WordQuizzle

Leonardo Vona
545042

22/10/2020

Il sistema di sfide di traduzione italiano – inglese WordQuizzle è implementato secondo un'architettura client – server.

Il client e il server comunicano tra di loro tramite i protocolli TCP e UDP, oltre che mediante RMI.

Server

Il server si occupa di:

- gestire le varie richieste provenienti dai client;
- accedere allo storage per la persistenza delle informazioni relative agli utenti registrati al servizio e per accedere al dizionario delle parole da sottomettere ai giocatori per la traduzione;
- comunicare con il servizio RESTful MyMemory tramite chiamate HTTP GET.;

In particolare tutte le richieste (ad eccezione della registrazione di un nuovo utente) provenienti dai client avvengono con una connessione TCP, che viene gestita tramite multiplexing dei canali mediante NIO, che permette una buona flessibilità e scalabilità dell'intero sistema.

Nel server sono presenti due tipi di selettore:

- Il primo tipo è specifico per la gestione della comunicazione con due client che si stanno sfidando. Di questo tipo, è presente un'istanza per ogni singola sfida.
- Il secondo tipo gestisce la comunicazione di tutte le altre richieste e risposte provenienti e dirette ai client. Di questo tipo è presente una singola istanza.

La registrazione di un nuovo utente viene gestita tramite RMI da parte del client, il quale chiama un metodo implementato dal server.

Al momento della registrazione di un nuovo utente, viene effettuato un hashing della password tramite algoritmo PBKDF2. L'attributo password dell'utente contiene il numero di iterazioni effettuate per effettuare l'hashing, il salt utilizzato e l'hash effettivo. I primi due parametri sono memorizzati per poter poi verificare al momento del login se la password comunicata è corretta.

La comunicazione tramite UDP avviene per l'invio di una richiesta di sfida dal server verso un client e per la relativa risposta. L'invio e la ricezione di pacchetti UDP viene gestita dal server tramite un metodo ad hoc per la specifica comunicazione.

L'accesso allo storage avviene attraverso Java IO API; infatti, per le performance che si vogliono ottenere e per il carico di lavoro che si prevede, non si è ritenuto necessario utilizzare NIO API.

Le informazioni relative agli utenti registrati nella piattaforma sono memorizzate all'interno di un file in formato JSON e la serializzazione e deserializzazione è gestita tramite la libreria esterna Gson (https://github.com/google/gson).

La comunicazione con il servizio MyMemory avviene ogni volta che una sfida viene accettata. Infatti il server recupera K parole dal dizionario ed effettua in sequenza K chiamate HTTP GET al servizio per ottenere le traduzioni.

Il recupero delle parole dal dizionario avviene attraverso un algoritmo di reservoir sampling.

Client

Il client si occupa di:

- interfacciarsi con l'utente tramite GUI;
- comunicare con il server per l'invio delle varie richieste provenienti dall'utente e ricevere le relative risposte

La gestione delle richieste di sfida avviene tramite un oggetto indipendente che si mette in ascolto (su una porta specificata nei parametri di configurazione) per la ricezione di datagrammi UDP. Al momento della ricezione di una richiesta, la comunica all'utente e invia al server tramite un nuovo datagramma se la richiesta è stata accettata o meno.

L'utente interagisce con il sistema attraverso la GUI fornita dal client. Gli eventi generati dalla GUI sono gestiti da uno specifico oggetto, il quale recupera le informazioni relative all'evento ed eventualmente prepara una richiesta da inviare al server. Al momento dell'invio della richiesta tale oggetto attende una risposta dal server e, una volta arrivata, la gestisce e comunica il risultato all'utente.

I pacchetti inviati e ricevuti dal server seguono uno specifico formato.

L'invio e la ricezione dei pacchetti con il server tramite TCP avviene attraverso uno specifico oggetto, il quale espone i metodi per leggere ed inviare un messaggio sul canale di comunicazione instaurato con il server.

Protocollo di comunicazione

Il server ed il client comunicano tramite TCP seguendo uno specifico protocollo.
Il client invia al server richieste della forma:

```
OPERAZIONE\nUSERNAME[\nDATI]
```

dove `OPERAZIONE` è l'operazione che si richiede, `USERNAME` è l'username dell'utente richiedente e `DATI` sono le informazioni necessarie per soddisfare la richiesta.
I messaggi che possono essere inviai al server sono:

- `LOGIN\nUSERNAME\nPASSWORD`: login nella piattaforma;
- `LOGOUT\nUSERNAME`: logout dalla piattaforma;
- `ADDFRIEND\nUSERNAME\nFRIENDUSERNAME`: aggiunge un amico;
- `FRIENDS\nUSERNAME`: ottiene la lista degli amici;
- `CHALLENGE\nUSERNAME\nFRIENDUSERNAME`: invia una richiesta di sfida;
- `POINTS\nUSERNAME`: recupera il punteggio dell'utente;
- `RANK\nUSERNAME`: recupera la classifica;
- `WORD\nUSERNAME\nTRADUZIONE`: invia la traduzione e richiede una nuova parola;
- `QUIT\nUSERNAME`: abbandona una sfida.

