RAPPORT PROJET JEE

CREATION APPLICATION WEB
DE GESTION DE SCOLARITE

Enseignant: HADDACHE MOHAMED

Présenté par :

Nour Ben Miled
Herman Sessou
Alexis Loyau
Stevie Mobue me Efangon
Matthis Ribeiro

Année : 2024-2025

Table des matières

I. Introduction

Le projet consiste en la création d'une application web de gestion de scolarité, permettant aux étudiants, enseignants et administrateurs de gérer les informations académiques. Cette application englobe plusieurs fonctionnalités telles que la gestion des étudiants, des enseignants, des cours, des inscriptions et des résultats. L'objectif principal est d'appliquer les concepts et technologies étudiés en cours de J2EE tout en respectant une architecture MVC (Modèle, Vue, Contrôleur) pour une meilleure séparation des préoccupations.

II. Organisation

Ayant choisi de travailler en groupe sur ce projet, nous avons décidé d'utiliser uniquement les deux outils d'organisation suivants :

- Trello : Cette application de gestion de projet en ligne utilise un tableau virtuel pour organiser les tâches et les projets. Elle nous a permis de suivre l'avancement du projet, avec des listes représentant les différentes phases et des cartes pour illustrer les tâches restantes.
- GitHub : La plateforme GitHub est un service d'hébergement de projets, ce qui nous a permis de réaliser des sauvegardes sécurisées du projet en cas de dysfonctionnement de la machine utilisée durant son développement.
- StarUML : Avec StarUML, nous avons pu démarrer la conception du projet en créant un diagramme de classes, qui a évolué au fil du temps.
- Ainsi, ces outils ont facilité notre collaboration et la gestion du projet tout au long de son développement.

III. Analyse des besoins

Les besoins fonctionnels et non fonctionnels du projet sont répartis en différentes catégories selon les rôles des utilisateurs :

- Étudiants :

Enregistrement et mise à jour des informations personnelles.
Consultation des cours inscrits et des résultats.
Inscription et désinscription aux cours.

- Enseignants :

Gestion des cours, affectation d'enseignants et saisie des notes.
Consultation des résultats des étudiants.

- Administrateurs :

Gestion complète des utilisateurs (étudiants, enseignants).

Gestion des cours, inscriptions et résultats.
Les exigences non fonctionnelles incluent la sécurité des données via une gestion des rôles et des permissions.

IV. Diagrammes

1-Modèle Conceptuel des Données (MCD)

Le MCD de l'application comprend plusieurs entités principales :

- Étudiant : Contient des informations comme le nom, prénom, date naissance,et un identifiant unique.
- Enseignant : Gère les informations des enseignants et leur affectation aux cours.
- Cours : Représente les cours, avec leurs matières et affectation aux enseignants.
- Inscription : Enregistre les étudiants inscrits à chaque cours.
- Résultat : Permet de saisir et d'afficher les notes des étudiants.

2.Diagramme de Classes

Le diagramme de classes représente les entités et leurs relations. Par exemple :
Un étudiant peut être inscrit à plusieurs cours, et un cours peut accueillir plusieurs étudiants.Chaque enseignant peut dispenser plusieurs cours, et chaque cours peut avoir plusieurs enseignants.

