République algérienne démocratique et populaire
Ministère de l'Enseignement supérieur et de la Recherche scientifique
Université M'Hamed Bougara de Boumerdes

Faculté des Sciences
Département d'informatique

Spécialité : Intelligence artificielle appliquée

PROJET BDA

Plateforme d'Optimisation des Emplois du Temps d'Examens Universitaires

Préparé par :
AZZABI Abdelhadi
GAHLOUZ Djamel
TALAHARI Yassine

Soumis en : **Janvier 2026**

Introduction

L'élaboration des emplois du temps d'examens universitaires constitue une tâche complexe et critique, en particulier dans les facultés à forte capacité d'accueil. Dans un établissement regroupant plus de 13 000 étudiants répartis sur plusieurs départements et formations, la planification manuelle entraîne fréquemment des conflits organisationnels.

Les principales difficultés rencontrées concernent la gestion des capacités des salles, la disponibilité des enseignants, les chevauchements d'examens pour les étudiants ainsi que le respect des contraintes pédagogiques. Ces limitations justifient la mise en place d'un système automatisé capable de produire des emplois du temps fiables et optimisés.

Notre projet vise à concevoir une plateforme informatique reposant sur une base de données relationnelle robuste et un algorithme d'optimisation automatique afin de générer des plannings d'examens cohérents en un temps réduit.

1 Analyse du problème et cahier des charges

Contexte général

La faculté étudiée comprend plus de 200 formations, chacune composée de 6 à 9 modules. La gestion simultanée de milliers d'inscriptions rend la planification manuelle inefficace et source d'erreurs.

1.1 Contraintes fonctionnelles

— Un étudiant ne peut passer qu'un seul examen par jour.
— Un professeur ne peut surveiller plus de trois examens par jour.
— La capacité des salles doit être respectée.
— Les enseignants surveillent en priorité les examens de leur département.

1.2 Contraintes non fonctionnelles

— Temps de génération inférieur à 45 secondes.
— Gestion d'un volume important de données.
— Fiabilité et intégrité des données.

2 Modélisation de la base de données

La modélisation de la base de données repose sur une approche relationnelle afin de garantir la cohérence et la performance du système.

Voyons les tables principales :

— **departements**
— **formations**
— **etudiants**
— **modules**
— **professeurs**
— **salles**
— **inscriptions**
— **examens**

L'intégrité des données est assurée par l'utilisation de clés primaires, de clés étrangères et de triggers permettant de vérifier automatiquement certaines contraintes critiques.

3 Implémentation et Optimisation

3.1 Implémentation de la base de données

La base de données a été implémentée sous PostgreSQL. Des index ont été ajoutés sur les colonnes fréquemment utilisées dans les jointures et les requêtes analytiques.

Des triggers PL/pgSQL ont été développés afin de :

— Limiter les examens à un par jour pour chaque étudiant.
— Restreindre le nombre d'examens surveillés par professeur.

3.2 Algorithme d'optimisation

L'algorithme d'optimisation repose sur une génération contrôlée des examens, suivie d'une vérification automatique des conflits. Chaque tentative invalide entraîne une réaffectation des ressources.

4 Architecture et Interface

4.1 Architecture de la plateforme

Notre plateforme repose sur une architecture en trois couches assurant une séparation claire entre la gestion des données, la logique métier et la présentation, garantissant performance, maintenabilité et évolutivité.

— Base de données : PostgreSQL

— Backend : Python

— Frontend : Streamlit

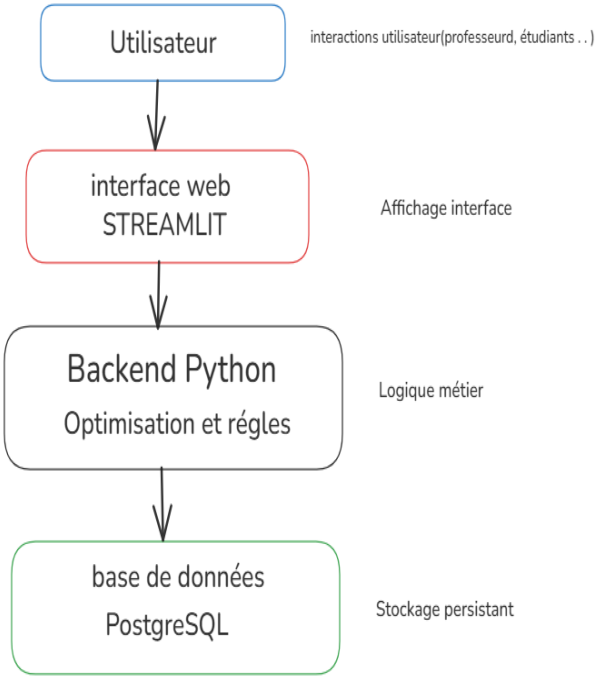

FIGURE 1 – Architecture globale de la plateforme

5 Interface web et visualisation

L'interface web permet une visualisation claire et interactive des données via le tableau de bord principal et les statistiques avancées.

