

RAPPORT DE MINI-PROJET

Application Web de Gestion d'un Centre de Formation

Realise dans le cadre de la formation d'Ingenieur

Presente par: [Votre Nom]

Encadre par: [Nom de l'encadrant]

Annee Universitaire: 2025-2026

Table des Matieres

1. Introduction
2. Analyse et specification des besoins
3. Architecture de la solution
4. Conception UML
5. Implementation
6. Securite et gestion des roles
7. Reporting et fonctionnalites avancees
8. Resultats et captures
9. Conclusion et perspectives

1. Introduction

Le projet 'Application web de gestion d'un centre de formation' s'inscrit dans une demarche de modernisation des outils de gestion pedagogique au sein d'une ecole d'ingenieurs. L'objectif est de centraliser toutes les informations relatives au cycle de vie d'une formation, de l'inscription des etudiants jusqu'a la validation des acquis par le biais de rapports de notes.

Objectifs pedagogiques :

- Maitriser l'ecosysteme Spring Boot pour le developpement backend.
- Comprendre l'architecture en couches et les principes de l'injection de dependances.
- Mettre en oeuvre une securite robuste basee sur les roles.
- Allier le rendu cote serveur (SSR) via Thymeleaf pour l'administration et le rendu cote client (CSR) pour une experience utilisateur fluide.

Interet de l'architecture Spring Boot :

Spring Boot offre une configuration simplifiee (auto-configuration), une gestion efficace de la persistance avec Spring Data JPA, et une integration facilitee de modules de securite. Sa modularite permet d'exposer des API REST tout en servant des vues classiques, repondant ainsi aux besoins hybrides du projet.

2. Analyse et specification des besoins

L'application doit repondre aux besoins de trois types d'acteurs principaux :

1. Administrateur (ADMIN) :

- Gestion du catalogue de formations (cours, sessions).
- Gestion des comptes utilisateurs (etudiants, formateurs).
- Affectation des formateurs aux cours.
- Surveillance globale des activites du centre.

2. Formateur (FORMATEUR) :

- Consultation de son emploi du temps.
- Saisie des notes pour les etudiants inscrits a ses cours.
- Gestion des supports pedagogiques.

3. Etudiant (ETUDIANT) :

- Inscription aux sessions de cours disponibles.
- Consultation de son emploi du temps personnalise.
- Acces a ses notes et telechargement du releve officiel.

Contraintes techniques :

- Securite JWT pour les API REST.
- Base de donnees relationnelle MySQL.
- Interface responsive utilisant Bootstrap.
- Disponibilite d'un systeme de notification par email.

3. Architecture de la solution

L'application repose sur une architecture multi-couches standardisee, optimisee pour la maintenabilite et l'evolutivite.

Architecture logique en couches :

- Couche Web (Controller) : Traite les requetes HTTP, gere les entrees utilisateur et delegue les traitements a la couche service.
- Couche Service (Business Logic) : Contient la logique metier, les validations complexes et la gestion des transactions.
- Couche Persistance (Repository) : Interagit avec la base de donnees via Spring Data JPA.
- Base de donnees : Stockage persistant sous MySQL.

Role de Spring IoC et de l'injection de dependances :

Le conteneur IoC (Inversion of Control) de Spring gere le cycle de vie des composants (Beans). L'injection de dependances permet de decoupler les couches. Par exemple, un contrôleur ne cree pas ses propres instances de services, mais les reçoit via son constructeur, ce qui facilite les tests unitaires.

Gestion des profils (dev / prod) :

L'utilisation des profils Spring permet de configurer des environnements distincts. En developpement, une base de donnees locale (ou H2) peut être utilisee avec une journalisation detaillee, tandis qu'en production, une instance MySQL configuree avec des parametres de securite stricts est privilegiee.

4. Conception UML

La phase de conception a permis de modeliser les interactions et les structures de donnees.