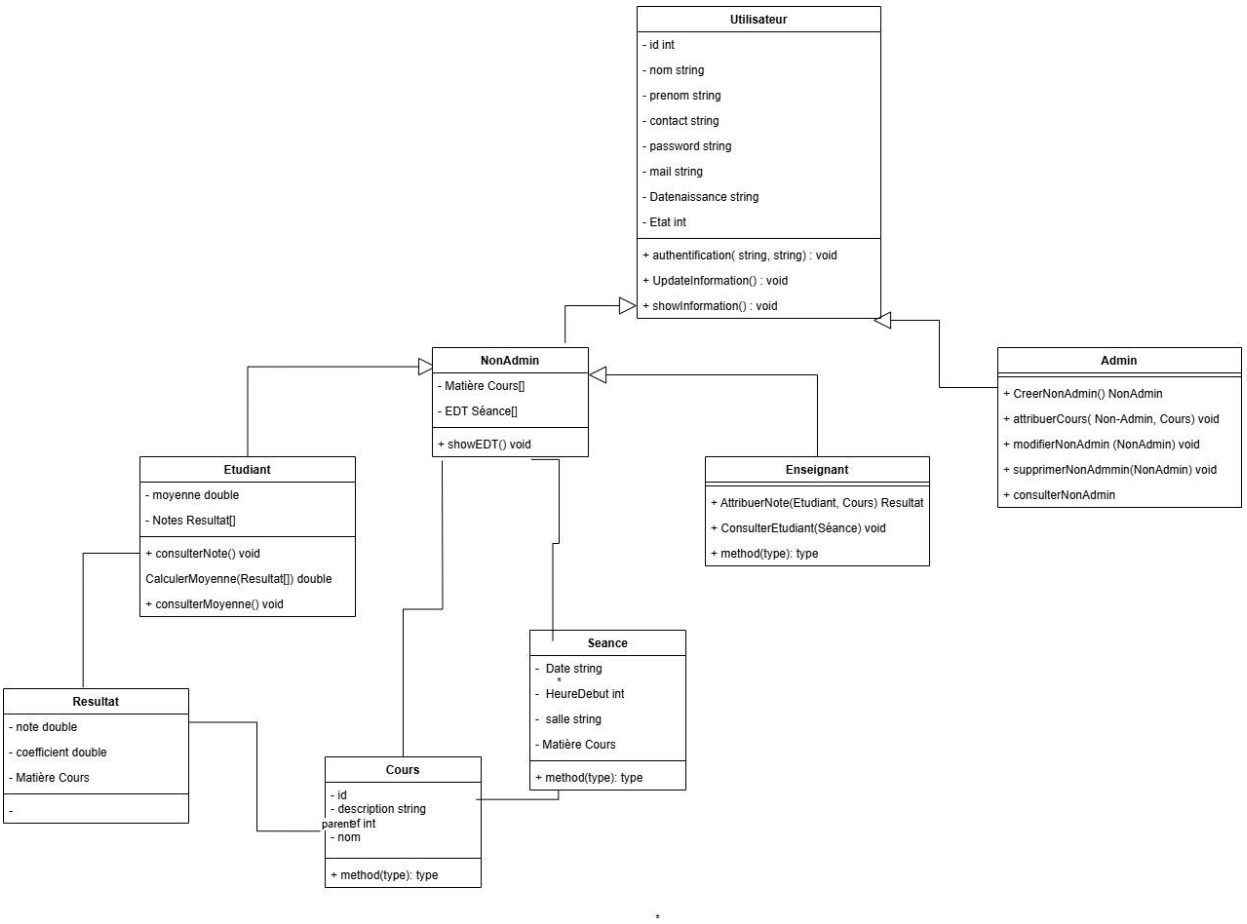

Figure 1 - Diagramme de classes

3.Diagramme de sequence

Le diagramme de séquence est un outil de modélisation utilisé en UML (Unified Modeling Language) pour décrire l'interaction dynamique entre les objets d'un système au fil du temps. Il met en évidence l'ordre des messages échangés entre les entités et permet de visualiser le déroulement chronologique d'un processus. Ce type de diagramme est particulièrement utile pour clarifier les scénarios d'interaction complexes et garantir une bonne compréhension des flux d'informations dans un système. Par exemple, dans le cadre de notre application de gestion de scolarité, nous avons réalisé un diagramme de séquence qui illustre le processus d'inscription des étudiants. Ce diagramme montrerait l'interaction entre l'utilisateur (l'étudiant), le serveur (qui gère la logique métier) et la base de données (où sont stockées les informations). En suivant cette même logique, ce diagramme pourrait facilement être adapté pour d'autres processus, tels que la gestion des

résultats ou la saisie des notes, en mettant en évidence les mêmes échanges d'informations entre les différents composants du système.

