

**Université M'hamed Bougara**

Faculté des Sciences  
Département d'Informatique

---

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

---

**Projet Base de Données Avancée**

Réalisé par :  
AISSOU Chaima  
FLICI Kenza

**Spécialité :**  
IA Appliquée

**Année :**  
2025/2026

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Contexte du Projet . . . . .	3
1.2	Problématique . . . . .	3
1.3	Objectifs . . . . .	3
<b>2</b>	<b>Architecture de la Base de Données</b>	<b>3</b>
2.1	Modèle Conceptuel . . . . .	3
2.2	Tables Principales . . . . .	3
2.2.1	Tables Entités . . . . .	3
2.2.2	Tables de Gestion . . . . .	4
<b>3</b>	<b>Mécanismes Avancés de Base de Données</b>	<b>4</b>
3.1	Vues Matérialisées . . . . .	4
3.1.1	Vue Statistiques Globales . . . . .	4
3.2	Triggers (Déclencheurs) . . . . .	4
3.2.1	Trigger de Détection de Conflits de Salles . . . . .	4
3.2.2	Trigger de Limitation Examens Étudiant . . . . .	4
3.3	Index Optimisés . . . . .	5
3.3.1	Index Partiels et Composés . . . . .	5
<b>4</b>	<b>Contraintes et Règles Métier</b>	<b>5</b>
4.1	Contraintes Implémentées . . . . .	5
4.1.1	Contraintes Étudiants . . . . .	5
4.1.2	Contraintes Professeurs . . . . .	5
4.1.3	Contraintes Salles . . . . .	5
4.2	Gestion des Priorités . . . . .	5
4.2.1	Système de Priorités . . . . .	5
<b>5</b>	<b>Algorithme d'Optimisation</b>	<b>6</b>
5.1	Approche Générale . . . . .	6
5.2	Phase 1 : Collecte des Données . . . . .	6
5.3	Phase 2 : Génération Initiale . . . . .	6
5.4	Phase 3 : Détection des Conflits . . . . .	6
5.5	Phase 4 : Résolution Partielle . . . . .	6
5.6	Complexité et Performance . . . . .	6
<b>6</b>	<b>Architecture Applicative</b>	<b>7</b>
6.1	Stack Technologique . . . . .	7
6.1.1	Backend . . . . .	7
6.1.2	Frontend . . . . .	7
6.1.3	Base de Données . . . . .	7
6.2	Architecture en Couches . . . . .	7
6.2.1	Couche Présentation . . . . .	7
6.2.2	Couche Métier . . . . .	7
6.2.3	Couche Données . . . . .	7
6.2.4	Utilitaires . . . . .	7
6.3	Configuration . . . . .	8

<b>7 Fonctionnalités par Acteur</b>	<b>8</b>
7.1 Vice-Doyen et Doyen . . . . .	8
7.1.1 Tableau de Bord Stratégique . . . . .	8
7.2 Administrateur Examens (Service Planification) . . . . .	8
7.2.1 Configuration de Génération . . . . .	8
7.2.2 Génération Automatique . . . . .	8
7.2.3 Gestion des Conflits . . . . .	8
7.2.4 Export et Sauvegarde . . . . .	9
7.3 Chef de Département . . . . .	9
7.3.1 Vue Département . . . . .	9
7.3.2 Validation Départementale . . . . .	9
7.4 Étudiants et Professeurs . . . . .	9
7.4.1 Consultation Planning . . . . .	9
<b>8 Sécurité et Gestion des Accès</b>	<b>9</b>
8.1 Authentification . . . . .	9
8.1.1 Système Multi-Rôles . . . . .	9
8.2 Gestion des Sessions . . . . .	10
8.2.1 Table Sessions Sécurisée . . . . .	10
8.3 Audit et Traçabilité . . . . .	10
8.3.1 Table Logs Connexion . . . . .	10
8.3.2 Table Logs Optimisation . . . . .	10
<b>9 Tests et Validation</b>	<b>10</b>
9.1 Jeu de Données de Test . . . . .	10
9.1.1 Dataset Minimal . . . . .	10
9.1.2 Dataset Réaliste (Optionnel) . . . . .	10
9.2 Scénarios de Test . . . . .	11
9.2.1 Tests Fonctionnels . . . . .	11
9.2.2 Tests de Performance . . . . .	11
9.3 Résultats . . . . .	11
<b>10 Améliorations Futures</b>	<b>11</b>
10.1 Fonctionnalités Additionnelles . . . . .	11
10.2 Optimisations Techniques . . . . .	12
<b>11 Conclusion</b>	<b>12</b>
11.1 Bilan du Projet . . . . .	12
11.2 Objectifs Atteints . . . . .	12
11.3 Compétences Acquises . . . . .	12
11.4 Perspectives . . . . .	12

# 1 Introduction

## 1.1 Contexte du Projet

Dans un environnement universitaire accueillant plus de 13 000 étudiants répartis sur 7 départements et proposant plus de 200 offres de formation, la planification manuelle des emplois du temps d'examens représente un défi considérable. Cette tâche complexe génère fréquemment des conflits logistiques et organisationnels.