Il server invia al client messaggi della forma:

```
ESITO\nDATI
```

dove `ESITO` è un intero positivo che comunica l'esito dell'operazione e `DATI` sono i dati in risposta o un messaggio di errore. I valori possibili di `ESITO` sono:

- `0`: richiesta soddisfatta;
- `1`: errore durante la richiesta;
- `2`: parole terminate (possibile solo durante una sfida);
- `3`: risultato sfida (possibile solo durante una sfida);

Nella comunicazione tramite UDP invece il server invia un messaggio contenente l'username dell'utente sfidante e il client risponde con "`OK`" se la sfida è accettata o con "`NO`" se la sfida non è accettata.

Strutture dati

Gli utenti registrati all'interno della piattaforma sono memorizzati all'interno di una HashMap dove la chiave è l'username e il valore è l'oggetto utente. La scelta di utilizzare una HashMap è data dal fatto che semplifica l'inserimento di nuovi utente e il recupero degli stessi tramite l'username.
All'interno dell'oggetto utente sono registrati gli username dei suoi amici all'interno di un HashSet. Viene utilizzato un Set per memorizzare gli amici in modo da semplificare l'inserimento di un nuovo amico, che fallisce nel caso sia già presente.

L'accesso in scrittura alla Map contenente gli utente avviene in mutua esclusione per evitare problemi potenzialmente causati da una cattiva gestione della concorrenza.

Per la serializzazione e deserializzazione dell'oggetto JSON che rappresenta una classifica, viene utilizzata la struttura dati di appoggio RankEntry, il quale identifica un singolo username con il relativo punteggio.

Thread

Relativamente alla componente Server, oltre al processo principale che si occupa della comunicazione TCP, sono presenti i seguenti thread:

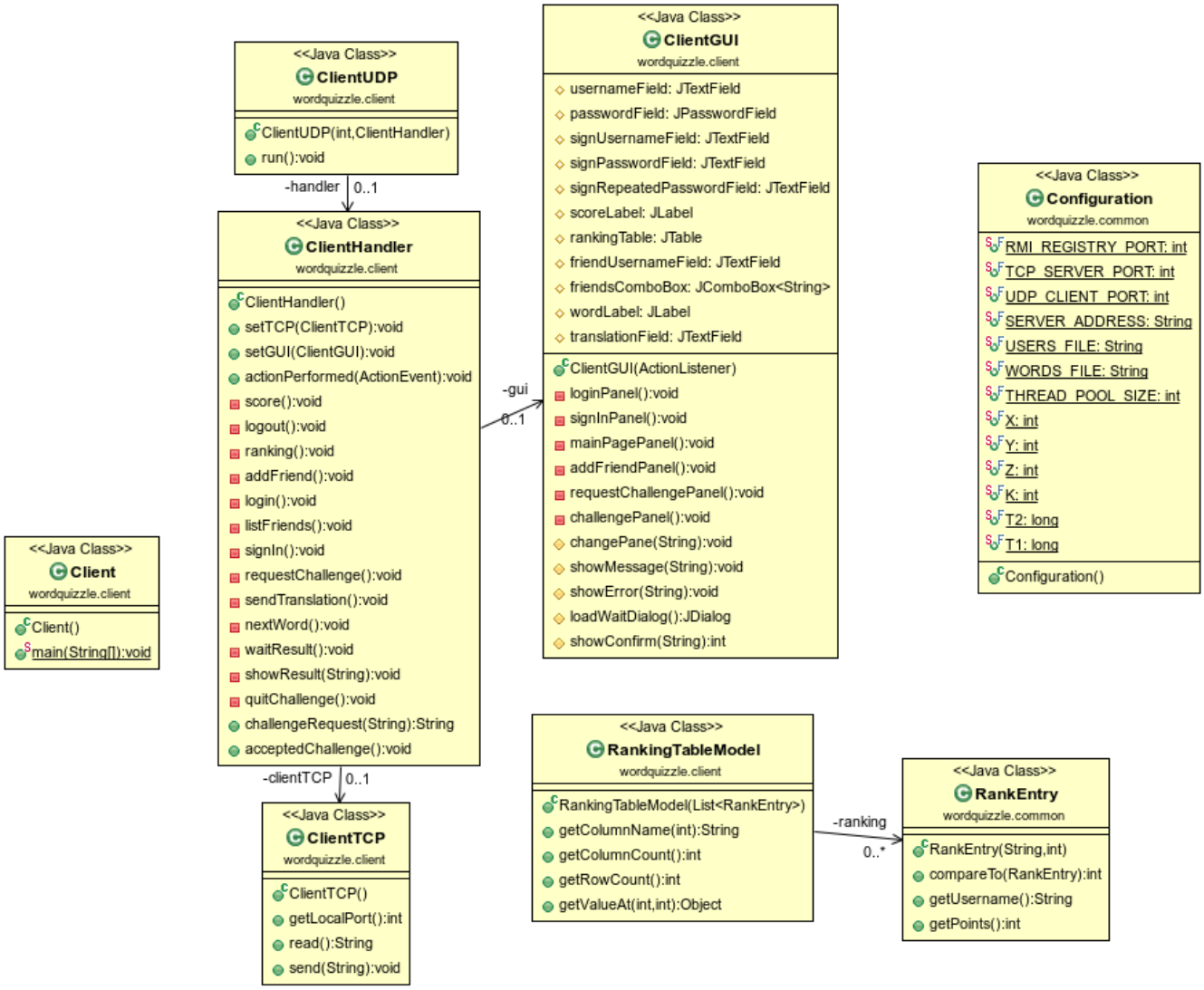
- Un thread che viene creato ogni volta che deve essere gestita una richiesta. Tali thread sono gestiti da un fixed Thread Pool. Nascono al momento in cui la richiesta viene letta e deve essere gestita e terminano quando viene elaborata una risposta.
- Un thread che viene creato ogni volta che deve essere gestita una sfida. Nasce al momento dell'accettazione di una sfida e termina al termine di essa.

