RELAZIONE PROGETTO LABORATORIO DI RETI: WORDLE 3.0

Diego Bernardini

Anno accademico 2022-2023

SOMMARIO

INTRODUZIONE

Panoramica

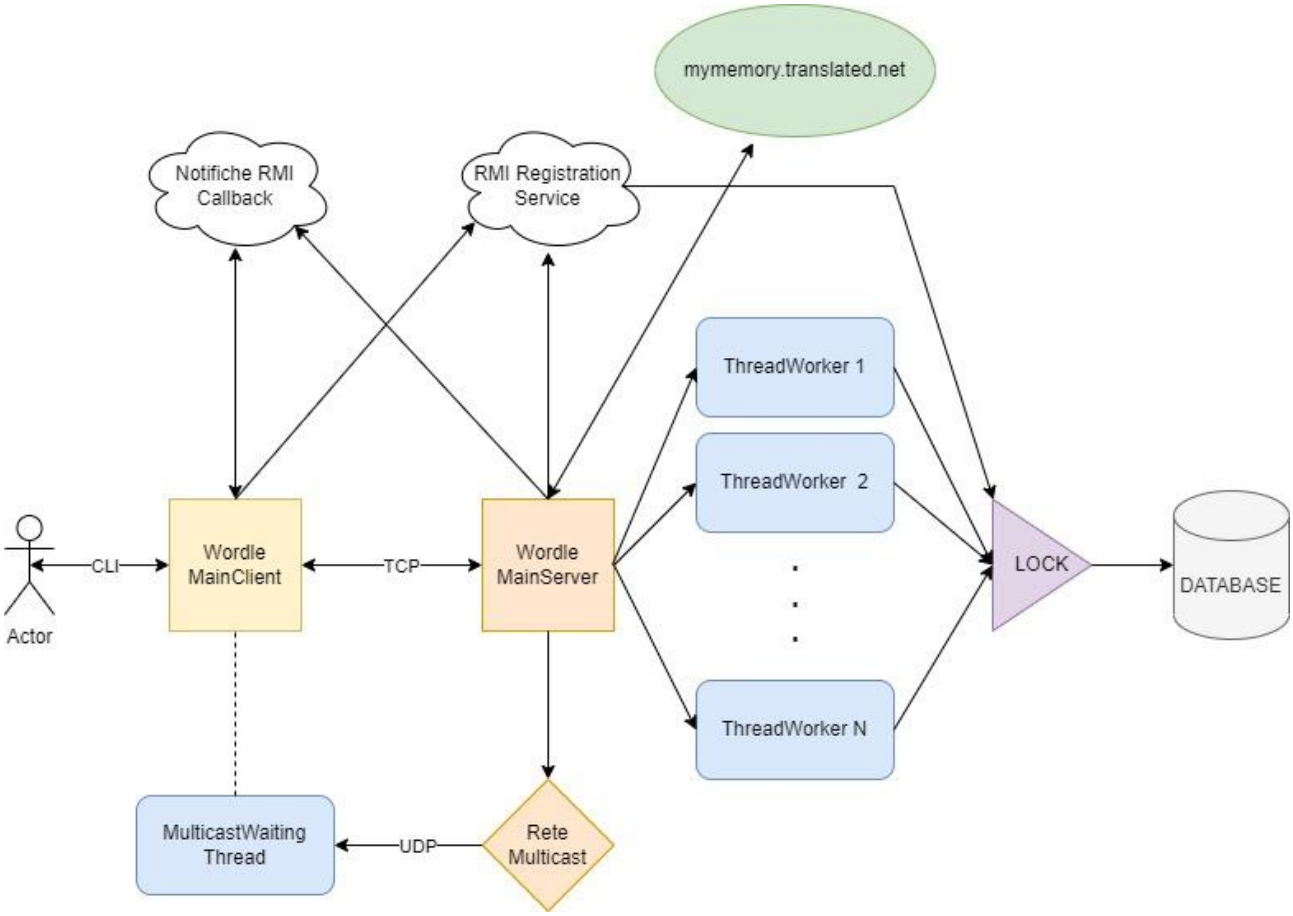
Il progetto consiste nella implementazione di una versione semplificata di WORDLE, un gioco di parole web-based.
Il gioco consiste nel trovare una parola inglese formata da 10 lettere, impiegando un numero massimo di 12 tentativi. WORDLE dispone di un vocabolario di parole di 10 lettere, da cui estrae casualmente una parola SW (Secret Word), che gli utenti devono indovinare. Ogni giorno viene selezionata una nuova SW, che rimane invariata fino al giorno successivo. L'utente propone una parola GW (Guessed Word) e il sistema inizialmente verifica se la parola è presente nel vocabolario. In caso negativo avverte l'utente che deve immettere un'altra parola. In caso la parola sia presente, il sistema fornisce all'utente alcuni indizi, utili per indovinare la parola.

