Projet n° 6

**Créer un site communautaire au tour de l’escalade**



Document fonctionnel

**Habib BAH**

Parcours Développeur d’Application Java

**Sommaire**

1. Introduction
2. Cahier des charges
3. Diagramme de classe
4. Architecture web
5. Model
6. La couche DAO
7. La couche métier
8. La couche web
9. Gestion des sessions

Introduction

Faisant partie d’un club d’escalade, je dois mettre en place une application web au tour de l’escalade à fin que les grimpeurs comme moi puissent partager notre passion commune autour d’une application web.

Pour ce faire, je vais utiliser une architecture dite MVC et utiliser le Framework Spring afin de pouvoir mettre en place de manière optimale tous les besoins et toute la sécurité dont ont besoin mes amis grimpeurs, pour pouvoir profiter au maximum de cette application et de partager cette expérience ainsi que des informations sur l’ensemble des sites d’escalade que chacun connait ou possède.

Dans les pages qui suivent, je vais détailler les différentes phases de création de notre application web et expliquer chaque module que comporte l’application.

Cahier des charges

D’après le sondage que j’ai effectué auprès de mes amis grimpeurs, notre futur application web doit permettre :

1. De partager des informations sur les sites, leurs secteurs et les voies de chaque secteur (hauteur, cotation, nombre de points…).
2. De faire des recherches sur des critères afin de trouver plus facilement le prochain site de grimpe.
3. De laisser des commentaires.
4. De présenter les topos qui existent et les sites/secteurs qu’ils couvrent.
5. D’avoir un espace de prêt de topo où les propriétaires de topo peuvent proposer le prêt de leur topo et les personnes intéressées peuvent voir les topos disponibles et les réserver pour une période.

Voici entre autres, ce que notre application web devra pouvoir proposer à nos utilisateurs.

Diagramme de classe

Le diagramme de classe présente l’ensemble des entités et des relations qui existent entre les entités.



Architecture web



L’architecture web mise en place se base sur l’architecture MVC.

L’application est déployée à l’aide d’un **server TomCat**, via le Framework **Spring Boot.**

Elle est composée de deux couches principales : la couche **web** et la couche **DAO.**

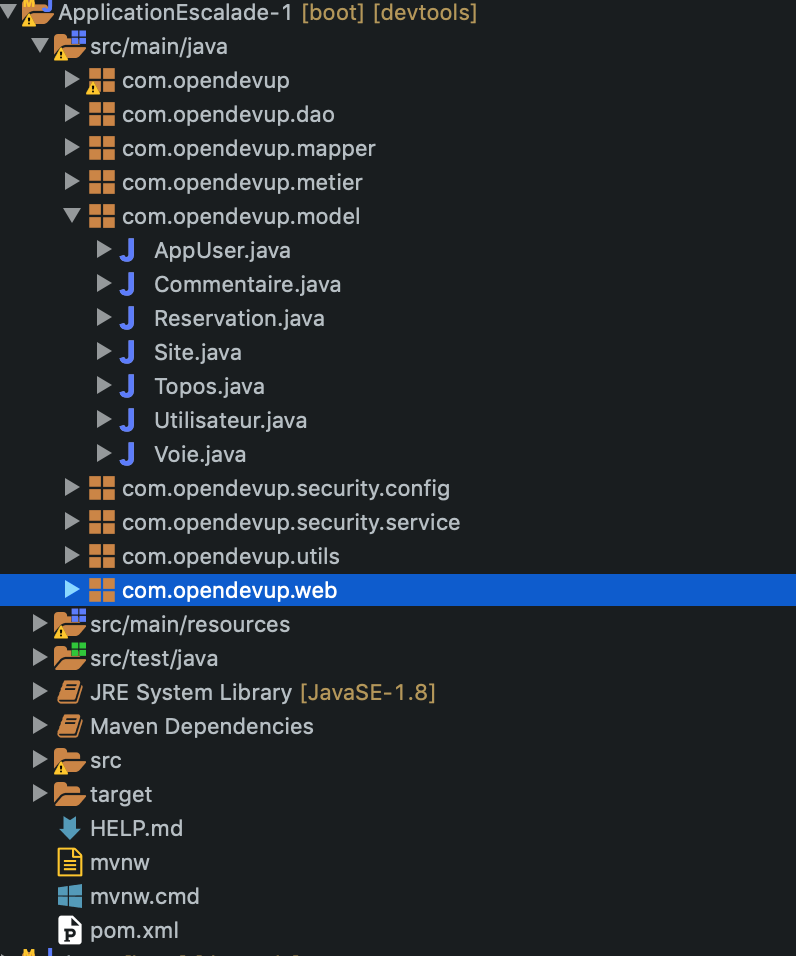
* La couche **web** : elle contient :