Relativamente alla componente Client, oltre alla gestione della GUI e al processo principale che gestisce la comunicazione TCP, è presente un thread separato per la gestione della comunicazione UDP.

Diagramma delle classi

Per questioni di leggibilità si riporta il diagramma delle classi in due sezioni, quella del client e quella del server. Inoltre, per il medesimo motivo, non sono stati riportati gli attributi privati delle classi.

Client



Server

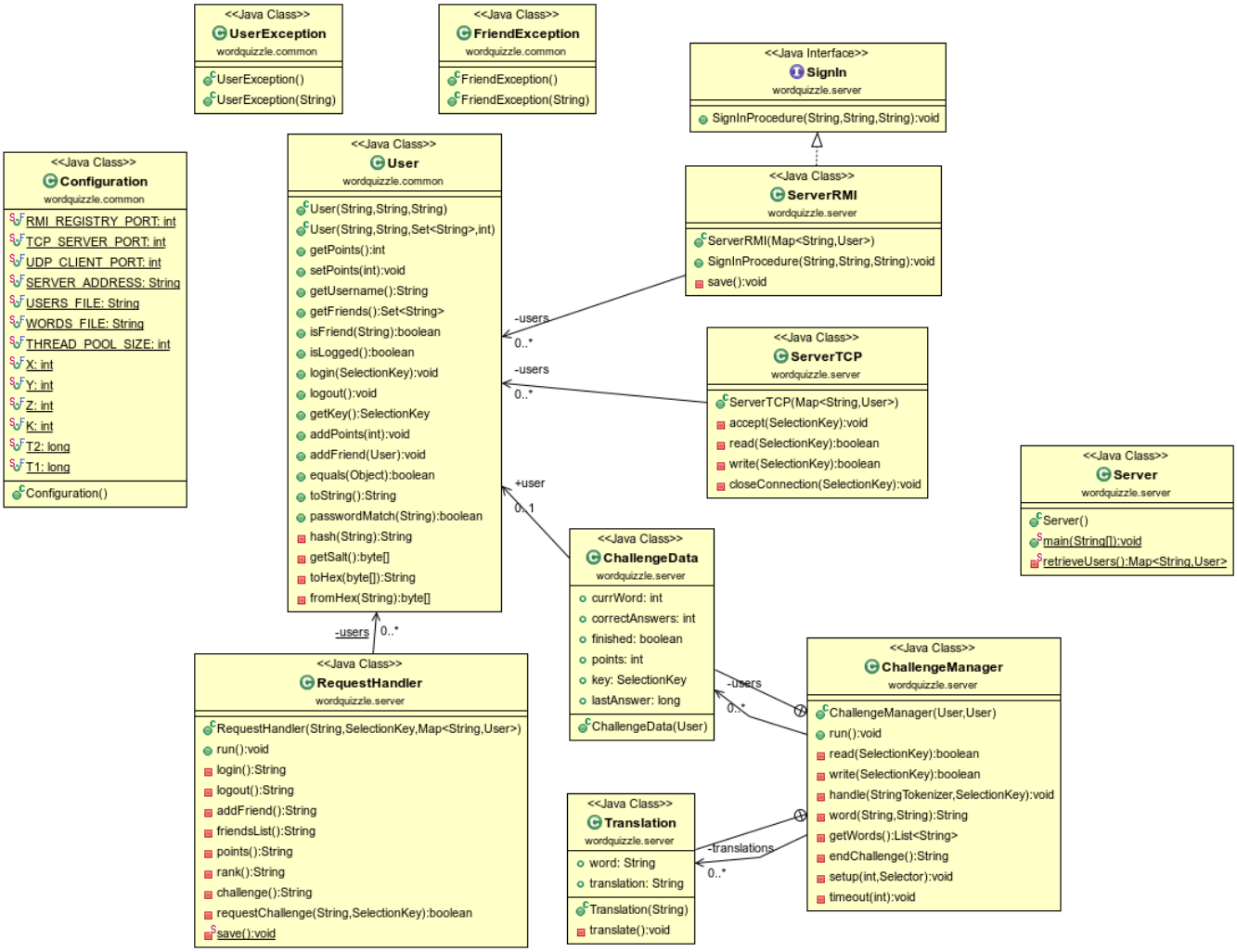
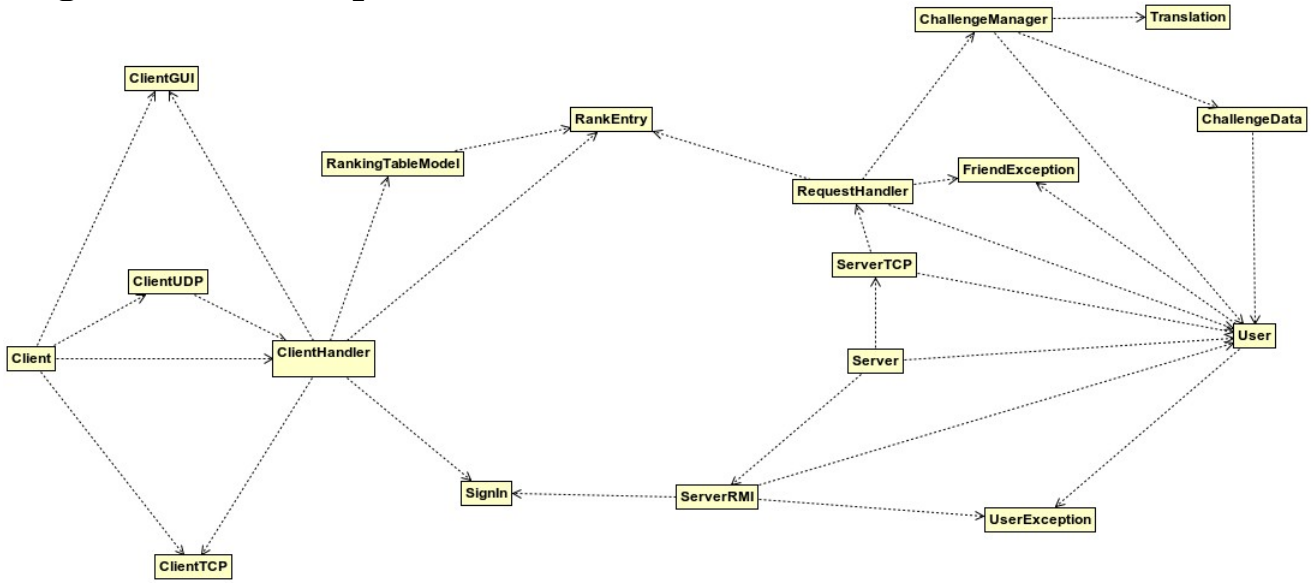


