Relazione Progetto Wordle 3.0

Andrea Carollo

January 10, 2023

UNIVERSITÀ DI PISA

Contents

1 Introduzione

Wordle 3.0 è una versione shell dell'omonimo e famoso gioco Wordle,rilasciato nel 2018 dal New York Times. Questa versione prevede interazione via CLI (Command Line Interface) e una pool di parole di 10 lettere,con 12 tentativi massimi.

2 Classi Principali

Le 2 componenti principali (Server e Client) vengono implementate grazie a diverse classi,di seguito vengono elencate le principali.

2.0.1 Server

- **WordleServerMain**:

  Classe Principale del Server, legge la sua configurazione dal file "server.properties",crea le strutture dati e gli oggetti per la corretta esecuzione del server stesso compreso un ThreadPool di oggetti "WordleClientHandler". All'interno

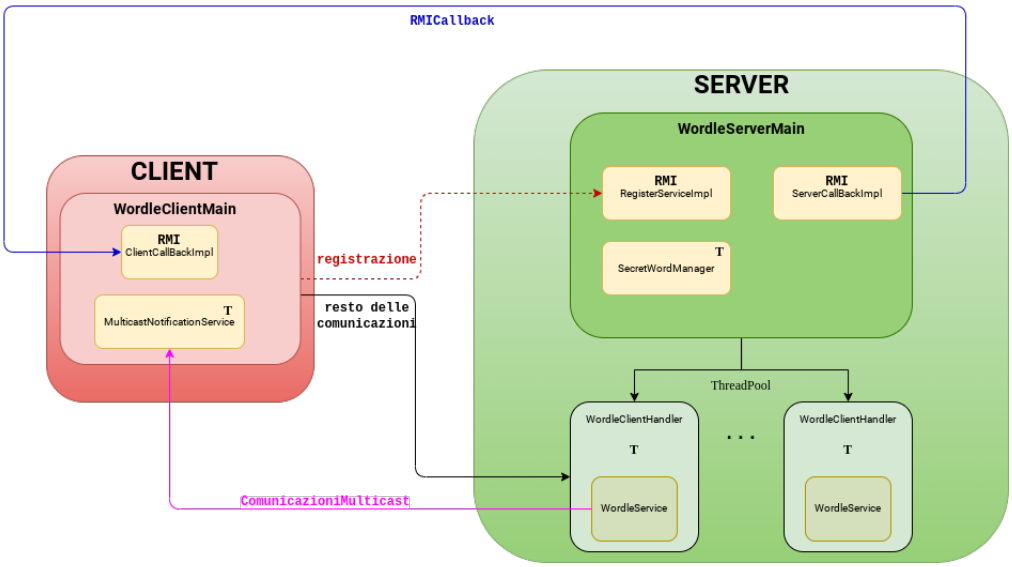


Figure 1: Il Simbolo "T" simboleggia che l'oggetto è stato creato come Thread Separato

del while loop principale accetta la comunicazione Socket con i client,passando il socket creato al thread facente parte del pool.

- **WordleClientHandler**:

  Oggetto respondabile della comunicazione con il singolo client,le istanze di questa classe fanno tutte parte del ThreadPool creato dal Server-Main. Ogni istanza si occupa di gestire un singolo client,soddisfacendo le richieste che arrivano tramite Socket. Ogni Oggetto WordleClientHandler crea un oggetto "WordleService" che implementa l'effettivo servizio di gioco "Wordle". WordleClientHandler è responsabile del logout dell'utente e della chiusura della connessione.

  .

- **WordleService**:

  Classe che implementa il servizio di gioco di "Wordle 3.0",si occupa del gioco e di tutte le funzionalità aggiuntive richieste da specifica. Interagisce con il client tramite Socket e modifica strutture dati comuni a tutte le altri classi facente parti del Server.

- **RegisterServiceImpl**:

  Classe che Implementa il servizio di regitrazione tramite RMI,effettua i controlli per non avere username duplicati e comunica con il file "Play-

ersData.json". Contiene anche una funzione per aggiornare lo stato e il punteggio di un giocatore.