FIGURE 2 – Dashboard principal

6 Détection des conflits et Performances

6.1 Détection des conflits

Le système détecte automatiquement les conflits étudiants, professeurs et les dépassements de capacité.

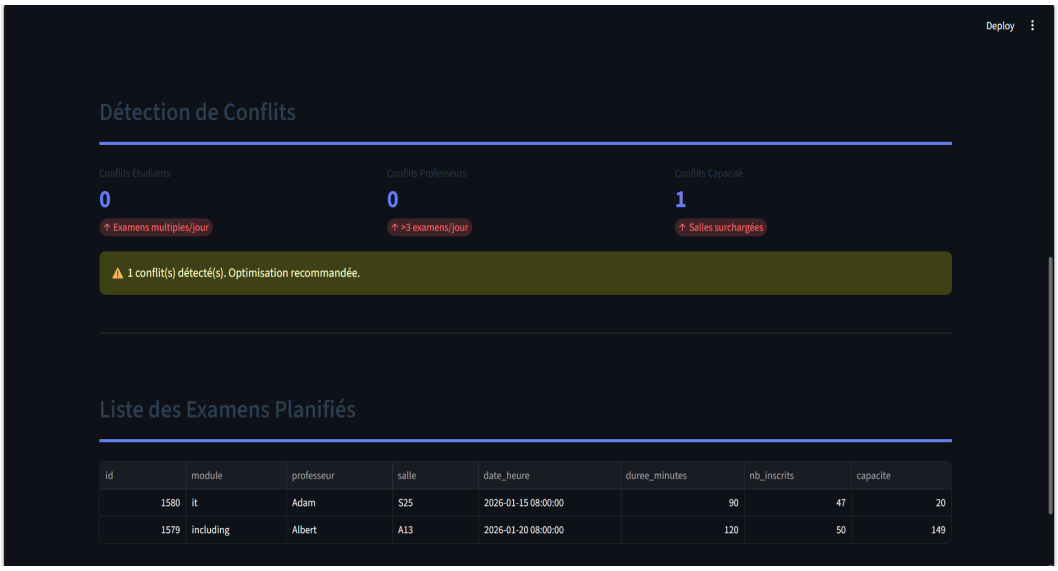

FIGURE 3 – Exemple de détection des conflits

6.2 Performances et benchmarks

Les tests sur 130 000 inscriptions montrent un temps de génération moyen inférieur à 45 secondes.

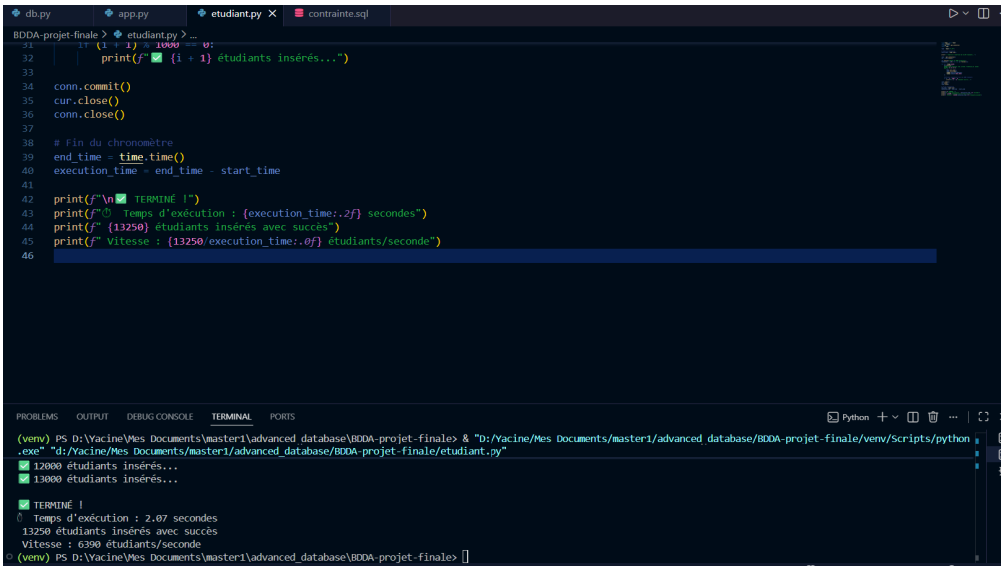

FIGURE 4 – Architecture globale de la plateforme

L'exécution de cette requête démontre que la génération a produit plus de 13 000 étudiants.