Diagramma delle dipendenze



Descrizione delle classi

Package server

Server

Classe che contiene il metodo main per avviare la componente server del servizio.
Si occupa di recuperare gli utenti da file JSON e di avviare la gestione di RMI e del server TCP.

SignIn

Interfaccia che estende Remote ed espone il metodo SignInProcedure come metodo remoto per la registrazione di un nuovo utente nella piattaforma.

ServerRMI

Implementa l'interfaccia SignIn e quindi gestisce la registrazione di nuovi utenti. Si occupa inoltre di salvare su file i nuovi utenti.

ServerTCP

Classe che gestisce la comunicazione tramite protocollo TCP del server. La gestione avviene tramite multiplexing dei canali. Crea un selettore e al momento della ricezione di un pacchetto affida la gestione della richiesta e la formulazione di una risposta ad un ThreadPool. Una volta formulata la risposta, la spedisce al client.

RequestHandler

Classe che implementa Runnable, che si occupa della gestione delle richieste provenienti dai client e la formulazione delle relative risposte. Analizza la richiesta identificando il comando richiesto dal client ed esegue di conseguenza le operazioni richieste.
Nel caso in cui un client voglia aggiungere un nuovo amico e se l'operazione va a buon fine, memorizza le modifiche su file JSON.
Nel caso in cui un client voglia effettuare una richiesta di sfida, inoltra al client sfidato la richiesta di sfida tramite protocollo UDP.
Se la richiesta di sfida viene accettata, inibisce le chiavi associate ai due client in sfida nel selettore principale, crea un thread separato per la gestione della sfida ed attende la terminazione di esso. Una volta terminato, riattiva le chiavi precedentemente inibite e salva le modifiche dei punteggi su file JSON.

ChallengeMananger

Classe che implementa Runnable, che si occupa della gestione di una sfida tra due client. Il metodo costruttore recupera K parole casuali dal dizionario ed effettua K richiesta al servizio MyMemory per la loro traduzione.

Al momento dell'invocazione del metodo run() viene creato un nuovo selettore specifico per l'interazione dei due client all'interno della sfida. Quando riceve un nuovo pacchetto, identifica la richiesta e l'origine e agisce di conseguenza. In questa fase un client può inviare unicamente i comandi "WORD" per inviare una traduzione e riceve una nuova parola e "QUIT" per abbandonare la partita.
Quando entrambi i client hanno terminato la partita, gli viene comunicato il risultato e aggiorna i punteggi degli utenti coinvolti nella sfida.