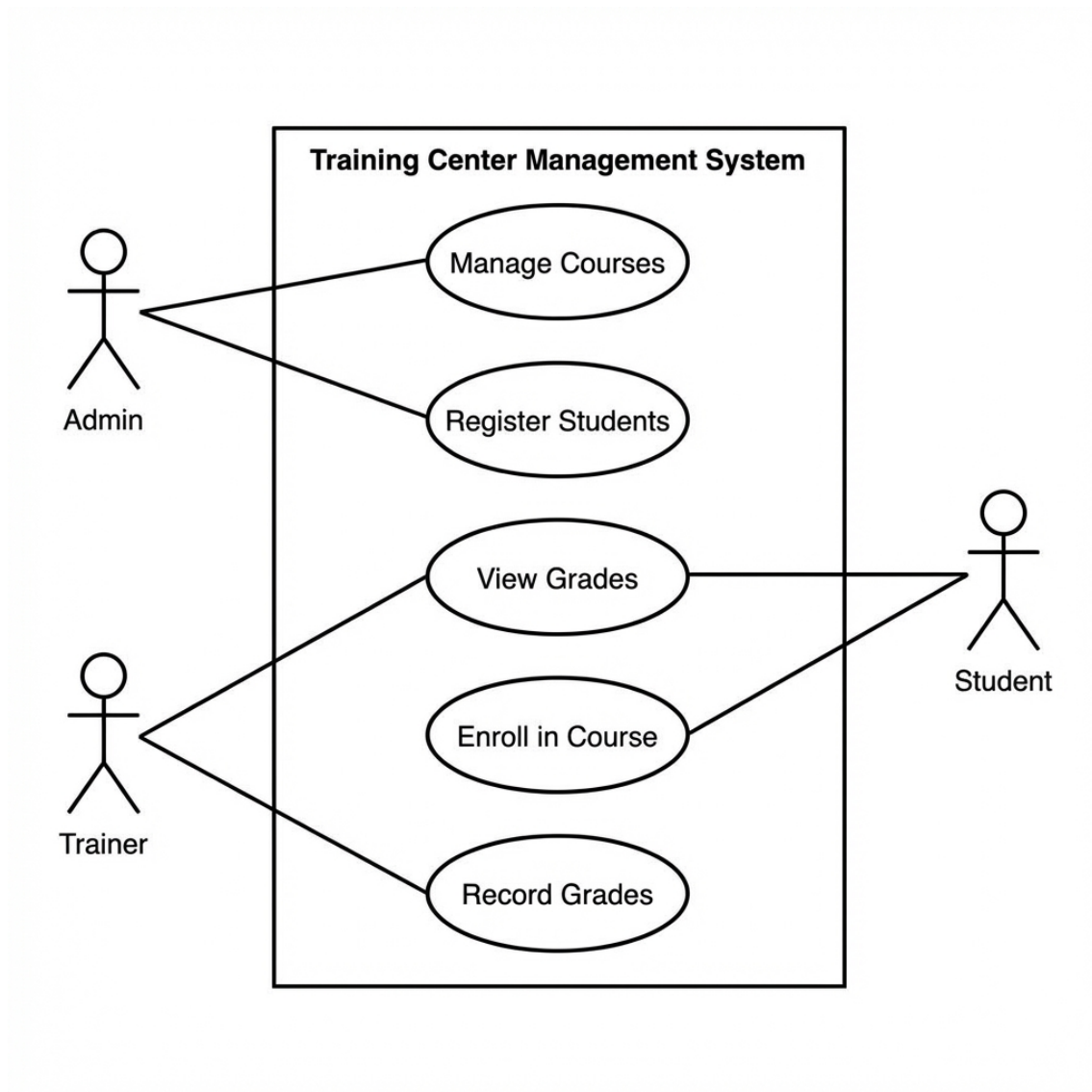


Figure: Diagramme de cas d'utilisation

UML CLASS DIAGRAM: TRAINING CENTER SYSTEM

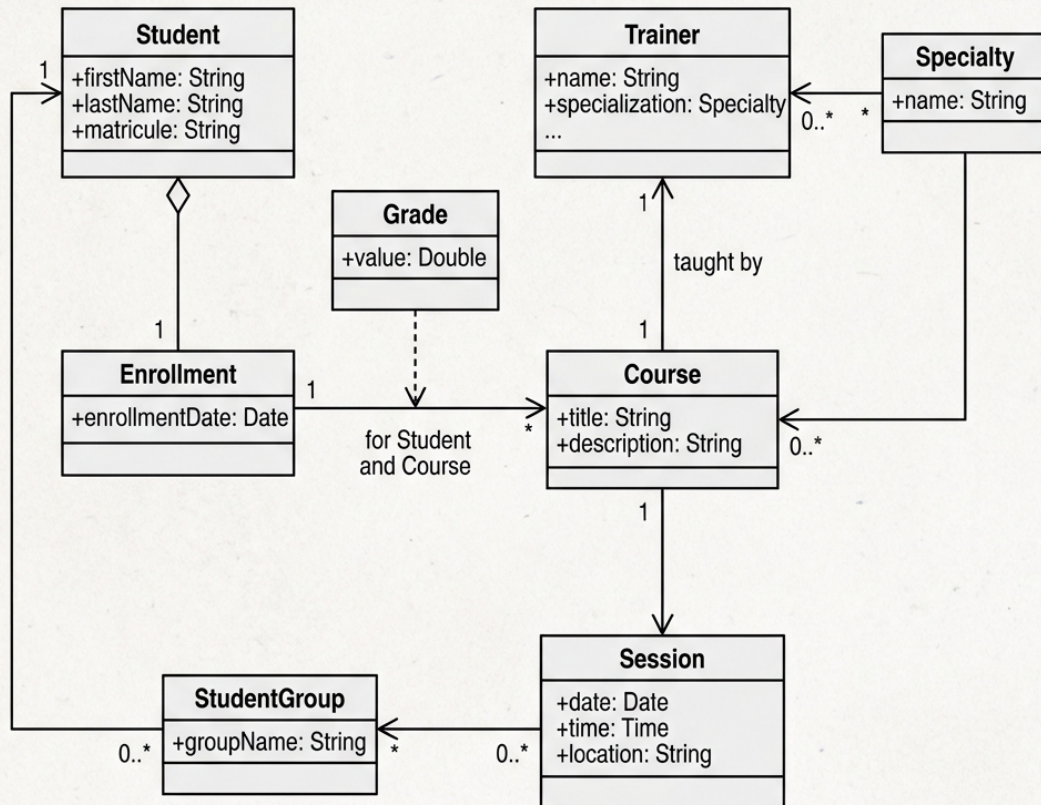