1. **Le Controller** qui a pour rôle de faire un aiguillage pour le traitement des requêtes des utilisateurs.
2. **Le model** dans le quel le traitement est effectué
3. **La vue** qui retourne une vue à l’utilisateur

* La couche **DAO** : elle contient :

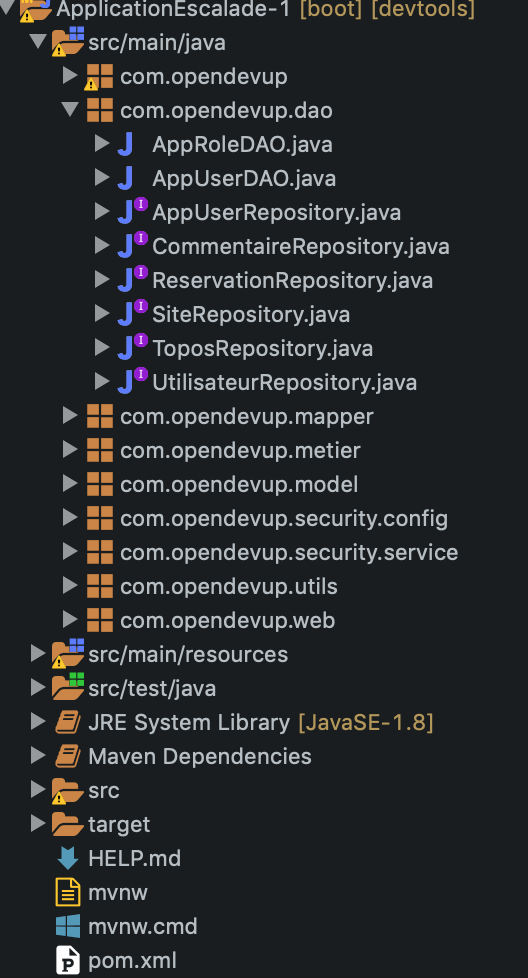
1. **Les entités** qui seront stockées dans notre base de données
2. **Spring DAO** qui est un outil spring qui fait lui-même appel à JPA
3. **JPA** qui va nous permettre de définir les règles de connexion grâce à Hibernate
4. **Hibernate** qui va réguler toutes ses règles afin qu’on puisse se connecter grâce aux informations se trouvant sur notre base de données

Les model



Notre package model regroupe l’ensemble de nos model à savoir : **Topo**, **AppUser**, **Site**, **Réservation**, **Commentaire** etc…

La couche DAO

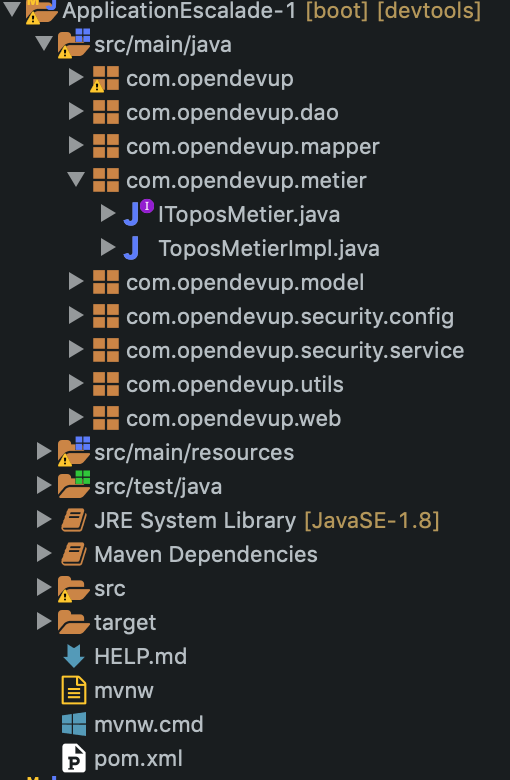


Dans cette couche, nous avons défini des interfaces qui étendent **JPARepository** pour certaines et **CrudRepository** pour d’autres, afin de pouvoir avoir accès aux méthodes nous permettant de faire des requêtes dans notre base de données, ses méthodes sont toutes préparées par spring.