Struttura delle directories

Panoramica sulla struttura delle classi e delle interfacce che sono state suddivise in package per una maggiore comprensione.

- Client: contiene le classi del client (ClientMainWordle e MulticastWaitingThread)
- Server: contiene le classi del server (ServerMainWordle, DatabaseSavingThread, ServerCtrlCHandler, WordExtractorThread e Worker) e le sottodirectory Database, che contiene le classi fondamentali per il database (Database, Account, Statistiche, User) e
- Models: contiene le classi che rappresentano l'entità fondamentali del progetto (BinarySearch, infoTraduzione, ScoreCalculator, Traduttore)
- Remote: contiene la parte implementativa e le interfacce dei servizi di registrazione e notifica (NotifyEventImpl, NotifyEventInterface, NotifyServerImpl, NotifyServerInterface, RemoteRegistrationImpl, RemoteRegistrationInterface)

ARCHITETTURA



DESCRIZIONE

ClientMainWordle è la classe che gestisce l'interazione con l'utente tramite una CLI (Command Line Interface) e comunica con il WordleServer per eseguire le azioni richieste dall'utente.
La fase di registrazione è implementata mediante RMI.
Dopo previa login effettuata con successo, l'utente:

1. si unisce a un gruppo di Multicast di cui fa parte anche il server con il quale viene implementato un meccanismo di condivisione dell'esito di una partita. Questa condivisione avviene in seguito ad uno specifico comando dell'utente grazie al quale il server invierà un messaggio UDP al gruppo. Per ricevere queste notifiche il client deve essere sempre in attesa e per questo motivo viene creato il 'MulticastWaitingThread', un thread che resta sempre in attesa di queste notifiche.
2. si registra a un servizio di notifica del server per ricevere aggiornamenti sulla classifica degli utenti, implementato con il meccanismo di RMI callback.
3. Instaura una connessione TCP persistente con WordleServerMain con il quale interagirà secondo il modello Client-Server (richieste/risposte) con il server inviando uno dei comandi disponibili.

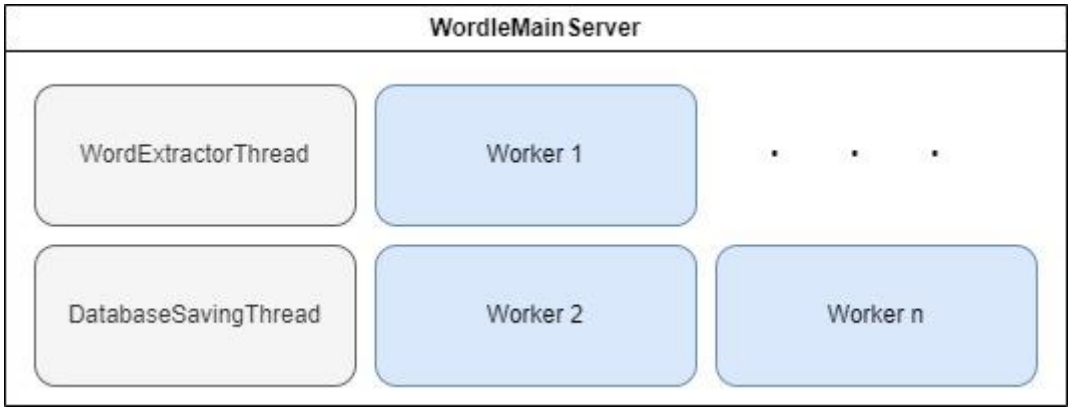
ServerMainWordle è la classe che gestisce la fase di registrazione e login degli utenti, memorizza tutti gli utenti registrati, propone periodicamente una nuova parola, interagisce con i diversi client

