**République Algérienne Démocratique et Populaire**

TAGDUDA TAMEGDAYT TAƐERRABT TAYERRAYT

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

AZELIF N USELIG AZELLAY AKE-D UNADI AMASSAN

---


**Université Abderrahmane Mira – Béjaïa**
**Faculté des Sciences Exactes**
**Département Informatique**

# Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master professionnel en Informatique
Option : Réseaux et Sécurité

*en vue De l'obtention du diplôme Master en informatique*
*option : Réseaux et sécurité*

## *Thème*

---

# Conception et réalisation d'un système de gestion pour le département informatique

---

**Réalisé par :**

Mr. BENIDIRI Nordine

**Devant le jury composé de :**

| | | |
|---|---|---|
| **Président** | Mr. AKILAL Karim | U.A/Mira Béjaïa |
| **Examinatrice** | Mme. YESSAAD Nawal | U.A/Mira Béjaïa |
| **Examinatrice** | Mme. HAMZA Lamia | U.A/Mira Béjaïa |
| **Examinatrice** | Mme. TASSOULT Nadia | U.A/Mira Béjaïa |
| **Encadrant** | Mr. DJEBARI Nabil | U.A/Mira Béjaïa |

---

**2024 / 2025**

# Dédicace

*Louange à Dieu Tout-Puissant, qui m'a permis de voir ce jour tant attendu et d'accomplir ce travail avec foi et persévérance.*

*Je dédie ce mémoire à mes très chers parents, véritables piliers de ma vie, dont l'amour, le soutien indéfectible et les sacrifices m'ont toujours porté vers le meilleur. Que Dieu vous protège et vous garde à mes côtés.*

*À mes frères et sœurs, ainsi qu'à tous les membres de ma famille, pour leur présence chaleureuse et leurs encouragements constants.*

*À mes amis, fidèles compagnons de route, pour leur soutien moral et leur confiance tout au long de ce parcours.*

*Je dédie également ce travail à tous mes enseignants, qui m'ont transmis leur savoir avec passion, et au personnel administratif pour leur aide, leur disponibilité et leur bienveillance.*

*Enfin, à toutes les personnes qui me sont chères, de près ou de loin – cette réussite est aussi la vôtre.*

# Remerciements

*Louange à **Dieu**, Tout-Puissant et Miséricordieux, pour m'avoir accordé la force, la patience et la volonté nécessaires afin de mener à bien ce travail. C'est avec une profonde gratitude que je reconnais Son soutien constant tout au long de cette aventure académique et personnelle.*

*Je tiens à exprimer ma reconnaissance la plus sincère à mon encadrant, **M. DJEBARI Nabil** pour sa disponibilité, ses conseils avisés, son accompagnement bienveillant et son implication continue. Son soutien a été d'une grande valeur dans l'élaboration de ce mémoire.*

*Je remercie également les membres du jury, **M.AKILAL Karim, Mme. YESSAAD Nawal, Mme. HAMZA Lamia et Mme. TASOULT Nadia**, pour le temps qu'ils ont consacré à l'évaluation de ce travail, ainsi que pour leurs observations constructives.*

*Je suis profondément reconnaissant envers ma famille et mes parents pour leur soutien moral, leur patience et leur confiance. Leur présence et leurs encouragements m'ont permis d'avancer avec sérénité.*

*Enfin, je remercie mes amis, proches ou lointains, pour leur appui constant, leur écoute et leur soutien durant cette étape importante de ma vie.*

# Table des matières

# Table des figures

# Liste des tableaux

# Acronymes

**AGs** Algorithmes génétiques

**API** Application Programming Interface

**CSS** Cascading Style Sheets

**HTML** HyperText Markup Language

**JSX** JavaScript XML

**PHP** Hypertext Preprocessor

**UML** Unified Modeling Language

**UP** Unified Process

# Introduction Générale

A l'heure actuelle, l'avancement de la technologie a profondément transformé les méthodes de gestion et d'organisation dans tous les secteurs, y compris celui de l'enseignement supérieur. Actuellement, l'informatique occupe une place essentielle dans l'amélioration de l'efficacité des établissements universitaires, leur permettant de mieux gérer leurs données, de gagner du temps, et de réduire les erreurs liées aux traitements manuels.

Dans ce contexte, les départements universitaires sont appelés à moderniser leur fonctionnement afin de mieux répondre aux besoins d'une population estudiantine en constante croissance. Parmi les tâches essentielles qui nécessitent une amélioration, figure la gestion des emplois du temps des cours, ainsi que le suivi des absences des étudiants et des enseignants.

