Allegato Tecnico

Gruppo SWEight - Progetto Colletta

SWEightGroup@gmail.com

Informazioni sul documento

Versione	1.0.0
Approvatore	Damien Ciagola
Redattori	Enrico Muraro Damien Ciagola Francesco Magarotto
Verificatori	Sebastiano Caccarto Alberto Bacco
Uso	Esterno
Distribuzione	Prof. Vardanega Tullio Prof. Cardin Riccardo Gruppo SWEight

Descrizione

Documento che ha lo scopo di illustrare la Product Baseline, ponendo attenzione alle scelte architetturali e alla copertura di use case e requisiti funzionali

Registro delle modifiche

Versione	Data	Descrizione	Autore	Ruolo
1.0.0	2019-04-05	Approvazione per il rilascio	Damien Ciagola	*Responsabile di Progetto*
0.4.0	2019-04-05	Verifica del documento	Enrico Muraro	*Verificatore*
0.4.0	2019-04-03	Stesura sezione §3	Francesco Magarotto	*Redattore*
0.3.0	2019-04-02	Stesura sezione §4	Damien Ciagola	*Redattore*
0.2.0	2019-04-02	Stesura sezione §5	Francesco Magarotto	*Redattore*
0.1.0	2019-03-31	Stesura sezione §1 e §2	Damien Ciagola	*Redattore*
0.0.1	2019-03-30	Creazione scheletro del documento	Damien Ciagola	*Redattore*

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di fornire agli sviluppatori uno specchietto informativo sul design strutturale e logico della piattaforma Colletta. Il documento sarà inoltre corredato da diagrammi UML 2.X delle principali scelte prese dal gruppo SWEight e descriverà le tecnologie utilizzate nella realizzazione dell'applicazione.

1.2 Scopo del prodotto

Il prodotto da realizzare consta in un'applicazione web che fornisca uno strumento per creare e svolgere esercizi di analisi grammaticale, e al contempo né raccolga i risultati. I dati raccolti verranno impiegati dagli sviluppatori dell'azienda proponente come strumento per il miglioramento di algoritmi di apprendimento automatico$_G$. Nello specifico il prodotto verrà utilizzato da tre tipologie di utenti: le/gli insegnanti che si occuperanno della creazione degli esercizi, gli allievi che potranno svolgere gli esercizi e ottenere delle valutazioni e gli sviluppatori che filtreranno i dati secondo alcuni criteri, e infine li scaricheranno.
Il prodotto si interfaccerà con un'applicazione di PoS-tagging$_G$, come FreeLing$_G$, a cui verrà delegata l'esecuzione dell'analisi grammaticale delle frasi.

1.3 Glossario

Al fine di rendere il documento il più comprensibile possibile e permetterne una rapida fruizione, viene allegato il *Glossario_v3.0.0* in cui sono presenti i termini contraddistinti dal pedice G. Tali termini includono abbreviazioni, acronimi, termini di natura tecnica, oppure sono fonte di ambiguità e pertanto necessitano di una definizione che renda il loro significato inequivocabile. Ogni termine, solo alla prima occorrenza per documento, verrà contrassegnato con la dicitura sopra indicata e rimanderà alla medesima definizione nel *Glossario_v3.0.0*.

2 Installazione ed esecuzione

Il codice relativo alla Product Baseline lo si può trovare al seguente link:

linkAllaRepo.

2.1 Maven project

Una volta fatto il clone della repository o dopo aver scaricato lo zip, posizionarsi nella directory "Mockup-V2/Backend" della repo e utilizzare i seguenti comandi:

```
mvn clean install
```

```
mvn exec:java
```

2.2 Node.js

Per lo sviluppo del front-end ci si avvale dell'ultima versione Long Term Support (LTS) di Node.js, che, al momento della stesura di questo documento, è la 10.15.1 LTS. Node.js è reperibile al seguente link:

Node.js.

Il file package.json contiene tutte le configurazioni e dipendenze del progetto. Per installare tutti i moduli necessari è necessario eseguire il seguente comando nella cartella contenente il file package.json:

```
npm install
```