ChallengeData

Classe privata di ChallengeManager che contiene al suo interno i dati di un utente in sfida. In particolare colleziona i dati dell'utente, la parola che attualmente deve tradurre, il numero di parole indovinate, un flag che indica se ha terminato con le traduzioni o meno e il tempo in cui ha dato l'ultima risposta.

Translation

Classe privata di ChallengeManager che identifica una coppia <parola, traduzione>. Al momento della creazione dell'oggetto riceve la parola e si occupa di contattare il servizio MyMemory per ottenere la traduzione.

Package client

Client

Classe che contiene il metodo main per avviare la componente client del servizio. Avvia il gestore delle richieste, il gestore della comunicazione TCP, il gestore della comunicazione UDP e il gestore della GUI.

ClientGUI

Si occupa della gestione della parte grafica dell'applicativo client. Crea le varie schermate che verranno visualizzate durante l'utilizzo, registra il Listener per la gestione degli eventi generati dalle componenti grafiche ed espone dei metodi per cambiare schermata e mostrare delle dialog di informazione, di errore, di conferma o di attesa.

RankingTableModel

Classe che estende AbstractTableModel e rappresenta il modello per la tabella contenente la classifica.

ClientTCP

Classe che si occupa di gestire la comunicazione con il server tramite TCP. Apre un canale di comunicazione con il server ed espone i metodi per scrivere e leggere dal canale.

ClientUDP

Classe che implementa Runnable e che si occupa di gestire la comunicazione con il server tramite UDP per l'accettazione di richieste di sfida. Si mette in attesa di un datagramma in entrata e al momento della ricezione passa la gestione a ClientManager che formulerà una risposta da comunicare al server.

ClientHandler

Classe che implementa ActionListener e si occupa di gestire gli eventi provenienti dai componenti grafici e quindi l'interazione con l'utente. Riconosce l'origine dell'evento da gestire e in base ad esso formula una richiesta da inviare al server e, una volta ricevuta una risposta, comunica all'utente l'esito della richiesta o agisce per completare la richiesta.

Package common

Configuration

Classe che contiene i parametri di configurazione della piattaforma.

RankEntry

Classe che rappresenta un elemento della classifica, composto da una coppia <username, punteggio>

User

Classe che rappresenta un utente della piattaforma. Contiene le informazioni relative allo username, la password, la lista degli amici, il punteggio, se è loggato o meno e la eventuale SelectionKey relativa al selettore principale del server.
Al momento della creazione di un nuovo utente verifica che i dati comunicati sia validi e crea un hash della password.
Espone, tra gli altri, i metodi per verificare se la password inserita è corretta e aggiungere un amico.

FriendException

Classe che estende Exception e che rappresenta un'eccezione lanciata da User durante l'aggiunta di un nuovo amico nel caso in cui i dati comunicati non siano corretti.

UserException

Classe che estende Exception e che rappresenta un'eccezione lanciata da User durante la creazione di un nuovo utente nel caso in cui i dati comunicati non siano corretti.

Istruzioni per la compilazione

Dalla directory principale del progetto (WordQuizzle) è possibile eseguire il seguente comando per effettuare la compilazione:

```
javac -d ./bin -cp gson-2.8.6.jar src/wordquizzle/common/*
src/wordquizzle/server/* src/wordquizzle/client/*
```

Sono inoltre allegati al progetto i file WordQuizzleServer.jar e WordQuizzleClient.jar.

Istruzioni per l'esecuzione

Per eseguire il server, è necessario eseguire dalla directory principale del progetto il seguente comando:

```
java -classpath ./bin:./gson-2.8.6.jar wordquizzle.server.Server
```

Similarmente, per eseguire il client, si deve eseguire il comando:

```
java -classpath ./bin:./gson-2.8.6.jar wordquizzle.client.Client
```

Screenshot dell’applicativo

Di seguito si riportano alcuni screenshot che mostrano l’interfaccia lato client dell’applicativo.