che vogliono partecipare al gioco fornendo gli indizi per la ricerca della parola segreta, gestisce le statistiche degli utenti, provvede, su richiesta, a inviare a tutti gli utenti iscritti al gruppo di condivisione, le informazioni relative alla partita conclusa da un utente tramite una rete Multicast e mantiene una classifica ordinata degli utenti. Inoltre, quando termina una sessione di gioco, il server deve fornire al client la traduzione italiana della parola segreta alla quale stava giocando, ottenuta accedendo al servizio presente alla URL https://mymemory.translated.net/doc/spec.php.

Database è la classe che contiene la classifica e le informazioni di tutti gli utenti che si sono registrati ed è la risorsa con cui i vari thread Worker e il servizio remoto di registrazione interagiranno quando necessario. Per risolvere problemi legati alla concorrenza viene utilizzata una Lock che limita i vari accessi.

SCELTE DI PROGETTAZIONE/IMPLEMENTAZIONE

ServerMainWordle



Il server una volta avviato:

1. ripristina lo stato del sistema utilizzando le informazioni presenti nel File ‘myDatabase.json’.
2. genera gli oggetti remoti che serviranno al client per le operazioni di registrazione e notifica.
3. entra nel gruppo Multicast di condivisione dei risultati delle partite.
4. Crea 2 thread, WordExtractorThread e DatabaseSavingThread, che verranno schedulati periodicamente da un singleThreadSchedulerExecutor per estrarre la nuova parola segreta e salvare le informazioni presenti nel Database sul file ‘myDatabase.json’.
5. si mette in ascolto sulla porta 8080 e attende che un client mandi una richiesta per accedere al servizio.

La comunicazione client-server è implementata utilizzando JAVA I/O e threadpool. Quest’ultimo è implementato come un CachedThreadpool, dato che non si conosce il numero di utenti che dovranno essere serviti e vista l'imprevedibilità dell’arrivo delle connessioni.
Il server si trova in uno stato di ciclo infinito durante il quale, quando arriva una richiesta, viene istanziato ed eseguito un nuovo thread ‘Worker’ al quale viene passato una socket con cui sarà in grado di leggere le operazioni richieste dal client e inviare le risposte opportune.

ClientMainWordle

È La classe che inizializza la connessione con il server e gestisce l'interazione con l'utente tramite la linea di comando. All'attivazione viene letto il file 'client.properties' e subito dopo viene visualizzato un primo menù ('welcomeMenu') nel quale si chiede quale azione l'utente vuol fare: registrarsi, effettuare il login o uscire. In seguito al login:

1. stabilisce la connessione con il server tramite Socket e inizializza gli Stream I/O per inviare e ricevere messaggi.
2. Crea il 'multicastWaitingThread', il quale entra a far parte della rete Multicast di condivisione dei risultati delle partite e resta in attesa di ricevere notifiche.
3. Si registra a un servizio di notifica del server per ricevere aggiornamenti sulla classifica degli utenti.

A questo punto viene visualizzato il menu principale ('menu') grazie al quale l'utente può decidere se giocare, visualizzare le proprie statistiche, le notifiche ricevute o la classifica oppure effettuare il logout.

[CLIENT] Benvenuto in WORDLE!
<1> -> Registrazione
<2> -> Login
<3> -> Esci

[CLIENT] MENU
<1> playWORDLE
<2> sendMeStatistics
<3> showMeSharing
<4> showMeRanking
<5> Logout

[CLIENT] SESSIONE DI GIOCO
<1> sendWord
<2> showMeSharing
<3> showMeRanking
<4> Logout

I menù gestiscono e controllano l'input dell'utente e chiamano le funzioni che implementano le operazioni richieste: login(), register(), logout(), playWordle(), sendWord(), share(), showMeSharing(), showMeRanking() e sendMeStatistics().

Worker

È il thread che gestisce la comunicazione con il client e implementa la logica del gioco lato server. Resta in attesa dei comandi da eseguire e invia le risposte al client in base allo stato dell'applicazione.

