CY TECH

30/11/2024

Rapport de Projet JEE

sColarity

ZEBOUDJ Nawel
JAGHNANE EL IDRISSI Rime
BELCAÏD Manâl
MOREAU Antoine
MALHERBE Axel

Professeur : M. Haddache
UE : Informatique
Année : ING2 - GSI2 - 2024/2025

Sommaire:

I. INTRODUCTION

Le développement d'applications web est devenu un domaine essentiel dans la création de solutions informatiques modernes. Ce projet a pour objectif de concevoir et de réaliser une application de gestion de données utilisant une architecture web. En l'occurrence, il s'agit de créer une application CRUD permettant de gérer des informations sur des étudiants. À travers ce projet, nous avons exploré différentes technologies et approches, allant des servlets Java pour la première version, puis en migrant vers Spring Boot pour bénéficier de ses nombreux avantages en termes de modularité, de maintenabilité et de flexibilité. Ce projet illustre ainsi l'importance de comprendre les bases du développement web tout en s'adaptant aux technologies actuelles.

II. Modélisation

1. Modèle Conceptuel de Données (MCD)

Le Modèle Conceptuel de Données (MCD) présenté ici est basé sur la structure de la base de données du système universitaire. Ce modèle illustre les entités principales du système, leurs attributs et les relations entre elles.

Les principales entités incluent :

- **Compte** : Représente les utilisateurs du système, qu'il s'agisse d'administrateurs, de professeurs ou d'étudiants. Chaque compte possède des attributs comme un nom d'utilisateur, un mot de passe et un rôle.
- **Administrateur**, **Professeur**, **Étudiant** : Ces entités sont des sous-ensembles du compte et contiennent des informations spécifiques comme le nom, la date de naissance, et l'email.
- **Cours** : Représente les cours offerts par l'université, avec des attributs comme le titre, la description, et le crédit du cours.
- **Inscription** : Relie les étudiants aux cours qu'ils suivent, et inclut des informations sur la date d'inscription et les professeurs responsables.
- **Résultat** : Contient les informations relatives aux notes des étudiants dans chaque cours, avec des attributs comme la note et le coefficient.

Chaque **Compte** est lié à un **Administrateur**, un **Professeur** ou un **Étudiant** (1:1).