Per eseguire il progetto è necessario usare il comando:

```
npm start
```

3 Architettura

3.1 MongoDB Database

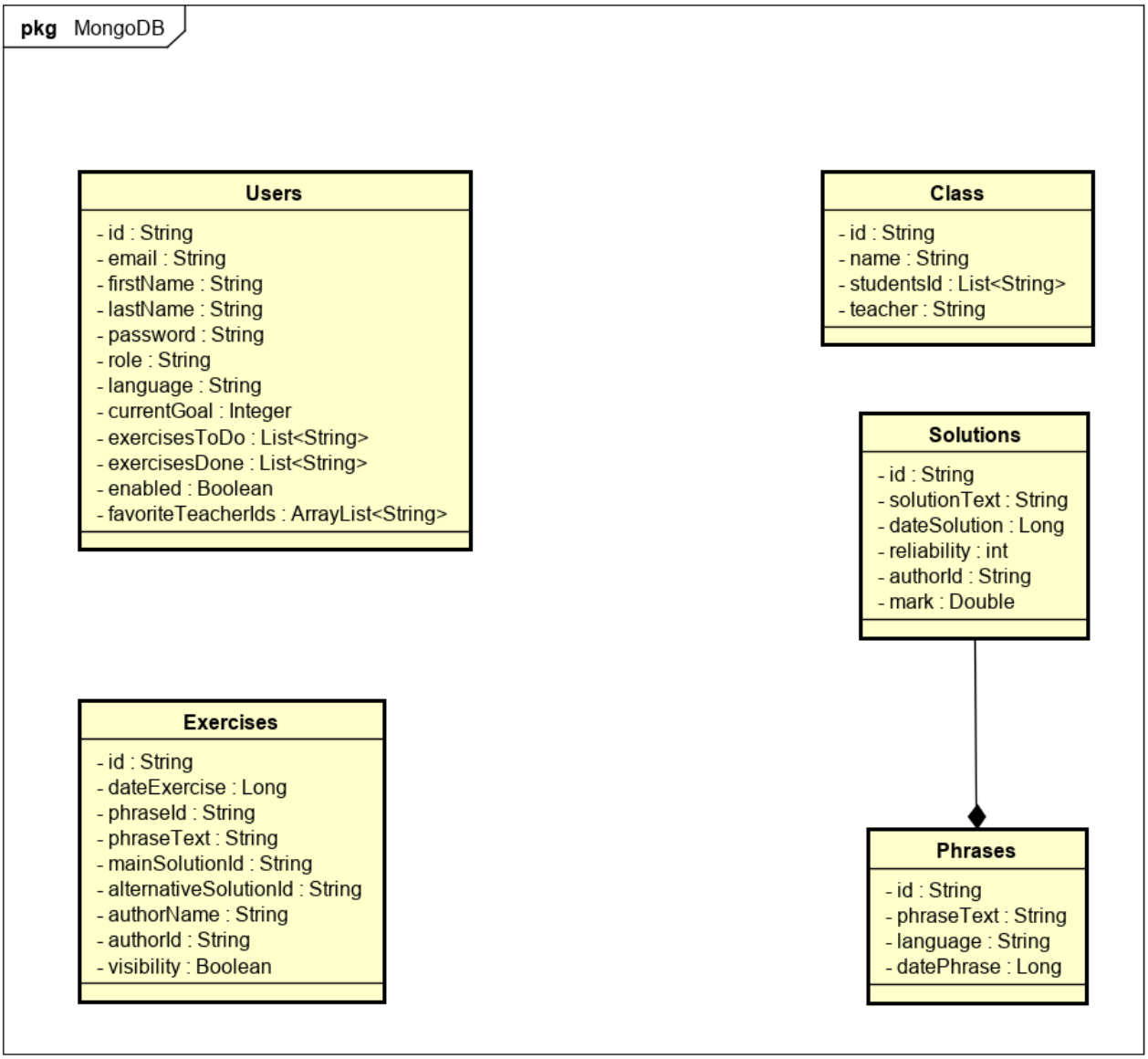

Figura 1: Exercise insert

3.2 Design pattern utilizzati

3.2.1 Backend

3.2.1.1 Adapter L'utilizzo della libreria di Freeling, per il pos-tagging, ha richiesto la creazione di un Adapter nella variante Object Adapter per poter adattare le funzionalità strettamente necessarie.