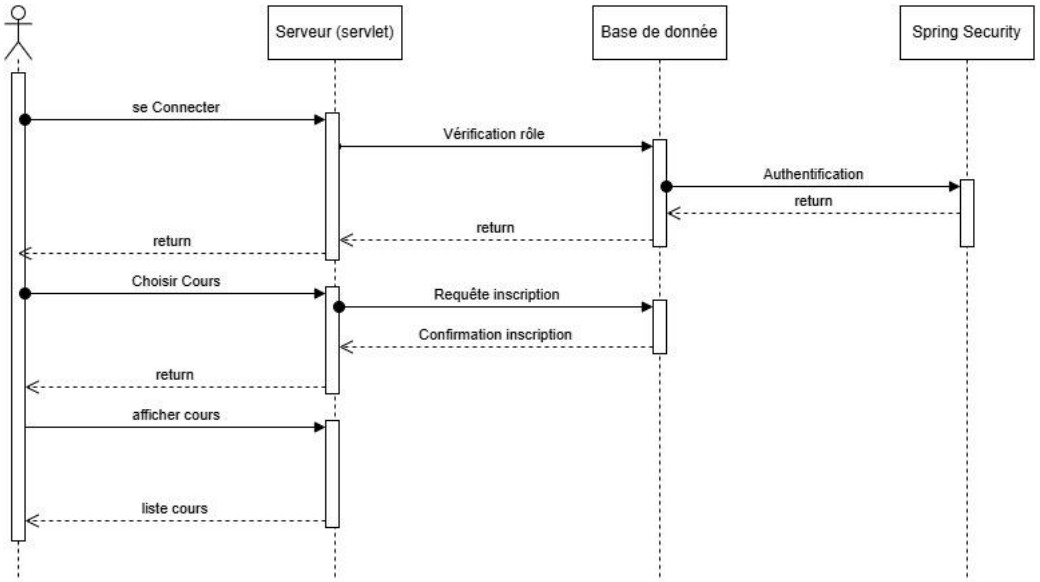

Figure 2 - Diagramme de sequences

V. Choix des technologies

Plusieurs technologies ont été employées afin de mener à terme le projet web. Pour le modèle, les données ont été stockées dans une base de données MySQL, et grâce au pilote mysql-connector ainsi qu'au framework Hibernate, nous avons pu assurer la persistance des données et la gestion des sessions.

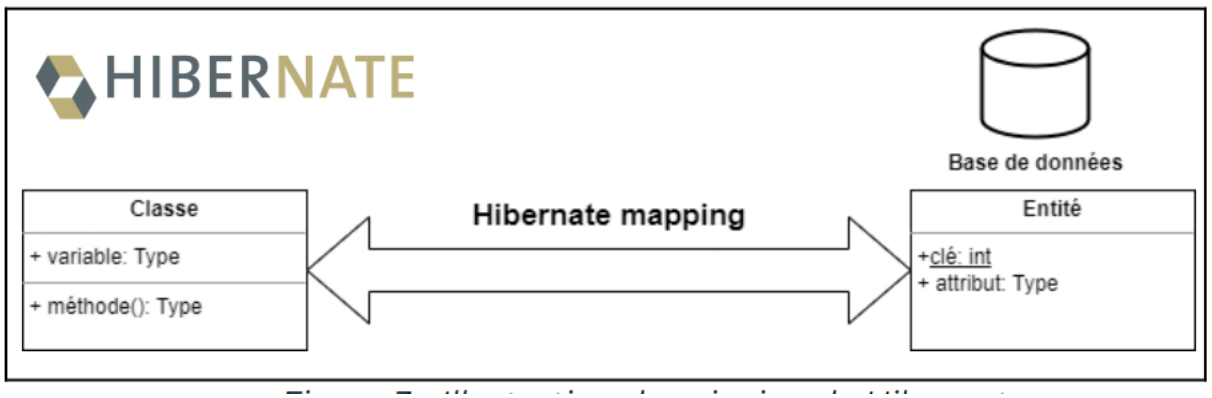

Figure 3 - Illustration du principe de Hibernate

Concernant la vue, plusieurs composants ont été utilisés. Tout d'abord, les pages JSP ont servi à livrer le contenu dynamique à afficher aux utilisateurs. Ensuite, pour apporter davantage de dynamisme et faciliter la décoration du côté client,

nous avons implémenté Bootstrap, un framework qui offre un ensemble de styles CSS prédéfinis permettant de concevoir des interfaces graphiques plus rapidement. Nous avons également ajouté JQuery pour simplifier la gestion des objets HTML et réaliser des requêtes AJAX, afin d'éviter de recharger l'intégralité de la page lorsqu'une modification est effectuée sur une section spécifique.

Figure 4 - Technologies de vue et contrôleur

Pour le déploiement du projet, le serveur d'Apache Tomcat a été employé. Ce serveur sert de conteneur des pages JSP ainsi que des servlets.