Dans le cadre de ce projet, nous avons utilisé la méthode **findBY** qui nous permet de faire une recherche par un critère que nous avons défini, la méthode **findAll** qui nous permet de trouver toutes les instances d’une entité.

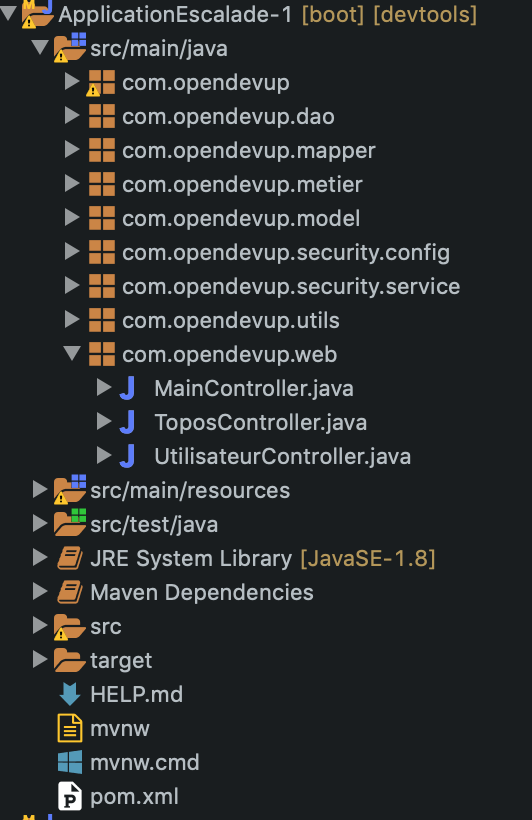
Les classes **AppUserDAO** et **APPRoleDAO** servent à trouver un utilisateur pour la première et à trouver le rôle de cet utilisateur pour la deuxième, cela nous permettra de pouvoir authentifier les utilisateurs entre autres.

La couche métier



Dans cette couche, nous définissons des classes qui implémentent nos interfaces. C’est dans cette couche que nous définissons nos méthodes que nous avons déclarés dans nos interfaces.

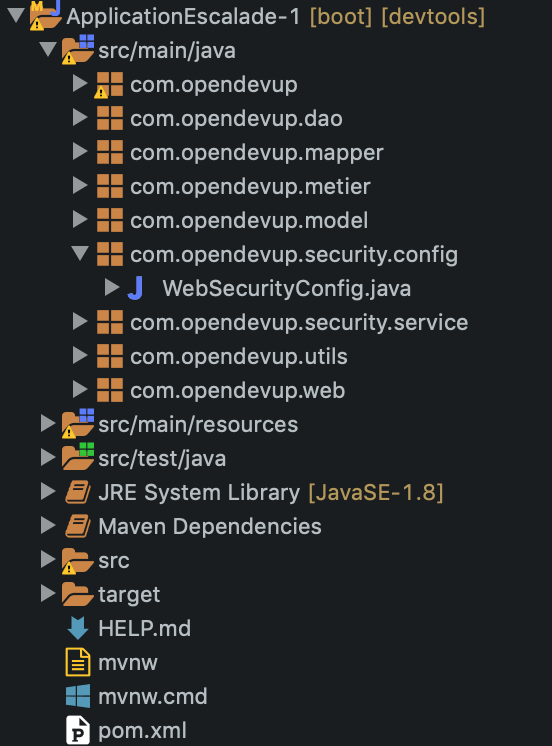
La couche web



Dans la couche web, nous avons défini nos Controller. Pour ce projet, nous avons mis en place trois Controller :

* **MainController :** Qui s’occupe de l’affichage des différentes pages de l’application, de la réservation de topo, des commentaires.
* **ToposController :** Qui s’occupe de tout ce qui est lié au topo : proposer, afficher la liste, faire une recherche
* **UtilisateurController :** Qui s’occupe de l’utilisateur : inscription, connexion

Gestion des sessions



La gestion des sessions est faite grâce à SpringSecurity.

Dans la classe **WebSecurityConfig**, nous définissons toutes les règles de connexion à notre application.

La classe **UserDetailsServiceImpl** nous permet d’authentifier un utilisateur à l’aide de son nom d’utilisateur et son mot de passe. Cette classe implémente l’interface **UserDetailsService** qui est fourni par spring security.