- **ServerCallbackImpl**:

  Implementa Il Servizio di Callback per gli aggiornamenti sulle prime posizioni della classifica ,comunicando ai client un messaggio contenente username e posizione raggiunta.

- **SecretWordManager**:

  Classe usata dal server per gestire la generazione di nuove parole segrete per il gioco,ogni "time" secondi viene prelevata una nuova parola da un ArrayList in cui sono state parsate tutte le parole del vocabolario

2.0.2 Client

- **WordleClientMain**:

  Classe principale del client,dopo aver settato la sua configurazione, letta dal file "client.properties" e aver creato gli oggetti per la comunicazione tramite RMI e UDP Interagisce con l'utente per la fase iniziale del Servizio Wordle. Nella fase iniziale l'utente potrà scegliere di registrare un nuovo profilo,loggare con un profilo già esistente o richiedere una spiegazione del gioco stesso.

  Una volta effettuato il login entra nel while loop principale dove,interagendo con il terminale,si potrà selezionare l'opzione desiderata tra:

  - *1) PlayWordle:* fa partire una sessione di gioco Wordle
  - *2) Logout:* Esegue il logout dell'utente
  - *3) Mostra Notifiche:* Mostra le notifiche ricevute tramite Multicast
  - *4) Condividi Risultato:* Condivide,se presente,i risultati dell'ultima partita effettuata
  - *5) Mostra Statistiche:* Chiede al server le statistiche aggiornate del giocatore e le stampa a schermo
  - *6) Mostra Classifica:* Chiede al server le classifica aggiornata,viene successivamente stampata a schermo

- **ClientCallbackImpl**:

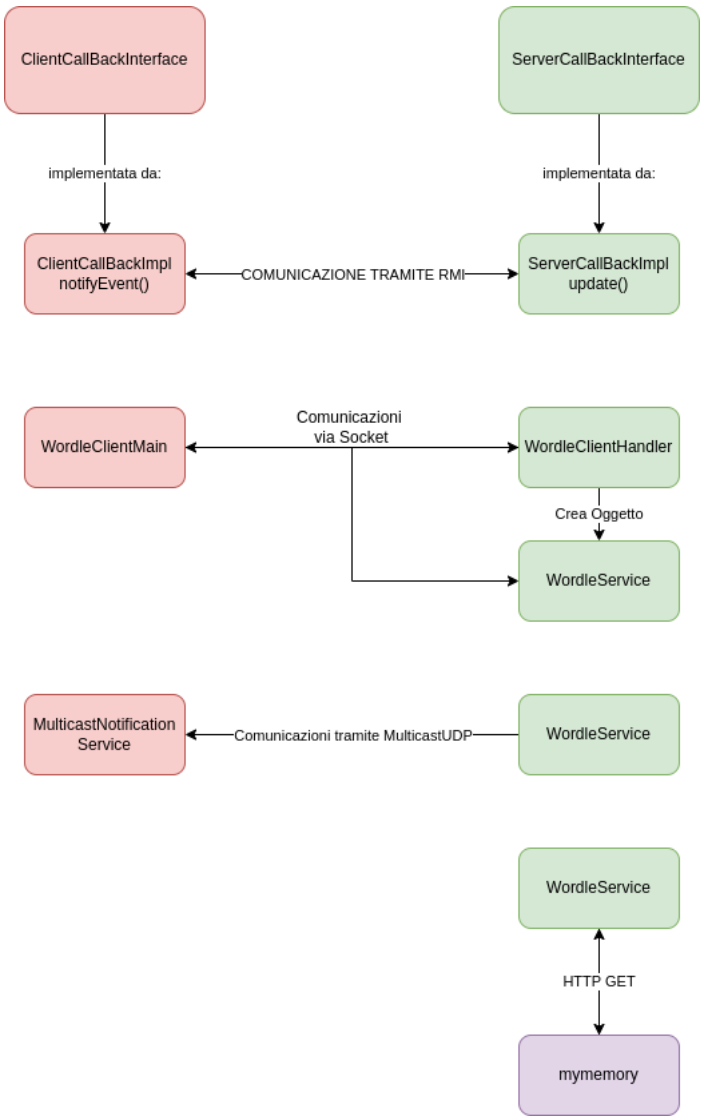
  Classe che implementa il servizio di CallBack per gli aggiornamenti delle prime posizioni della classifica