VI. SpringBoot

1.Migration vers Spring Boot

Spring Boot est un vaste projet du framework Spring qui vise à simplier le processus de développement des applications Java. Il offre des fonctionnalités et des conventions par défaut pour accélérer le développement d'applications basées sur Spring. L'objectif principal de SPRING Boot est de faciliter la création d'applications autonomes, prêtes à l'emploi, avec une conguration minimale. En offrant un début de projet modulable tout en minimisant le nombre de chiers de conguration.xml.Spring Boot simplifie le processus de développement en éliminant la configuration complexe et en intégrant directement des composants comme Spring Data JPA pour la gestion de la base de données.La structure du projet Spring Boot repose sur une organisation modulaire avec des Controllers pour gérer les requêtes HTTP, des Services pour la logique métier, et des Repositories pour l'accès aux données.

Figure 5 - Logo de SpringBoot

2.Structure du projet

Le projet est divisé en plusieurs packages, incluant :

- Controller : Contient les servlets réécrites en utilisant les annotations Spring.
- Service : Gère la logique métier.
- Domain : Contient les entités comme Étudiant, Enseignant, Cours, etc.
- Repository : Interfaces permettant l'interaction avec la base de données via des méthodes CRUD.

3.Sécurisation et Authentification

Une gestion des rôles a été soigneusement mise en place dans le système afin d'assurer une sécurité optimale en régulant l'accès aux différentes fonctionnalités selon le type d'utilisateur. Ce système repose sur trois principaux rôles : étudiant, enseignant et administrateur, chacun ayant des niveaux d'accès distincts et spécifiques à leurs besoins.

- Étudiant : Ce rôle est limité aux fonctionnalités qui lui permettent de consulter les cours, de soumettre des devoirs et de suivre sa progression. Les étudiants n'ont pas accès aux fonctionnalités d'administration ou à la gestion du contenu.
- Enseignant : Les enseignants ont un rôle étendu, leur permettant non seulement de consulter les cours, mais aussi de créer du contenu, d'évaluer les travaux des étudiants et de suivre les performances des élèves. Toutefois, l'accès à certaines sections administratives reste restreint.
- Administrateur : L'administrateur dispose d'un accès complet à toutes les fonctionnalités du système, y compris la gestion des utilisateurs, des cours, des rôles et des permissions. Il est responsable de la configuration générale et de la

maintenance du système.

Pour garantir une sécurité robuste et efficace, Spring Security a été utilisé pour gérer l'authentification des utilisateurs ainsi que les permissions d'accès. Grâce à Spring Security, nous avons mis en place une authentification forte qui vérifie l'identité de l'utilisateur à chaque tentative de connexion, en utilisant des mécanismes tels que l'authentification par formulaire ou l'authentification basée sur des tokens (JWT, par exemple). Ce framework permet également de gérer les autorisations et les rôles de manière centralisée, en définissant des règles précises pour chaque type d'utilisateur.

Les permissions sont gérées de manière fine grâce à Spring Security, ce qui permet de définir des règles d'accès détaillées. Par exemple, seules les personnes avec le rôle d'enseignant ou d'administrateur peuvent modifier le contenu des cours, tandis que les étudiants ne peuvent que consulter les informations. Cette gestion des accès permet d'assurer une séparation claire des responsabilités et une meilleure sécurité du système en limitant les risques d'accès non autorisés à des fonctionnalités sensibles.

En résumé, cette approche combinée de gestion des rôles et d'utilisation de Spring Security garantit non seulement la sécurité des données et des utilisateurs, mais aussi une flexibilité dans l'attribution des droits d'accès en fonction des besoins spécifiques de chaque type d'utilisateur.

VII. Les Interfaces

Cette partie a pour objectif de présenter le produit final. C’est la phase de réalisation de cette application web. Dans cette partie on va présenter quelques interfaces graphiques dans le but de vous situer dans le projet.

1.Interface connexion

C’est elle qui permet l’accès proprement dit à l’application par les différents utilisateurs. Elle a bien sûr pour rôle de vérifier si l’utilisateur est effectivement présent dans la base de données et ensuite le redirige vers ses différents droits d’accès.