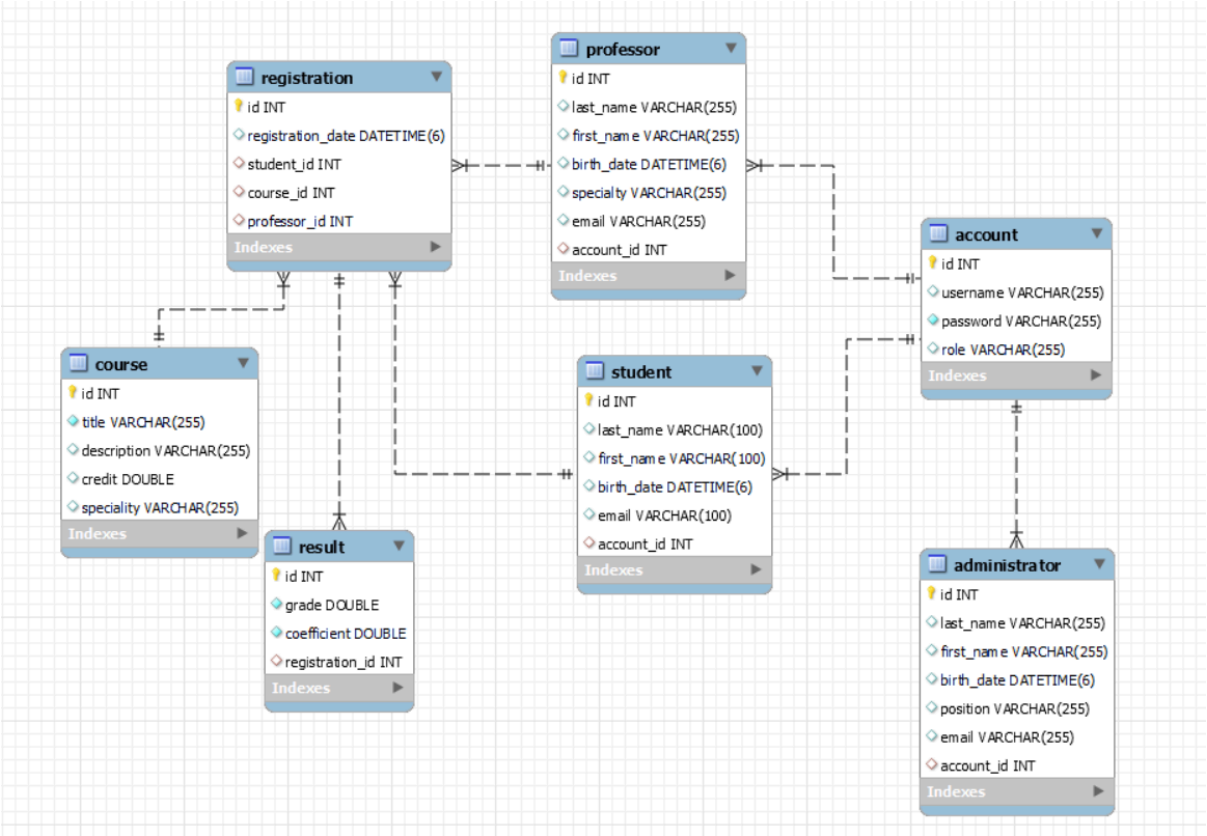
Chaque **Étudiant** peut avoir plusieurs **Inscriptions** (1:N).

Chaque **Cours** peut être lié à plusieurs **Inscriptions** (1:N).

Chaque **Professeur** peut être associé à plusieurs **Inscriptions** (1:N).

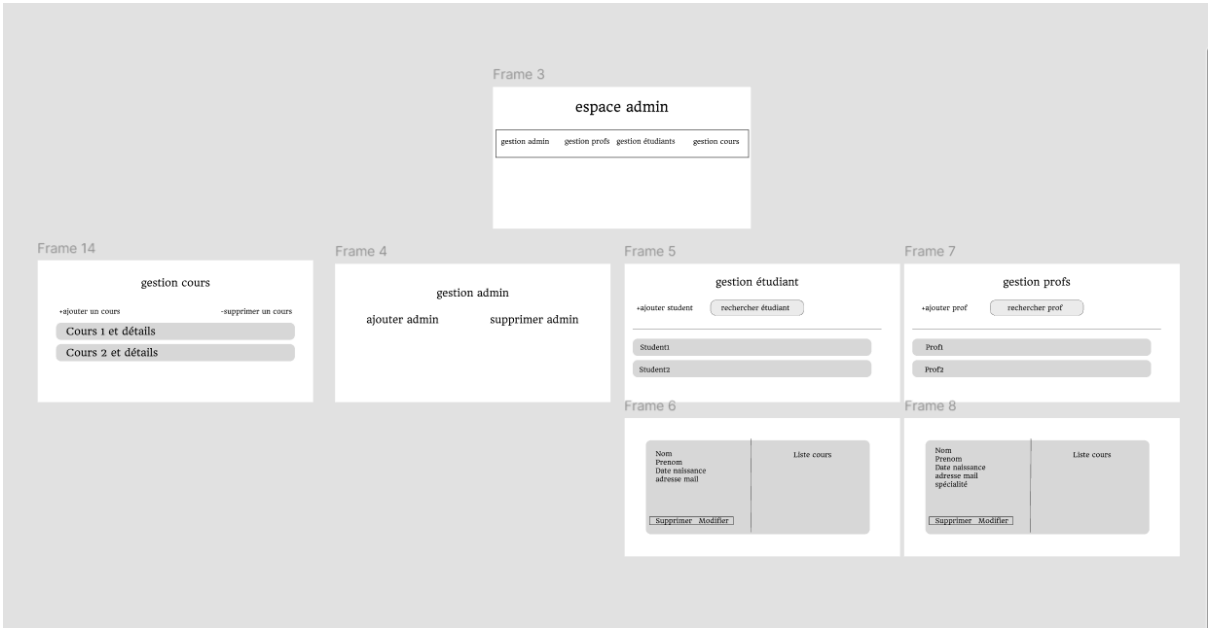
Chaque **Inscription** peut être liée à plusieurs **Résultats** (1:N).

