DÉPARTEMENT D'INFORMATIQUE ET GÉNIE LOGICIEL

RAPPORT DE PROJET DE FIN DE SEMESTRE

Système de Gestion des Cours (LMS)

Conception et Implémentation d'une Solution Fullstack

Technologie : *Spring Boot 3 & React.js*

Date limite : *7 Janvier 2026*

Table des Matières

Chapitre 1 : Introduction et Analyse des Besoins

1.1 Contexte du Projet

L'éducation numérique est devenue un pilier central de l'enseignement moderne. Les établissements scolaires nécessitent des outils performants pour gérer le cycle de vie académique : de la création des cours à l'évaluation des étudiants. Ce projet, "Système de Gestion des Cours", vise à fournir une solution web centralisée pour répondre à ces besoins.

1.2 Objectifs Fonctionnels

Le système a été conçu autour de trois modules interconnectés, définis comme suit :

- **Module Administrateur :** Il représente l'autorité centrale. Ses responsabilités incluent la gestion des utilisateurs (inscription, validation, bannissement) et la structuration académique (création des matières).
- **Module Instructeur :** Dédié au corps enseignant. Il permet la diffusion de contenu pédagogique, la création de devoirs (projets), et, , point crucial, l'évaluation des travaux rendus par les étudiants avec un système de notation et de commentaires.
- **Module Étudiant :** L'interface finale pour les apprenants. Elle permet de consulter les cours, de soumettre des fichiers (PDF, ZIP) pour les évaluations, et de suivre sa progression via les notes reçues.

1.3 Analyse des Flux

Un flux typique dans l'application se déroule ainsi : *L'Administrateur crée un cours (ex: "Java Avancé"). L'Instructeur y ajoute un "Projet Final". L'Étudiant télécharge le sujet, réalise le*

travail, et upload sa solution via la plateforme. L'Instructeur reçoit une notification, télécharge la copie, et attribue une note.

Chapitre 2 : Architecture et Choix Technologiques

2.1 Architecture N-Tiers

Nous avons opté pour une architecture distribuée, séparant clairement le Front-end (Client) du Back-end (Serveur), communiquant exclusivement via une API RESTful.

Technologies Backend

- **Java 17 & Spring Boot 3 :** Choisi pour sa robustesse, son injection de dépendances et son écosystème mature.
- **Spring Security :** Pour la sécurisation des endpoints et le hachage des mots de passe (BCrypt).
- **Maven :** Pour la gestion des dépendances et le build automatisé.

Technologies Frontend

- **React.js (Vite) :** Framework JavaScript pour créer une Single Page Application (SPA) réactive.
- **Tailwind CSS :** Framework CSS utilitaire pour un design rapide et cohérent.
- **Axios :** Client HTTP pour consommer l'API REST.

Chapitre 3 : Conception de la Base de Données

La persistance des données est assurée par MySQL. L'ORM Hibernata (via Spring Data JPA) a été utilisé pour mapper les classes Java vers les tables relationnelles.

3.1 Dictionnaire de Données

ENTITÉ	ATTRIBUTS PRINCIPAUX	RELATIONS & CARDINALITÉS
Utilisateur	`id` (PK), `email` (Unique), `password` (Hash), `role` (ENUM), `status`	- 1 Utilisateur -> N Cours (Instructeur) - 1 Utilisateur -> N Message (Auteur)
Course	`id` (PK), `code` (Unique), `titre`, `description`	- N Cours -> 1 Utilisateur (Instructeur) - 1 Cours -> N Projets
Project	`id` (PK), `titre`, `deadline` (Date)	- N Projets -> 1 Cours
Submission	`id` (PK), `note` (Double), `fileUrl` (String), `submitDate`	- N Soumissions -> 1 Projet - N Soumissions -> 1 Etudiant

Note technique : Une contrainte d'unicité composite `@UniqueConstraint(columnNames = {"etudiant_id", "projet_id"})` a été placée sur la table Submission pour empêcher un étudiant de soumettre deux fois le même devoir.

Chapitre 4 : Implémentation Back-end (Spring Boot)

4.1 Couche de Contrôle (Controllers)

Les contrôleurs exposent l'API REST. Voici un exemple pour la création de projets :

```
@PostMapping public ResponseEntity createProject(@RequestBody Project project) { // Vérification que l'utilisateur est bien l'instructeur du cours if (! permissionService.isInstructorOfCourse(user, project.getCourse())) { throw new AccessDeniedException("Non autorisé"); } return ResponseEntity.ok(projectService.save(project)); }
```

4.2 Gestion des Exceptions

Pour garantir des réponses d'erreur propres (JSON), nous utilisons un `@RestControllerAdvice`. Cela permet de transformer une exception Java (ex: `EntityNotFoundException`) en une réponse HTTP 404 claire pour le client Front-end.

4.3 Sécurité

La classe `SecurityConfig` configure la chaîne de filtres. Actuellement, l'authentification est gérée via des sessions stateful pour le développement, mais l'architecture est prête pour migrer vers JWT (JSON Web Tokens). Le CSRF a été désactivé pour faciliter les tests Postman.

Chapitre 5 : Implémentation Front-end (React.js)

5.1 Gestion de l'État et Navigation

L'application utilise **React Router** pour la navigation côté client. L'état global (utilisateur connecté) est géré, assurant que la barre de navigation (`Navbar.jsx`) s'adapte dynamiquement (montrer "Dashboard Admin" vs "Mes Cours").

5.2 Composants Clés

Visualisation et Identité

Le composant `MoroccanPattern.jsx` a été créé pour donner une identité visuelle unique, utilisant des SVGs pour créer un fond texturé léger et performant.

Interaction Modale

Pour améliorer l'UX (Expérience Utilisateur), nous évitons les rechargements de page. Le `SubmitAssignmentModal.jsx` permet à un étudiant d'uploader son fichier directement depuis la liste des projets. De même, `FeedbackModal.jsx` permet à l'enseignant de noter "à la volée".

Chapitre 6 : Tests et Validation

La validation a été effectuée à deux niveaux :

1. **Tests API (Postman) :** Chaque endpoint a été testé unitairement pour vérifier les codes de retour (200 OK, 201 Created, 400 Bad Request).
2. **Tests d'Intégration (Manuel) :** Des scénarios complets (Création compte -> Connexion -> Création Cours -> Soumission Projet) ont été joués pour valider la chaîne complète.

Chapitre 7 : Conclusion et Perspectives

Le système délivré est fonctionnel et répond au cahier des charges. Il offre une base solide pour une gestion académique digitalisée.

Améliorations futures envisagées :

- **Notifications Temps Réel :** Utilisation de WebSockets pour alerter un étudiant dès qu'une note est publiée.
- **Stockage Cloud :** Migration du stockage de fichiers local vers AWS S3 ou Azure Blob Storage.
- **Analytique :** Tableau de bord graphique pour visualiser la moyenne de la classe par cours.