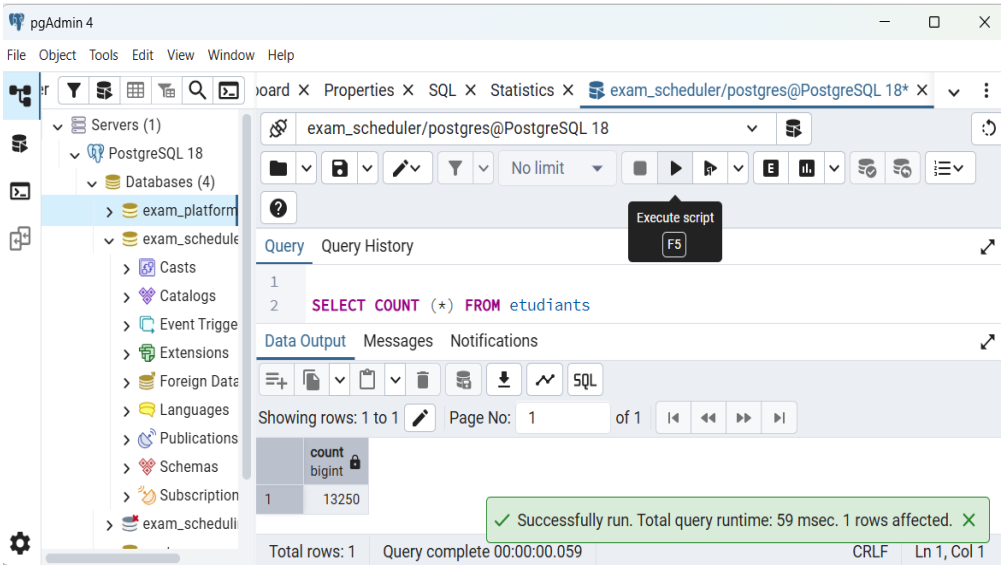

FIGURE 5 – Test de query

7 Stack technique

Voici les technologies utilisées dans la réalisation de notre projet.

7.1 Python

Python est le langage de programmation principal utilisé pour développer la logique métier, le traitement des données et l'orchestration des différents services. Il offre une grande flexibilité et un écosystème riche pour l'intégration de bibliothèques spécialisées.

FIGURE 6 – Logo Python

7.2 PostgreSQL

PostgreSQL est utilisé comme système de gestion de base de données relationnelle pour stocker de manière structurée et persistante les données critiques du projet, garantissant ainsi l'intégrité et la cohérence des informations.

FIGURE 7 – Logo PostgreSQL

7.3 Streamlit

Streamlit est le framework utilisé pour concevoir l'interface utilisateur de notre plateforme. Il permet de visualiser les emplois du temps générés, de suivre les statistiques d'occupation des salles et d'afficher les alertes en cas de détection de conflits, le tout via une application web interactive et intuitive.

FIGURE 8 – Interface de visualisation Streamlit

8 Défis, bilan et perspectives

La réalisation de ce projet a soulevé plusieurs défis techniques et organisationnels. Parmi les principaux obstacles rencontrés figurent la gestion d'un volume important de données, la modélisation de contraintes complexes et parfois contradictoires (étudiants, enseignants, salles), ainsi que l'optimisation des performances afin de respecter le temps de génération imposé.

Malgré ces contraintes, les objectifs principaux du projet ont été atteints. La base de données relationnelle a été correctement modélisée et implémentée, garantissant l'intégrité et la cohérence des données. Les mécanismes de détection des conflits, basés sur des contraintes et des triggers, permettent d'assurer la validité des emplois du temps générés. De plus, l'algorithme développé est capable de produire des plannings fonctionnels en un temps compatible avec les exigences du cahier des charges.

En perspective, plusieurs améliorations peuvent être envisagées. L'intégration d'algorithmes d'intelligence artificielle ou de méthodes d'optimisation avancées (permutation, heuristiques, algorithmes génétiques) pourrait permettre d'améliorer la qualité des solutions générées. Par ailleurs, l'ajout de nouvelles contraintes pédagogiques, une meilleure prise en compte des préférences des enseignants et une extension de la plateforme à d'autres établissements constitueraient des axes d'évolution pertinents pour ce système.

Conclusion

Ce projet a permis de concevoir et de mettre en œuvre une plateforme complète d'optimisation des emplois du temps d'examens universitaires. En s'appuyant sur une base de données relationnelle robuste et des mécanismes de contrôle adaptés, la solution proposée répond efficacement aux problématiques de planification et de gestion des ressources.

Les résultats obtenus démontrent l'intérêt d'une automatisation du processus d'élaboration des emplois du temps, tant en termes de fiabilité que de gain de temps. Ce travail constitue ainsi une base solide pouvant être enrichie et améliorée dans un contexte académique ou institutionnel réel.