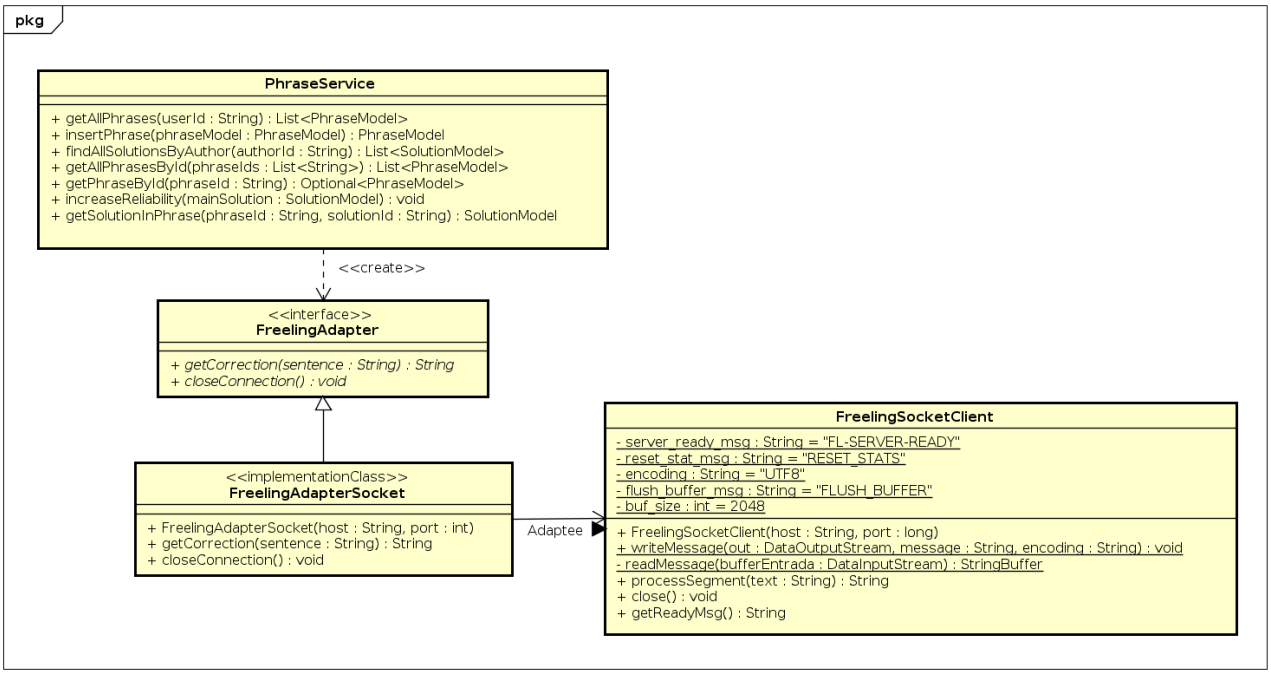

Figura 2: Diagramma delle classi Adapter

La classe FreelingSocketClient viene fornita dai creatori della libreria e si occupa di realizzare la connessione con il server Freeling scritto in C++.

3.2.1.2 Controller - Service - Repository - Model L'architettura realizzata all'interno di Spring Web consta nella presenza di:

- **Controller**: un a cui è delegato il compito di gestire le richieste provenienti dalla parte frontend e alla cattura delle eccezioni;
- **Service**: realizzano la business-logic;
- **Repository**: realizzano il layer di persistenza gestendo la base di dati;
- **Model**: rappresentano oggetti Plain Old Java Object, un'istanza di un model rappresenta un documento di una collezione.

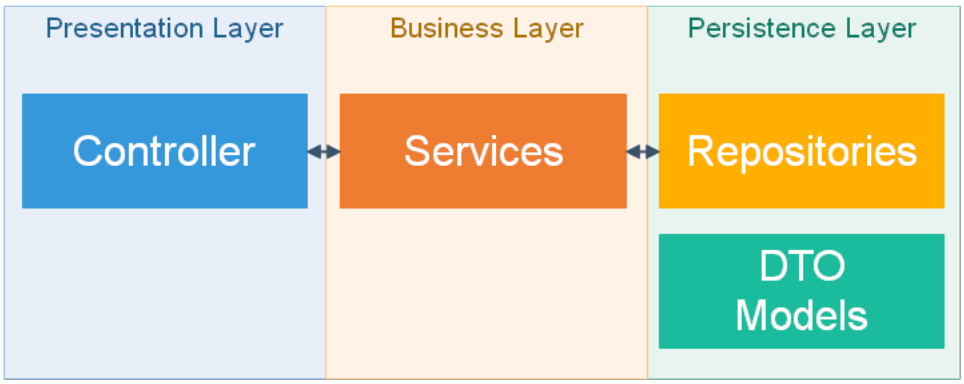

Figura 3: Scherma generale architettura in Spring

3.2.2 Frontend

3.2.2.1 Flux

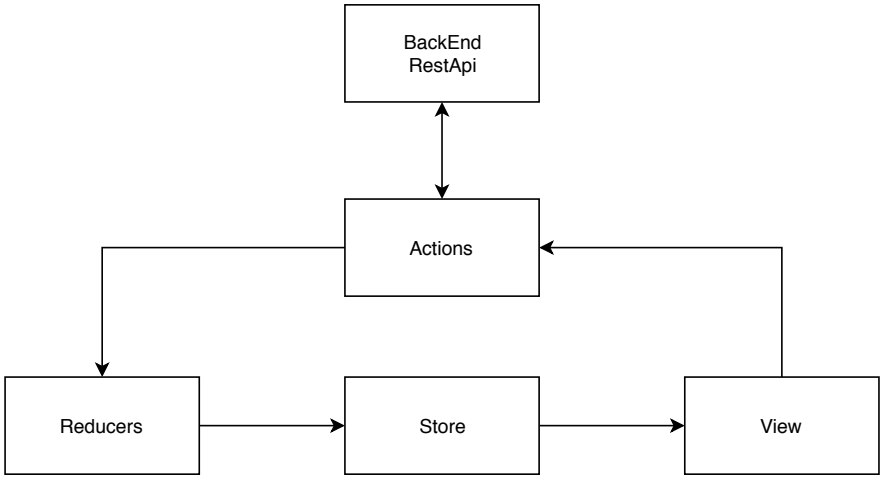

Figura 4: Schema del design pattern Flux

Per la frontend viene utilizzato React con design pattern Redux, un'evoluzione del design pattern Flux. In Redux, tutti i dati scorrono in modo unidirezionale attraverso i seguenti componenti:

- **Store:** oggetto immutabile che contiene l'intero stato dell'applicazione in maniera centralizzata;
- **Reducers:** sono funzioni pure. Ogni volta che i Reducer ricevono una nuova azione, processano l'azione ricevuta e, in caso sia necessario apportare delle modifiche allo stato, restituiscono un nuovo oggetto
- **Action creators:** funzioni che facilitano la gestione del dispatch (creazione di azioni da mandare ai reducers);
- **View:** componenti grafiche, il loro contenuto dipende dallo store. Sono implementate attraverso un *Observer Pattern* sullo store stesso. Ad ogni cambiamento di stato prodotto da un reducers vengono renderizzate le componenti collegate ad esso.

