UNIVERSITÀ DI PISA

Relazione progetto Word Quizzle
Laboratorio di Programmazione di Reti a.a. 2019/20

Gennaro Baratta (mat. 517339)

1. Introduzione

WordQuizzle è gioco multiplayer dove gli utenti possono sfidarsi in traduzioni dall'italiano all'inglese. È presente la gestione di una rete sociale, in cui è data la possibilità di creare amicizie e vedere i punteggi dei propri amici. La comunicazione fra server e client avviene tramite protocollo TCP, mentre per la registrazione è utilizzato Remote Method Invocation. L'inoltro delle notifiche di sfida avviene con protocollo UDP. Ho scelto di implementare la persistenza dei dati su file JSON con l'aiuto di una libreria esterna fasterXML Jackson. Le traduzioni delle parole vengono richieste dal servizio MyMemory esterno utilizzando il protocollo HTTP.

Per l'implementazione dell'interfaccia grafica ho deciso di utilizzare JAVAFX, che permette di sfruttare il design pattern Model-View-Controller separando le view in appositi file FXML.

2. Gestione concorrenza

Il mio principale obiettivo riguardo la concorrenza è stato quello di minimizzarla ove possibile. Nel caso delle operazioni di login e registrazione avere strutture dati centralizzate e quindi in concorrenza è una necessità. Nel caso delle sfide invece basta non permettere l'avvio di accettazione di due sfide diverse lato client e avere come flag un AtomicBoolean per avere il feedback di una sfida in corso in modo thread-safe.

