Rapport Technique – Architecture et Choix Technologique

Master 1 SRT – 2024-2025

Projet : Plateforme de gestion d'exercices avec correction automatique par IA

Cours : SGBD

Membre du groupe :

- ✓ **Hamit Amir MAHAMAT**
- ✓ **Mouhamadou Moustapha BA**
- ✓ **Ibrahim FALL**
- ✓ **Amadou DIALLO**

Introduction

Dans le cadre de notre projet de fin de cours de SGBD, nous avons développé une plateforme web interactive dédiée à la gestion des exercices entre professeurs et étudiants. Ce projet avait pour ambition d'intégrer une correction automatisée via l'intelligence artificielle (IA), tout en respectant les exigences du cahier des charges fourni. Le présent rapport retrace les **choix technologiques** que nous avons faits ainsi que **l'architecture globale** mise en place. Nous partageons également les **problèmes rencontrés** lors du déploiement sur AWS.

Architecture du système

Notre solution repose sur une **architecture web en trois couches**, qui permet une séparation claire des responsabilités :

1. Frontend (Interface utilisateur - React.js)

- Utilisé pour concevoir les interfaces des étudiants et des professeurs.
- Les étudiants peuvent consulter les exercices, soumettre leurs réponses au format PDF et consulter leurs notes/feedback.
- Les professeurs peuvent publier des exercices sous forme de texte, voir les soumissions, et consulter les notes attribuées par l'IA.

2. Backend (API - Django + Django REST Framework)

- Gère la logique métier : exercices, utilisateurs, soumissions, correction IA.
- Implémente une API RESTful sécurisée par des jetons JWT (SimpleJWT).
- Permet la gestion automatique des utilisateurs, des rôles (professeur/étudiant), et des fichiers.

3. Service de correction IA (Ollama + DeepSeek)

- Lorsque l'étudiant soumet un fichier PDF, le serveur extrait son contenu via **PyPDF2**, puis l'envoie à **Ollama** utilisant le modèle **DeepSeek-Coder** pour obtenir :
    - Une correction textuelle
    - Un feedback personnalisé
    - Une note sur 20

4. Stockage des fichiers

- Les fichiers PDF (sujets et soumissions) sont stockés en local, avec une structure prête à migrer vers **MinIO** ou **Amazon S3** en production cloud.

Choix technologiques justifiés

Composant	Technologie choisie	Raisons du choix
Interface utilisateur	React.js	Framework moderne, composantiel, idéal pour les interfaces dynamiques
Backend	Django + DRF	Puissance, rapidité de développement, gestion ORM, authentification intégrée
Authentification	JWT (via SimpleJWT)	Permet un accès sécurisé à l'API REST
IA de correction	DeepSeek via Ollama local	Réponse rapide, correction précise de type SQL, fonctionne en local
Extraction PDF	PyPDF2	Simple, efficace, adapté à nos besoins
Serveur de production	Gunicorn + NGINX	Pour servir l'application en production de manière fiable
Déploiement	AWS EC2 (Ubuntu 22.04)	Offre gratuite, accessible, flexible

Problèmes rencontrés lors du déploiement AWS

Malgré le bon fonctionnement de notre application en local, le déploiement sur AWS EC2 a présenté plusieurs difficultés majeures qui ont empêché la mise en ligne complète du site. Voici les principaux problèmes rencontrés :

Port déjà utilisé

- Lors de l'exécution de Gunicorn, le message Address already in use apparaissait.

Cela indiquait que le port 8000 était déjà utilisé par un autre processus.

Solution : nous avons utilisé la commande sudo lsof -i :8000 suivie de kill -9 <PID> pour libérer le port.

- **Conflit avec la configuration NGINX**
  - Au lieu de charger notre projet, le navigateur affichait la page par défaut de NGINX.

Cela provenait du fait que la configuration par défaut n'avait pas été désactivée.

Solution : tentative de suppression du fichier /etc/nginx/sites-enabled/default.

- **Blocage du pare-feu AWS**
  - L'adresse IP publique **(http://16.171.171.4)** restait inaccessible depuis un navigateur externe.

NB : Malgré toutes les configurations effectuées, notre application ne s'affiche toujours pas en ligne sur l'adresse publique.

Nous n'avons pas encore trouvé de solution définitive à ce problème de déploiement.

Perspectives

Nous prévoyons de :

- Reprendre toute la configuration NGINX en repartant de zéro.
- Tester avec une redirection du frontend vers S3 ou un service statique.
- Envisager une alternative de déploiement comme **Render**, **Railway**, ou **Heroku** si les problèmes AWS persistent.