3.3 Diagrammi dei package

3.3.1 Data Transfer Object

Le classi Helper rappresentano Data Transfer Object (DTO), vengono utilizzate dalla classe `Controller.java` per fornire un oggetto per il trasferimento dati dalla frontend senza ricorrere a JSON troppo complessi.

3.3.2 Builder

I model sono dotati ognuno di un builder, tale classe interna non è codificata ma realizzata tramite un plugin denominato lambok. A seguire viene mostrato un esempio di un builder creato automaticamente.

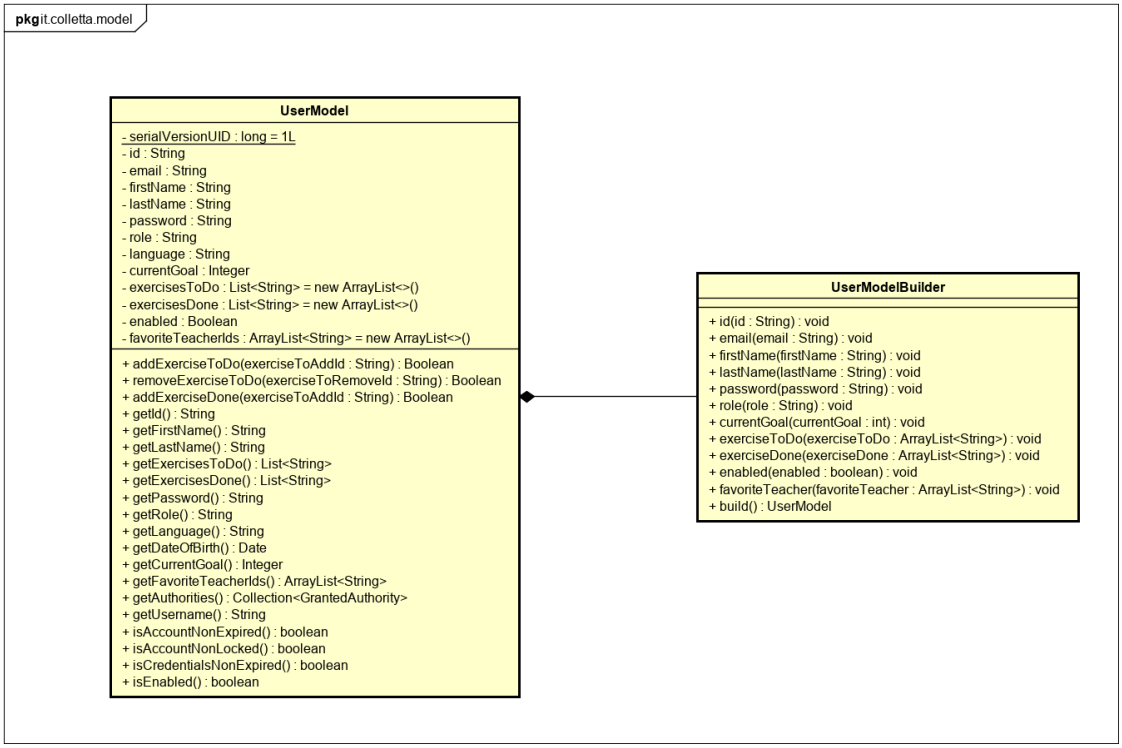

Figura 5: Esempio di builder presente nella classe UserModel

3.3.3 Model

Viene di seguito riportato il diagramma delle classi del package model.

