EHEI Oujda / GI & IG

Année académique 2025 - 2026

Fiche Projet de Fin d'Année

CommitEd : PLATEFORME D'ÉVALUATION INTELLIGENTE

Une approche Dev-First pour l'enseignement supérieur

Réalisé par :

EL AISSAOUI Iliass
MAKOURI Loqman
SGHIOURI IDRISSI Youness
TALEB Fayza

Sous l'encadrement de :

M. MOUHIB Imad

01 Décembre 2025

Table des matières

Liste des Illustrations

Figures

Tableaux

1 Proposition de Projet

1.1 Problématique

Les plateformes éducatives actuelles (type Google Classroom) manquent de spécificité pour l'enseignement de l'informatique. Elles traitent le code comme du texte brut, ignorant tout l'écosystème de développement (Tests, Versioning, Qualité).

Le besoin est de créer une plateforme qui intègre le workflow professionnel (Git, GitHub) directement dans le processus pédagogique, tout en offrant aux professeurs des outils d'analyse automatisés pour suivre la progression réelle des étudiants.

1.2 Méthodologie et Organisation

Pour mener à bien ce projet complexe en équipe, nous adoptons une approche rigoureuse :

- **Méthodologie Scrum :** Développement itératif par « Sprints » courts pour valider chaque module fonctionnalité par fonctionnalité.
- **Collaboration DevOps :** Utilisation intensive de Git & GitHub pour la gestion de versions.
- **Documentation Centralisée :** Mise en place d'un portail `/docs` (type Swagger/ OpenAPI) pour documenter nos APIs et faciliter l'interconnexion de nos micro-services.

1.3 Plan de Développement (Roadmap)

Le projet suivra 5 phases distinctes pour assurer une montée en charge progressive :

Phase 1 - Le Socle « Classroom » Implémentation des fonctionnalités fondamentales : gestion des utilisateurs, création de classes, publication de devoirs simples et soumission de fichiers.

Phase 2 - Statistiques de Base Développement d'un dashboard permettant au professeur de visualiser les taux de soumission, les retards et l'engagement basique des étudiants.

Phase 3 - Intégration GitHub (GitOps) L'automatisation du workflow technique. Le professeur publie un « Template », l'étudiant « Fork » le projet, et la soumission se fait via un « Push ». La plateforme gère les liaisons API avec GitHub.

Phase 4 - Analyse de Code (IA vs Fallback) Deux approches sont envisagées :

- *Plan A (Agentique) :* Un agent IA analyse le code étudiant (respect des principes SOLID, Clean Code) et génère un rapport automatique.
- *Plan B (Fallback) :* Un système de notation paramétrique permettant au professeur d'évaluer manuellement le code sur des métriques précises.

Phase 5 - Modèle Prédictif (ML) Développement d'un modèle ML en Python qui croise les rapports d'analyse et les notes pour prédire les difficultés futures des étudiants et suggérer des actions correctives.

1.4 Architecture & Stack Technologique

Nous avons opté pour une architecture **Micro-services** afin de découpler les responsabilités métier. Cette approche « Polyglotte » nous permet d'appliquer le principe du **« Best Tool**

for the Job » (le meilleur outil pour la tâche) plutôt que de nous enfermer dans un écosystème unique.

L'architecture globale est illustrée dans la Fig. 1.

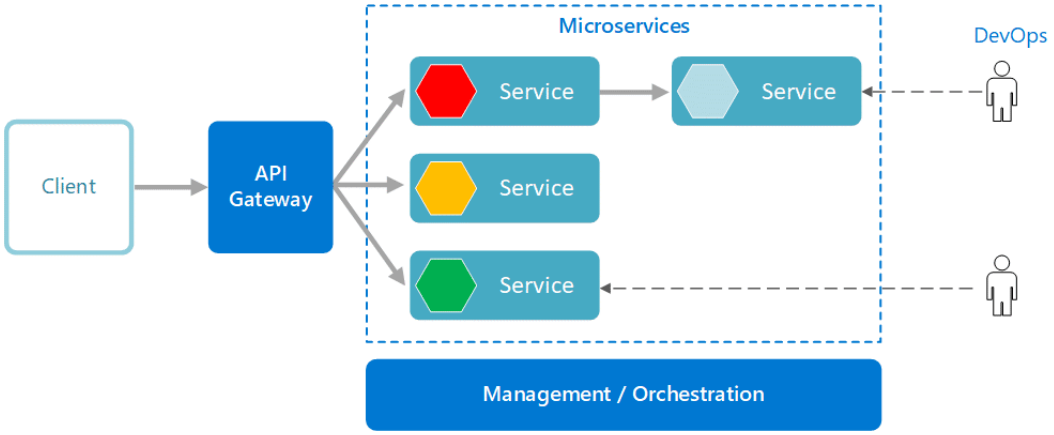

Fig. 1. – Diagramme d'Architecture Micro-services

1.4.1 Justification des Choix Technologiques

Chaque micro-service a des contraintes spécifiques (performance, rapidité de développement, écosystème IA) qui ont dicté le choix de la technologie :

Tableau 1. – Matrice de décision technologique

Service & Techno	**Justification du Choix**
Frontend *React.js*	**Pourquoi React ?** Son architecture à base de composants et le DOM virtuel offrent une fluidité indispensable pour une SPA. **Pourquoi ici ?** Le dashboard nécessite une gestion d'état complexe (temps réel, graphiques) que React gère nativement.
Service Core *Symfony (PHP)*	**Pourquoi Symfony ?** Framework mature pour le développement rapide avec un ORM (Doctrine) puissant. **Pourquoi pour le Core ?** Ce service gère la logique « administrative ». Symfony permet de développer ces CRUD 2x plus vite qu'en Java.
Service Orchestrator *Spring Boot (Java)*	**Pourquoi Spring ?** L'écosystème Java est inégalé pour la stabilité, le typage fort et la gestion du multithreading. **Pourquoi pour l'Orchestrator ?** Ce service est le moteur critique (Git, Docker, I/O intensifs). La robustesse de la JVM est impérative ici.
Service Analytics *ASP.NET Core (C#)*	**Pourquoi .NET ?** Performance d'exécution proche du C++ et la bibliothèque LINQ, atout majeur pour la data. **Pourquoi pour l'Analytics ?** LINQ nous permet d'écrire des requêtes de filtrage complexes de manière plus lisible et performante que du SQL pur.
Service Intelligence *Python (FastAPI)*	**Pourquoi Python ?** La lingua franca de l'IA et de la Data Science (Pandas, Scikit-learn).
Infrastructure *Docker & Oracle*	**Docker :** Indispensable pour l'isolation et la création de « Sandboxes » éphémères pour le code étudiant. **Oracle Database :** Choisi pour sa conformité ACID stricte et pour simuler un environnement « Grand Compte ».

1.5 Répartition Prévisionnelle des Tâches

- **Membre 1 :** Architecture Frontend & Intégration UX.
- **Membre 2 :** Backend Core (Symfony) & Base de données.
- **Membre 3 :** Backend Orchestrator (Spring) & Intégration GitHub API.
- **Membre 4 :** Services Analytics (.NET) & Intelligence Artificielle (Python).

2 Annexe : Journal de Suivi (Logbook)

2.1 Séance 1 : Validation du Sujet

Date : 01 Décembre 2025
Présents : Tout le groupe

Remarques du Superviseur :

Décisions prises :