## 1.2 Problématique

Les principales difficultés rencontrées dans la gestion manuelle incluent :

- Surcharge des amphithéâtres avec capacités variables
- Limitation des salles à 20 étudiants maximum en période d'examen
- Chevauchements de disponibilités étudiants/professeurs
- Contraintes d'équipements spécifiques
- Respect des règles de surveillance équitable

## 1.3 Objectifs

Ce projet vise à développer une solution complète permettant de :

- Générer automatiquement des plannings optimaux en moins de 45 secondes
- Déetecter et résoudre les conflits potentiels
- Optimiser l'utilisation des ressources (salles, surveillants)
- Fournir une interface web intuitive pour tous les acteurs

# 2 Architecture de la Base de Données

## 2.1 Modèle Conceptuel

La base de données a été conçue selon une approche relationnelle normalisée (3NF) pour garantir l'intégrité et la cohérence des données.

## 2.2 Tables Principales

### 2.2.1 Tables Entités

Table	Description
departements	Contient les 7 départements de la faculté avec leur code, nom et responsable
formations	Répertorie les 200+ formations avec effectifs, niveau (L1-M2) et nombre de modules
etudiants	Données personnelles, matricule unique, formation et promotion de chaque étudiant
professeurs	Informations sur les enseignants incluant département, spécialité et contraintes de surveillance
modules	Modules d'enseignement avec crédits, coefficients, pré-requis et volume horaire

Table	Description
lieu_examen	Salles et amphithéâtres avec capacité, type, localisation et équipements disponibles
examens	Planning des examens avec date, durée, salle et surveillant principal
inscriptions	Relations étudiants-modules pour calculer les effectifs réels

### 2.2.2 Tables de Gestion

- administrateurs : Gestion des comptes admin (Doyen, Vice-doyen, Service planification)
- exam\_surveillance : Affectation multiple de surveillants par examen
- contraintes\_speciales : Gestion des exceptions et contraintes particulières
- logs\_optimisation : Historique des générations avec métriques de performance
- logs\_connexion : Audit des connexions et actions utilisateurs
- sessions : Gestion sécurisée des sessions web avec tokens

## 3 Mécanismes Avancés de Base de Données

### 3.1 Vues Matérialisées

#### 3.1.1 Vue Statistiques Globales

Cette vue agrège en temps réel les indicateurs clés de performance (KPIs) :

- Nombre total d'étudiants, professeurs, examens
- Capacité moyenne et totale des salles
- Statistiques par département et formation

**Rôle :** Optimisation des requêtes du tableau de bord décideurs en évitant les calculs répétitifs sur des données volumineuses.

### 3.2 Triggers (Déclencheurs)

#### 3.2.1 Trigger de Détection de Conflits de Salles

**Nom :** trigger\_check\_exam\_conflict

**Déclenchement :** BEFORE INSERT sur table examens

**Fonctionnement :**

- Vérifie qu'une salle n'est pas déjà occupée au même créneau horaire
- Compare les intervalles temporels (date\_heure + durée)
- Lève une exception SQL en cas de conflit détecté

**Impact :** Garantit l'intégrité du planning au niveau de la base de données, empêchant toute double réservation de salle.

#### 3.2.2 Trigger de Limitation Examens Étudiant

**Nom :** trigger\_limit\_exams\_per\_student

**Déclenchement :** BEFORE INSERT sur table examens

**Fonctionnement :**

- Compte le nombre d'examens déjà planifiés pour un étudiant le même jour
  - Vérifie via la table `inscriptions` quels étudiants sont concernés
  - Bloque l'insertion si la contrainte "maximum 1 examen/jour" est violée
- Impact :** Respect automatique des règles pédagogiques sans nécessiter de vérifications applicatives.

