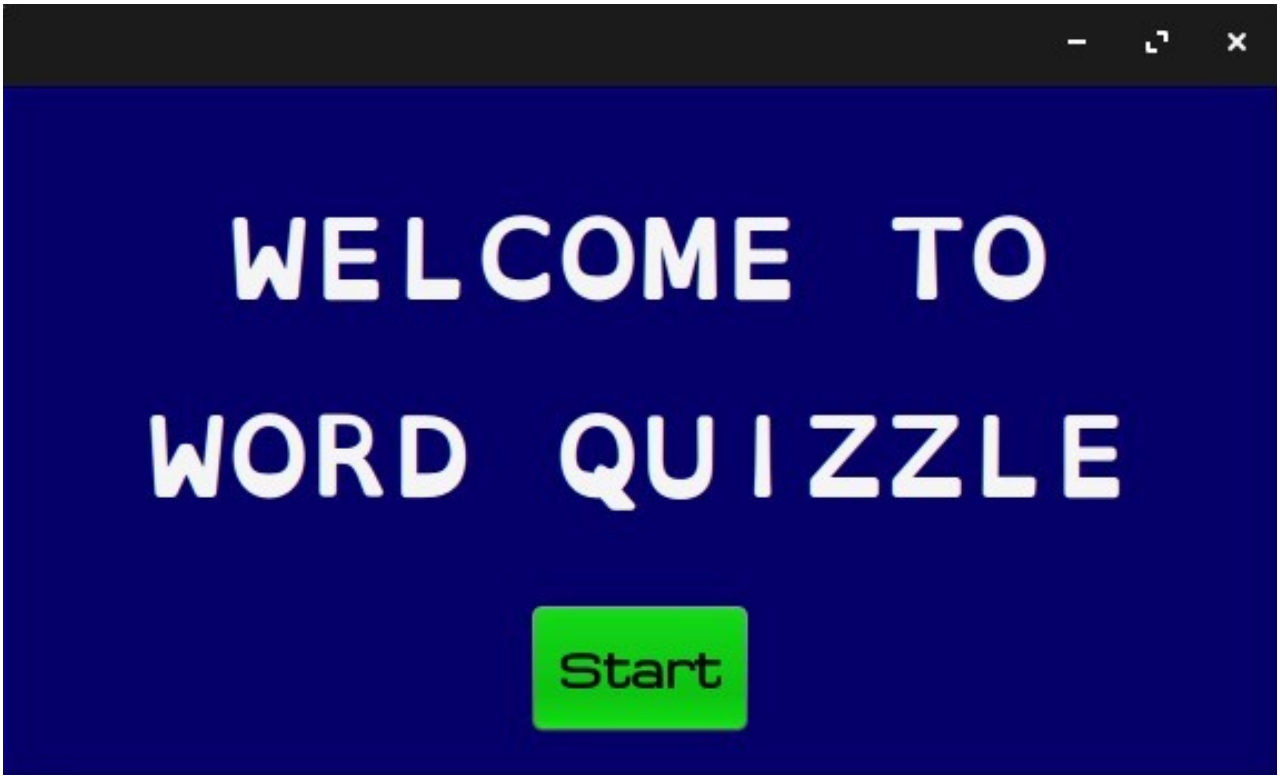
Il thread “Notify” implementa, inoltre, la comunicazione, verso il server, delle risposte alle richieste di sfida ricevute dall’utente.

GUI

Di seguito vengono brevemente descritte e illustrate le interfacce grafiche realizzate mediante l’utilizzo delle librerire grafiche “java.awt” e “javax.swing”.

Start View

Interfaccia di avvio del gioco.