Properties File

La specifica stabilisce che non è consentito leggere i parametri in modo "interattivo" e per questo motivo vengono utilizzati i due file server.properties e client.properties, letti rispettivamente da WordleServerMain e WordleClientMain.

Database

Per gestire i dati persistenti necessari per il corretto funzionamento dell'applicazione vengono utilizzati due File:

- words.txt
- myDatabase.json

words.txt è il file che contiene parole inglesi di dieci lettere ordinate lessicograficamente. Svolge il ruolo di dizionario: è il file dal quale vengono estratte periodicamente le parole segrete.

Il file json mantiene le informazioni degli utenti. La struttura del file è realizzata in modo che i dati possano essere deserializzati e convertiti in oggetti a partire dalle classi 'User', 'Account' e 'Statistiche'. Inoltre, siccome la specifica stabilisce che l'username è univoco e che il server deve mantenere una classifica, in questo file i vari 'user' sono ordinati in modo crescente in base al proprio score perché vale la regola secondo cui il giocatore più bravo è quello con punteggio minore. E.g. L'utente A con punteggio = 1,5 precede sicuramente l'utente B con punteggio = 2,5.

Dipendentemente dall'azione richiesta dal client, il server può leggere questi File per recuperare alcune informazioni oppure può sovrascrivere i dati presenti nel file json. Inoltre, periodicamente (la frequenza è specificata nel file server.properties) il database virtuale viene scritto sul file 'myDatabase.json'.

Gestione della concorrenza

Come si vede dall'architettura, chiunque voglia interagire con il database deve prima ottenere una 'Lock'. L'implementazione sfrutta un oggetto di tipo ReentrantLock che viene creato dal server all' avvio e passato come argomento ad ogni thread Worker al momento della creazione. Così facendo, si garantisce mutua esclusione attraverso una gestione esplicita e si evitano problemi legati a possibili accessi concorrenti di più Thread.
Durante l'interazione con il Database si utilizzano il metodo TryLock per bloccare e Unlock per sbloccare. Quando invochiamo il primo impostiamo anche un timer di X secondi (parametro di configurazione 'delay'), così da poter inviare un messaggio al client in caso di mancata acquisizione, comunicandogli che l'operazione richiesta non è andata a buon fine.

COMUNICAZIONE

Esistono due modi per il client di interagire con il server, uno attraverso RMI oppure RMI callback e l'altro attraverso brevi richieste al server principale mediante Socket, ma per quanto riguarda lo scambio dei messaggi che avviene in questo progetto vengono utilizzati tre modi:

- Messaggio sottoforma di semplice stringa
- Datagramma
- Stringa JSON

Messaggi come stringhe: utilizzati per la comunicazione tra client e server. L'idea è quella di inviare i comandi richiesti dal client sotto forma di semplici stringhe in cui il comando ed eventuali altre informazioni legate ad esso, sono separate dal carattere '#'. Stesso discorso per le risposte, in cui la stringa conterrà un numero che rappresenterà l'esito dell'operazione richiesta ed eventuali altre informazioni separate da '#'.

Datagramma: Utilizzato per l'invio dei messaggi tra client con UDP Multicast. Viene ricevuto un DatagramPacket che viene utilizzato per generare una stringa che contiene la data e l'esito della partita di un utente.

Stringa JSON: Utilizzata durante l'accesso al servizio 'mymemory.translated.net' per ottenere la traduzione italiana di una parola. A tal proposito viene utilizzata la classe 'infoTraduzione', per catturare la traduzione all'interno della risposta in formato json inviata dal servizio nel campo "translated-text".


```
{
  "responseData": {
    "translatedText": "Ciao Mondo!",
    "match": 1
  },
  "quotaFinished": false,
  "mtLangSupported": null,
  "responseDetails": "",
  "responseStatus": 200,
  "responderId": null,
  "exception_code": null,
  "matches": [
    {
      "id": "723178549",
      "segment": "Hello World!",
      "translation": "Ciao Mondo!",
      "source": "en-GB",
      "target": "it-IT",
      "quality": 74,
      "reference": null,
      "usage-count": 2,
      "subject": "",
      "created-by": "MateCat",
      "last-updated-by": "MateCat",
      "create-date": "2023-06-25 12:36:32",
      "last-update-date": "2023-06-25 12:36:32",
      "match": 1
    }
  ]
}
```