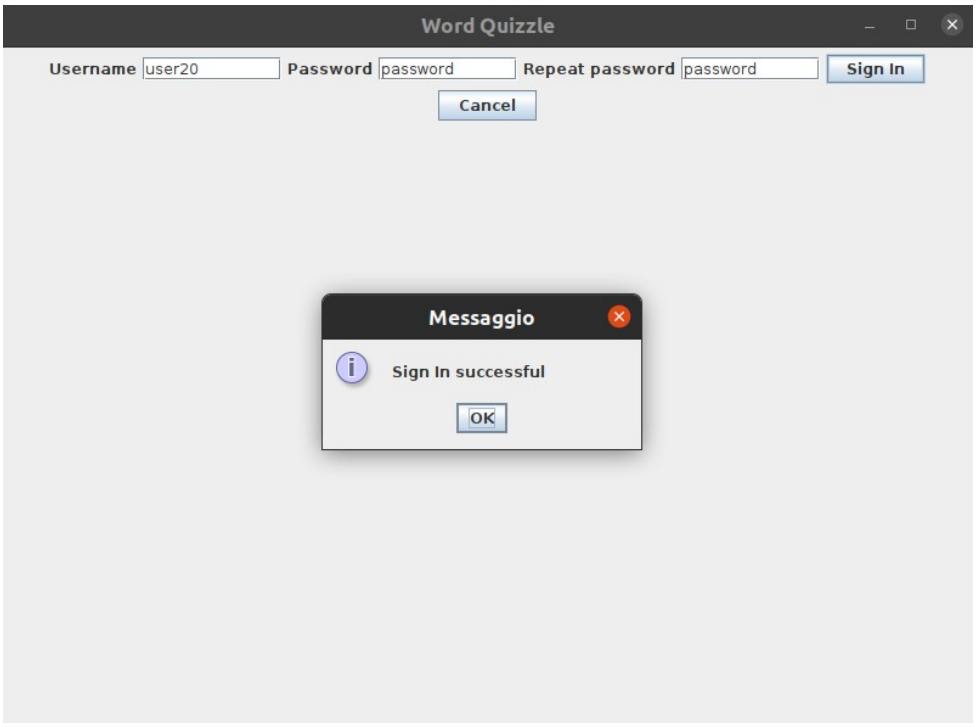


Figura 1: Registrazione

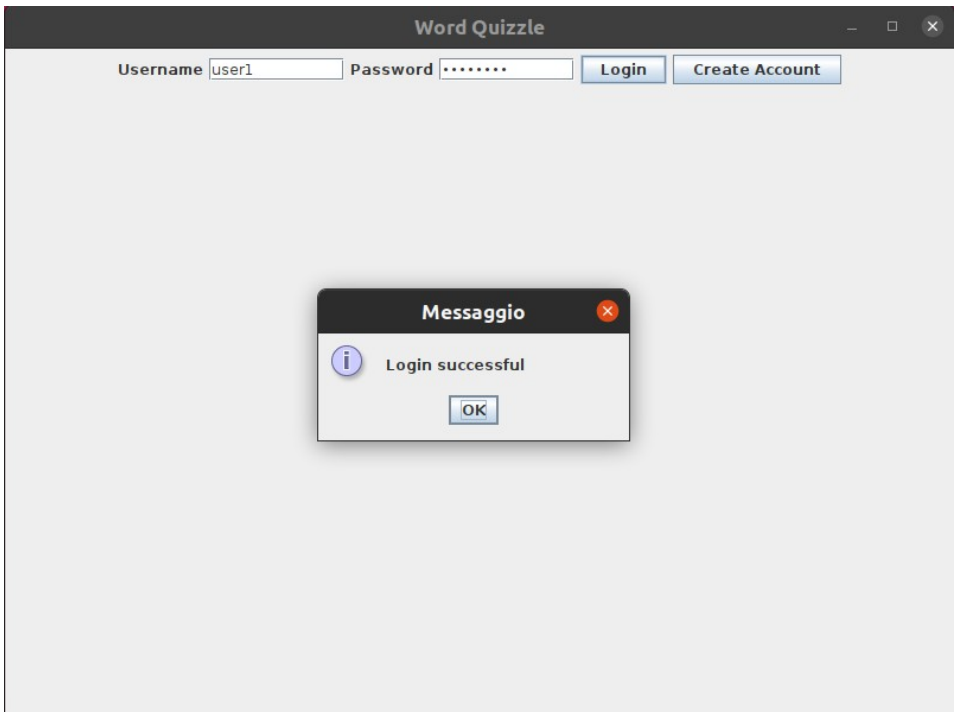


Figura 2: Login

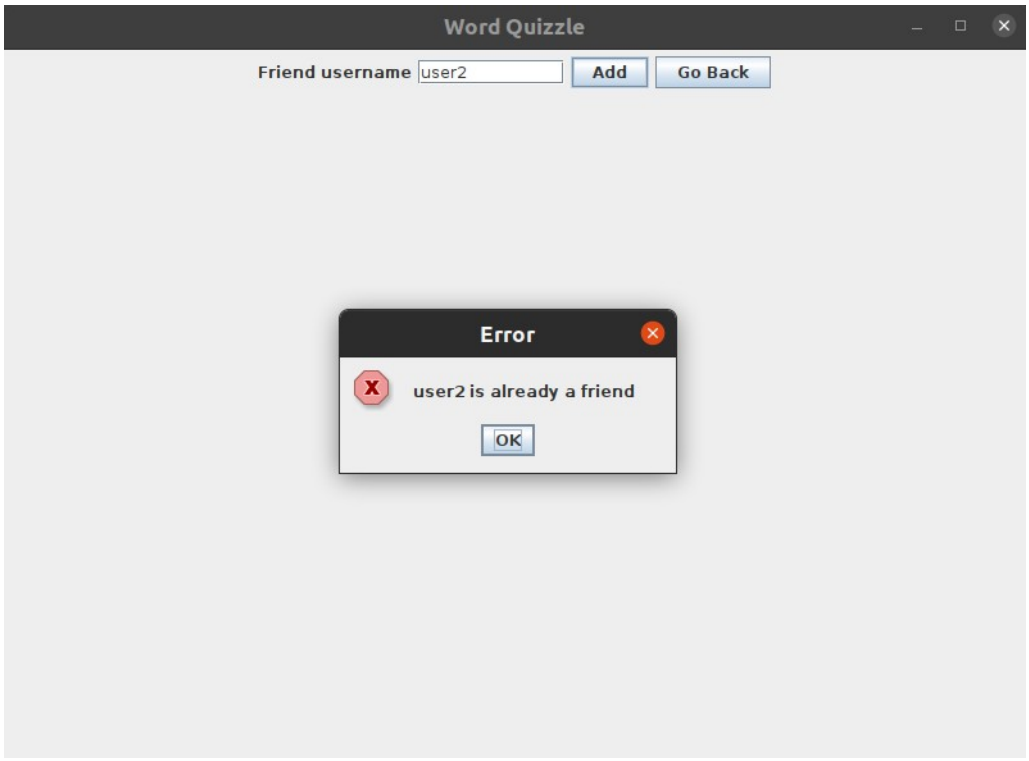