- **MulticastNotificationService**:

  Classe che implementa il servizio di notifica dei giocatori,quando richiesto,tramite l'opzione "Mostra Notifiche" sarà possibile consultare le notifiche ricevute.

3 Comunicazioni

Come da Specifica,sono state utilizzate varie tecnologie per la comunicazione tra Client e Server,di seguito uno schema che le riassume:



Le comunicazioni più frequentemente avvengono tra il Main del client e le istanze della classe "WordleService",questi scambiano i risultati delle richieste del client e eventuali messaggi di errore,gestendoli. Il Client comunica con gli oggetti "WordleCLientHandler" principalmente per la comunicazione della richiesta da inoltrare all'oggetto WordleService.

4 Strutture Dati e MultiThreading

Nello Sviluppo del progetto ho posto attenzione ad utilizzare,dove opportuno, strutture dati thread-safe,tra le strutture principali più interessanti troviamo:

- $SynchonizedList< String >loggeduser$: ArrayList sincronizzato contentente gli utenti loggati al server
- $Hashtable< String, ArrayList < String >> playedwords$: Hashtable (ThreadSafe per costruzione) contenente per ogni username la lista di parole giocate.
- $SynchronizedSortedSet< Player > ranking$: TreeSet sincronizzato utilizzato per implementare la classifica dei giocatori
- $SynchonizedList< String >words$: ArrayList sincronizzato contentente le parole del vocabolario

Dove necessario e richiesto è stato utilizzato un approccio multithreading per la gestione delle funzionalità più importanti o esose,in particolar modo:

- Il Server crea un CachedThreadPool di oggetti "WordleClientHandler" per la gestione delle connessioni,un thread per ogni client connesso
- Il Server crea un thread per la gestione della parola segreta
- Il Client cera un thread per la gestione delle notifiche Multicast da parte del Server

5 Dati Permanenti e Configurazione

I file di configurazione "client.properties" e "server.properties" sono utilizzati rispettivamente da Client e Server per costruire i parametri di configurazione,al loro interno contengono i valori delle porte utili per la comunicazione e altri valori utili (come il tempo di aggiornamento della

parola segreta). I file verranno poi parsati grazie ad una funzione specifica,utilizzando i metodi della classe "Properties".
.
I dati relativi agli utenti vengono tutti memorizzati all'interno del file "PlayersData.json" che funziona da Database. La classifica viene anche essa memorizzata e aggiornata all'interno di un file, "Ranking.json".
Entrambi i file json vengono opportunamente parsati per ricostruire gli oggetti relativi all'accensione del server.

6 Scelte Implementative

Di seguito una serie di scelte implementative:

- TreeSet è stato preferito ad altre strutture dati per mantenere più facilmente l'ordine desiderato,implementato grazie al Comparator "PlayerRankingComparator"
- È stato scelto di rendere la classifica permanente grazie al file "Ranking.json"
- Per ordine del codice ho preferito implementare i metodi richiesti da specifica in una classe separata da quella che gestisce la connessione con il client,ovvero WordleService
- Per evitare la latenza data dalla connessione con il servizio mymemory,ho preferito separare la sua esecuzione da quella del metodo "playwordle",creando una task Callable da assegnare ad un nuovo thread. Il risultato verra' successivamente letto da un oggetto di tipo FutureTask
- Dove possibile ho cercato di ridurre gli accessi ai file esterni,poichè computazionalmente costosi

StringBuilder poiche non immutable TreeSet per mantere ordine classifica permanente

7 Test

8 Compilazione ed Esecuzione