Projet Semestriel

Système de Gestion Académique

Application Web (React + API REST + JWT) pour Administrateur, Enseignant et Étudiant

---

Réalisé par : Maatouk Nour

Ben Ammeur Mohamed Mehdi

Année universitaire 2025–2026

Table des matières

Table des figures

Résumé

Ce projet consiste à concevoir et développer une application web de **gestion académique** permettant :

— à l'**administrateur** de gérer les étudiants, enseignants, cours, départements, inscriptions, affectations et notes ;

— à l'**enseignant** de consulter ses cours, la liste des étudiants inscrits et de gérer les notes ;

— à l'**étudiant** de consulter ses cours et ses notes.

Le système repose sur une interface **React (TypeScript)** et une **API REST** sécurisée via **JWT**. L'objectif est d'assurer une gestion fiable, rapide et centralisée des processus académiques.

Chapitre 1

Introduction générale

1.1 Contexte

Les établissements d'enseignement doivent gérer un grand volume d'informations : étudiants, enseignants, cours, départements, inscriptions, affectations des enseignants et notes. Une solution informatique centralisée permet de réduire les erreurs, d'améliorer la traçabilité et d'accélérer l'accès à l'information.

1.2 Problématique

Comment mettre en place un système web qui :

— centralise les données académiques ;

— contrôle l'accès selon les rôles (admin/enseignant/étudiant) ;

— facilite les opérations courantes (CRUD, inscriptions, notes) ;

— assure la sécurité via authentification/autorisation ?

1.3 Objectifs

— Développer une plateforme web multi-rôles.

— Offrir des fonctionnalités CRUD complètes.

— Implémenter l'authentification et l'autorisation par JWT.

— Proposer des interfaces modernes (Tailwind) et des tableaux de bord.

1.4 Organisation du rapport

Le rapport présente : analyse des besoins, conception, réalisation, tests et conclusion.

Chapitre 2

Analyse des besoins

2.1 Acteurs

— **Administrateur :** gère tout le système (départements, cours, utilisateurs, inscriptions, affectations, notes).

— **Enseignant :** consulte ses cours, les étudiants inscrits, gère les notes.

— **Étudiant :** consulte ses cours et ses notes.

2.2 Besoins fonctionnels

2.2.1 Fonctionnalités Administrateur

— Authentification.

— Gestion des départements (CRUD).

— Gestion des cours (CRUD).

— Gestion des étudiants (CRUD).

— Gestion des enseignants (CRUD).

— Gestion des inscriptions (enrôlement, modification statut/grade, suppression).

— Gestion des affectations (assigner un enseignant à un cours par semestre).

— Gestion des notes (consultation par cours, statistiques, mise à jour d'une note).

2.2.2 Fonctionnalités Enseignant

— Authentification.

— Tableau de bord (résumé des cours, étudiants, notes en attente).

— Consultation des cours affectés.

— Consultation de la liste des étudiants par cours.

— Gestion des notes (ajout/modification).

2.2.3 Fonctionnalités Étudiant

— Authentification.

— Tableau de bord (résumé des inscriptions + moyenne).

— Consultation des cours (par statut).

— Consultation des notes (moyenne simple + moyenne pondérée).

2.3 Besoins non fonctionnels

— **Sécurité :** JWT + contrôle d'accès par rôle.

— **Ergonomie :** interface claire, responsive.

— **Performance :** chargement optimisé, états de loading.

— **Maintenabilité :** structure modulaire (features), types TypeScript.

— **Fiabilité :** gestion d'erreurs (messages explicites).