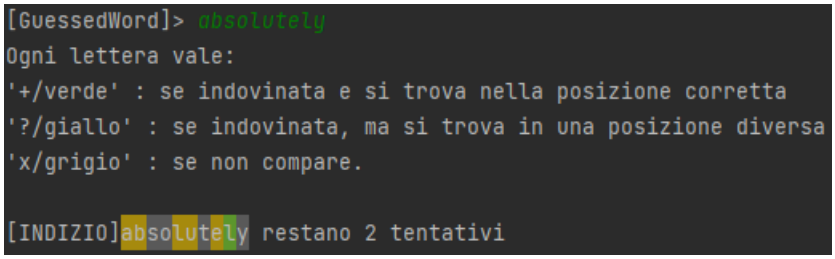
RMI

In totale ci sono due funzionalità implementate con questa tecnica:

- Registrazione di un utente
- Notifiche degli aggiornamenti sulla classifica degli utenti

La registrazione utente implementa un unico metodo che controlla nel database se esiste già un utente registrato con l'username appena inserito, in caso negativo inserisce lo inserisce.
Per quanto riguarda il sistema di notifica viene utilizzata la tecnica RMI Callback con la quale si dà la possibilità al client di riceverne una ogni volta che si verifica lo specifico evento. Nello specifico, per abilitare questo sistema, un client si registra ad un oggetto remoto una volta effettuato con successo il login e si disiscrive al momento del logout tramite due metodi specifici. La funzione che esegue la callback è una funzione asincrona che scorre con un iteratore tutti i client e invia un messaggio.

Suggerimento



Quando un client tenta di indovinare la secretWord e fallisce, il server invia un suggerimento che viene calcolato in base alla posizione dei caratteri della guessedWord e di quelli della secretWord., tenendo conto anche del numero di occorrenze di ogni lettera.
L' i-esimo carattere dell'indizio indica se l'i-esima lettera della guessedWord è:

- Presente nella secretWord nella posizione corretta (colore verde o '+')
- Presente nella secretWord ma in posizione errata (colore giallo o '?')
- Assente nella parola segreta (nessun colore o 'x')

CONCLUSIONI

Miglioramenti

Modificare l'attuale protocollo di comunicazione tra client e server, basato sull'invio di stringhe, utilizzando invece una classe 'messaggio', che implementi l'interfaccia 'serializable' e che sia l'astrazione di un messaggio generico che può essere inviato tra le due entità. Questa classe dovrebbe contenere al suo interno tutte le variabili di istanza possibili che potremmo utilizzare per le operazioni del progetto, e tutti i metodi 'getter' e 'setter' specifici. Così facendo quando creeremo un nuovo messaggio l'unico campo che dovrà essere sempre impostato sarà quello relativo al comando richiesto dal client e inizializzeremo i campi opportuni per uno specifico comando.

MANUALE D'USO

Il progetto è stato sviluppato con i seguenti strumenti:

- IDE: IntelliJ IDEA
- OS: Windows 10
- Liberia GSON versione 2.10.1
- Maven

Esecuzione Maven

Con Maven posizionarsi nella cartella WordleBernardini ed eseguire il seguente comando:
<<mvn exec:java -Dexec.mainClass="Server.ServerMainWordle" >> per avviare il server,

oppure il seguente comando:
<<mvn exec:java -Dexec.mainClass="Client.ClientMainWordle" >> per avviare il client.

Esecuzione Jar

Posizionarsi nella cartella WordleBernardini\jar\client, assicurarsi che sia presente la cartella META_INF, contenente il file manifest, e il file di configurazione ed eseguire il seguente comando:
<<java -jar ClientMainWordle.jar>> per avviare il client.

Posizionarsi nella cartella WordleBernardini\jar\server, assicurarsi che sia presente la cartella META_INF, contenente il file manifest, il file di configurazione e il vocabolario ed eseguire il seguente comando:
<<java -jar ServerMainWordle.jar>> per avviare il Server.

Nota bene: entrambe le esecuzioni richiedono l'utilizzo di due finestre di terminale distinte, una per il server e una per il client