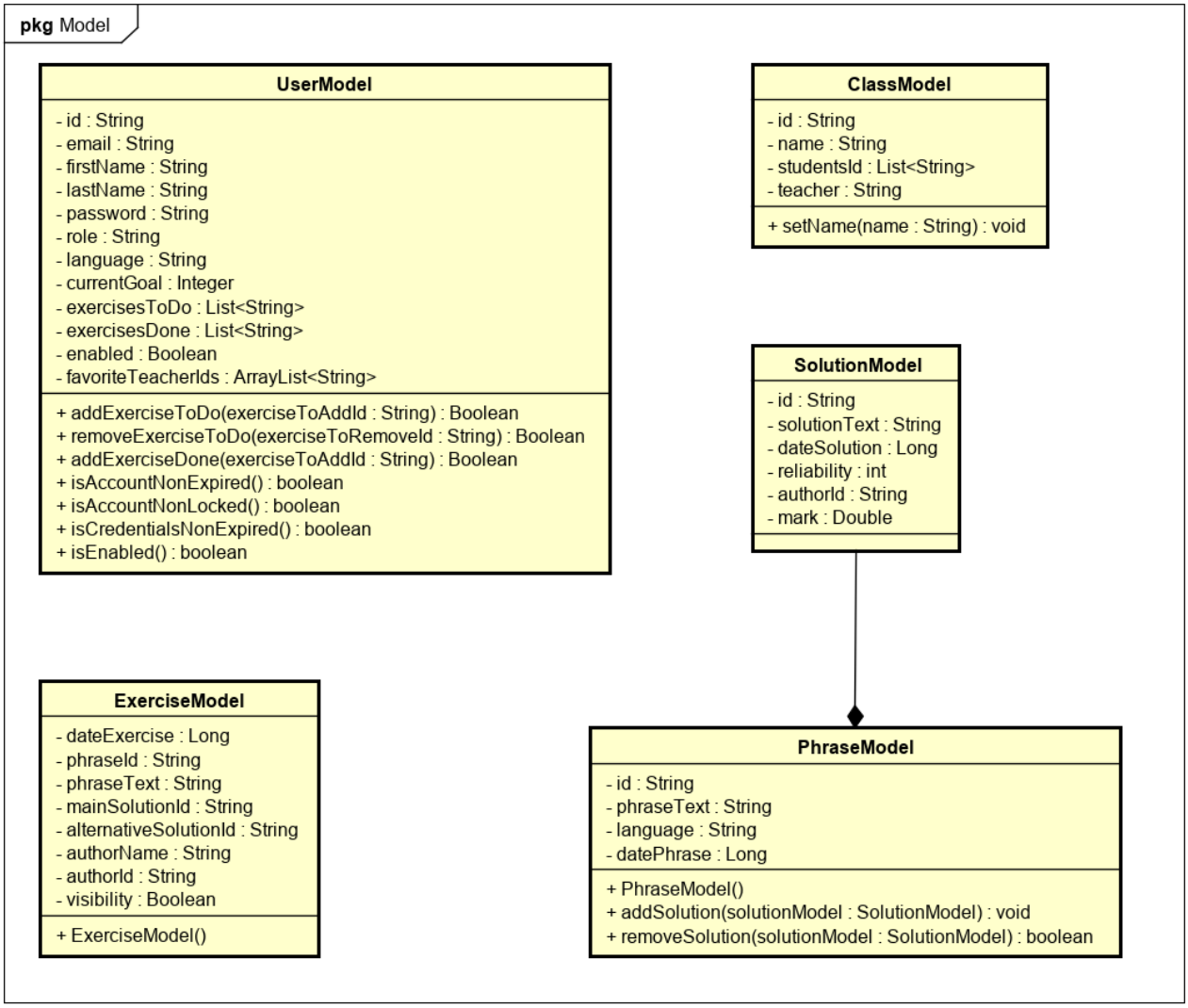

Figura 6: Model

3.3.4 Controller e service

Viene di seguito riportato il diagramma delle classi di package controller e service.

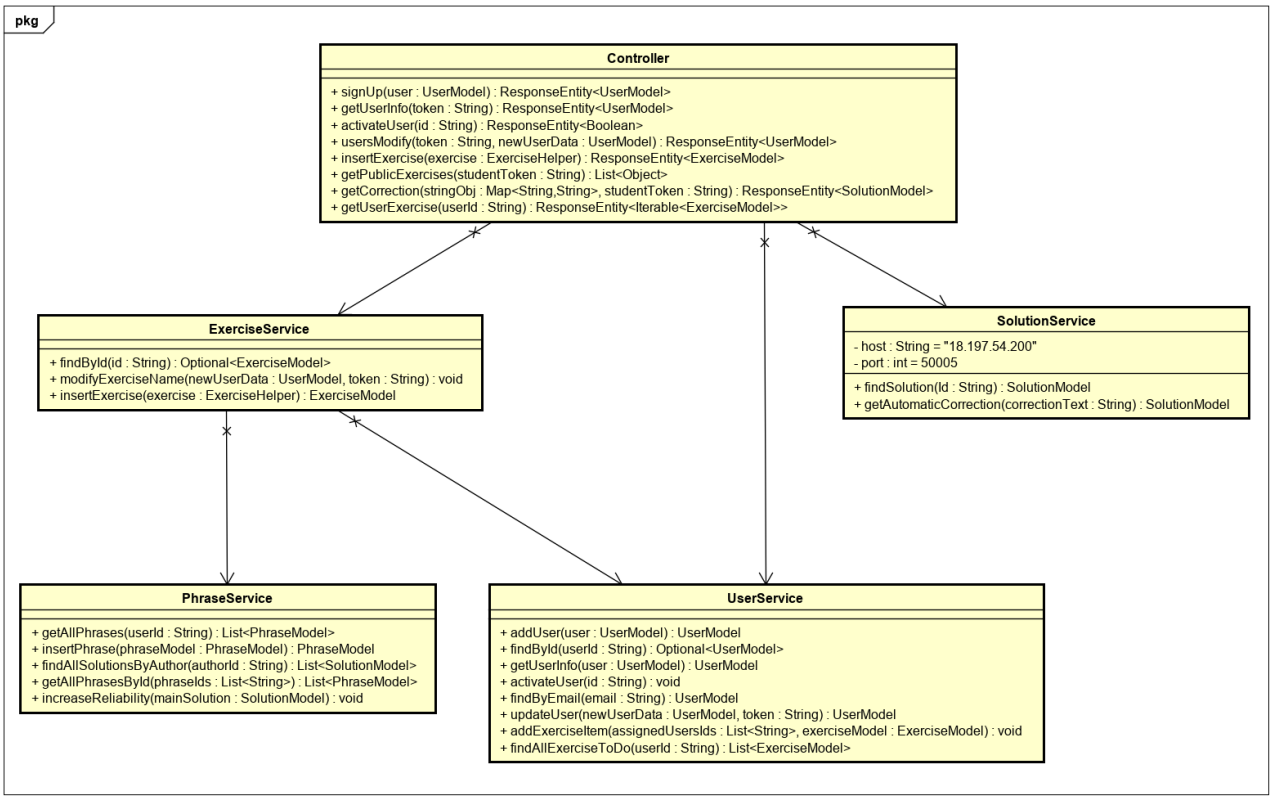

Figura 7: Controller e Service

3.3.5 Service e repository

Viene di seguito riportato il diagramma delle classi di package service e repository.

