Laboratorio di Reti, Corsi A e B
Word Quizzle (WQ)
Progetto di Fine Corso
A.A. 2019/20

1. Descrizione del Problema

Il progetto consiste nell'implementazione di un sistema di sfide di traduzione italiano-inglese tra utenti registrati al servizio. Gli utenti registrati possono sfidare i propri amici ad una gara il cui scopo è quello di tradurre in inglese il maggiore numero di parole italiane proposte dal servizio. Il sistema consente inoltre la gestione di una rete sociale tra gli utenti iscritti. L'applicazione è implementata secondo una architettura client server.

2. WQ: specifica delle operazioni

Di seguito sono specificate le operazioni offerte dal servizio WQ. In sede di implementazione è possibile aggiungere ulteriori parametri se necessario.

Registrazione di un utente: per inserire un nuovo utente, il server mette a disposizione una operazione *registra_utente(nickUtente,password).* Il server risponde con un codice che può indicare l'avvenuta registrazione, oppure, se il nickname è già presente, o se la password è vuota, restituisce un messaggio d'errore. Come specificato in seguito, le registrazioni sono tra le informazioni da persistere.

utilizzare file/database per mem info utente così se il server crasha si mantengono, non importa sicurezza pswd

login(nickUtente, password): Login di un utente già registrato per accedere al servizio. Il server risponde con un codice che può indicare l'avvenuto login, oppure, se l'utente ha già effettuato la login o la password è errata, restituisce un messaggio d'errore.

logout(nickUtente): effettua il logout dell'utente dal servizio.

aggiungi_amico (nickUtente, nickAmico): registrazione di un'amicizia: aggiungere un amico alla cerchia di amici di un utente. Viene creato un arco non orientato tra i due utenti (se A è amico di B, B è amico di A). Il Server risponde con un codice che indica l'avvenuta registrazione dell'amicizia oppure con un codice di errore, se il nickname del nodo destinazione/sorgente della richiesta non esiste, oppure se è stato richiesto di creare una relazione di amicizia già esistente. Non è necessario che il server richieda l'accettazione dell'amicizia da parte di *nickAmico.*

lista_amici(nickUtente): utilizzata da un utente per visualizzare la lista dei propri amici, fornendo le proprie generalità. Il server restituisce un oggetto JSON che rappresenta la lista degli amici.

sfida(nickUtente, nickAmico): l'utente nickUtente intende sfidare l'utente di nome nickAmico. Il server controlla che nickAmico appartenga alla lista di amicizie di nickUtente, in caso negativo restituisce un codice di errore e l'operazione termina. In caso positivo, il server invia a nickAmico una richiesta di accettazione della sfida e, solo dopo che la richiesta è stata accettata, la sfida può avere inizio (se la risposta non è stata ricevuta entro un intervallo di tempo T1 si considera la sfida come non accettata). La sfida riguarda la traduzione di una lista di parole italiane in parole inglesi, nel minimo tempo possibile.
Il server sceglie, in modo casuale, K parole da un dizionario contenente N parole italiane da inviare successivamente, una alla volta, ai due sfidanti. La partita può durare al massimo un intervallo di tempo T2. Il server invia ai partecipanti la prima parola. Quando il giocatore invia la traduzione (giusta o sbagliata), il server invia la parola successiva a quel giocatore.
Il gioco termina quando entrambi i giocatori hanno inviato le traduzioni alle K parole o quando scade il timer.
La correttezza della traduzione viene controllata dal server utilizzando un servizio esterno, come specificato nella sezione seguente. Ogni traduzione corretta assegna X punti al giocatore; ogni traduzione sbagliata assegna Y punti negativi; il giocatore con più punti vince la sfida ed ottiene Z punti extra. Per ogni risposta non inviata (a causa della scadenza del timer) si assegnano 0 punti. Il punteggio ottenuto da ciascun partecipante alla fine della partita viene chiamato punteggio partita.
I valori espressi come K, N, T1, T2, X, Y e Z sono a discrezione dello studente.

Esempio di svolgimento della partita:
I giocatori U1 e U2 si sfidano. A inizio partita il server seleziona le parole "Bottiglia", "Quarantadue" e "Rete" dal dizionario. Interroga il servizio e memorizza le traduzioni. Infine il server setta un timeout della partita di 1 minuto.
Il server invia "Bottiglia" ad entrambi i giocatori. U1 risponde con "Bottle": il server assegna 2 punti a U1 (punteggio corrente di U1: 2) e invia "Quarantadue". Nel frattempo U2 risponde con "Botle", sottrae 1 punto a U2 (punteggio corrente di U2: -1) e invia "Quarantadue" a U2.
Supponendo che U1 sbagli le 2 parole successive, alla fine della partita totalizza un punteggio di 0 punti (+2 -1 -1), e supponendo che U2 indovini entrambe le parole successive, totalizza un punteggio di 3 punti (-1 +2 +2). Il server dichiara U2 vincitore ed assegna ad U2 3 punti extra.

mostra_punteggio(nickUtente): il server restituisce il punteggio di nickUtente (chiamato "punteggio utente") totalizzato in base ai punteggi partita ottenuti in tutte le sfide che ha effettuato.

mostra_classifica(nickUtente): Il server restituisce in formato JSON la classifica calcolata in base ai punteggi utente ottenuti da nickUtente e dai suoi amici.