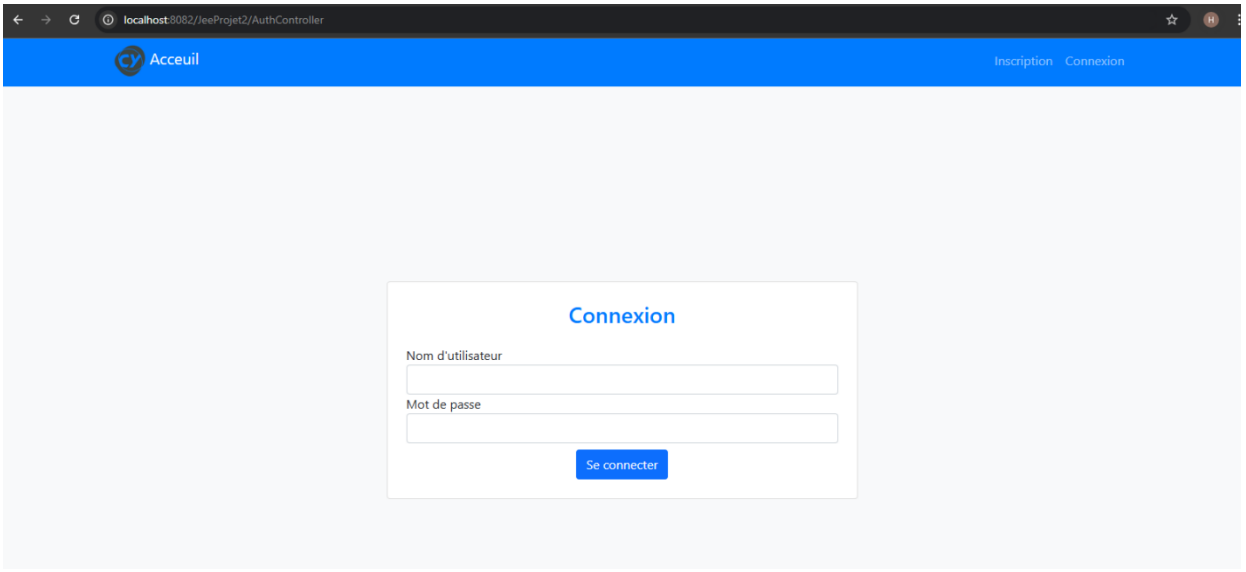

Figure 6 - Interface connexion

2. Interface administrateur

Après son authentification réussie, l'administrateur a un accès très spécial qui lui permet de gérer toutes les différentes rubriques de l'application et des fonctionnalités de gestion des etudiants et enseignant qui y sont présents.