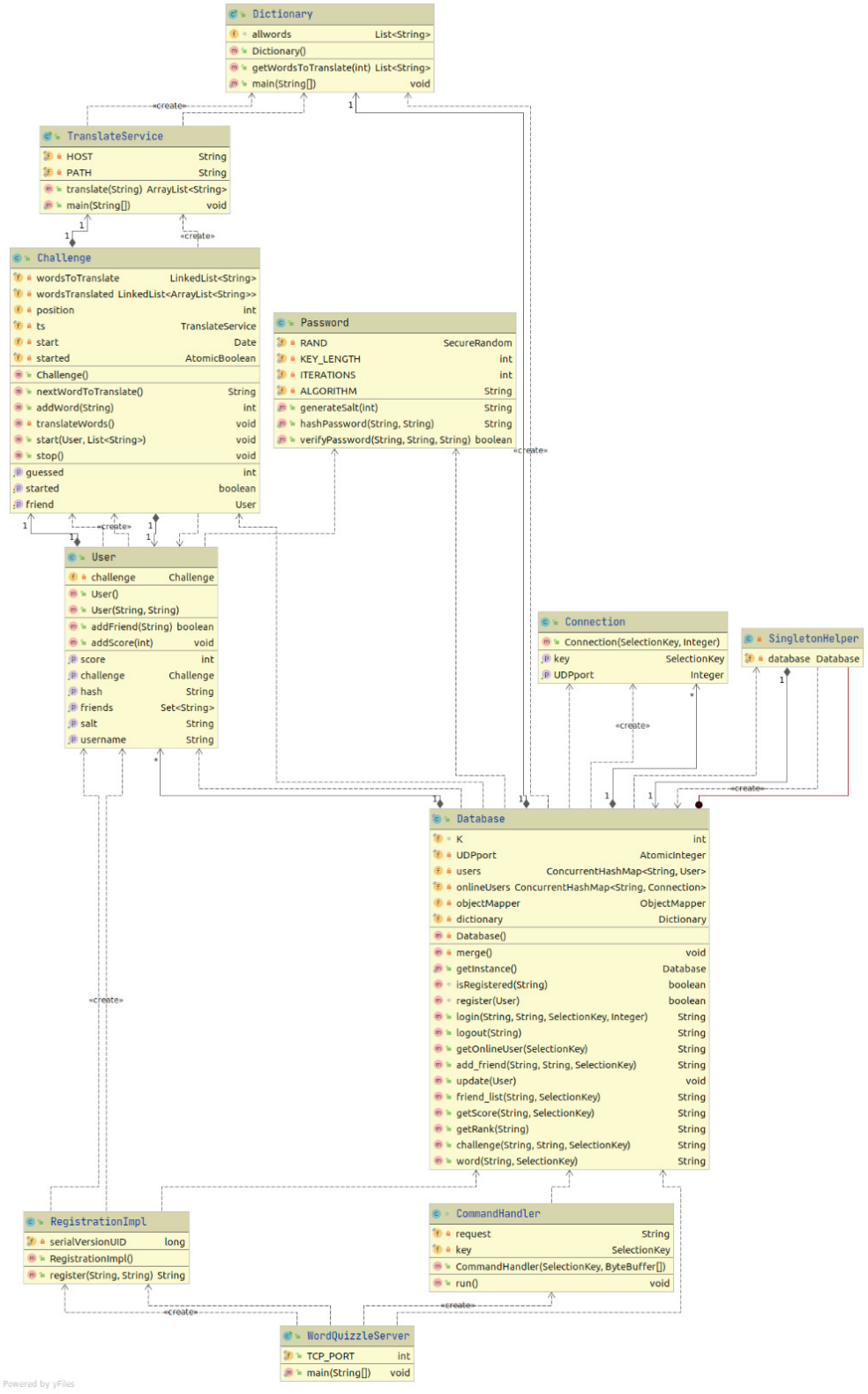
3. Persistenza

Quando viene istanziata la classe Database, la funzione merge si occupa di leggere i vari file che rappresentano gli utenti dalla cartella users e di unirli in un unico file users.json, che è utilizzato successivamente per inizializzare la ConcurrentHashMap degli utenti registrati. La separazione in file diversi per ogni utente permette di ottenere aggiornamenti al tempo di esecuzione più performanti e frequenti rispetto al mantenimento di un unico file. Il merging è effettuato per la comodità di

assegnamento degli utenti registrati tramite l'objectMapper della libreria jackson ed è una operazione, benché potenzialmente costosa, una tantum.

4. Classi

4.1 Modulo Server



WordQuizzleServer

È la classe con il metodo main, inizializza un ThreadPoolExecutor su cui vengono eseguite le richieste dei vari client e l'RMI per la registrazione. Le richieste sono smistate tramite un ciclo in cui si utilizzano Channels NIO. In particolare, il ServerSocketChannel è configurato in modalità non bloccante per sfruttare al massimo il singolo thread su cui viene eseguito l'ascolto delle connessioni in arrivo. Dopo aver accettato una connessione, verrà allacciato un buffer su cui eseguire letture e scritture al socket del client. Quando sarà possibile un'operazione di lettura, verrà eseguito un CommandHandler nel ThreadPool avendogli passato come argomenti la SelectionKey del client e il suo buffer. Quando il CommandHandler avrà soddisfatto la richiesta del client e popolerà il buffer con l'esito dell'operazione, essa verrà comunicata tramite una scrittura sul socket client.

CommandHandler

E un'implementazione della classe Runnable e chiama funzioni della classe Database dopo un opportuno parsing.

Database

È la classe più grande del backend su cui effettuerei un eventuale refactoring per estendere il progetto e renderlo più mantenibile. Gestisce le strutture dati utili a conservare lo stato dell'applicazione: due ConcurrentHashMap per utenti registrati e connessi attualmente, un AtomicInteger per utilizzare più porte UDP nell'invio e ricezione del risultato di richieste di sfide.

TranslateService

Si tratta della classe che invia le richieste HTTP per le traduzioni al servizio esterno MyMemory. Effettua un parsing del JSON della risposta e restituisce un ArrayList di stringhe.

Dictionary

Contiene l'implementazione della funzione che sceglie le parole da un dizionario di 1160 parole comuni (https://github.com/napolux/paroleitaliane/blob/master/paroleitaliane/1000_parole_italiane_comuni.txt licenza MIT).