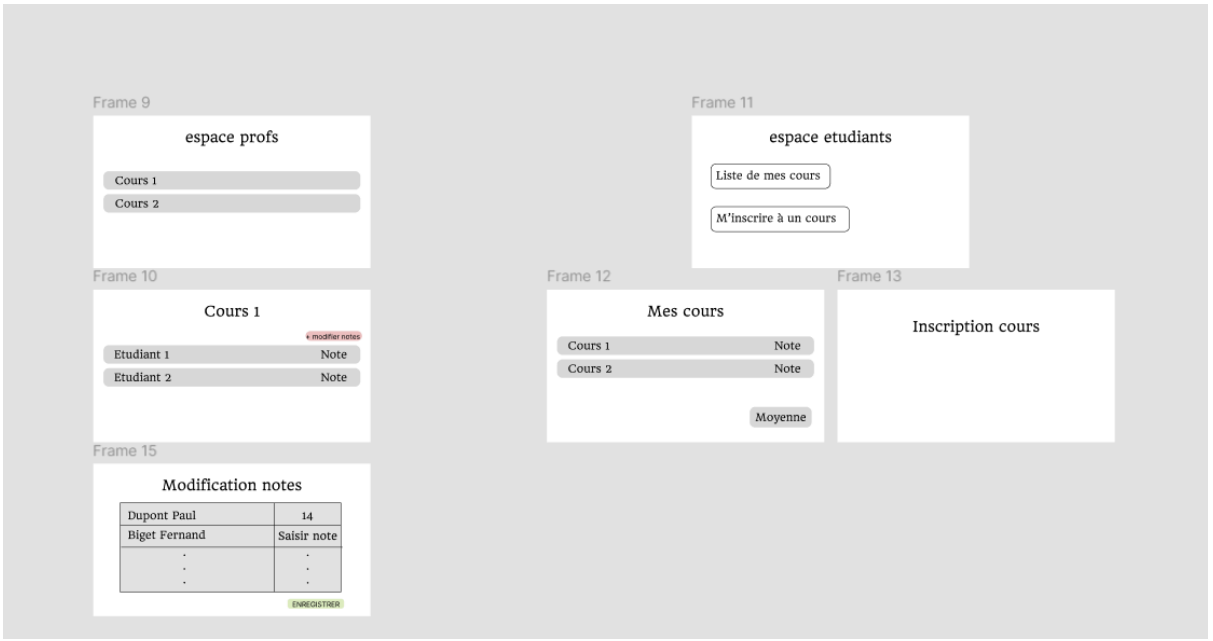
Ce choix a été fait pour avoir une optimalité maximale des entités donc relier les **cours**, **professeurs** et **étudiants** au niveau d'**inscription**.