### 3.3 Index Optimisés

#### 3.3.1 Index Partiels et Composés

Pour améliorer les performances sur un dataset de 130 000+ inscriptions :

- `idx_examens_date` : Accélère les recherches par plage de dates
- `idx_examens_salle` : Optimise la détection de conflits de salles
- `idx_examens_module` : Accélère les jointures examens-modules
- `idx_inscriptions_etudiant` : Optimise les requêtes par étudiant
- `unique_surveillance` : Index composite garantissant l'unicité (examen, professeur)

**Résultat :** Réduction du temps d'exécution des requêtes complexes de 800ms à 45ms en moyenne.

## 4 Contraintes et Règles Métier

### 4.1 Contraintes Implémentées

#### 4.1.1 Contraintes Étudiants

- **Maximum 1 examen par jour** : Implémenté via trigger BEFORE INSERT
- **Respect des prérequis** : Colonne `pre_requis_id` dans table `modules`
- **Capacité des salles** : Validation côté application avec alerte si capacité dépassée

#### 4.1.2 Contraintes Professeurs

- **Maximum 3 surveillances par jour** : Implémenté dans la logique de génération
- **Équilibrage des surveillances** : Algorithme d'affectation round-robin
- **Priorité département** : Les professeurs surveillent prioritairement leur département

#### 4.1.3 Contraintes Salles

- **Disponibilité** : Colonne `disponibilite` (boolean)
- **Équipements requis** : Champ JSON stockant liste d'équipements
- **Type adapté** : Distinction Amphi/Salle/Laboratoire selon module

### 4.2 Gestion des Priorités

#### 4.2.1 Système de Priorités

Table `contraintes_speciales` permettant de définir :

- Indisponibilités exceptionnelles (salles, professeurs)
- Dates bloquées pour maintenance

- Contraintes spécifiques par entité avec niveau de priorité (1-5)

## 5 Algorithme d'Optimisation

### 5.1 Approche Générale

L'algorithme de génération automatique utilise une approche heuristique en plusieurs phases :

### 5.2 Phase 1 : Collecte des Données

- Récupération des modules à planifier depuis la base
- Chargement des salles disponibles avec capacités
- Récupération de la liste des professeurs et leurs contraintes
- Calcul des effectifs réels via table `inscriptions`

### 5.3 Phase 2 : Génération Initiale

Stratégie :

1. Tri des modules par ordre décroissant d'effectif (modules les plus critiques d'abord)
2. Attribution séquentielle des créneaux horaires (08 :00, 10 :30, 14 :00, 16 :30)
3. Affectation des salles selon capacité requise
4. Sélection du surveillant principal selon département et disponibilité

### 5.4 Phase 3 : Détection des Conflits

Types de conflits détectés :

- **Conflit de salle** : Même salle, créneaux qui se chevauchent
- **Conflit de surveillant** : Même professeur affecté simultanément
- **Conflit étudiant** : Étudiant inscrit à 2+ examens le même jour
- **Capacité insuffisante** : Effectif > capacité salle

### 5.5 Phase 4 : Résolution Partielle

Stratégies de résolution automatique :

- Déplacement vers créneaux adjacents disponibles
- Réaffectation de salle de capacité supérieure
- Changement de surveillant si disponible
- Signalement des conflits non résolubles pour intervention manuelle

### 5.6 Complexité et Performance

**Complexité théorique** :  $O(n \log n)$  pour le tri +  $O(n^2)$  pour la détection de conflits

**Performances mesurées** :

- 50 examens : 12-18 secondes
- 100 examens : 28-35 secondes
- 200 examens : 38-45 secondes (objectif atteint)

## 6 Architecture Applicative

### 6.1 Stack Technologique

#### 6.1.1 Backend

- **Python 3.10+** : Langage principal
- **mysql-connector-python** : Connecteur base de données
- **Pandas** : Manipulation de données tabulaires
- **Plotly** : Visualisations interactives

#### 6.1.2 Frontend

- **Streamlit** : Framework web Python pour interfaces réactives
- **CSS personnalisé** : Design professionnel et responsive
- **JavaScript (Plotly.js)** : Graphiques interactifs

#### 6.1.3 Base de Données

- **MySQL 8.0** : SGBD relationnel avec support triggers et vues
- **phpMyAdmin** : Interface de gestion base de données

### 6.2 Architecture en Couches

#### 6.2.1 Couche Présentation

- `pages/` : Interfaces utilisateur par rôle
- Connexion.py : Authentification multi-rôles
  - Admin.py : Interface service planification
  - Decideurs.py : Tableaux de bord KPIs
  - Chef\_Dept.py : Validation départementale
  - Consultation.py : Visualisation planning public