Chapitre 3

Conception UML

3.1 Diagramme de cas d'utilisation

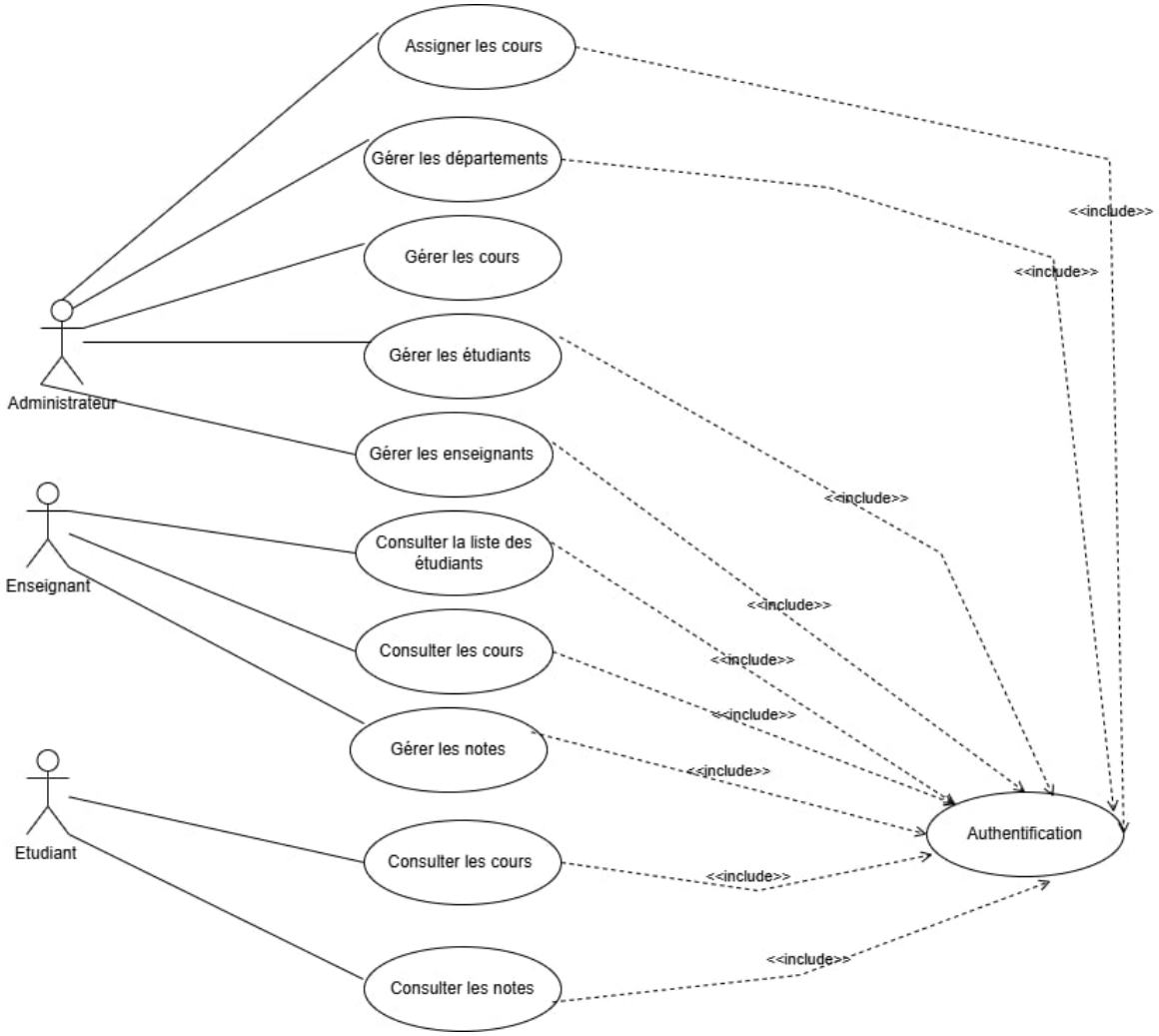

FIGURE 3.1 – Diagramme de cas d'utilisation

3.2 Cas d'utilisation détaillé : GérerCours

Cas d'Utilisation : GérerCours

Acteur : Administrateur
Objectif : Permettre à l'administrateur de gérer les cours académiques
Pré-conditions : Administrateur authentifié
Post-conditions : Les cours sont ajoutés/modifiés/supprimés dans le système

Scénario Principal

1. L'administrateur se connecte au système.
2. L'administrateur accède à la section *"Gestion des Cours"*.
3. Le système affiche la liste des cours existants.
4. L'administrateur choisit une action (*Ajouter/Consulter/Modifier/Supprimer*).
5. Le système exécute l'action demandée.
6. Le système met à jour la base de données.
7. Le système confirme l'opération.

Extensions

— **4a. Ajouter un cours :**

a) L'administrateur saisit les informations du cours (code, nom, crédits, département).
b) Le système valide les données.
c) Le système crée le cours dans la base.

— **4b. Modifier un cours :**

a) L'administrateur sélectionne un cours à modifier.
b) Le système affiche les informations actuelles.
c) L'administrateur modifie les champs nécessaires.
d) Le système sauvegarde les modifications.

