**Présentation du Projet E-Learning**

Introduction

Nous avons développé une application E-Learning complète, composée d'un backend en Spring Boot et d'un frontend en Angular. Cette application permet de gérer des étudiants, des inscriptions et des classes, avec des fonctionnalités CRUD (Create, Read, Update, Delete) et une authentification sécurisée.

1. Architecture du Projet

Backend (Spring Boot)

* Technologies utilisées :
  + Spring Boot (API REST)
  + Spring Data JPA (Gestion des données)
  + H2 Database (Base de données en mémoire pour le développement)
  + Spring Security (Authentification et autorisation)
  + Lombok (Réduction du code boilerplate)
  + Swagger (Documentation de l'API)
* Structure du projet :
  + Modèles : Student, Registration, Classes.
  + Repositories : Interfaces pour interagir avec la base de données.
  + Services : Logique métier pour gérer les étudiants, les inscriptions et les classes.
  + Contrôleurs : Endpoints REST pour exposer les fonctionnalités.
  + Sécurité : Configuration de Spring Security pour l'authentification.

Frontend (Angular)

* Technologies utilisées :
  + Angular (Framework frontend)
  + HttpClient (Communication avec l'API)
  + Angular Material (UI components)
  + RxJS (Gestion des observables)
* Structure du projet :
  + Modèles : Interfaces TypeScript pour représenter les données (Student, Registration, Classes).
  + Services : Services Angular pour interagir avec l'API.
  + Composants : Composants pour afficher et gérer les étudiants, les inscriptions et les classes.
  + Routing : Navigation entre les différentes vues.

2. Fonctionnalités Implémentées

Backend

1. Gestion des Étudiants :
   * Créer, lire, mettre à jour et supprimer des étudiants.
   * Validation des données avec @Valid et annotations Jakarta.
2. Gestion des Inscriptions :
   * Associer un étudiant à une classe via une inscription.
   * Gérer les dates et descriptions des inscriptions.
3. Gestion des Classes :
   * Créer, lire, mettre à jour et supprimer des classes.
   * Associer des inscriptions à une classe.
4. Authentification et Autorisation :
   * Configuration de Spring Security pour sécuriser les endpoints.
   * Utilisation de JWT (JSON Web Tokens) pour l'authentification.
   * Endpoints protégés par rôles (par exemple, ROLE\_ADMIN, ROLE\_USER).
5. Documentation de l'API :
   * Utilisation de Swagger pour documenter les endpoints.
   * Accès à la documentation via http://localhost:8080/swagger-ui/.

Frontend

1. Affichage des Étudiants :
   * Liste des étudiants avec possibilité de suppression.
   * Formulaire pour ajouter ou modifier un étudiant.
2. Affichage des Inscriptions :
   * Liste des inscriptions avec détails (étudiant, classe, date).
   * Formulaire pour créer ou mettre à jour une inscription.
3. Affichage des Classes :
   * Liste des classes avec description.
   * Formulaire pour ajouter ou modifier une classe.
4. Authentification :
   * Page de connexion pour les utilisateurs.
   * Gestion des tokens JWT pour les requêtes authentifiées.
   * Protection des routes Angular en fonction des rôles.

3. Authentification et Sécurité

Backend

* Spring Security :
  + Configuration des règles d'accès aux endpoints.
  + Utilisation de JwtAuthenticationFilter pour valider les tokens JWT.
  + Endpoint /api/auth/login pour générer un token JWT après authentification.
* JWT (JSON Web Tokens) :
  + Génération d'un token JWT après une connexion réussie.
  + Validation du token pour chaque requête protégée.

4. Démonstration

Backend

* Swagger : Montrer les endpoints disponibles et les tester.
* Base de données H2 : Afficher les données persistées via la console H2.

Frontend

* Navigation : Montrer les différentes pages (étudiants, inscriptions, classes).
* Authentification : Démontrer la connexion et la déconnexion.
* CRUD : Ajouter, modifier et supprimer des étudiants, des inscriptions et des classes.

5. Prochaines Étapes

1. Améliorer l'interface utilisateur :
   * Utiliser Angular Material pour un design plus moderne.
   * Ajouter des notifications (toasts) pour les actions réussies ou les erreurs.
2. Tests :
   * Implémenter des tests unitaires et d'intégration pour le backend et le frontend.
3. Déploiement :
   * Déployer l'application sur un serveur cloud (AWS, Heroku, etc.).
   * Configurer une base de données PostgreSQL pour la production.
4. Fonctionnalités supplémentaires :
   * Ajouter la pagination et la recherche pour les listes.
   * Implémenter un tableau de bord pour les administrateurs.

Conclusion

Nous avons développé une application E-Learning complète, sécurisée et scalable. Le backend en Spring Boot fournit une API robuste, tandis que le frontend en Angular offre une interface utilisateur intuitive. L'ajout de l'authentification et de la gestion des rôles garantit la sécurité des données. Ce projet peut être étendu pour répondre à des besoins plus complexes, comme la gestion des cours, des modules, et des évaluations.