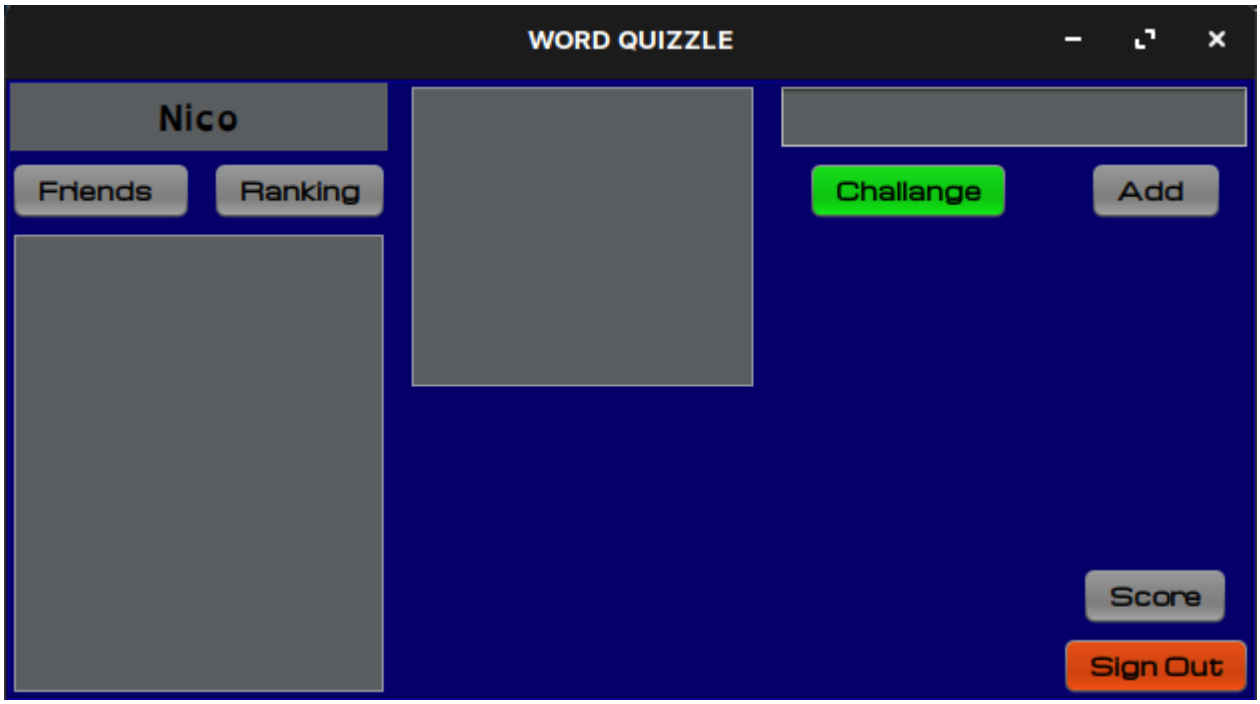
Access View

Interfaccia dove è possibile registrarsi e accedere al gioco.

Main View

Interfaccia centrale del gioco, permette di:

- Vedere la lista dei propri amici
- Controllare la classifica
- Vedere il proprio punteggio
- Vedere le notifiche
- Sfidare un amico
- Aggiungere un amico



Challenge View

Interfaccia di gioco.

Compilazione ed Esecuzione

Compilazione

Per la compilazione è sufficiente spostarsi nella cartella contenente i file (es. src) ed eseguire il seguente comando:

- javac -classpath forms_rt.jar:gson-2.8.2.jar:json-simple-1.1.1.jar: *.java

Assicurarsi che i file “jar” siano all’interno della stessa cartella dei file “java”, altrimenti è necessario specificare il path dopo l’opzione “-classpath”.

Esecuzione

Per l’esecuzione è sufficiente eseguire i comandi di cui sotto.

Server:

- java -classpath gson-2.8.2.jar:json-simple-1.1.1.jar: Server

Client:

- java -classpath forms_rt.jar:gson-2.8.2.jar:json-simple-1.1.1.jar: Client

Notare che per una correta esecuzione è necessario che il file “dictionary.json” sia all’interno della cartella dove sono presenti tutti gli altri file.
Progetto testato su sistemi Unix: Ubuntu, Zorin OS, Manjaro.