Figure: Diagramme de classes

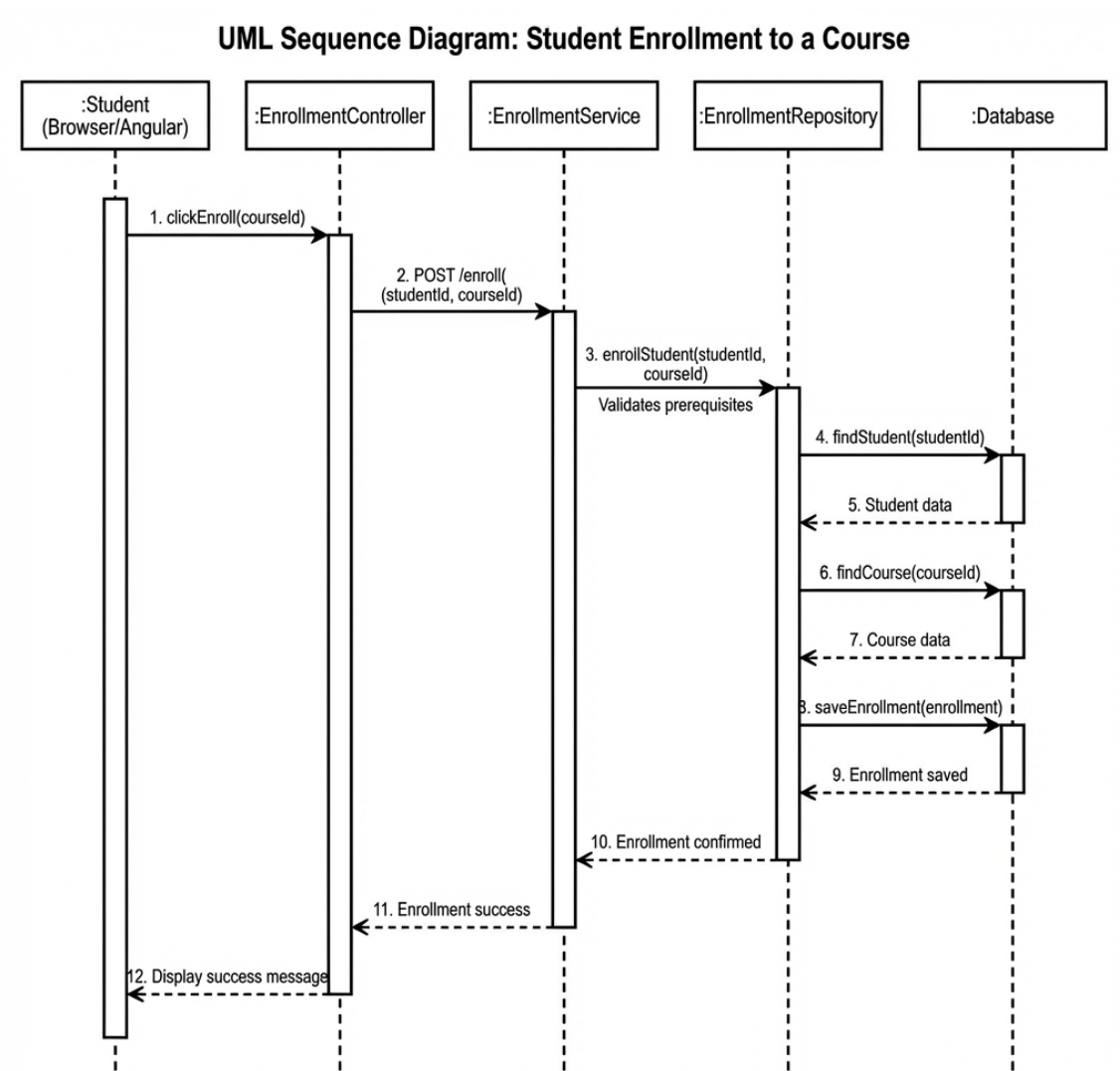


Figure 1: Sequence of operations for a student enrolling in a course within a tiered architecture.