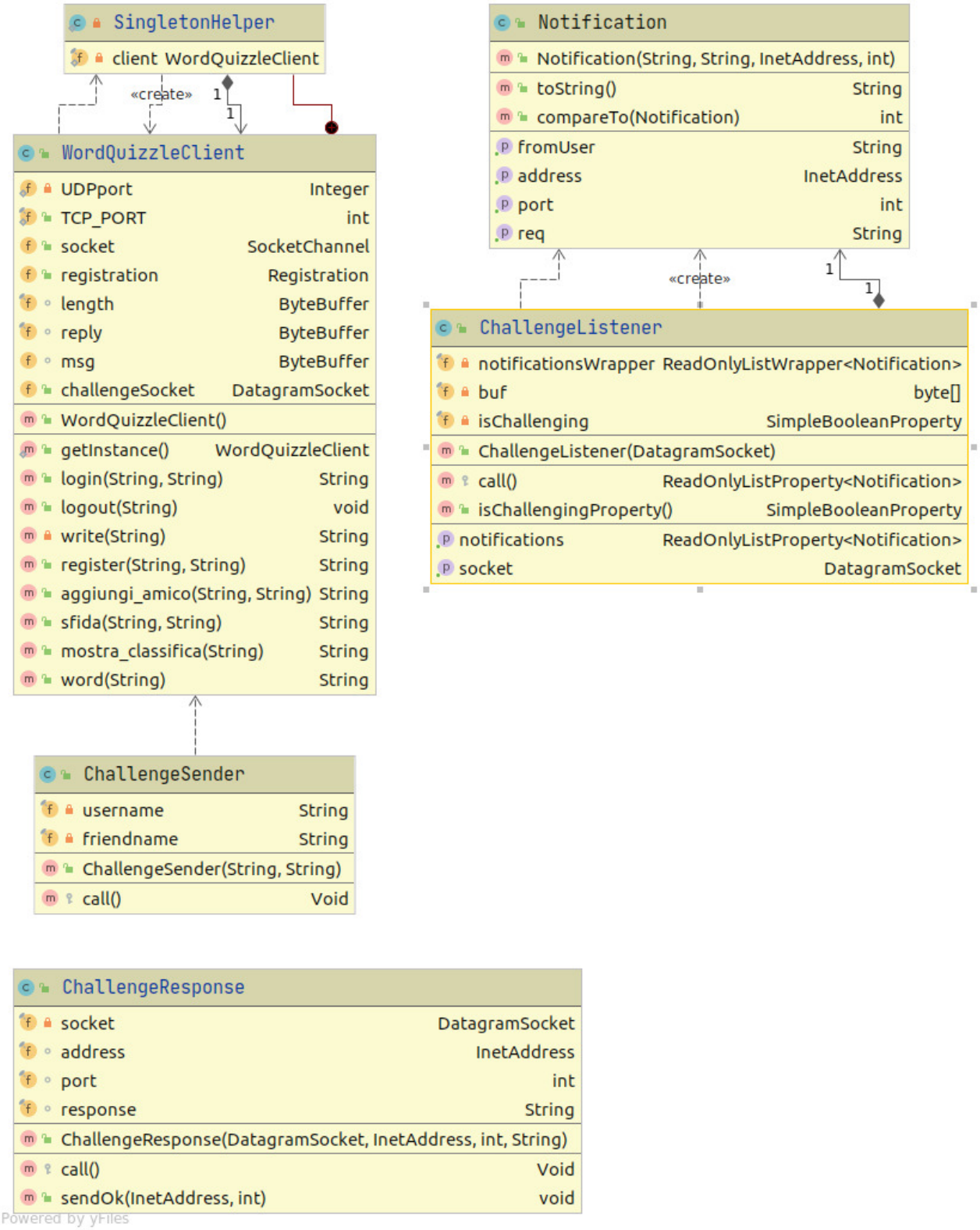
Password

La classe per verificare la correttezza delle password. Utilizza SHA512 con salt per evitare di salvare le credenziali in chiaro.

Challenge

Gestisce la logica della sfida e richiede le traduzioni a inizio sfida.

4.2 Modulo Client



4.2.1 Logica

WordQuizzleClient

Singoletto con tutte le funzioni richieste dal progetto, inizializza socket TCP e UDP e gestisce la scrittura su socket TCP.

Notification

Modello per notifiche.

