Wordle

Reti di Calcolatori e Laboratorio

Marco Perugini – Matricola: 581493

Il Progetto

Consiste nello sviluppo di Wordle, un gioco virale del 2021 in cui gli utenti sono chiamati ad indovinare una parola di lingua inglese con un massimo di 5 tentativi. Al termine di ogni partita viene anche mostrata una traduzione in italiano della parola che doveva essere indovinata.

Oltre alla possibilità di giocare una partita l'utente può anche visualizzare le proprie statistiche, condividere una partita con gli altri utenti, vedere le partite condivise e vedere la classifica con i tre giocatori migliori.

L'applicazione deve essere implementata utilizzando un'architettura Client-Server e con l'utilizzo di TCP e UDP nonché tramite RMI.

Tutte le informazioni riguardo agli utenti devono essere salvate in un file JSON che viene letto dal server ad ogni nuovo avvio e aggiornato prima della chiusura.

L'interfaccia di entrambi è stata implementata tramite CLI.

Funzionalità implementate

Il sistema è quindi diviso in Client e Server. Il server si occupa di offrire tutti i servizi per far si che i vari client possano giocare a Wordle.

Server

Il server si avvia automaticamente una volta fatto partire l'eseguibile, prende tutte le informazioni utili per l'avvio dal file config.json (es. indirizzo, numero di porta, ogni quanto generare una nuova parola, ...). Stamperà quindi sulla CLI vari messaggi sul suo stato così da rendere chiaro cosa stia facendo e quali azioni sono richieste dai vari client. In caso dovessero presentarsi degli errori verranno anch'essi stampati.

Client

All'avvio il client utilizza sempre il file di configurazione per reperire le informazioni su come connettersi al server e poi presenta all'utente un semplice menù in cui poter effettuare una registrazione o un login, dopodiché l'utente avrà accesso ad un menù secondario in cui poter effettuare tutte le altre operazioni.

Di seguito la lista delle funzionalità implementate:

REGISTRAZIONE

Come richiesto la fase di registrazione è stata implementata utilizzando RMI. Al suo avvio il server utilizza la porta successiva a quella indicata nel file di configurazione per l'interfaccia di

registrazione, quando un client riceve una richiesta per una nuova registrazione utilizzerà quella porta e la chiave “*WordleService*” per completare l’operazione.
In caso di username già utilizzato o password vuota il server ritorna un errore

LOGIN / LOGOUT

La fase di login è invece gestita tramite lo scambio di messaggi TCP (ogni messaggio ha la struttura messaggio:risultato). In particolare dopo aver letto da input username e password il client invia al server un messaggio di tipo *login:username password* il quale si occupa di controllare se ci sono errori e invia poi un messaggio con l’esito.

Se l’operazione è completata con successo allora il client mostrerà un secondo menù con tutte le operazioni disponibili agli utenti loggati.

NUOVA PARTITA

Quando un utente avvia una nuova partita, il client invia una richiesta al server per controllare che l’utente non abbia già giocato una partita con la secret word attuale (in quel caso il client riceverebbe un messaggio di errore e chiederebbe all’utente di aspettare fino alla generazione di una nuova parola).

Se la richiesta va a buon fine client e server iniziano una serie di scambi di messaggi con i tentativi dell’utente e i suggerimenti (o il messaggio di vittoria) per l’utente. Al termine della partita vengono aggiornati i dati dell’utente sul server e i suoi tentativi vengono salvati (lasciando solo i suggerimenti) nel caso in cui volesse condividerli con gli altri utenti.

MOSTRA STATISTICHE UTENTE

Il client invia una richiesta al server per le statistiche relative all’utente loggato al momento. Il server risponde con una stringa contenente tutte le informazioni, sarà quindi il client ad effettuare il parsing così da mostrarle correttamente all’utente.

CONDIVIDI PARTITA

Al suo avvio il client si collega ad un gruppo multicast in cui è presente anche il server. Ad ogni richiesta di condivisione di una partita il client invia un messaggio al server il quale si occuperà di avvertire tutti gli altri client che una nuova partita è stata condivisa tramite un messaggio UDP.

Ogni client resta in ascolto per queste notifiche ed aggiorna la propria lista di partite condivise ogni volta che riceve un messaggio.

MOSTRA PARTITE CONDIVISE

Il client tiene salvate le partite condivise dagli altri utenti in un ArrayList, in caso di richiesta dell’utente stampa le partite nell’ordine in cui le ha ricevute.

MOSTRA CLASSIFICA

La classifica è gestita tramite il meccanismo delle RMI Callback, in particolare ad ogni avvio il client si registra sul server per ricevere delle callback con gli aggiornamenti della classifica. Il server si occupa quindi di aggiornare tutti i client registrati ogni volta che c'è un cambiamento nelle prime tre posizioni.