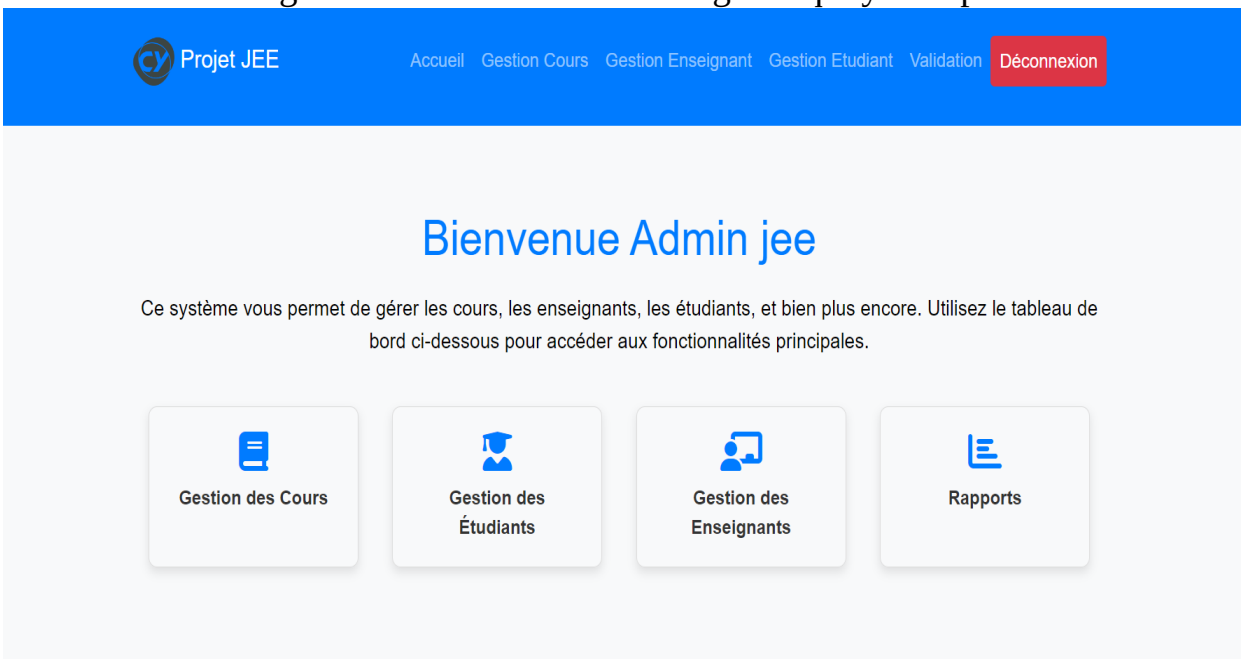

Figure 7 - Interface administrateur

3.Interface etudiant

L'interface étudiant est une partie essentielle d'une application de gestion de

scolarité, offrant aux étudiants un accès personnalisé et sécurisé à leurs informations académiques. Après une authentification réussie, l'étudiant peut consulter ses cours, ses notes, ses emplois du temps et suivre sa progression. Contrairement à l'interface administrateur, qui permet une gestion complète de toutes les rubriques et des fonctionnalités liées aux étudiants et aux enseignants, l'interface étudiant se concentre principalement sur les services et les informations pertinents pour lui.

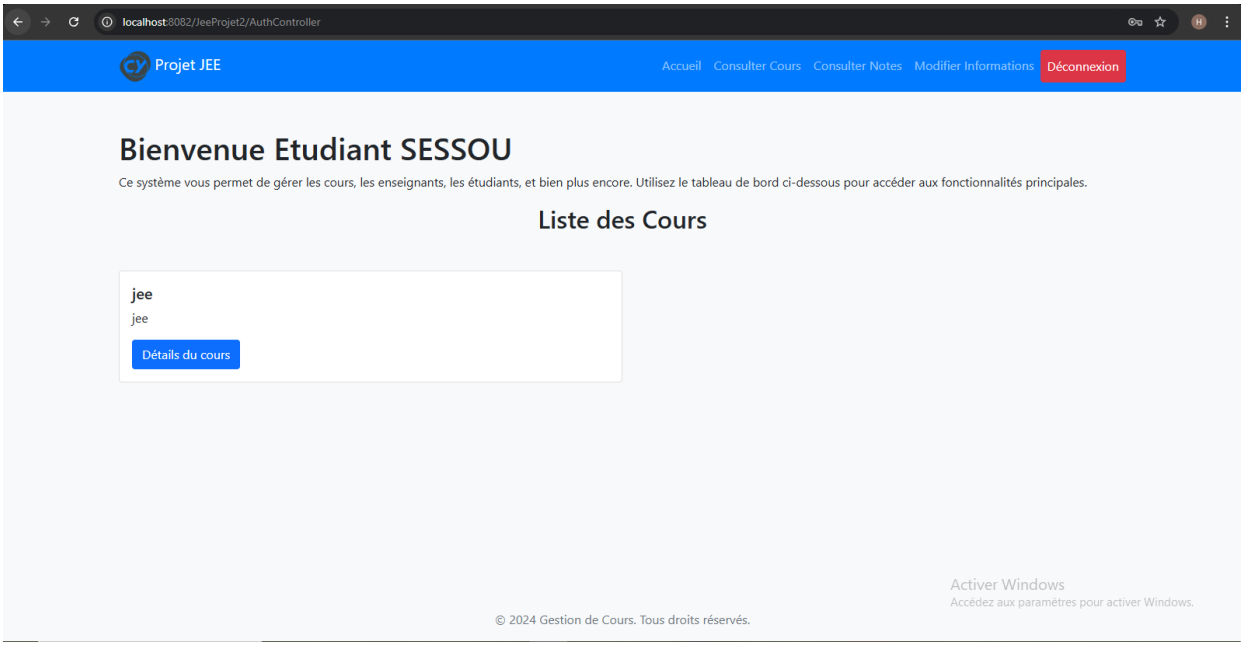

Figure 8 - Interface etudiant

4.Interface enseignant

L'interface enseignant joue un rôle crucial dans une application de gestion de scolarité, offrant aux enseignants un accès facilité et sécurisé à leurs outils pédagogiques. Après une authentification réussie, l'enseignant peut consulter ses cours, saisir et modifier les notes des étudiants, ainsi que suivre la progression de chacun d'eux. Contrairement à l'interface étudiant, qui se concentre sur les informations et services personnels de l'élève, l'interface enseignant permet également la gestion de l'organisation des cours, l'ajout de contenus pédagogiques, la planification des examens et la communication avec les étudiants. Cette interface facilite ainsi l'optimisation de la gestion des matières enseignées et améliore l'interaction entre les enseignants et les étudiants, tout en simplifiant les tâches administratives liées à l'enseignement.

