**Rapport de projet**

**Java EE**

****

**« EasyStock, une plateforme de gestion de machines à café collaborative »**

**Paritosh Sharma MAHABIRSINGH**

**Malik BENCHEMAM**

Table des matières

[Introduction 3](#_Toc31986389)

[Architecture logicielle 3](#_Toc31986390)

[Base de données 4](#_Toc31986391)

[Stratégie de tests 5](#_Toc31986392)

[Comparaison des technologies pour la partie interface utilisateur 6](#_Toc31986393)

[Implémentation de l’authentification 6](#_Toc31986394)

[Manuel partie Administration / Manuel utilisateur 6](#_Toc31986395)

Introduction

Ce projet constitue une plateforme de gestion de machine à café collaborative. Il nous a permis de réaliser ensemble un ensemble de composants permettant d’implémenter une partie d’administration.

Chaque utilisateur peut créer son propre compte, puis éditer ses informations.

Un administrateur (**username : admin, password : admin**) peut accorder des privilèges aux utilisateurs existants, effectuer des opérations CRUD sur les utilisateurs.

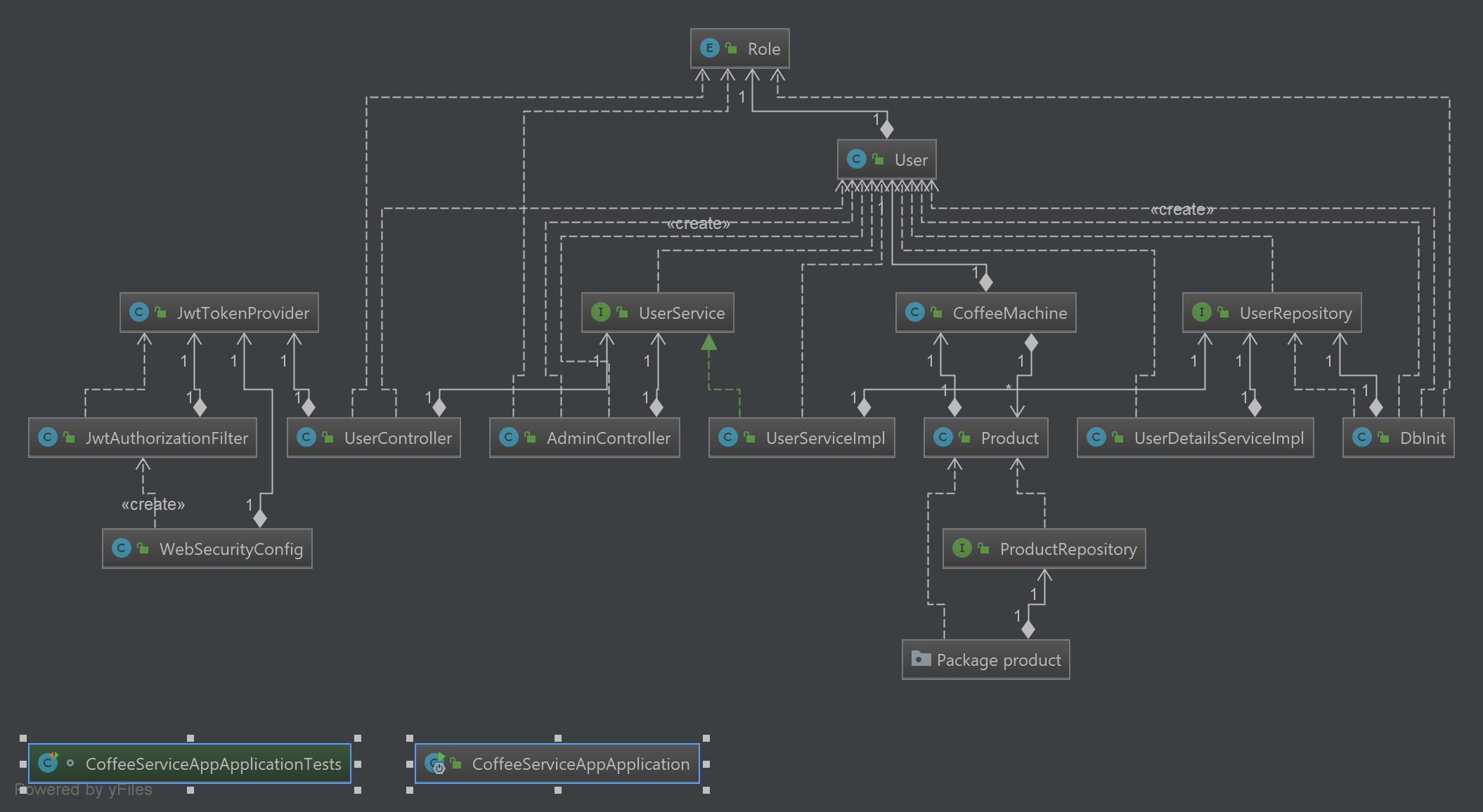
Une notice de déploiement est fourni en annexe afin configurer l’application.

