Laboratorio di Reti
Wordle: un gioco di parole 3.0

Giovanni Guerrazzi 580384

2022/2023

Contents

1 Server

1.1 Intro

Il server è rappresentato dalla classe WordleServerMain. Riesce a gestire le varie comunicazioni con i diversi client grazie al thread pool. Per ogni nuova connessione stabilita da parte di un nuovo client il server manda in esecuzione un thread che gestisce interamente la richiesta di quest'ultimo. I thread sono implementati dalla classe 'Worker' presente all'interno del Server. Per quanto riguarda il thread pool ho scelto di implementarlo come se fosse un *'CachedThreadPool'* in modo da renderlo più dinamico: pochi thread attivi contemporaneamente in base alle connessioni attive presenti che verrano aumentati nel caso in cui ci fossero più richieste (più client). Client e server si scambiano messaggi sulla connessione TCP creata un volta che il client è riuscito a effetturare correttamente il login. Per quanto riguarda lo scambio di messaggio tra client e server ho utilizzato Java IO quindi gli stream,in particolare ho utilizzato *ObjectOutputStream* e *ObjectInputStream* in quanto le informazioni scambiate tra client e server contenevano anche oggetti e quindi c'è stata la necessità di serializzare questi ultimi per inviarli sulla socket stream e deserializzarli per leggerli.

1.2 Strutture Dati

Le strutture dati principali del server sono le seguenti:

- **ConcurrentHashMap<String,String> DataBase**
- **ArrayList<User> UsersStats**
- **LinkedList <UserRank> Rank**

La struttura DataBase contiene al suo interno tutti gli utenti che si sono correttamente registrati al server tramite RMI. Ad ogni utente (chiave) che è univoco in quanto non è possibile che siano presenti più utenti con lo stesso username, è associata la sua password (valore) che servirà successivamente in fase di login per autenticarsi.

La struttura UserStats contiene tutte le 'schede' giocatore dove vengono memorizzate le informazioni di quest'ultimo come le partite disputate,partite vinte,perse,win rate... Sostanzialmente questa struttura dati serve per memorizzare tutte le statistiche degli utenti che vengono aggiornate dopo ogni partita.

La struttura Rank contiene la classifica degli utenti che hanno giocato a Wordle.Le prime posizioni appartengono all'utente che ha un 'wordleScore' minore (un valore di wordleScore minore implica una posizione in classifica più alta).

Le strutture dati UsersStats e Rank all'interno del server vengono accedute in mutua esclusione grazie alle reentrant lock 'StatsUserLock' e 'RankUserLock' in quanto potrebbero esserci più thread che ci accedono in contemporanea andando a generare un'inconsistenza nelle strutture e quindi si andrebbero a manomettere le strutture dati.

1.3 Avvio

Al primo lancio il server proverà a ricaricare il proprio stato tramite la funzione 'loadServerStatus'.Ovviamente al primo avvio non saranno presenti i file contenenti lo stato del server e quindi il server non avrà uno stato (le sue strutture dati saranno vuote). Durante l'esecuzione, ci sarà un thread (threadSaver) che andrà a memorizzare le strutture dati del server sui file appositi e quindi nei seguenti avvii il server ricostruirà il proprio stato andando a leggere da questi file ripristinando le proprie strutture dati.

Il vocabolario di Wordle ovvero words.txt verrà anche questo caricato grazie alla funzione 'loadVocabolary' e memorizzato nella struttura
ArrayList<String> Vocabolary

1.4 Threads

All' interno della classe WordleServerMain vengono avviati quattro diversi thread una volta che il server viene lanciato:

- **threadGenWord**
- **threadListener**
- **threadSaver**
- **threadEnder**

1.4.1 threadGenWord

Questo è il thread che si occupa di generare ogni 'newWordTime' millisecondi (parametro passato al file di configurazione) una nuova parola segreta che gli utenti connessi al server che vogliono giocare dovrenno indovinare. Questo thread chiama la funzione 'generateNewSecretWord' che si assicura che la parola estratta non sia già precedentemente uscita.

1.4.2 threadListener

Questo è il thread del server che ha lo scopo di rimanere in ascolto sul DatagramSocket dove i client manderanno i pacchetti contenenti i loro risultati delle partite effettuate. A questo punto il thread invia sul gruppo di multicast i pacchetti in modo che questi ultimi arrivino a tutti i client che si sono correttamente loggati.

1.4.3 threadSaver

Questo è il thread che si occupa del salvataggio delle strutture dati principali del server Database,UsersStats e Rank su dei file appositi chiamati rispettivamente DataBase.json, UsersStats.json e Rank.json. Il salvataggio di queste strutture dati avviene ogni 'saveTime' grazie alla funzione 'save()' chiamata dal thread. Alla terminazione del server, viene chiamata un' ultima volta la funzione 'save()'.

1.4.4 threadEnder

Questo è il thread responsabile della terminazione del server. Il thread prende in input una stringa inviata da linea di comando. Se quest'ultima coincide con la stringa 'end' allora inizia la procedura di terminazione settando la variabile globale 'end' a true.

1.5 Worker

Questa è la classe contenuta all'interno del file WordleServerMain che implementa i thread che gestiscono le connessioni con i vari client. Ogni volta che un nuovo client decide di connettersi al server per iniziare una nuova partita viene lanciato un nuovo thread 'Worker' il quale permetterà al server di gestirlo interamente.