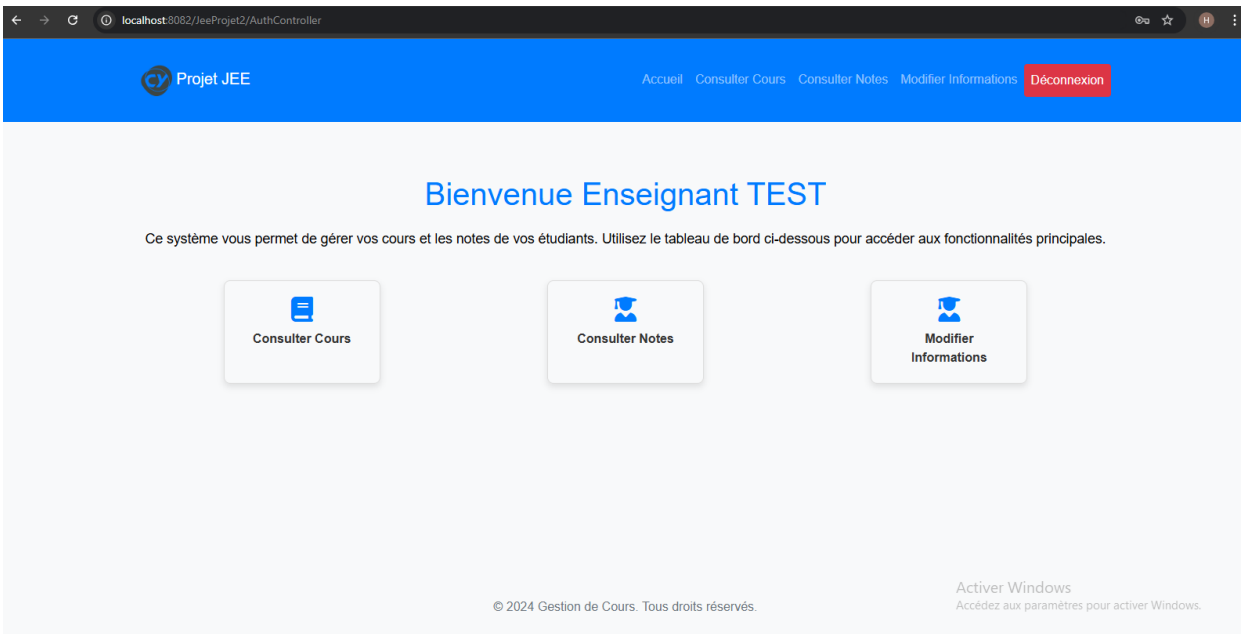

Figure 9 - Interface enseignant

VIII. Conclusion

Ce projet a permis de développer une application complète de gestion de scolarité en utilisant les technologies Java EE et Spring Boot. L'implémentation du modèle MVC a facilité l'organisation du projet et l'interaction entre les différentes couches de l'application. L'utilisation de Spring Boot a simplifié la gestion des dépendances et des configurations, tandis qu'Hibernate a facilité l'accès à la base de données.

Les objectifs ont été atteints, et l'application offre désormais toutes les fonctionnalités nécessaires à la gestion académique des étudiants et enseignants. Cette expérience a également permis de mettre en pratique des compétences en développement web, gestion de projet, et utilisation des technologies Spring et Hibernate, tout en respectant les bonnes pratiques de développement logiciel.

En résumé, cette expérience a été enrichissante pour nous, approfondissant nos compétences en développement JEE et nous offrant l'opportunité de découvrir de nouvelles technologies prometteuses. Elle a également mis en évidence l'importance de l'analyse approfondie des besoins, du choix judicieux des technologies et de l'adaptation continue aux évolutions du domaine du développement web.

Néanmoins, nous pensons que des modifications et améliorations peuvent être apportées à notre application afin de répondre convenablement à des besoins précis en fonction de la demande pour être adapter à un etablissement pour la gestion de scolarite.