— **4c. Supprimer un cours :**

a) L'administrateur sélectionne un cours à supprimer.
b) Le système demande confirmation.
c) Le système supprime le cours (s'il n'a pas d'inscriptions).

3.3 Diagramme de classes

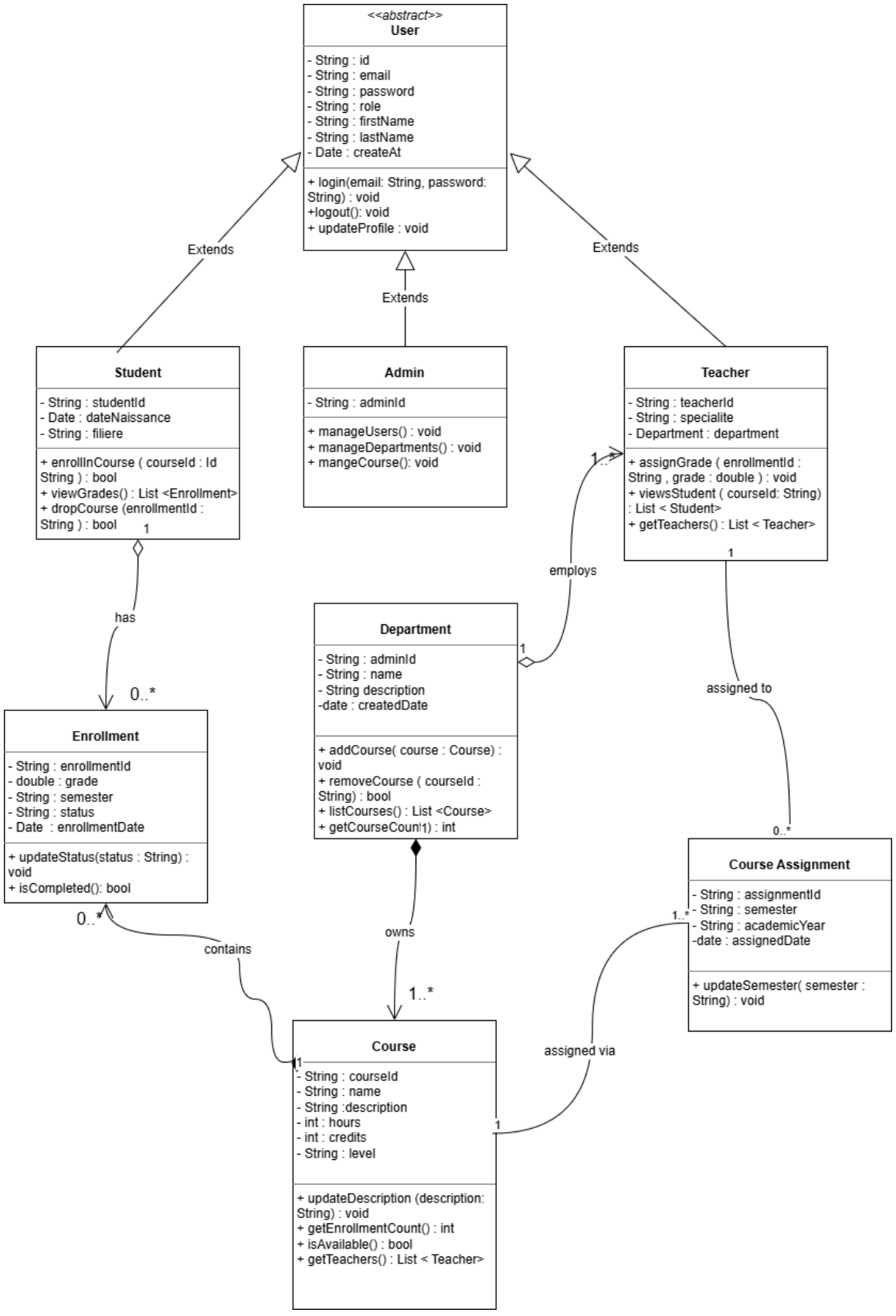

FIGURE 3.2 – Diagramme de classes

3.4 Diagrammes de séquence

3.4.1 Séquence : Ajouter un cours

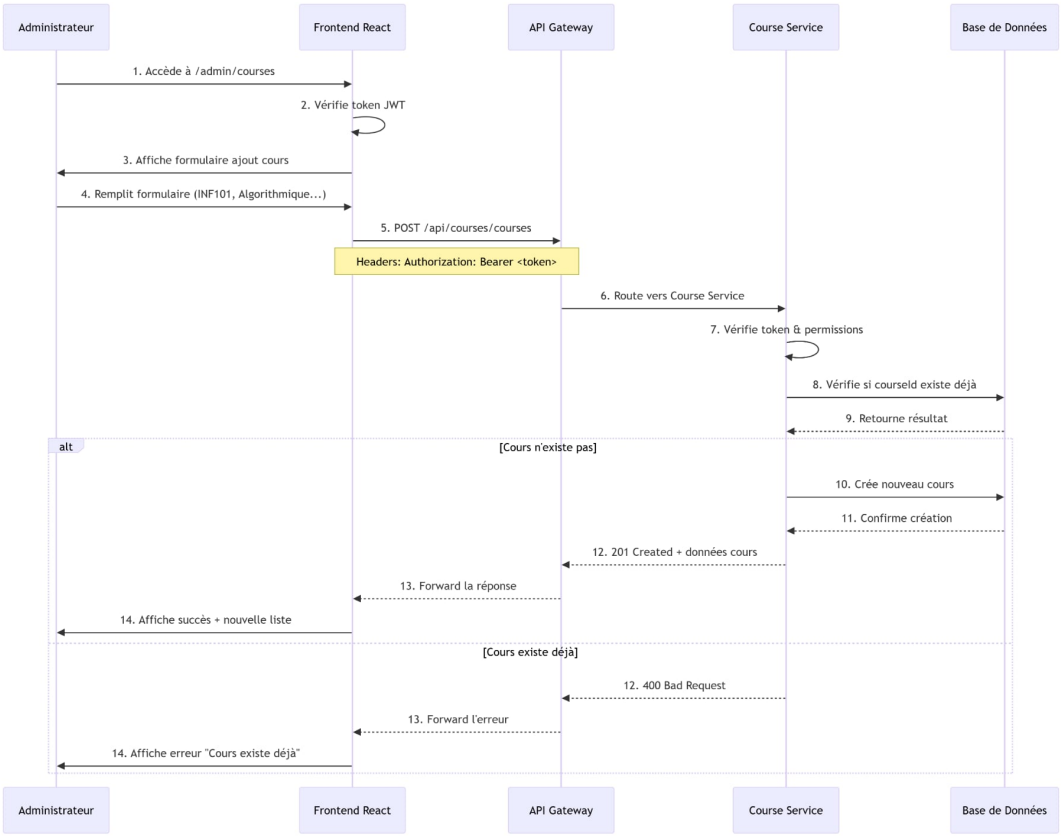

FIGURE 3.3 – Diagramme de séquence : ajout d'un cours (Admin)

Chapitre 4

Architecture et conception technique

4.1 Architecture globale

Le système est une application web structurée en deux parties :

— **Frontend :** React + TypeScript, React Router, Tailwind CSS.

— **Backend :** API REST sécurisée, gestion des rôles, accès base de données.

4.2 Sécurité (JWT)

4.2.1 Authentification

Après connexion, le serveur renvoie un token JWT contenant notamment :

— `userId`

— `role` (admin/teacher/student)

— `email`

— `exp` (expiration)

4.2.2 Autorisation

— Le frontend protège les routes (composant ProtectedRoute) selon le rôle.

— Les requêtes API incluent `Authorization: Bearer <token>`.

4.3 Technologies utilisées

Technologie	**Rôle**
React + TypeScript	Développement de l'interface

React Router	Navigation, routes protégées
Tailwind CSS	Design et responsive
Axios	Appels API
JWT	Sécurité (auth/roles)
API REST	Services (users/courses/enrollments/grades...)
Base de données	Stockage des données

Chapitre 5

Description détaillée du projet

5.1 Module Authentification

5.1.1 Fonctionnement

— L'utilisateur saisit email, mot de passe, et sélectionne un rôle.

— Le backend renvoie un token JWT et le rôle réel de l'utilisateur.

— Si le rôle choisi ne correspond pas au rôle renvoyé, le système affiche une erreur.

— Sinon, le token est sauvegardé, puis redirection vers le dashboard.

5.1.2 Règles

— Token stocké en `localStorage`.

— Vérification d'expiration avant accès aux pages.

— Redirection automatique vers `/login` si token invalide.

5.2 Module Départements (Admin)

5.2.1 Objectif

Permettre la création et l'organisation des départements académiques.