#### 6.2.2 Couche Métier

- `services/` : Logique applicative
- `auth_service.py` : Gestion authentification
  - `scheduler_service.py` : Algorithme génération planning
  - `conflict_detector.py` : Détection conflits

#### 6.2.3 Couche Données

- `models/` : Accès base de données
- `db_manager.py` : Requêtes SQL et ORM simple

#### 6.2.4 Utilitaires

- `utils/` : Fonctions transverses
- `styles.py` : CSS personnalisé
  - `helpers.py` : Fonctions auxiliaires (session, formatage)

## 6.3 Configuration

config/database.py : Paramètres connexion MySQL centralisés

# 7 Fonctionnalités par Acteur

## 7.1 Vice-Doyen et Doyen

### 7.1.1 Tableau de Bord Stratégique

- **Indicateurs globaux :**
  - Nombre total d'examens programmés
  - Taux d'occupation global des salles et amphithéâtres
  - Nombre de conflits restants
  - Taux de validation par département
- **Visualisations interactives :**
  - Graphique circulaire : Répartition examens par département
  - Graphique en barres : Charge par formation
  - Courbe temporelle : Nombre d'examens par jour
- **Validation finale :**
  - Bouton de validation définitive du planning
  - Génération PDF du planning officiel
  - Publication automatique vers tous les utilisateurs

## 7.2 Administrateur Examens (Service Planification)

### 7.2.1 Configuration de Génération

- Sélection période : date début et fin examens
- Choix durée créneaux (90, 120, 150, 180 minutes)
- Sélection modules à planifier avec cases à cocher
- Affectation surveillant principal par module

### 7.2.2 Génération Automatique

- Lancement algorithme optimisation avec barre de progression
- Affichage temps d'exécution en temps réel
- Métriques post-génération :
  - Nombre d'examens planifiés
  - Conflits détectés avec niveau de gravité
  - Temps d'exécution total

### 7.2.3 Gestion des Conflits

- Liste détaillée des conflits avec type et description
- Code couleur : rouge (erreur) / orange (avertissement)
- Actions : résolution manuelle ou relance génération

#### 7.2.4 Export et Sauvegarde

- Export CSV du planning complet
- Sauvegarde automatique en base de données
- Versioning des plannings (historique)

### 7.3 Chef de Département

#### 7.3.1 Vue Département

- Filtrage automatique examens de son département
- Métriques spécifiques :
  - Nombre d'examens département
  - Modules concernés
  - Alertes départementales

#### 7.3.2 Validation Départementale

- Consultation planning détaillé par module
- Ajout remarques et observations
- Validation ou demande modification
- Export planning département (CSV)

### 7.4 Étudiants et Professeurs

#### 7.4.1 Consultation Planning

- **Filtres avancés :**
  - Par département
  - Par formation
  - Recherche textuelle (module, salle, professeur)
- **Affichage détaillé :**
  - Cartes visuelles par examen
  - Date, heure, durée
  - Salle avec capacité
  - Surveillant principal
  - Code module et formation
- **Export personnalisé :**
  - Téléchargement CSV planning filtré
  - Possibilité import calendrier électronique

## 8 Sécurité et Gestion des Accès

### 8.1 Authentification

#### 8.1.1 Système Multi-Rôles

- Authentification par matricule + nom complet
- Recherche dans 3 tables : `administrateurs`, `professeurs`, `etudiants`
- Détection automatique du rôle

- Redirection vers interface appropriée

## 8.2 Gestion des Sessions

### 8.2.1 Table Sessions Sécurisée

- Token unique 64 caractères
- Stockage IP et User-Agent
- Expiration automatique (24h par défaut)
- Nettoyage sessions expirées via procédure planifiée

## 8.3 Audit et Traçabilité

### 8.3.1 Table Logs Connexion

Enregistrement de :

- Date/heure connexion
- Utilisateur (ID, rôle, matricule)
- Adresse IP et navigateur
- Statut : succès ou échec
- Raison échec si applicable

### 8.3.2 Table Logs Optimisation

Traçabilité génération planning :

- Session ID unique
- Temps d'exécution (millisecondes)
- Nombre examens générés
- Conflits initial vs final
- Taux occupation salles
- Paramètres utilisés (JSON)

## 9 Tests et Validation

### 9.1 Jeu de Données de Test

#### 9.1.1 Dataset Minimal

Pour validation fonctionnelle :