Dans ce mémoire, nous partons avec l'objectif de concevoir et de réaliser une solution pour automatiser la gestion des emplois du temps et des absences dans le département Informatique de l'université de Bejaia. Pour structurer ce projet qui est confié dans le cadre d'un mémoire de fin d'études, on a choisi de suivre la méthode du Processus Unifié (UP), connue pour sa rigueur et sa capacité à guider les projets logiciels de manière itérative et efficace.

Le système qu'on a réalisé permet d'automatiser plusieurs tâches du département, notamment :

- **Pour l'administrateur :**
  - La génération automatique des emplois du temps.
  - La gestion des ressources pédagogiques : enseignants, étudiants, salles, l'enseignement, etc.
  - Le suivi et la validation des absences des étudiants.
  - La personnalisation des emplois du temps et la résolution des conflits horaires.
- **Pour l'enseignant :**
  - La consultation de son emploi du temps personnel.
  - La gestion de ses indisponibilités.

— **Pour l'étudiant :**

  — La consultation de son emploi du temps.
  — La justification de ses absences en ligne.

Pour cela, on a organisé ce rapport comme suit :

Dans le premier chapitre intitulé « Étude préalable », on présente le contexte de la gestion universitaire et les solutions existantes, tout en identifiant les limites des méthodes actuelles utilisées dans notre département.

Dans le deuxième chapitre intitulé « Analyse des besoins », on procède à une modélisation UML à l'aide de diagrammes de cas d'utilisation et de diagrammes de séquence afin de décrire les interactions entre les acteurs et le système.

Dans le troisième chapitre « Conception », on développe l'architecture du système à l'aide de diagrammes de classe et d'un schéma relationnel complet. On présente également l'algorithme génétique utilisé pour la génération optimisée des emplois du temps dans se système.

Dans le dernier chapitre intitulé « Réalisation », on détaille les outils, les langages et les technologies utilisés pour implémenter la plateforme, en présentant des captures d'écran illustrant les interfaces principales de l'application.

Enfin, on clôture ce mémoire par une conclusion générale, dans laquelle nous revenons sur les résultats obtenus et proposons quelques perspectives d'amélioration pour l'avenir.

# Chapitre 1

# Étude préalable

## 1.1 Introduction

Toute analyse doit commencer par l'étude de ce qui existe. L'analyse de l'existant ou l'étude préalable constitue une étape fondamentale dans l'analyse des besoins. Dans ce chapitre, on présentera l'étude préalable qui a été réalisée pour le projet « système de gestion pour le département informatique ».

Le but de ce projet est d'automatiser la gestion des emplois du temps et le suivi des absences qui se fait actuellement de manière manuelle. Nous allons commencer par la problématique et les raisons pour lesquelles ce système est devenu une nécessité pour notre département. Puis, nous allons analyser le système existant et identifier ses limites. Enfin, nous allons définir les besoins et présenter la solution proposée.

## 1.2 Problématique

Dans la majorité des universités algériennes, la gestion des emplois du temps et le suivi des absences représentent un véritable défi pour l'administration. Le département d'informatique n'échappe pas à cette réalité et se heurte à de nombreuses difficultés liées à l'utilisation de méthodes manuelles et dépassées. Parmi les problèmes rencontrés, on peut citer :

— La planification et l'organisation des emplois du temps.
— Le suivi des absences des étudiants.
— La gestion des ressources (salles, enseignants, étudiants, leçons).
— La gestion des enseignants et la prise en compte de leurs disponibilités.

Face à ces limitations, nous avons choisi de travailler sur la conception et la mise en place d'un système de gestion automatisé, spécifiquement adapté aux besoins d'un département universitaire.

## 1.3 Étude de l'existant

Comme mentionné précédemment, notre département utilise actuellement des méthodes de gestion anciennes qui ne répondent plus aux exigences actuelles. Par exemple, les emplois du temps sont générés à l'aide de l'application **OpenCTT**, tandis que le suivi des absences se fait manuellement et en présentiel, à l'aide de feuilles de présence et de stylos.

— **OpenCTT :**
Open Course Timetabler (OpenCTT) est une application libre et open source utilisée pour générer les emplois du temps dans les établissements scolaires et universitaires. Elle est développée en langage C# et fonctionne sous l'environnement Windows. L'outil est disponible en téléchargement via le lien suivant :
https://sourceforge.net/projects/openctt/

## 1.4 Évaluation détaillée de l'application « OpenCTT »

Dans cette section, on va procéder à une évaluation de l'application OpenCTT, le système actuellement utilisé pour la gestion des emplois du temps dans notre département.

**Le tableau 1.1** résume les résultats de l'évaluation, Cette analyse permettra d'identifier ses points forts et ses insuffisances, afin de s'inspirer pour concevoir un système plus adapté.

