Relazione Progetto Wordle 3.0

Andrea Carollo

January 11, 2023

UNIVERSITÀ DI PISA

Contents

1 Introduzione

Wordle 3.0 è una versione shell del famoso gioco "Wordle", rilasciato nel 2018 dal New York Times. Questa versione prevede interazione via CLI (Command Line Interface) e un vocabolario di parole di 10 lettere.
12 tentativi massimi.

2 Classi Principali

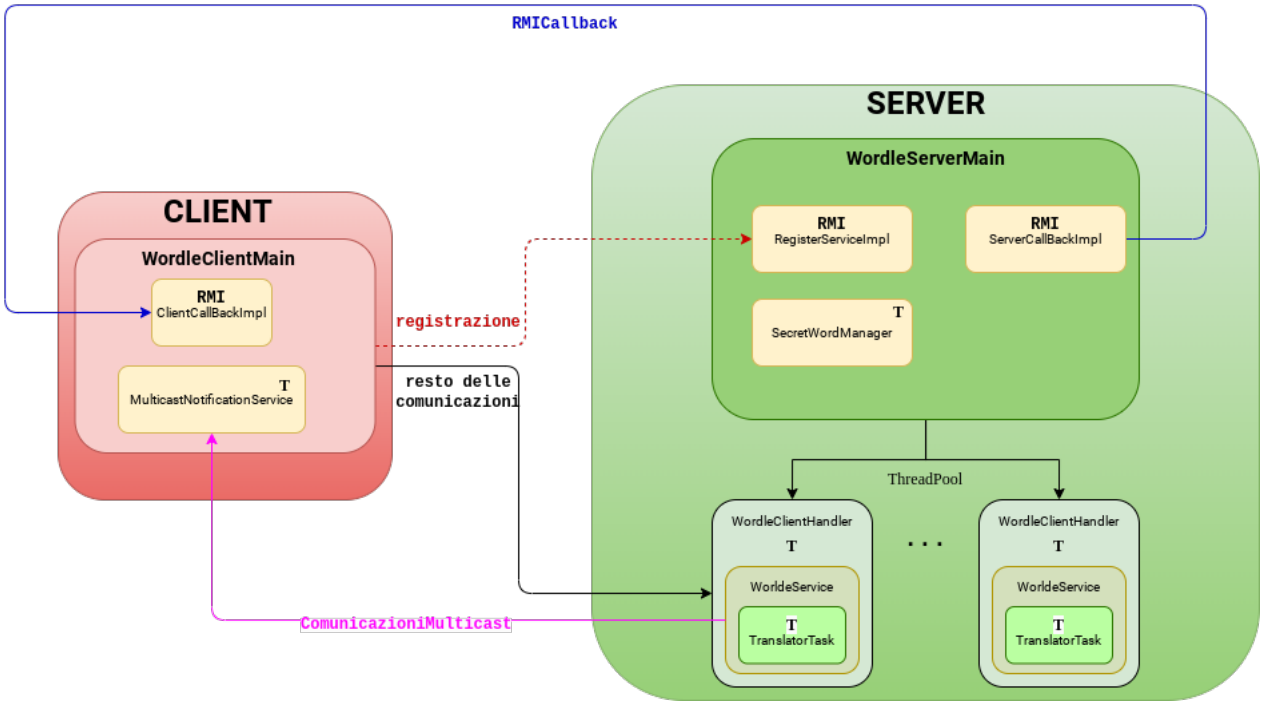


Figure 1: Il Simbolo "T" simboleggia che l'oggetto è stato creato come Thread Separato

Ho implementato le due componenti principali, Client e Server, suddividendo le funzionalità in numerose classi, di seguito vengono elencate le più significative:

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.
  Crea una ThreadPool di oggetti "WordleClientHandler".

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

  .

- **WordleClientHandler**:

  Classe respondabile della comunicazione con il singolo client,le istanze di questa classe fanno tutte parte del ThreadPool creato dal ServerMain.

  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 gioco "Wordle 3.0",Implementa inoltre le funzionalità aggiuntive richieste da specifica.

  Interagisce con il client tramite Socket e modifica le strutture dati comuni a tutto il Server.

- **RegisterServiceImpl**:

  Classe che Implementa il servizio di regitrazione tramite RMI,effettua i controlli per non avere username duplicati e comunica con il file "PlayersData.json",per la configurazione iniziale e per l'aggiornamento dei dati degli utenti.