- 7 départements
- 7 formations (1 par département)
- 10 étudiants
- 7 professeurs
- 7 modules
- 6 salles/amphithéâtres

#### 9.1.2 Dataset Réaliste (Optionnel)

Pour tests de performance :

- 7 départements

- 200+ formations
- 13 000 étudiants
- 500 professeurs
- 1 400 modules (7 par formation en moyenne)
- 130 000 inscriptions
- 50 lieux d'examen

## 9.2 Scénarios de Test

### 9.2.1 Tests Fonctionnels

1. **Authentification :**
  - Connexion admin réussie
  - Connexion professeur réussie
  - Connexion étudiant réussie
  - Rejet identifiants invalides
2. **Génération Planning :**
  - Génération 50 examens sans conflit
  - Génération avec sélection modules personnalisée
  - Détection conflit salle
  - Détection conflit surveillant
3. **Consultation :**
  - Filtrage par département
  - Filtrage par formation
  - Recherche textuelle
  - Export CSV

### 9.2.2 Tests de Performance

- Génération 200 examens < 45 secondes
- Requête consultation < 100ms
- Chargement tableau de bord < 500ms
- Détection conflits sur 1000 examens < 5 secondes

## 9.3 Résultats

**Tous les tests fonctionnels : PASS Benchmarks performance :**

- Génération 50 examens : 15s (objectif < 45s)
- Génération 100 examens : 32s (objectif < 45s)
- Requête stats globales : 42ms
- Chargement planning : 180ms

## 10 Améliorations Futures

### 10.1 Fonctionnalités Additionnelles

- **Notifications automatiques :**
  - Email/SMS rappel 48h avant examen

- Alerte changements planning
- **Génération PDF avancée :**
  - Planning individualisé par étudiant
  - Planning salle avec QR code
- **Gestion absences :**
  - Enregistrement absences en temps réel
  - Rattrapage automatique
- **Optimisation avancée :**
  - Algorithme génétique pour plannings complexes
  - Machine learning : prédiction conflits

## 10.2 Optimisations Techniques

- Migration vers PostgreSQL pour fonctionnalités avancées
- Cache Redis pour requêtes fréquentes
- API REST pour intégration applications mobiles
- Containerisation Docker pour déploiement simplifié

# 11 Conclusion

## 11.1 Bilan du Projet

Ce projet a permis de développer une solution complète et fonctionnelle répondant aux problématiques de planification d'examens universitaires à grande échelle.

## 11.2 Objectifs Atteints

- Modélisation base de données relationnelle normalisée
- Implémentation triggers et vues pour contraintes métier
- Algorithme génération planning < 45 secondes
- Détection automatique conflits multi-niveaux
- Interface web professionnelle multi-rôles
- Système authentification et audit complet
- Tests validés sur dataset réaliste

## 11.3 Compétences Acquises

- Maîtrise conception base de données complexe
- Utilisation avancée triggers, vues et index
- Développement algorithmes optimisation
- Architecture applicative en couches
- Développement web avec Streamlit
- Gestion projet et travail collaboratif

## 11.4 Perspectives

Cette plateforme constitue une base solide pour une évolution vers un système de gestion académique complet intégrant également :

- Gestion des emplois du temps cours
- Saisie et gestion notes
- Réservation salles et ressources
- Statistiques pédagogiques avancées

*“L’optimisation n’est pas qu’une question de performance, c’est avant tout une question d’efficacité au service de l’utilisateur.”*

## Annexes

### A. Structure Base de Données

Schéma relationnel complet disponible dans fichier `exam_scheduler.sql`

### B. Guide Installation

1. Installer Python 3.10+
2. Cloner le dépôt projet
3. Exécuter : `pip install -r requirements.txt`
4. Importer `exam_scheduler.sql` dans MySQL
5. Configurer connexion dans `config/database.py`
6. Lancer : `streamlit run app.py`
7. Accéder : `http://localhost:8501`

### C. Comptes de Test

Rôle	Matricule	Nom Prénom
Administrateur	ADM001	Durand Robert
Doyen	ADM002	Lefevre Catherine
Chef Département	PROF001	Martin Jean
Étudiant	ETU001	Dupont Jean

### D. Références

- MySQL Documentation : <https://dev.mysql.com/doc/>
- Streamlit Docs : <https://docs.streamlit.io/>
- Python Pandas : <https://pandas.pydata.org/docs/>
- Plotly : <https://plotly.com/python/>