1.6 Terminazione

Per quanto riguarda la terminazione del server basta digitare da linea di comando 'end'. Questo comando imposterà una variabile globale (end == true) che permetterà al server di poter cominciare le operazioni di terminazione. Infatti c'è un thread apposito, il 'threadEnder' che si occupera di avviare le pratiche di terminazione (vedi dopo per quanto riguarda le specifiche del threadEnder). Il server proverà a terminare subito: se non sono presenti client collegati cosi sarà. Se invece sono presenti dei client collegati, il server non termina subito ma attende un 'terminationDelay' in millisecondi nel quali i client possono finire le proprie operazioni. Se, aspettato questo tempo ci sono ancora client collegati, il server forzerà una chiusura terminando. Alla terminazione i dati contenuti nelle strutture dati principali vengono memorizzati su dei file appositi che serviranno per avvii futuri a ricostruire lo stato del server.

2 Client

2.1 Intro

Il client è rappresentato dalla classe WordleClientMain. Una volta lanciato il client l'utente ha a disposizione varie opzioni che vengono stampate a schermo. Se un utente vuole avere più informazioni può digitare il comando 'help'. La prima operazione che un client deve fare per poter giocare a Wordle è registrarsi. La registrazione avviene mediante RMI e solo dopo quest'ultima l'utente potrà loggarsi e iniziare una partita.

2.2 Strutture Dati

Le strutture dati principali del client sono le seguenti:

- **ArrayList<String> OwnResult**
- **ArrayList<String> AllResult**

La struttura OwnResult permette al client di tenere traccia dei propri risultati dopo ogni partita disputata. I risultati consistono nelle stringhe inviate dal server come aiuto per indovinare la parola segreta. Un esempio può essere "+++??+xx??" dove:

- **x** indica che la lettera non appartiene alla parola segreta.
- **?** indica che la lettera appartiene alla parola da indovinare ma si trova nella posizione sbagliata.
- **+** indica che la lettera appartiene alla parola segreta e si trova nella giusta posizione.

In questo modo, quando l'utente ha intenzione di condividere i propri risultati può farlo senza andare a svelare la parola segreta.

La struttura AllResult è simile a OwnResult però non contiene solo i propri risultati ma contiene invece quello di TUTTI gli utenti che hanno disputato delle partite e hanno voluto condividere i propri risultati sul gruppo di multicast.

2.3 Thread

Nel client è presente un thread che in modalità del tutto asincrona rispetto al resto del programma rimarrà in attesa di pacchetti inviati sul gruppo di multicast. Questo compito è svolto dal 'threadShare'. I risultati ottenuti verranno memorizzati nella struttura dati 'AllResults'.

3 RMI

In questo progetto sono presenti 3 differenti classi che vanno ad implementare le corrispettive RMI interface.

- **RegistrationRmi**
- **CallbackRmi**
- **NotifyEventRmi**

  Le prime due classi sono esportate dal server e utilizzate dal client. In particolare la 'RegistrationRmi' è utilizzata dal client per effettuare la registrazione al server Wordle. La classe 'CallbackRmi' è sempre utilizzata dal client e questa serve per registrarsi al sistema di notifica del cambiamento delle prime tre posizioni in classifica.

L'ultima classe 'NotifyEventRmi' invece non viene esportata dal client ma comunque viene utilizzata dal server per aggiungere i nuovi client che effettuano il login in una struttura apposita chiamata 'Clients' in modo tale da avere una lista di client registrati che devono essere notificati che c'è stato un cambiamento della top 3.

4 WordleScore

Per quanto riguarda il calcolo del punteggio di un utente che sarà quello che poi ne determinerà la sua posizione in classifica ho deciso di adottare Il Wordle Average Score (WAS). Questo score tiene di conto del numero complessivo di partite giocate (e quindi anche delle parole che non sono state indovinate) ed è basato sulla guess distribution. La guess distribution di un utente è un vettore di interi avente lunghezza pari al numero massimo di tentativi. La posizione k-esima del vettore contiene il numero di parole indovinate dall'utente con esattamente k tentativi. Facendo in questa maniera non sono avvantaggiati gli utenti che giocano più partite, bensì quelli che giocano meglio, indovinando con meno tentativi la parola segreta. Conta di più la qualità che la quantita!

5 Utility

Ho utilizzato la classe Utility come classe di supporto per la lettura e la scrittura delle strutture dati sui file di memorizzazione (.json) . In particolare :

- **DataBase.json** conterrà la struttura DataBase
- **UsersStats.json** conterrà la struttura UsersStats
- **Rank.json** conterrà la struttura Rank

6 Compilazione ed Esecuzione

Per la **compilazione** del progetto bisognerà collocarsi nella cartella *'src'* e digitare il seguente comando da terminale:

- *javac -cp ".;./gson-2.8.9.jar" *.java* oppure
- *javac -cp ".:./gson-2.8.9.jar" *.java*

Per quanto riguarda **l'esecuzione** invece bisognerà aprire 2 terminali differenti e digitare per quanto riguarda il server:

- *java -cp ".;./gson-2.8.9.jar" WordleServerMain* oppure
- *java -cp ".;./gson-2.8.9.jar" WordleServerMain*

  mentre per quanto riguarda il client:

- *java -cp ".;./gson-2.8.9.jar" WordleClientMain* oppure
- *java -cp ".:./gson-2.8.9.jar" WordleClientMain*