La struttura del codice

Il codice è principalmente diviso tra Client e Server ma ci sono delle classi e delle risorse condivise:

- **Game:** è una classe utilizzata per gestire una partita, in particolare chi l'ha effettuata e i suoi tentativi
- **User:** è una classe utilizzata per gestire tutte le informazioni riguardo ad un utente, tra le altre cose implementa anche la funzione compareTo così da facilitare il confronto di due utenti secondo il metodo stabilito
- **Config.json:** è il file che contiene tutte le informazioni utili per le connessioni tra server e client
- **Users:** contiene la lista di tutti gli utenti iscritti e le loro statistiche
- **Words.txt:** contiene la lista di tutte le parole che possono essere usate per il gioco

Server

Di seguito tutte le classi e interfacce utilizzate per l'implementazione del server:

WordleRegistrationInterface

Questa è l'interfaccia utilizzata sia per la registrazione di nuovi utenti (con *register*) sia per la registrazione dei client per le callback (con *registerForCallback*).

WordleServer

Classe centrale del progetto lato server in quanto implementa tutte le funzionalità richieste, dall'autenticazione dell'utente, alla gestione delle partite fino all'aggiornamento dei client per il nuovo ranking.

WordleServerMain

Come dice il nome stesso contiene il main lato server, quello che fa è semplicemente creare un'istanza del server.

WordleServerTask

Thread che viene lanciato in caso di connessione di un nuovo client, si occupa della comunicazione con esso analizzando le richieste e inviando i risultati.

WordPicker

Thread che si occupa di estrarre una nuova parola dalla lista presente in words.txt ogni volta che scade l'intervallo di tempo preimpostato.

Client

Di seguito tutte le classi e interfacce utilizzate per l'implementazione del client:

NotificationHandler

Thread che si occupa di stare in ascolto per eventuali nuove partite condivise nel gruppo multicast con tutti i client e il server. Ogni partita viene salvata in coda ad un ArrayList<Game>

RankingUpdater

Thread per la registrazione del client agli aggiornamenti del ranking tramite RMI Callback

WordleCallback

Interfaccia contenente *updateRanking*, il quale verrà utilizzato dal server per ogni aggiornamento della classifica.

WordleClient

Classe centrale del progetto lato client in quanto implementa tutte le funzionalità necessarie per comunicare con il server e eseguire le richieste da parte dell'utente.

WordleClientMain

Come dice il nome stesso contiene il main lato client in cui è implementato un semplice menù che permette all'utente di navigare le varie funzionalità offerte.

Strutture dati

Di seguito le strutture dati più importanti lato server e client:

Server

- **List<WordleCallback> clients:** utilizzata per tenere traccia di tutti i client che sono registrati per gli aggiornamenti del ranking
- **HashMap<String, User> users:** una mappa che associa ad ogni username un oggetto di tipo utente con tutte le informazioni riguardo ad esso

Client

- **ArrayList<Game> games:** lista di tutte le partite condivise dagli utenti nel gruppo multicast

Condivise

- **List<User> rankedUsers:** questa è la lista che viene passata ad ogni client ogni volta che il ranking viene aggiornato

Thread utilizzati

Di seguito i thread utilizzati lato server e client:

Server

- **Thread principale (*WordleServerMain*):** crea un'istanza del server che gestirà tutto il resto
- **Thread di comunicazione con il client (*WordleServerTask*):** gestisce lo scambio di messaggi tra client e server per le principali funzionalità (login, richiesta statistiche, avvio e gestione della partita)
- **Thread per la scelta di una nuova parola (*WordPicker*):** si occupa di pescare una nuova parola dalla lista fornita allo scadere del tempo.

Client

- **Thread principale (*WordleClientMain*):** implementa l'interfaccia che permette all'utente di giocare a Wordle
- **Thread per le notifiche delle partite (*NotificationHandler*):** si occupa di registrare il client al gruppo multicast in cui vengono condivise le partite e resta poi in ascolto per riceverle
- **Thread per la registrazione per gli aggiornamenti del ranking:** registra il client per le callback di aggiornamento sul server tramite RMI

Librerie utilizzate

L'unica libreria esterna utilizzata nel progetto è **GSON (gson-2.10.jar)** che è stata già inclusa nei JAR allegati al progetto nonché nella cartella *lib*.

Questa libreria è stata utilizzata per facilitare la gestione (serializzazione e trasformazione) di JSON e le loro conversioni in oggetti JAVA, oltre che per la lettura di file con la stessa estensione.

Istruzioni per compilazione ed avvio

Il progetto è stato sviluppato con l'utilizzo di IntelliJ IDEA come IDE. Per la compilazione è necessario specificare la libreria esterna gson che si trova all'interno della cartella *lib*.

Per l'esecuzione dei JAR è sufficiente eseguire il comando *java -jar nome-del-file.jar* senza l'aggiunta di parametri in quanto sia Client che Server prendono tutte le informazioni necessarie dal file *config.json*.

Struttura delle cartelle

Ecco una breve descrizione di cosa contengono le varie cartelle consegnate:

- **lib:** contiene l'unica libreria esterna utilizzata (*gson-2.10.jar*)
- **out:** contiene il codice compilato e i file JAR
    - **artifacts:** al suo interno sono presenti due cartelle per i JAR di client e server
    - **production:** contiene il codice compilato
- **src:** contiene tutte le cartelle con il codice (non compilato)