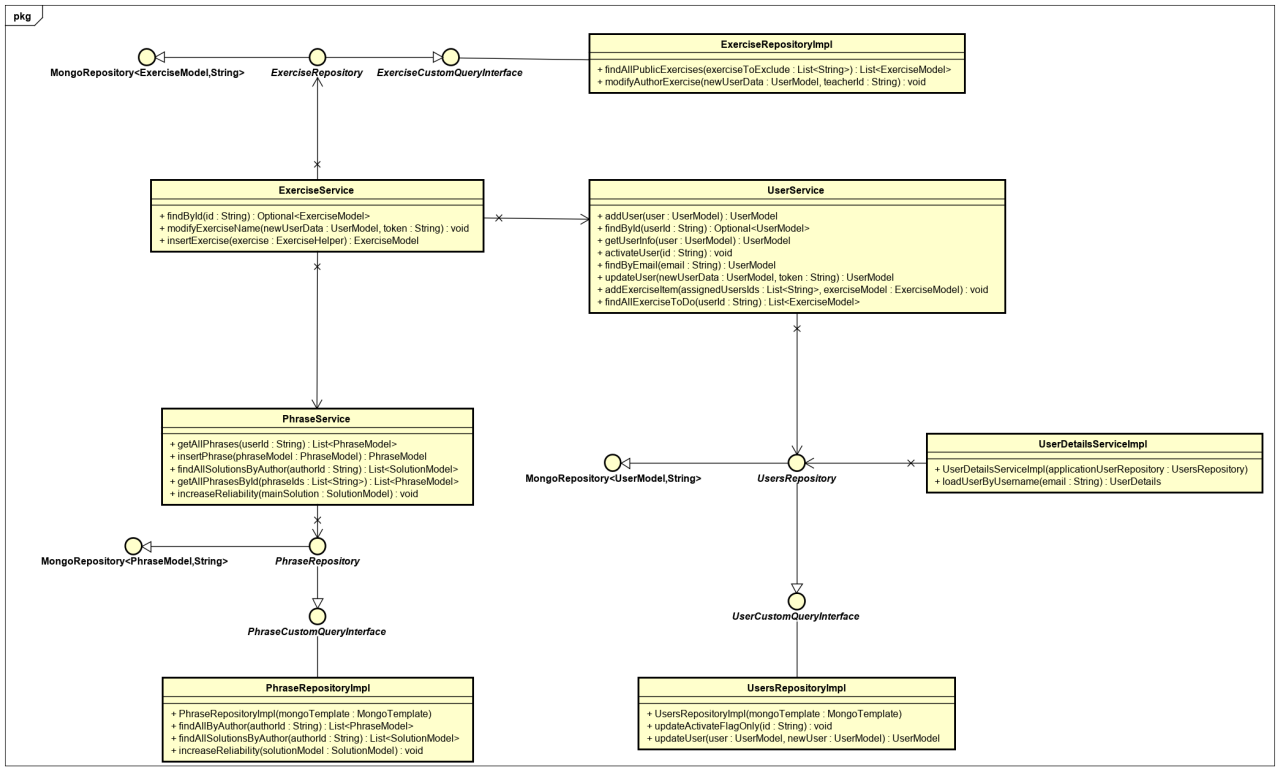

Figura 8: Service e Repository

3.4 Diagrammi di sequenza

3.4.1 Inserimento di un utente

Il diagramma di sequenza rappresenta l'azione di inserimento di un esercizio nel sistema