- **ServerCallbackImpl**:

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

- **SecretWordManager**:

  Classe usata dal server per gestire la generazione di nuove parole segrete.

  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 grazie al file "client.properties", crea gli oggetti per la comunicazione tramite RMI e UDP.

  Completata la configurazione interagisce con l'utente nella fase di benvenuto,qui l'utente potrà scegliere se registrare un nuovo profilo,loggare con un profilo già esistente o richiedere una spiegazione del gioco.

  Effettuato il login entra nel while loop principale dove,interagendo con il terminale,saranno selezionabili varie opzioni:

  - *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 presenti,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

- **ClientCallbackImpl**:

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

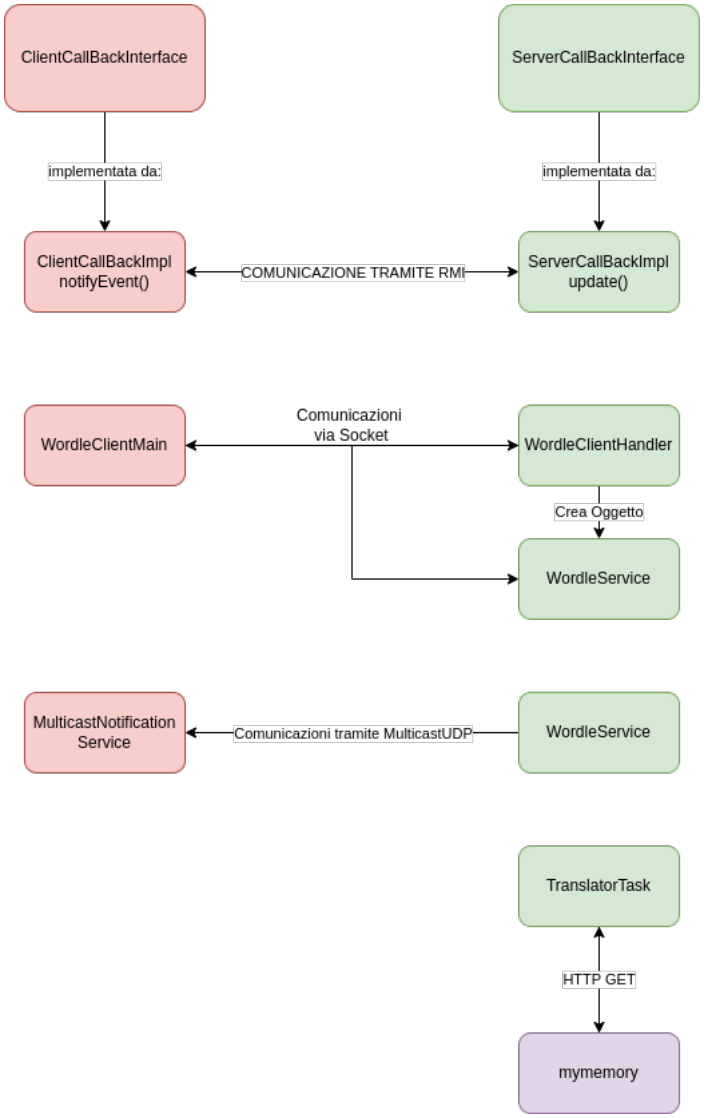
- **MulticastNotificationService**:

  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 avvengono più frequentemente tra il Main del client e le istanze della classe "WordleService",questi scambiano i risultati delle richieste del client e eventuali messaggi di errore.
Il Client comunica con gli oggetti "WordleCLientHandler" per la comunicazione della richiesta da inoltrare all'oggetto WordleService.
Il thread che esegue la task "Translator Task" richiede le risorse del servizio mymemory