2. Maquettes Figma

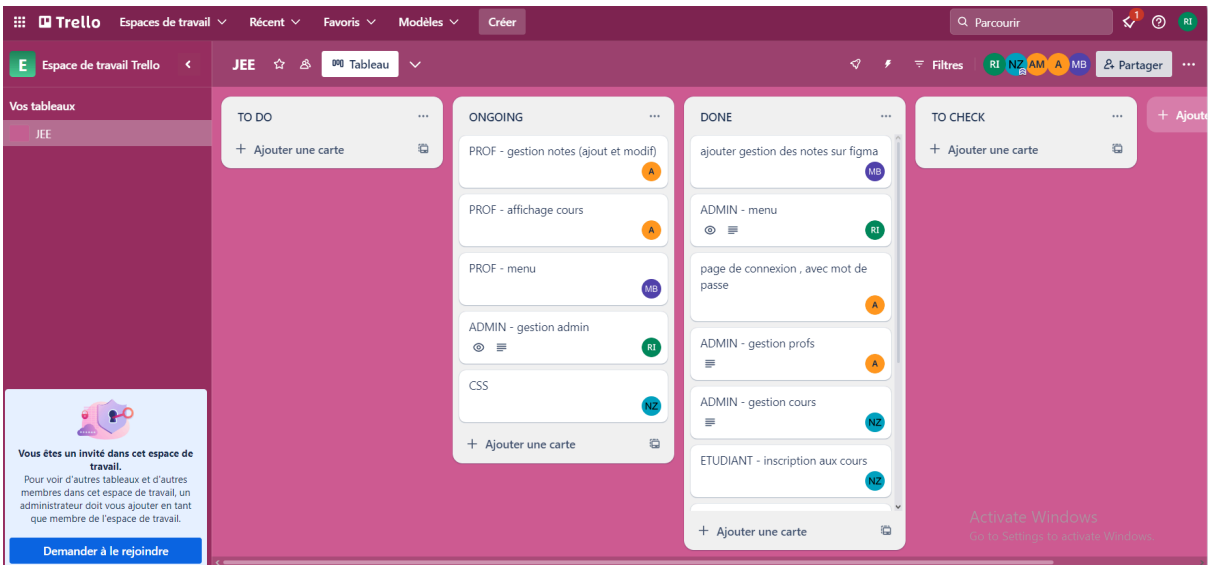
Les images ci-dessus présentent les différentes maquettes des interfaces utilisateurs (UI) pour l'application de gestion universitaire. Ces maquettes montrent l'architecture et les interactions attendues entre les différents utilisateurs du système : Administrateurs, Professeurs, et Étudiants.





3. Organisation et répartition des tâches

Trello a été utilisé dans le cadre de notre projet pour faciliter la gestion des tâches et assurer une collaboration fluide entre les membres de l'équipe. Grâce à sa structure visuelle de tableaux, de listes et de cartes, nous avons pu suivre l'avancement des différentes étapes du projet, attribuer des responsabilités et garantir que toutes les tâches étaient bien couvertes dans les délais impartis.



III. Implémentation

1. Architecture du projet

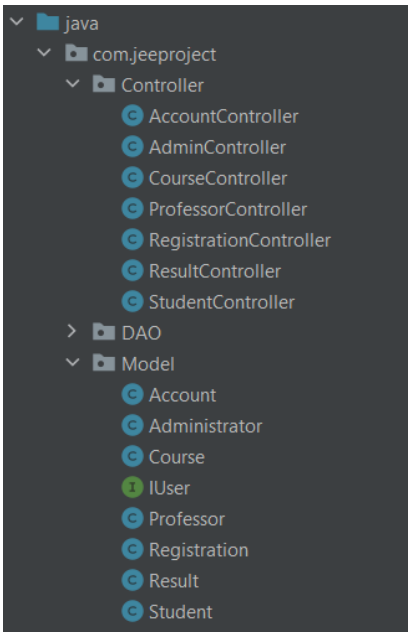
L'architecture de notre application respecte le modèle MVC (Modèle-Vue-Contrôleur):

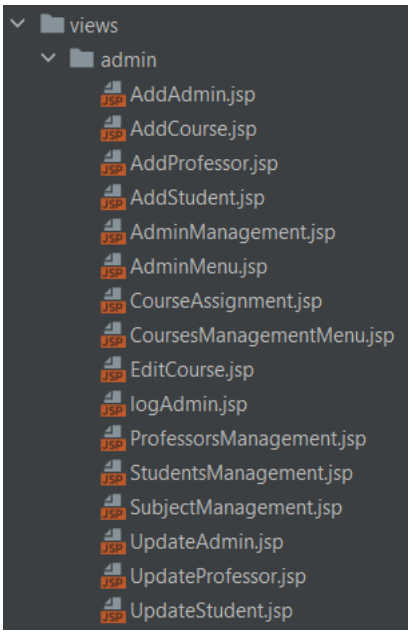
Structure générale:

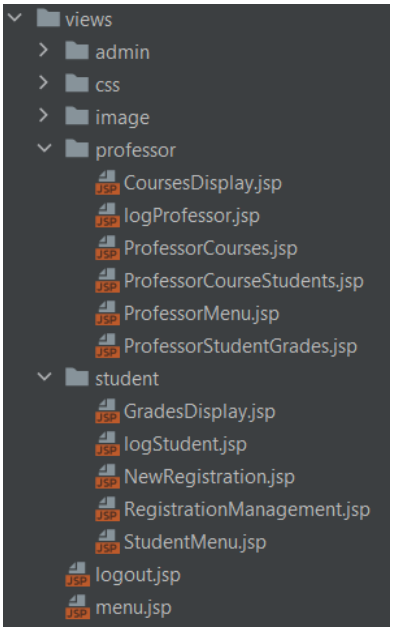
- ***java/*** : Contient tout le code métier, y compris les modèles, les DAO (Data Access Objects), les contrôleurs et les utilitaires.
- ***resources/*** : Inclut les configurations essentielles, comme celles de Hibernate et Log4j, ainsi que les fichiers de mapping nécessaires.
- ***webapp/*** : Renferme les éléments liés à la présentation tels que les fichiers JSP.

Architecture MVC:

- ***java/com.jeeproject/Model***: Ce dossier contient les classes Java représentant les entités, telles que Student ou Registration par exemple.
- ***java/com.jeeproject/Controller***: Ce répertoire contient les controllers responsables de gérer les requêtes HTTP, d'appeler les services via les DAO et de transmettre les données aux vues.
- ***webapp/views***: Les pages JSP pour afficher les données sont ici. Les JSP sont répartis dans les dossiers */admin*, */professor* et */student* afin de mieux gérer les interfaces des différents utilisateurs.

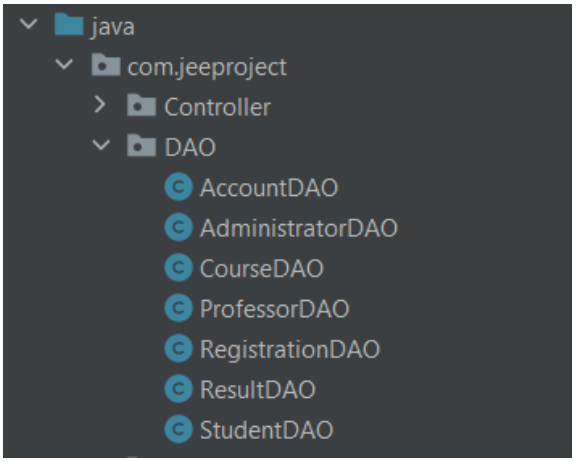






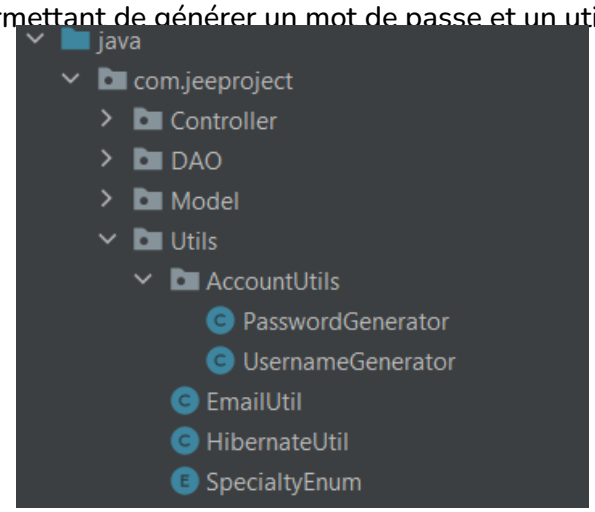
Couche d'Accès aux Données (DAO):

- ***java/com.jeeproject/DAO***: Les DAO gèrent les interactions avec la base de données. Ils gèrent notamment les opérations CRUD pour chaque entité.