5.2.2 Fonctionnalités

— Afficher la liste des départements (deptId, nom, description, date).

— Ajouter un département via formulaire modal.

— Modifier un département (sans modifier deptId).

— Supprimer un département après confirmation.

5.3 Module Étudiants (Admin)

5.3.1 Fonctionnalités

— Afficher les étudiants avec recherche (nom, email, studentId, filière).

— Ajouter un étudiant (email, password, prénom, nom, studentId, dateNaissance, filière, department).

— Modifier un étudiant (inline edit panel).

— Supprimer un étudiant.

5.4 Module Enseignants (Admin)

5.4.1 Fonctionnalités

— Afficher les enseignants avec recherche (nom, email, teacherId, spécialité).

— Ajouter un enseignant (email, password, prénom, nom, teacherId, spécialité, department).

— Modifier un enseignant.

— Supprimer un enseignant.

5.5 Module Cours (Admin)

5.5.1 Fonctionnalités

— Afficher la liste des cours.

— Ajouter un cours (courseId, nom, description, crédits, heures, département).

— Modifier un cours.

— Supprimer un cours (si aucune contrainte d'inscription).

5.6 Module Inscriptions (Admin)

5.6.1 Fonctionnalités

— Inscrire un étudiant à un cours (`studentId`, `courseId`, `semester`).

— Afficher les inscriptions par cours.

— Modifier : statut (enrolled/completed/dropped) et note.

— Supprimer une inscription.

5.7 Module Affectations (Admin)

5.7.1 Objectif

Associer un enseignant à un cours pour un semestre donné.

5.7.2 Fonctionnalités

— Liste des affectations.

— Création : `teacherId`, `courseId`, `semester`.

— Suppression d'une affectation.

5.8 Module Notes (Admin)

5.8.1 Fonctionnalités

— Sélection d'un cours.

— Affichage des étudiants inscrits avec leurs notes.

— Statistiques : total, graded, average, min, max.

— Ajout/modification d'une note via modal.

5.9 Espace Étudiant

5.9.1 Dashboard

— Affiche les informations de profil (studentId, department, niveau...).

— Résumé des inscriptions (total/enrolled/completed).

— Résumé des notes (moyenne, graded, pending).

5.9.2 Mes cours

— Liste des cours de l'étudiant.

— Filtre par statut (all/enrolled/completed/dropped).

5.9.3 Mes notes

— Affiche les notes par cours.

— Calcule moyenne simple.

— Calcule moyenne pondérée (GPA) en fonction des crédits.

— Affiche crédits acquis, cours notés, etc.

5.10 Espace Enseignant

5.10.1 Dashboard

— Affiche profil enseignant (teacherId, spécialité, department).

— Résumé cours total / semestre courant.

— Nombre total d'étudiants et notes en attente.

5.10.2 Mes cours

— Affiche les cours affectés à l'enseignant.

— Possibilité de dérouler la liste des étudiants inscrits.

— Lien direct vers la gestion des notes du cours.

5.10.3 Gestion des notes

— Sélection d'un cours affecté.

— Liste des étudiants inscrits (studentId, nom, statut).

— Ajout/modification de la note (0–20).

Chapitre 6

Tests et validation

6.1 Scénarios de test

— **Authentification :** login correct, rôle incorrect, token expiré.

— **Admin :** CRUD départements, cours, étudiants, enseignants.

— **Admin :** inscription, mise à jour statut et note, suppression.

— **Enseignant :** consultation cours, liste étudiants, saisie notes.

— **Étudiant :** consultation cours, consultation notes et moyennes.

6.2 Résultats attendus

— Accès uniquement aux pages du rôle.

— Données persistées et mises à jour dans la base.

— Messages d'erreur affichés correctement.

— Notes limitées à 0–20.

Chapitre 7

Conclusion générale et perspectives

7.1 Conclusion

Le projet a permis de réaliser une application web complète de gestion académique multi-rôles. Les modules développés couvrent l'essentiel des besoins (gestion des entités, inscriptions, affectations, notes, dashboards), tout en respectant les exigences de sécurité via JWT.

7.2 Perspectives

— Ajouter export PDF/Excel (relevés, listes).

— Ajouter pagination + filtres avancés côté API.

— Notifications (email) lors de publication de notes.

— Gestion d'emplois du temps et salles.

— Historique des modifications (audit logs).