Figura 3: Aggiunta amico

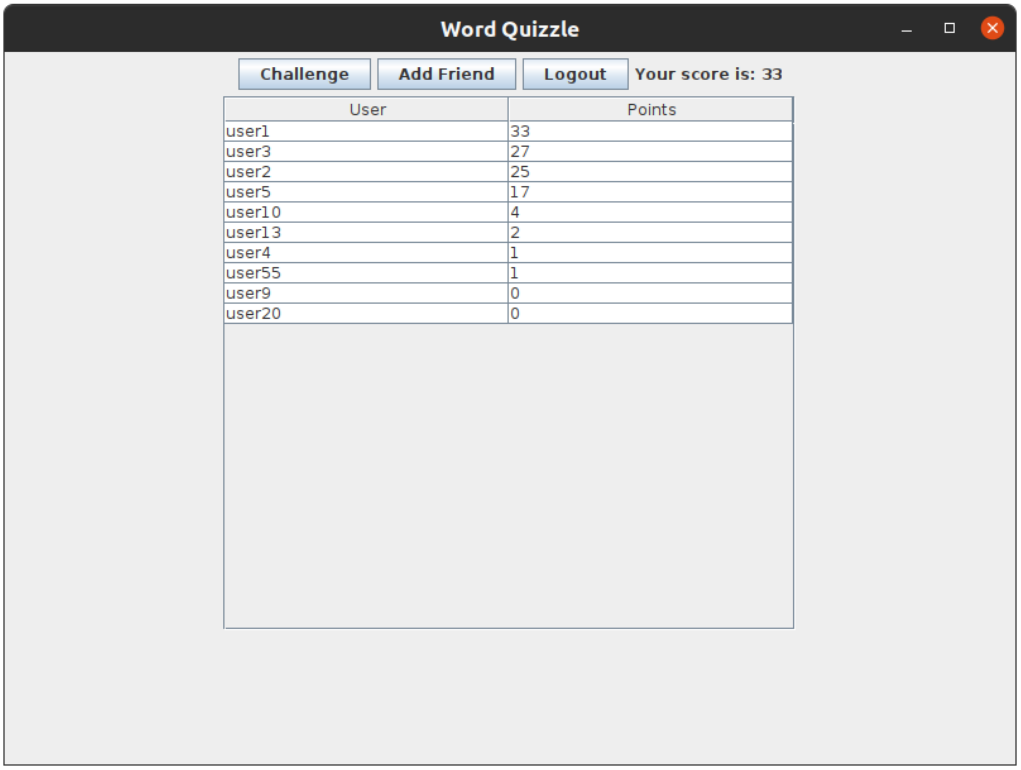


Figura 4: Pagina principale

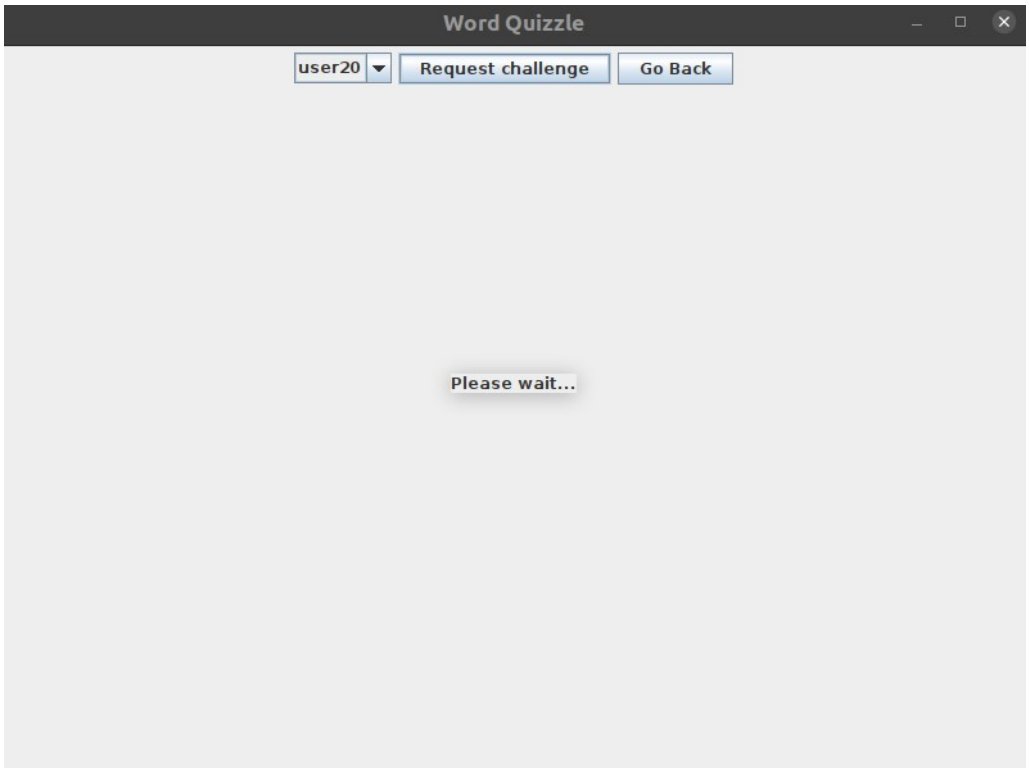


Figura 5: In attesa di accettazione sfida