Utilitaires (Utils):

- ***java/com.jeeproject/Utils***: Le répertoire Utils contient HibernateUtil qui centralise la gestion des sessions Hibernate. On trouve aussi EmailUtil, qui fournit les fonctionnalités liées à l'envoi de mails. Le sous-répertoire */AccountUtils* regroupe les classes permettant de générer un mot de passe et un utilisateur.



Configuration et Ressources:

- ***resources/hibernate.cfg.xml***: Ce fichier définit la connexion à la base de données. Il configure également les classes mappées et les paramètres de transaction.
- ***resources/mapping****:* Ce dossier contient les fichiers .hbm.xml pour le mapping.

IU. Springboot

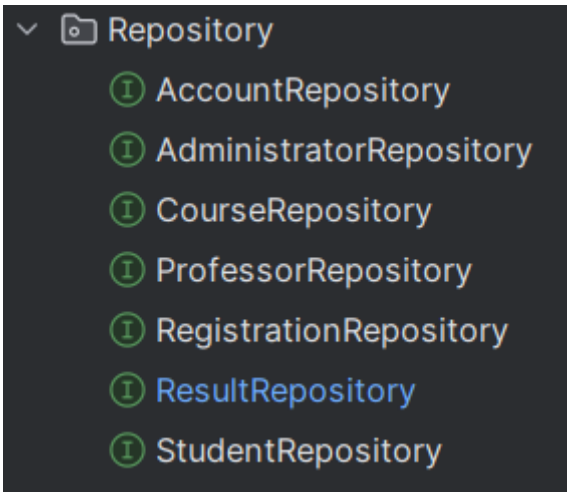
Les étapes de migration du projet JEE en springboot:

1.Création du projet Spring Boot

- Initialiser un nouveau projet Spring Boot avec Maven.
- Ajouter les dépendances nécessaires dans pom.xml (Spring Web, Spring Data JPA, MYSQL Driver).

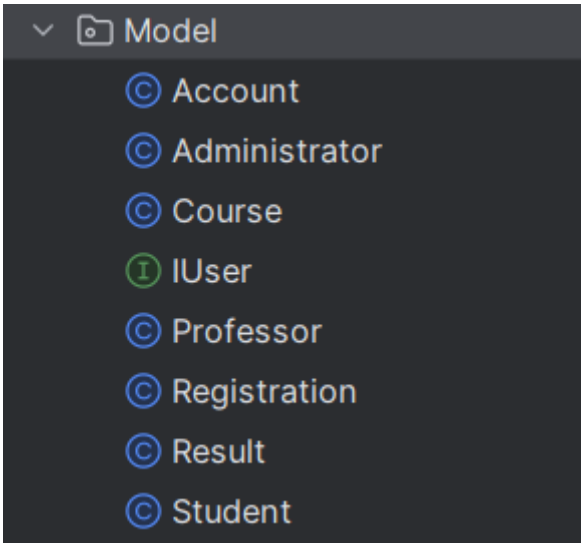
2. Conversion des DAO en Repositories Spring Data JPA

- Remplacer les DAO classiques par des repositories Spring Data JPA en étendant JpaRepository.
- Supprimer la gestion manuelle des transactions, Spring Boot gérant cela automatiquement avec @Transactional.