**Authentification d’un utilisateur**

**Création d’un nouvel utilisateur**

**Gestion des utilisateurs**

**Architecture** logicielle

*****Diagramme UML de l’application*

Ce Diagramme est disponible en annexe.

**Controller**

**Service**

**Repository**

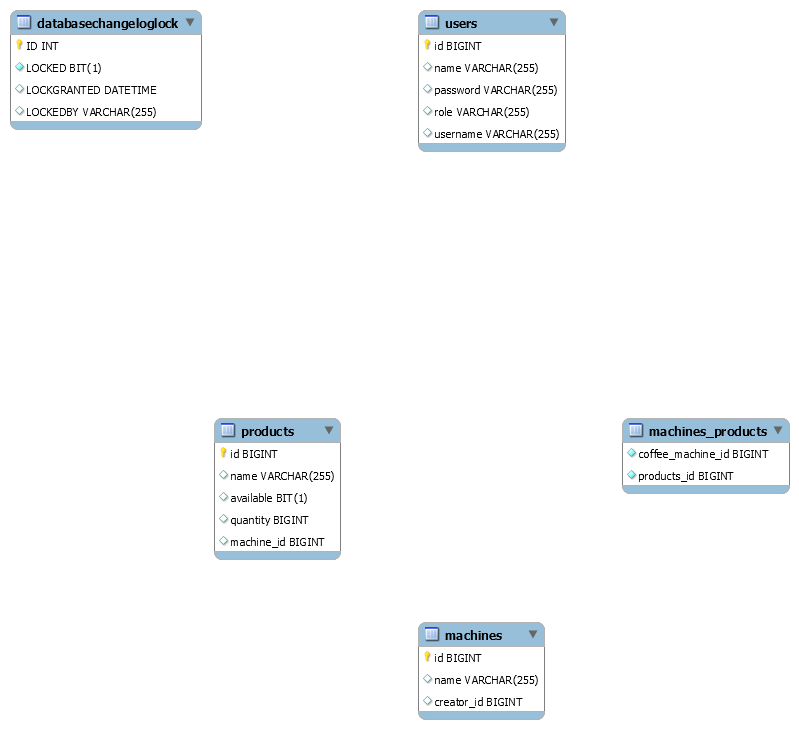
**DTO**

**Entity**

Base de données

Pour la base de données, nous avons adopté la méthodologie « **Coding First** », en déléguant la création de la base à notre ORM (**O**bject **R**elational **M**apping).

Nous avons utilisé le concept JPA Java Persistence API afin de générées automatiquement grâce aux annotations présentes dans les entités.



*Schéma relationnel*

Stratégie de tests

La stratégie de test adoptée fut les tests unitaires, des tests d’IHM et des tests d’API.