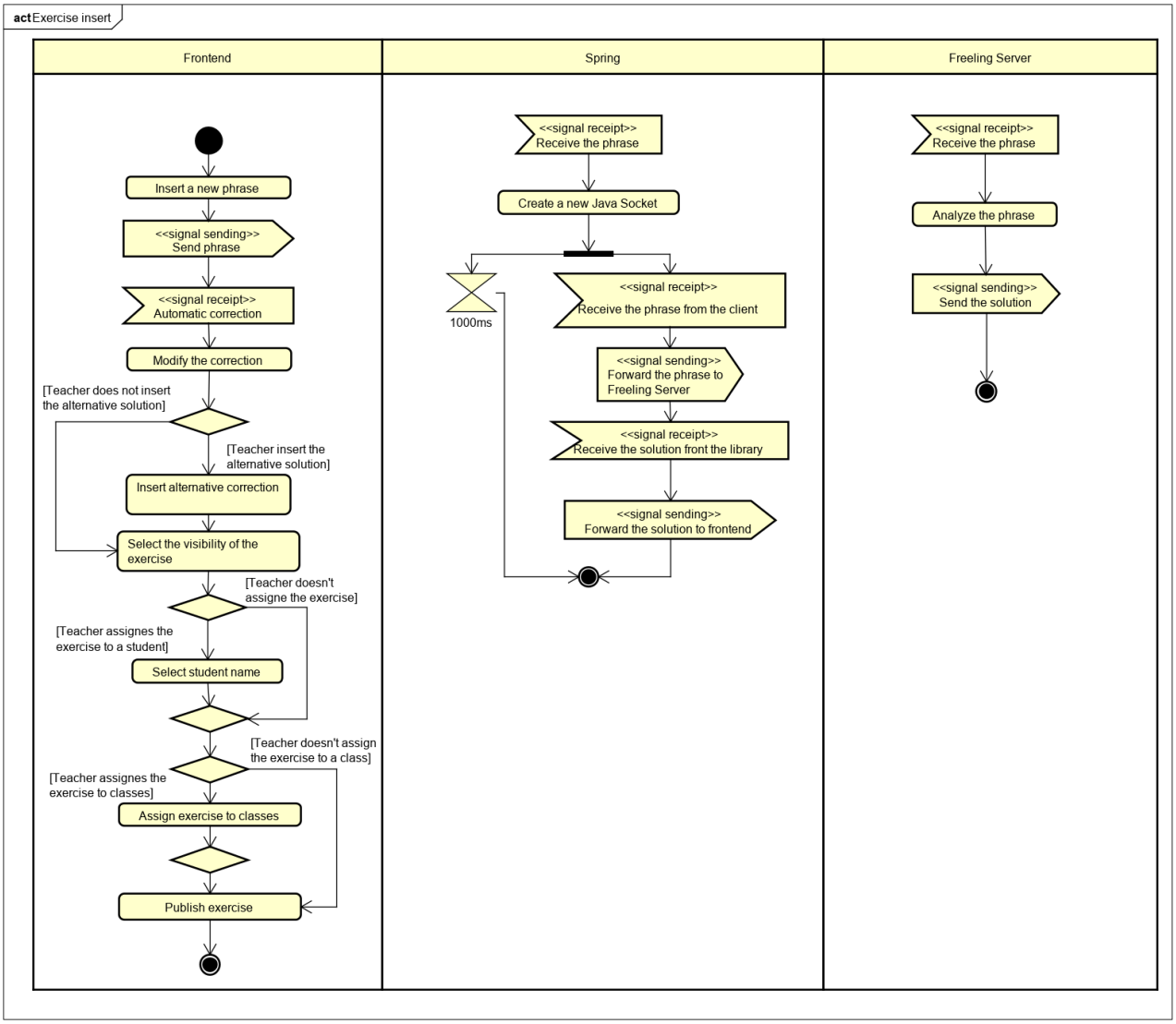

Figura 9: Exercise insert

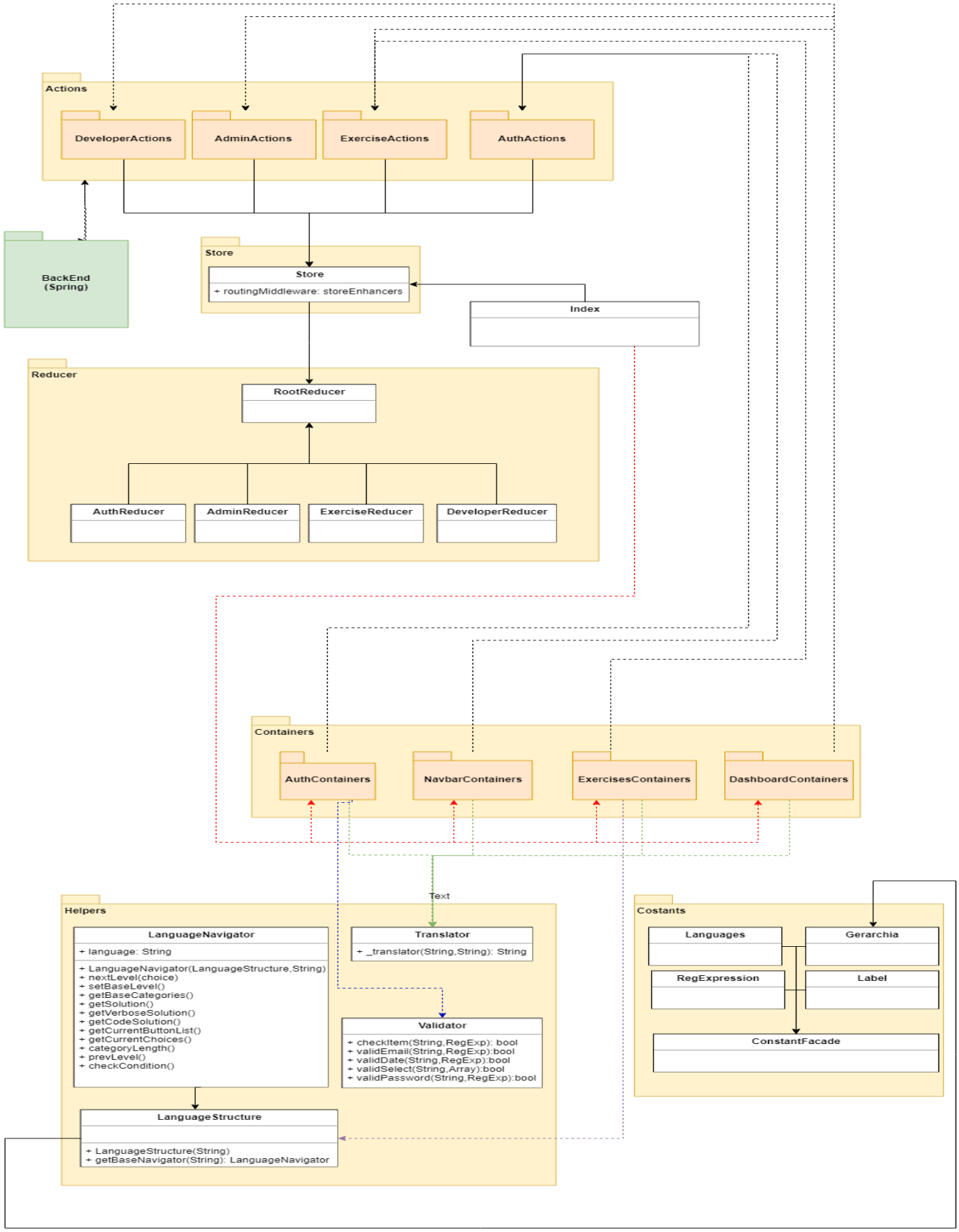

Figura 10: UML - FrontEnd

3.4.2 Login

Il diagramma di sequenza riportato qui