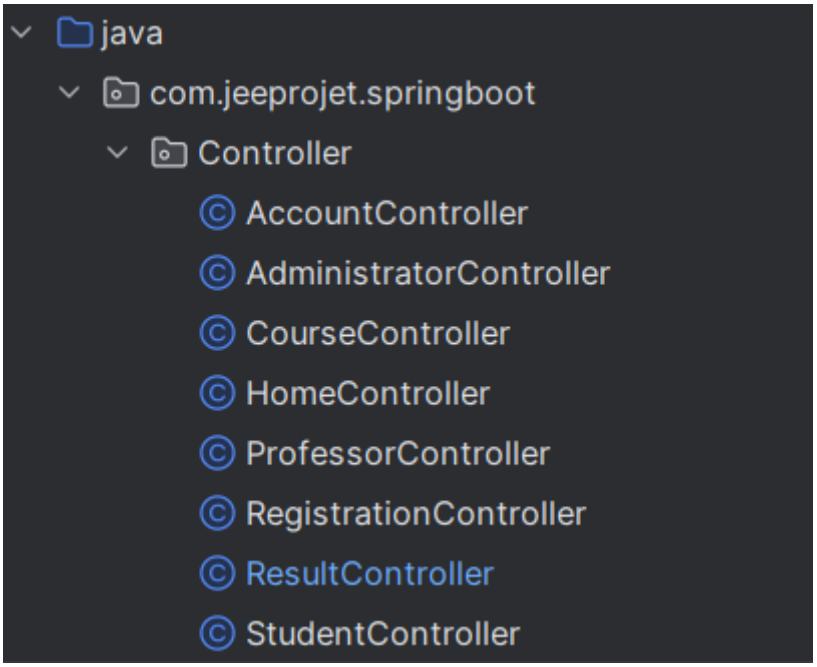
3. Refactorisation des classes de modèle

- Adapter les classes de modèle pour travailler avec JPA en ajoutant les annotations @Entity, @Table, @Id, etc.
- Définir les relations entre les entités (ex. @OneToMany, @ManyToOne).



4. Migration des servlets vers des contrôleurs Spring MVC

- Convertir les anciens servlets en contrôleurs Spring MVC avec les annotations @Controller.
- Mapper les requêtes HTTP avec @RequestMapping, @GetMapping, @PostMapping, etc.
- Utiliser @ModelAttribute et @RequestParam pour lier les paramètres aux modèles.

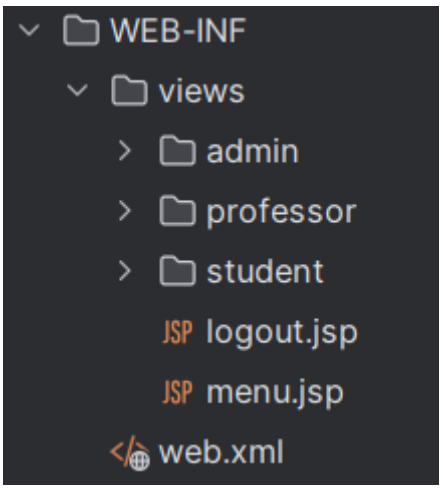


5. Configuration des propriétés Spring Boot

- Déplacer les configurations de l'application (connexion à la base de données, variables d'environnement) vers application.properties.
- Ajouter les dépendances pour la gestion des JSP.

6. Utilisation de JSP pour les vues

- Conserver nos fichiers JSP dans le dossier /WEB-INF/views/.(comme mentionné précédemment)


WEB-INF
views
admin
professor
student
logout.jsp
menu.jsp
web.xml

V. CONCLUSION

Ce projet a permis de mettre en pratique des concepts fondamentaux du développement web, notamment la gestion des bases de données, la création d'interfaces utilisateur et l'implémentation de la logique d'application avec Java. La réécriture avec Spring Boot a permis d'adopter une architecture plus modulaire et évolutive, simplifiant ainsi l'intégration de nouvelles fonctionnalités à l'avenir. Ce travail nous a également offert une meilleure compréhension des meilleures pratiques en matière de gestion des erreurs, de validation des entrées et de structuration d'applications web. Les connaissances acquises au cours de ce projet seront précieuses pour les futurs défis en développement web, en particulier avec des technologies modernes comme Spring Boot.