3. WQ: specifiche per l'implementazione

Nella realizzazione del progetto devono essere utilizzate molte tecnologie illustrate durante il corso. In particolare:

- la fase di registrazione viene implementata mediante RMI.
- La fase di login deve essere effettuata come prima operazione dopo aver instaurato una connessione TCP con il server. Su questa connessione TCP, dopo previa login effettuata con successo, avvengono le interazioni client- server (richieste/risposte).
- Il server inoltra la richiesta di sfida originata da nickUtente all'utente nickAmico usando la comunicazione UDP.
- Il server può essere realizzato multithreaded oppure può effettuare il multiplexing dei canali mediante NIO.
- Il server gestisce un dizionario di N parole italiane, memorizzato in un file. Durante la fase di setup di una sfida fra due utenti il server seleziona K parole a caso su N parole presenti nel dizionario. Prima dell'inizio della partita, ma dopo che ha ricevuto l'accettazione della sfida da parte dell'amico, il server chiede, tramite una chiamata HTTP GET, la traduzione delle parole selezionate al servizio esterno accessibile alla URL https://mymemory.translated.net/doc/spec.php. Le traduzioni vengono memorizzate per tutta la durata della partita per verificare la correttezza delle risposte inviate dal client.
- L'utente interagisce con WQ mediante un client che può utilizzare una semplice interfaccia grafica, oppure una interfaccia a linea di comando, definendo un insieme di comandi, presentati in un menu.
- Il server persiste le informazioni di registrazione, relazioni di amicizia e punteggio degli utenti su file json.

4. Modalità di svolgimento e consegna

Il materiale consegnato deve comprendere:

- il codice dell'applicazione e di eventuali programmi utilizzati per il test delle sue funzionalità. E' importante che il codice segua le convenzioni JAVA e le linee guida date per gli assegnamenti;
- la relazione in formato pdf che deve contenere:
    - una descrizione generale dell'architettura complessiva del sistema, in cui sono motivate le scelte di progetto;
    - uno schema generale dei threads attivati da ogni componente e delle strutture dati utilizzate, con particolare riferimento al controllo della concorrenza;
    - una descrizione sintetica delle classi definite ed indicazioni precise sulle modalità di esecuzione.

- una sezione di istruzioni su come compilare ed eseguire il progetto (librerie esterne usate, argomenti da passare al codice, sintassi dei comandi per eseguire le varie operazioni...). Questa sezione deve essere un manuale di istruzioni semplice e chiaro per gli utilizzatori del sistema.
- L'organizzazione e la chiarezza della relazione influiranno sul voto finale.

Il codice che non compila non verrà considerato. Per essere considerato sufficiente e quindi ammissibile per la discussione, il progetto deve implementare tutte le funzioni precedentemente illustrate
Relazione e codice sorgente devono essere consegnati su Moodle. La relazione deve anche essere consegnata presso la portineria del Dipartimento.

Per domande di chiarimento sul progetto verrà attivato un forum di discussione su moodle per ciascuno dei moduli di Laboratorio A e B su cui potrete pubblicare le vostre domande.

5. Esempio di interfaccia CLI

Lista dei comandi

Di seguito la lista dei comandi che il programma client offre all'utente. La CLI deve prevedere i seguenti comandi con i rispettivi argomenti per interagire con il server WQ.

```
$ wq --help
usage : COMMAND [ ARGS ...]
Commands:
 registra_utente <nickUtente > <password > registra l' utente
 login <nickUtente > <password > effettua il login
 logout effettua il logout
 aggiungi_amico <nickAmico> crea relazione di amicizia con nickAmico
 lista_amici mostra la lista dei propri amici
 sfida <nickAmico > richiesta di una sfida a nickAmico
 mostra_punteggio mostra il punteggio dell'utente
 mostra_classifica mostra una classifica degli amici dell'utente (incluso
l'utente stesso)
```

Di seguito un esempio di una possibile interazione client server.
Compilazione ed esecuzione del Client di WQ.

```
$ javac MainClassWQClient.java
$ java MainClassWQClient
```

L'utente si registra al servizio WQ.

```
> registra_utente Mario 1234
Registrazione eseguita con successo.
```

L'utente effettua il login con username e password.

```
> login Mario 1234
Login eseguito con successo
```

L'utente crea un’amicizia

```
> aggiungi_amico Luisa
Amicizia Mario-Luisa creata
```

L'utente chiede di visualizzare la lista delle amicizie.

```
> lista_amici
Luisa, Marco, Giovanni
```

L'utente sfida un amico

```
> sfida Luisa
Sfida a Luisa inviata. In attesa di accettazione
```

Mostra il punteggio dell'utente

```
> mostra_punteggio
Punteggio: 16
```

```
Mostra la classifica della cerchia di amici dell’utente
> mostra_classifica
Classifica: Luisa 31, Marco 25, Mario 16, Giovanni 10
```

L'utente effettua il logout.

```
> logout
Logout eseguito con successo
```

Svolgimento del gioco

Di seguito un esempio di svolgimento del gioco. Il server invia una parola e l’utente risponde con la traduzione. Dopo la risposta l’utente riceve la parola successiva, così fino a che non risponde a tutte le parole oppure il tempo scade.

```
Via alla sfida di traduzione!
Avete 60 secondi per tradurre correttamente 8 parole.
Challenge 1/8: Bottiglia
> Bottle
...
Challenge 8/8: Bicchiere
```

> Glass

Il server invia l’esito della sfida
Hai tradotto correttamente 4 parole, ne hai sbagliate 3 e non risposto a 1.
Hai totalizzato 5 punti.
Il tuo avversario ha totalizzato 3 punti.
Congratulazioni, hai vinto! Hai guadagnato 3 punti extra, per un totale di 8 punti!