di seguito raffigura il processo di login, durante il quale l'utente che vuole accedere può essere autenticato dal sistema.

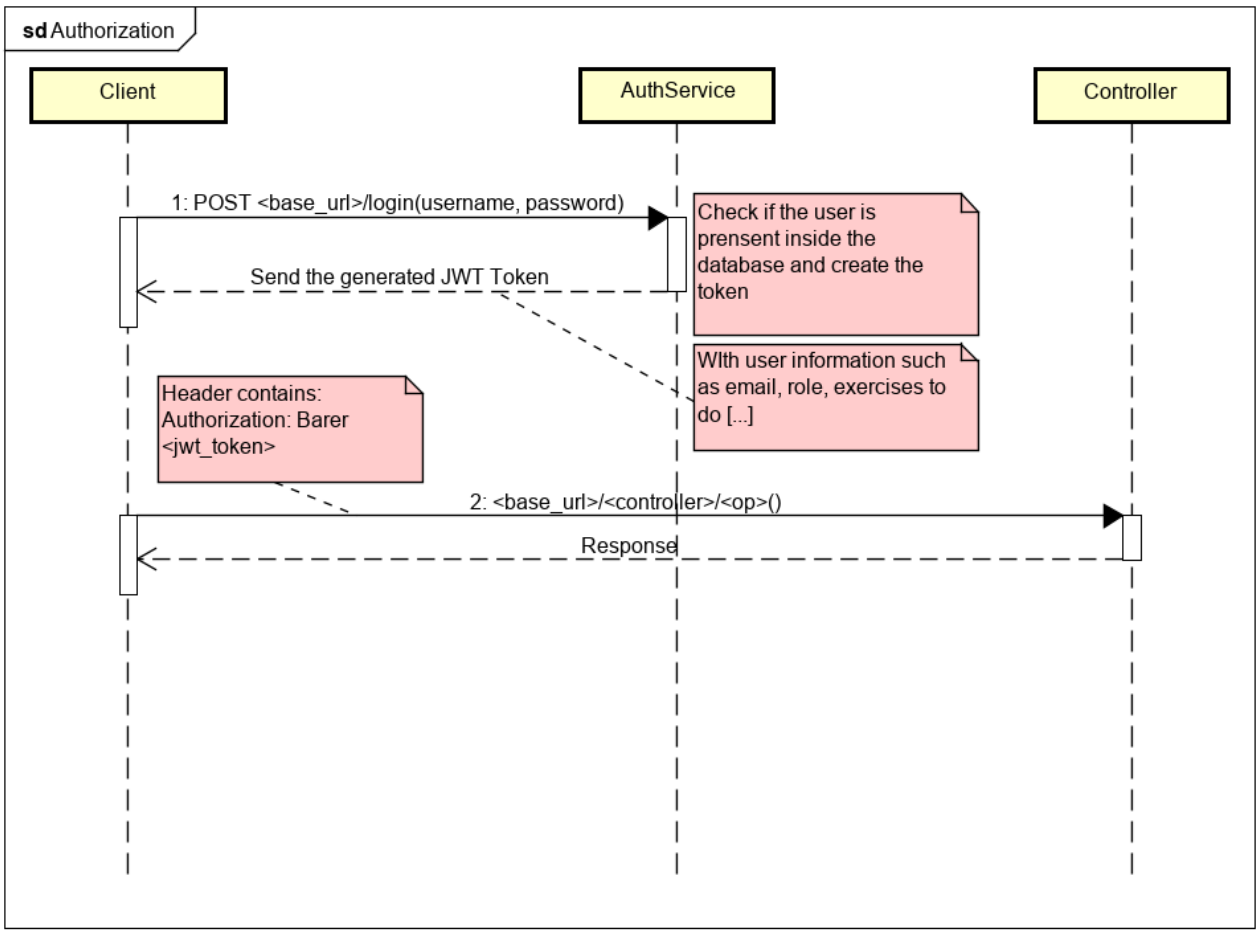

Figura 11: Authorization