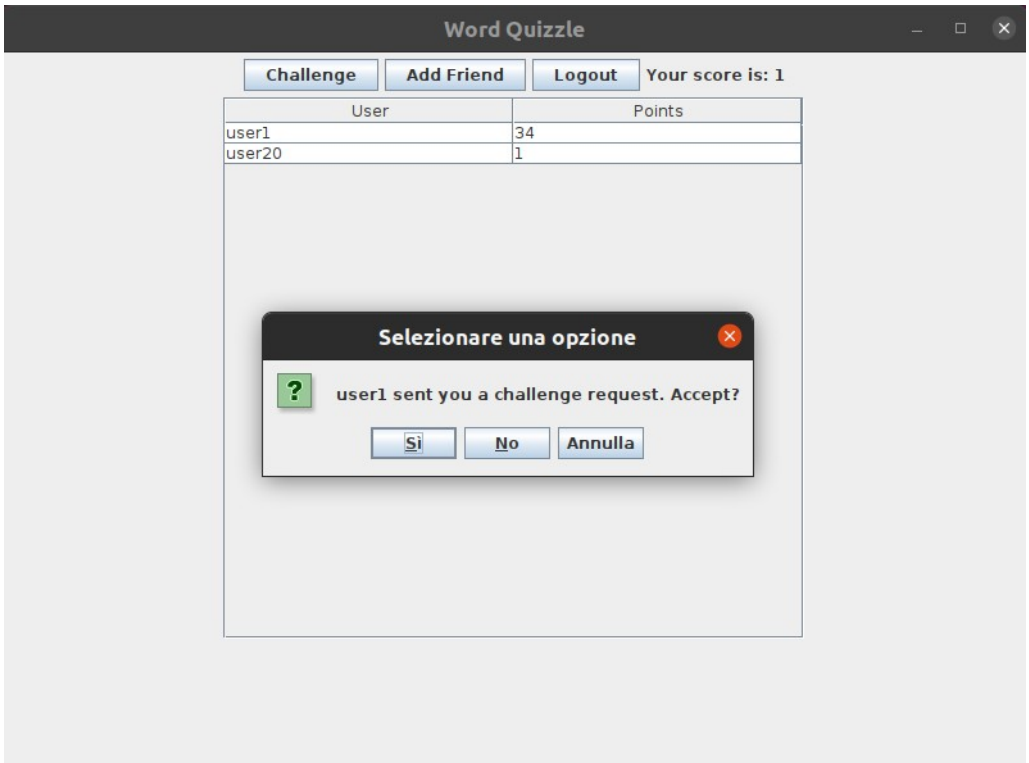


Figura 6: Richiesta di sfida

Figura 7: Sfida in corso

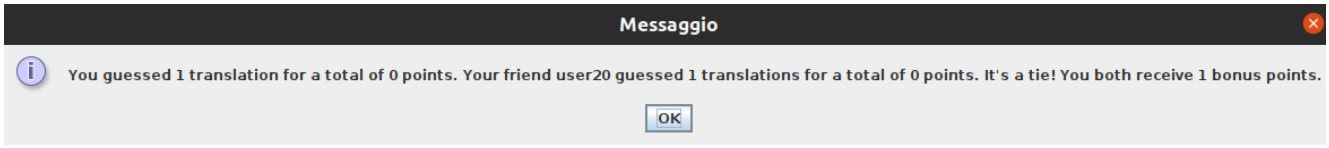


Figura 8: Risultato sfida