Figure: Diagramme de sequence - Inscription

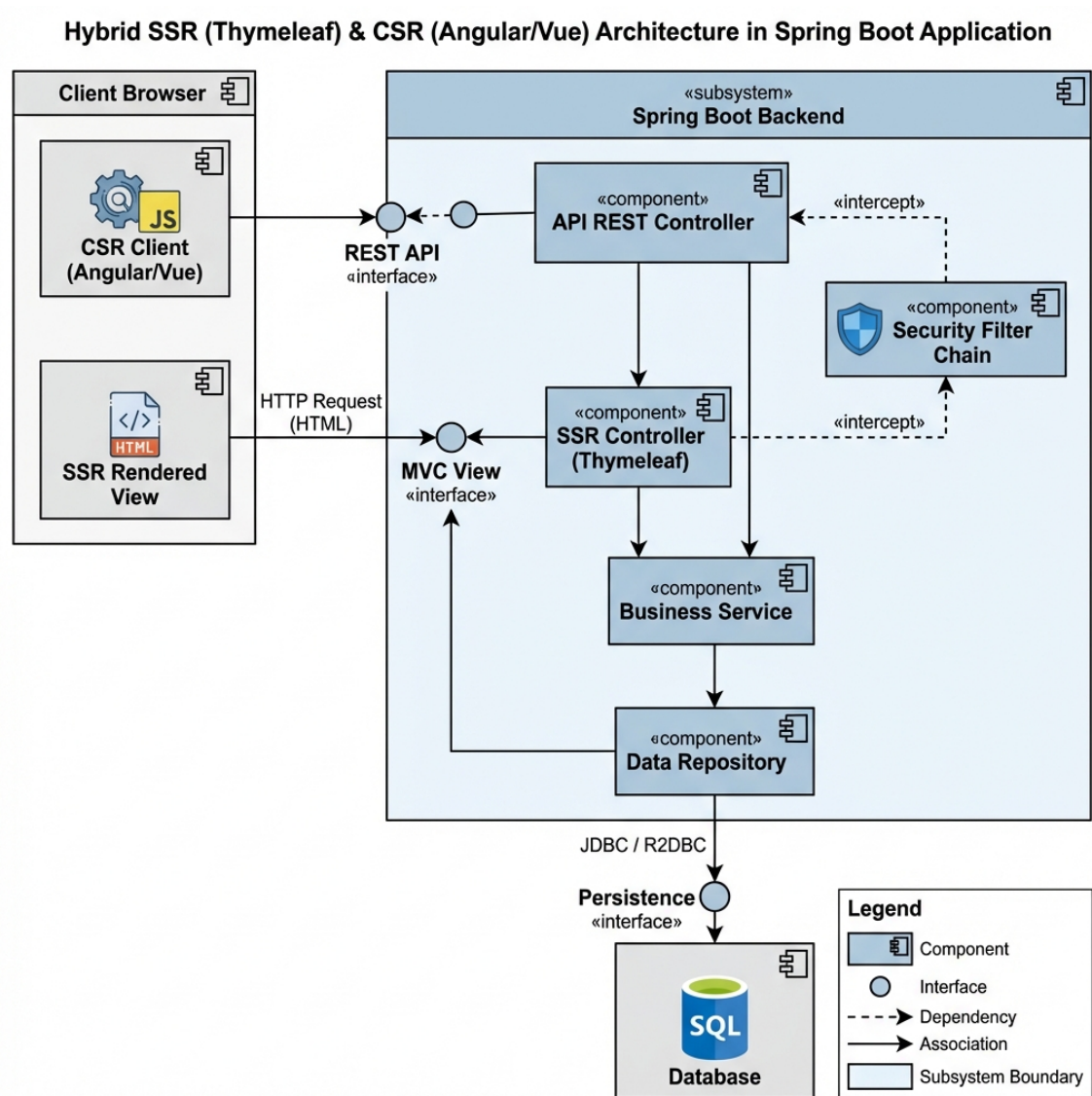


Figure: Diagramme de composants

5. Implementation

Technologies utilisees :

- Backend : Spring Boot 3.4, Spring Data JPA, Hibernate, Spring Security.
- Frontend : Thymeleaf (SSR), Bootstrap 5, Angular/React (CSR).
- Base de donnees : MySQL 8.0.
- Gestion de versions de BD : Liquibase.

Description des entites et relations ORM :

Le modele de donnees est centre sur l'entite User, declinee en Student et Trainer via un heritage ou des jointures. La relation entre Student et Course est geree par l'entite Enrollment, permettant de stocker la date d'inscription. Les notes (Grade) sont liees a la fois a un etudiant et a un cours.

Exposition des API REST :

Les API REST sont exposees via des RestControllers, retournant des DTO (Data Transfer Objects) pour eviter les cycles dans le JSON et securiser les transferts de donnees sensibles.

6. Securite et gestion des roles

La securite est un pilier de l'application. Elle est implementee avec Spring Security 6.

Authentification :

Pour les parties SSR, l'authentification se fait via un formulaire classique avec gestion de session. Pour les parties CSR, un système JWT (JSON Web Token) est mis en place, permettant une authentification stateless adaptee aux clients modernes.

Autorisation par roles :

- ROLE_ADMIN : Acces total aux interfaces de gestion.