4 Strutture Dati e MultiThreading

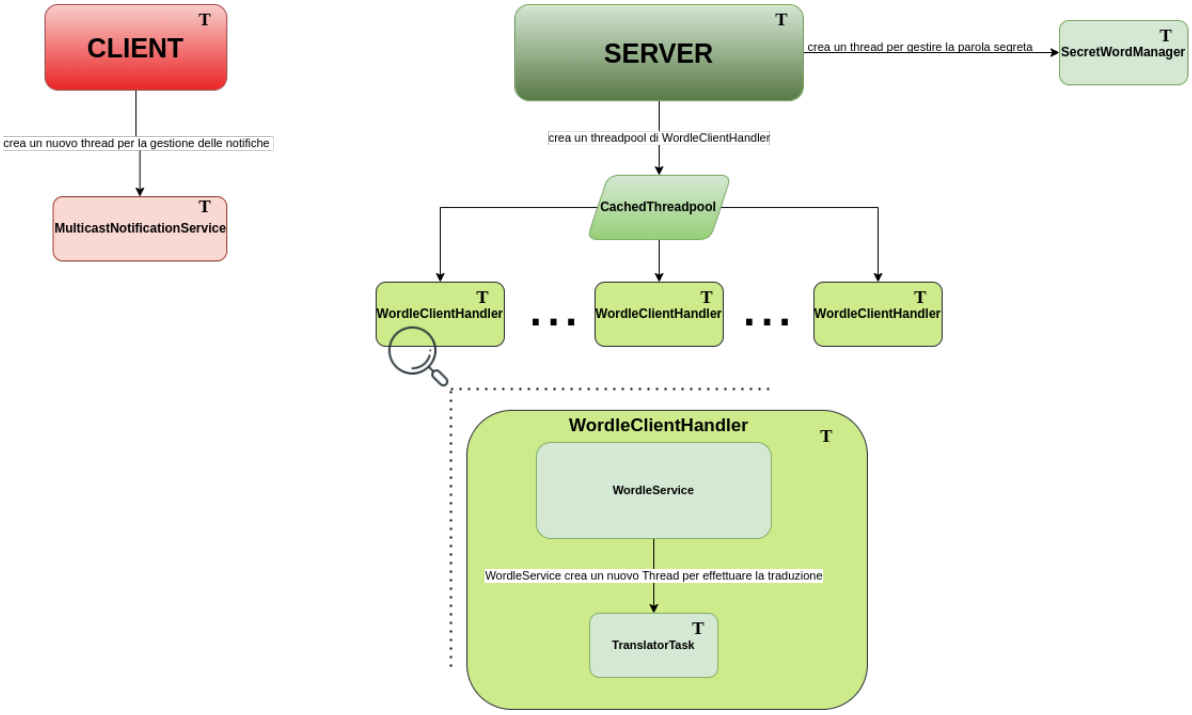


Figure 2: Il simbolo "T" indica che si tratta di un thread separato

Nello Sviluppo del progetto ho posto attenzione ad utilizzare,dove opportuno, strutture dati thread-safe.

.

Tra le strutture principali 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

- *SynchonizedList< String >notification*: ArrayList sincronizzato contentente le notifiche ricevute dal server
- *StringBuffer secretword*: StringBuffer è stato preferito a StringBuilder per la sua Thread-safeness

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

- Il Server crea un CachedThreadPool di oggetti "WordleClientHandler" per la gestione delle connessioni
- Il Server crea un thread per la gestione della parola segreta
- Il Client crea un thread per la gestione delle notifiche Multicast da parte del Server
- Il Server crea un thread per la traduzione della parola segreta,evitando la latenza di connessione al servizio "mymemory"

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 del file, "Ranking.json".
.
Entrambi i file json vengono opportunamente parsati per ricostruire le strutture dati 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(circa 2s),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
- Ho scritto un Makefile per rendere compilazione ed esecuzione piu' agevole

StringBuilder poiche non immutable TreeSet per mantere ordine classifica permanente

7 Compilazione ed Esecuzione

Per la compilazione e l'esecuzione del progetto bisogna recarsi all'interno della cartella "java" del progetto,dove risiede il Makefile.

7.1 Compilazione

è possibile eseguire i seguenti comandi per la compilazione del Server e del Client:

- **make comp_server**
- **make comp_client**

Inoltre è possibile eliminare i file .class (frutto della compilazione) con il comando **make rm_comp**

7.2 Esecuzione

Per eseguire il progetto è possibile usare i file .jar nella cartella "java".

- **make clientjar**
- **make serverjar**

  Altrimenti è sempre possibile effettuare l'esecuzione tramite i comandi:

- **java -cp "./json-simple-1.1.1.jar":"./gson-2.10.jar":"./" ./WordleClientMain.java ../resources/client.properties**
- **java -cp "./json-simple-1.1.1.jar":"./gson-2.10.jar":"./" ./WordleServerMain.java ../resources/server.properties**