

RAPPORT TECHNIQUE – Projet P02

MANTOR : Axel Demontoux

Auteur : Quang NGUYEN Date :

Janvier 2026

Table des matières

1. INTRODUCTION.....	2
2. MISE EN PLACE DE L'ARCHITECTURE	2
2.1 Architecture générale.....	2
2.2 Architecture Back-end (Spring Boot).....	2
2.3 Architecture Front-end (Angular)	3
2.4 Base de données MySQL (Docker).....	3
3. EXERCICES EFFECTUÉS	3
3.1 Analyse du code existant	3
3.2 Correction de l'API d'authentification	3
3.3 Implémentation de l'interface d'authentification (Angular).....	4
3.4 Ajout des fonctionnalités CRUD (backend)	4
3.5 Implémentation des écrans CRUD (frontend)	4
3.6 Campagne de tests	4
4. CONCLUSION.....	5

1. INTRODUCTION

Le projet P02 vise à analyser, corriger et améliorer une application full-stack existante composée d'un **backend Spring Boot**, d'un **frontend Angular**, et d'une base **MySQL** exécutée via Docker. L'objectif était de rendre l'application fonctionnelle, sécurisée, maintenable et conforme aux attentes pédagogiques d'OpenClassrooms.

Le travail réalisé couvre :

- l'analyse de l'architecture existante
- la correction de l'API d'authentification
- l'implémentation des écrans d'inscription et de connexion
- l'ajout des fonctionnalités CRUD pour les étudiants
- la sécurisation front/back via JWT
- la mise en place d'une base MySQL Dockerisée
- la réalisation d'une campagne de tests complète

2. MISE EN PLACE DE L'ARCHITECTURE

2.1 Architecture générale

L'application repose sur une architecture full stack moderne :

- **Front-end Angular** : interface utilisateur, formulaires, navigation, appels API
- **Back-end Spring Boot** : logique métier, sécurité, API REST
- **Base MySQL Dockerisée** : stockage persistant
- **Communication** : HTTP + JSON + token JWT

2.2 Architecture Back-end (Spring Boot)

Le backend suit une architecture en couches :

- **Controllers** : réception des requêtes HTTP
- **Services** : logique métier (authentification, CRUD étudiants)
- **Repositories** : accès à la base via Spring Data JPA
- **Security** : filtres JWT, règles d'accès, hashing BCrypt

Flux d'authentification :

1. Vérification des identifiants
2. Hashing du mot de passe (BCrypt)
3. Génération d'un token JWT

4. Stockage du token côté front
5. Validation du token pour chaque requête protégée

2.3 Architecture Front-end (Angular)

Le front utilise :

- **Composants** : Login, Register, StudentList, StudentDetail, StudentEdit
- **Services** : AuthService, RegisterService, StudentService
- **Routing** : navigation entre les écrans
- **Formulaires réactifs** : validation, gestion des erreurs

Flux utilisateur :

1. Saisie des informations
2. Envoi au backend via un service Angular
3. Stockage du token JWT
4. Accès aux écrans CRUD si authentifié

2.4 Base de données MySQL (Docker)

- Exécution via Docker Compose
- Volume persistant
- Charset UTF-8
- Configuration via .env

3. EXERCICES EFFECTUÉS

3.1 Analyse du code existant

- Installation des outils (Java, Node, Angular CLI, Maven, Docker)
- Démarrage du backend et du frontend
- Création d'un utilisateur test
- Analyse de l'architecture front/back
- Compréhension du flux JWT

3.2 Correction de l'API d'authentification

- Correction du filtre JWT
- Correction du hashing BCrypt
- Correction des imports et du bean AuthenticationManager

- Test Postman validé
- Token JWT fonctionnel

3.3 Implémentation de l'interface d'authentification (Angular)

- Création du composant Register
- Formulaire réactif + validation
- Routing vers /register
- Création de RegisterService
- Tests Cypress validés

3.4 Ajout des fonctionnalités CRUD (backend)

- POST /students
- GET /students
- GET /students/{id}
- PUT /students/{id}
- DELETE /students/{id}
- Sécurisation via JWT
- Tests Postman validés

3.5 Implémentation des écrans CRUD (frontend)

- Liste, détail, ajout, modification, suppression
- Sécurisation via token JWT
- Tests Cypress (Home + Register)
- Couverture E2E : 100 %

3.6 Campagne de tests

Backend :

- Authentification
- Erreurs générées proprement
- Token valide/invalidé
- CRUD étudiants
- Unicité du login

Frontend :

- Cypress E2E
- Navigation
- Formulaires
- Sécurisation

4. CONCLUSION

Ce projet m'a permis de :

- comprendre une architecture full stack complète
- corriger une API d'authentification
- implémenter des écrans Angular professionnels
- sécuriser un front et un back via JWT
- manipuler Docker pour la base MySQL
- réaliser une campagne de tests complète

Tout est nouveau pour moi, ces nouvelles technologies. Je ne me sens pas maîtriser encore ces technologies. Mais Je me sens désormais un peu plus confiant pour aborder les projets suivants, notamment ceux qui demanderont des tests unitaires et d'intégration plus avancés.