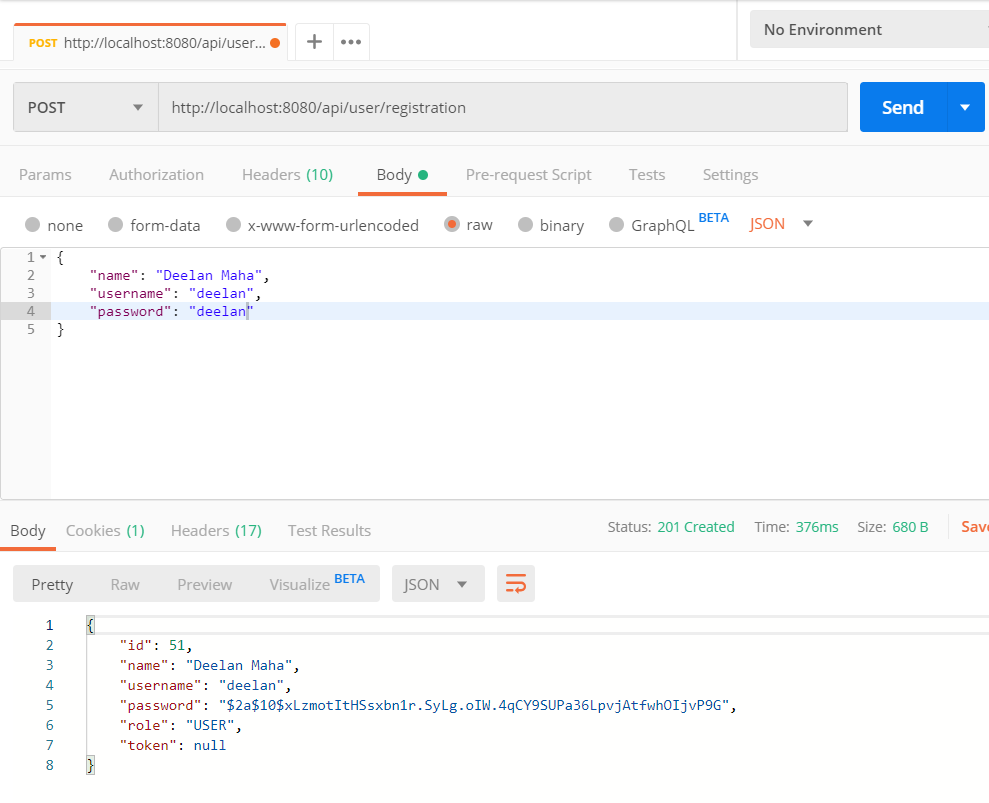
Les tests unitaires ont été générés automatique avec le plugin Intellij SquareTest. Cependant, ces tests n’ont pas été concluants.

**Tests IHM**

Nous avons développé une application Single Page Application avec Angular, les erreurs de vue ont pu être corrigé grâce à ces tests, ainsi que les problèmes entre les composants, et les problèmes portant sur les évènements.

**Test d’API**

Les tests unitaires n’étant pas concluant, nous avons donc décidé d’effectuer des tests d’API. Nous avons donc effectué des requêtes avec plusieurs utilisateurs, possédant un rôle d’administrateur ou non.



Comparaison des technologies pour la partie interface utilisateur

Ce projet nous a permis de manipuler le Framework Angular, cependant, plusieurs Framework étaient possibles.

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Vue.js**

**Angular**

**ReactJS**

Après avoir pesé le pour et le contre, nous avons opté pour Angular, en effet le projet Annuel étant basé sur cette technologie, nous avons décidé de nous former sur cette technologie pour ce projet JEE.

Afin de parfaire au mieux notre auto-formation d’Angular, nous sommes parti d’une vue **from scratch** et non d’une api open-source modifié.

Implémentation de l’authentification

Manuel partie Administration / Manuel utilisateur