ChallengeListener

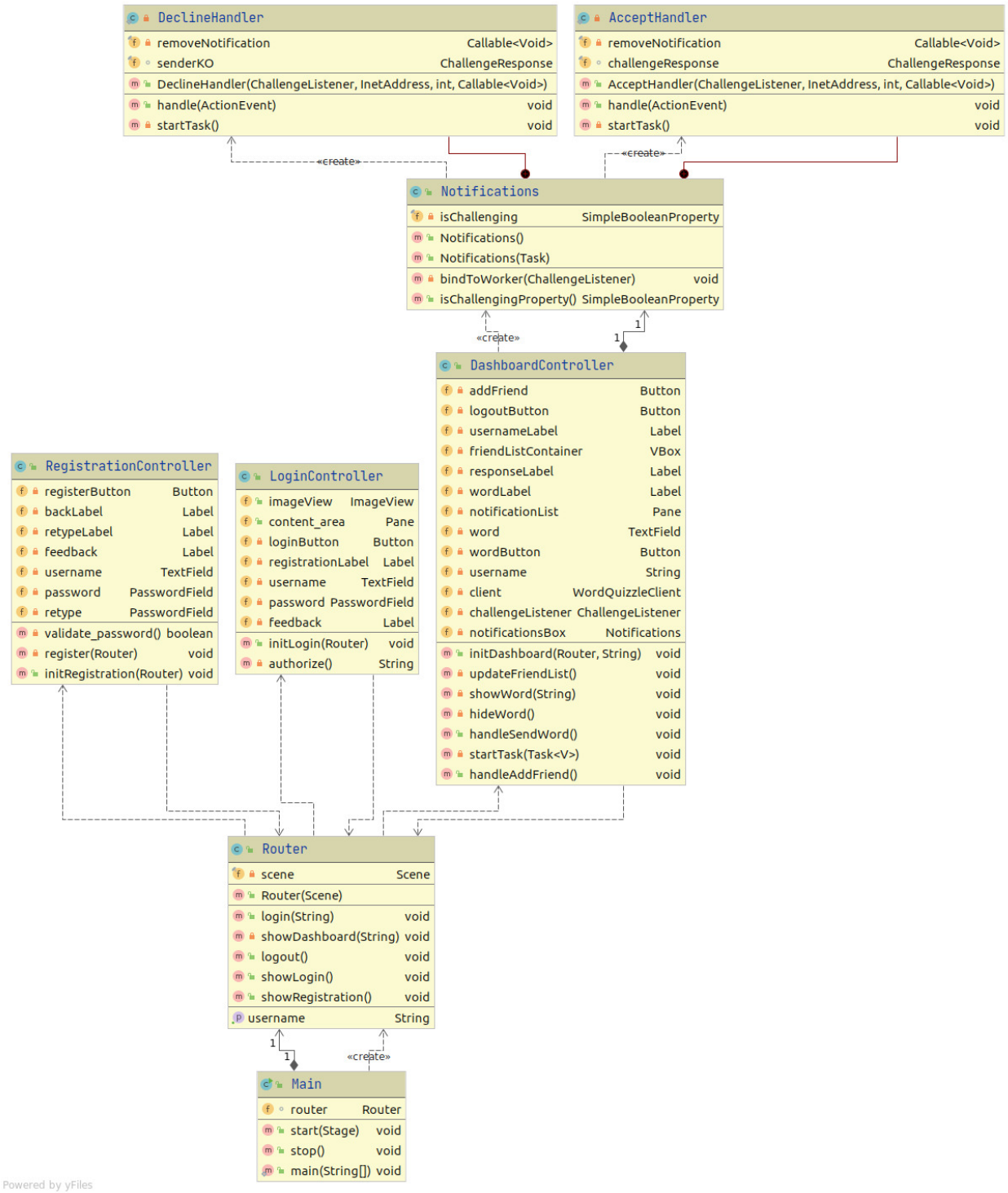
Task che attende sfide tramite pacchetti UDP, ha una ReadOnlyListProperty<Notification> per comunicare con DashboardController.

ChallengeResponse

Task per rispondere ad una richiesta di sfida.

ChallengeSender

Task per inviare una richiesta di sfida, utilizza MessageProperty per notificare il risultato con DashboardController.



4.2.2 Javafx

Main

Esegue l'applicazione, crea la scena iniziale utilizzata da Router per visualizzare la GUI. In chiusura chiama la funzione logout.

Router

Gestisce il cambio di scene fra login, registrazione e dashboard.

LoginController

Chiama la funzione login, inizializza username usato come variabile globale durante la sessione di gioco, da un feedback per errori su login

RegistrationController

Chiama funzione register, effettua una validazione lato client della conferma password in modo responsive, da feedback per errori sulla registrazione.

DashboardController

Inizializza ChallengeListener passato come argomento a Notifications, crea programmaticamente la classifica chiamando mostra_classifica e posizionando a lato di ogni Label dei nomi degli amici un bottone per sfidarli. Fa comparire dei controlli grafici quando inizia una sfida utilizzati per chiamare la funzione word.

Notifications

Classe che estende una VBox. Creata programmaticamente, gestisce le notifiche, inizializza ChallengeResponse ed espone una SimpleBooleanProperty per comunicare con DashboardController l'inizio della sfida in caso in cui ne venga accetta una.

5. Istruzioni di compilazione ed esecuzione su ubuntu

Ho deciso di utilizzare l'ambiente di sviluppo integrato Intellij e Apache Maven per il build automatico.

```
$sudo apt update

$sudo apt install maven

$sudo apt install default-jdk

…/WordQuizzleUnipi:$cd ./Server && mvn clean compile
exec:java

…/WordQuizzleUnipi:$cd ./Client && mvn clean compile
javafx:run
```