- ROLE_FORMATEUR : Acces restreint a ses cours et a la saisie des notes.
- ROLE_ETUDIANT : Acces a ses inscriptions et a ses resultats uniquement.

Gestion du profil utilisateur :

Chaque utilisateur peut mettre a jour ses informations personnelles et changer son mot de passe, lequel est stocke sous forme de hachage BCrypt en base de donnees.

7. Reporting et fonctionnalites avancees

Le systeme integre des fonctionnalites de calcul et de reporting pour faciliter le suivi pedagogique.

Calcul des moyennes :

Un service dedie parcourt les notes d'un etudiant pour calculer sa moyenne generale ponderee ou simple. Des statistiques par cours sont egalement generees (taux de reussite, ecart-type).

Generation de rapports PDF :

Grâce a iText, l'application genere dynamiquement des releves de notes au format PDF. Ces documents sont formates de maniere professionnelle, incluant le logo du centre et un recapitulatif clair des resultats.

Notifications par email :

L'integration de Spring Mail permet d'envoyer des notifications automatiques (confirmation d'inscription, alerte en cas de nouvelle note publiee, rappel de session).

8. Resultats et captures

Les interfaces ont ete concues pour être intuitives. La partie administration (Thymeleaf) privilegie l'efficacite operationnelle, tandis que le tableau de bord etudiant mise sur la clarte visuelle.

Voici un aperçu du tableau de bord principal :

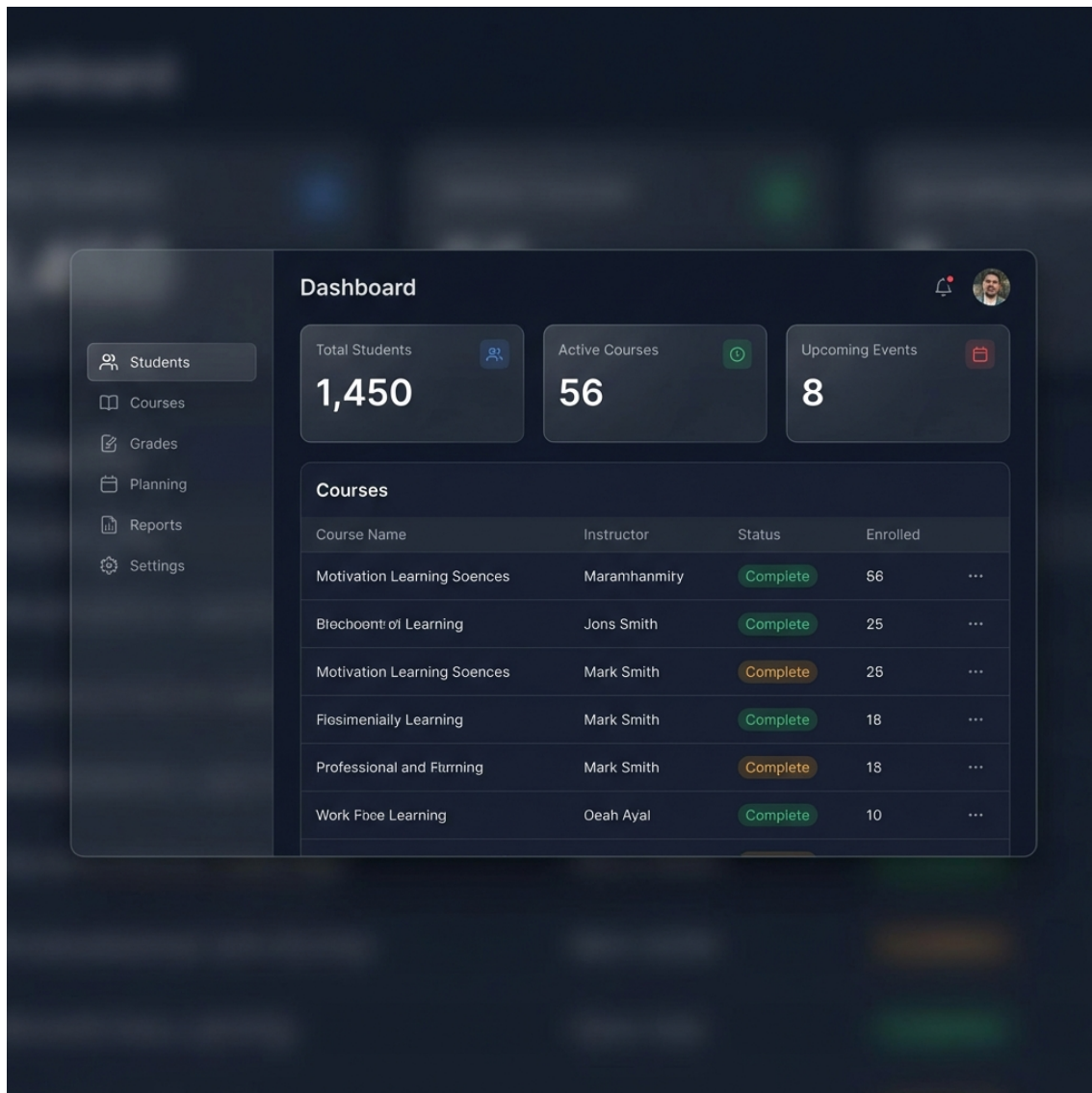


Figure: Tableau de bord de gestion (SSR/Thymeleaf)

9. Conclusion et perspectives

Ce projet a permis de concretiser les concepts theoriques du developpement web with Spring Boot. L'application est fonctionnelle et repond aux exigences initiales de gestion d'un centre de formation.

Bilan du travail realise :

- Mise en place d'une architecture solide et securisee.
- Developpement d'une interface hybride SSR/CSR.
- Automatisation du reporting par PDF.

Limites du projet :

- Absence d'un module de paiement pour les inscriptions.
- Manque de gestion multi-filiales.

Ameliorations futures :

- Ajout d'une application mobile hybride utilisant les API REST.
- Integration d'un module de visioconference pour les cours a distance.
- Mise en place d'analyse predictive sur les performances des etudiants (IA).