TABLE 1.1 – Évaluation de l'application OpenCTT

| **Avantage** | **Insuffisances** |
|---|---|
| • Logiciel gratuit et open source. | • Application desktop uniquement, nécessitant une installation sur chaque poste. |
| • Prise en compte des contraintes de base (disponibilité des enseignants, capacité des salles). | • Absence totale de fonctionnalités web, impossible d'accéder aux emplois du temps à distance. |
| • Export possible en formats PDF pour impression. | • Pas de gestion d'absences possible. |
| • Stabilité du logiciel pour les opérations basiques. | • Interface utilisateur dépassée (design des années 2000), peu intuitive et démotive les utilisateurs. |
| | • Pas de synchronisation en temps réel, pas de possibilité de collaboration sur la gestion. |

## 1.5 Solution proposée

Pour remédier aux problèmes rencontrés par le département, on a proposé la mise en place d'un système de gestion, qui est capable de couvrir :

— La génération automatique des emplois du temps et la liberté de modification sous une interface fluide et intuitif.
— La gestion automatisée des absences.
— La gestion des ressources (salles, enseignant, leçons, étudiants).
— L'accès aux emplois du temps et aux absences à distance par le département, les étudiants et les enseignants.
— L'envoi par mail des emplois du temps.
— Possibilités de collaborations pour la gestion des emplois du temps et des absences par les administrateurs du système.

## 1.6 Méthodologie de développement

Pour mener à bien le projet du système de gestion, il faut choisir une méthodologie de développement adaptée. Dans cette section on vas présenter une comparaison des principales méthodes utilisées dans le développement logiciel, afin de sélectionner celle qui correspond à nos besoins.

### 1.6.1 Comparaison entre les méthodes de développement logiciel

Le tableau I.2, présente une comparaison entre les différentes méthodes de développement logiciel telles que UP, XP et Scrum [1].

TABLE 1.2 – Comparaison entre les méthodes de développement logiciel [1]

| **Critères** | **UP** | **XP** | **Scrum** |
|---|---|---|---|
| Structure | Une structure plus formelle et plus définie. | Une structure moins formelle. | Un Framework structuré avec des rôles bien définis (Product Owner, Scrum Master, Équipe) et des événements (sprints, revues de sprint, réunions quotidiennes). |
| Approche de développement | L'approche la plus traditionnelle avec une attention particulière qui porte sur l'analyse, sur la conception et sur la modélisation. | Se concentre sur des pratiques telles que la programmation en binôme, les tests automatisés et une conception émergente pour faciliter le développement et la livraison de fonctionnalités de haute qualité. | Se concentre sur des itérations (sprints) pour développer et livrer des fonctionnalités, avec une priorisation claire du Product Owner. |
| Gestion du temps | Pas de durée fixe pour les itérations, mais peut avoir des phases avec des jalons spécifiques pour gérer le temps. | Utilise des itérations courtes (1-3 semaines) pour gérer le temps et des livraisons rapides. | Utilise des Sprints de durée fixe (généralement 2-4 semaines). |
| Interaction avec le client | Implique une interaction avec le client, principalement lors des phases d'analyse et de validation. | Une collaboration continue avec le client, favorisant une communication constante et des feedbacks réguliers. | Interaction régulière via le Product Owner et des événements tels que les revues de sprint. |

### 1.6.2 Choix de la méthodologie et des outils de modélisation

Après l'analyse et la comparaison des différentes méthodes de développement logiciel, Le choix était évident vu que **UML** et le processus **UP** sont les plus utilisés dans ce genre de projet, car cette combinaison offre la structure et la rigueur nécessaires pour un projet de cette envergure.

Avant de commencer le projet, on présente dans les sections suivantes un bref aperçu de ces deux outils.

#### 1.6.2.1 Langage UML

L'UML est un langage de modélisation graphique et textuelle largement utilisé. Ce langage est désormais la référence en modélisation objet, ou programmation orientée objet. Son objectif est de représenter des éléments du monde réel ou virtuel en utilisant des entités informatiques appelées "objets". L'utilisation de l'UML permet de comprendre et décrire les besoins, spécifier et documenter les systèmes, concevoir des solutions, et communiquer les différents aspects d'un système d'information [2].

#### 1.6.2.2 La méthode UP

La méthode du Processus Unifié est un guide méthodologique qui aide à la réalisation de logiciels en conseillant et en pilotant l'équipe dans ses différentes activités afin de réduire la complexité des projets. Ce processus permet de clarifier les rôles et les responsabilités de chacun, permettant ainsi à chaque membre de l'équipe de connaître sa place dans le processus de production du logiciel. Il se caractérise par une approche itérative et incrémentale, basée sur les cas d'utilisation, axée sur la réduction des risques et centrée sur l'architecture logicielle et les modèles UML [3].

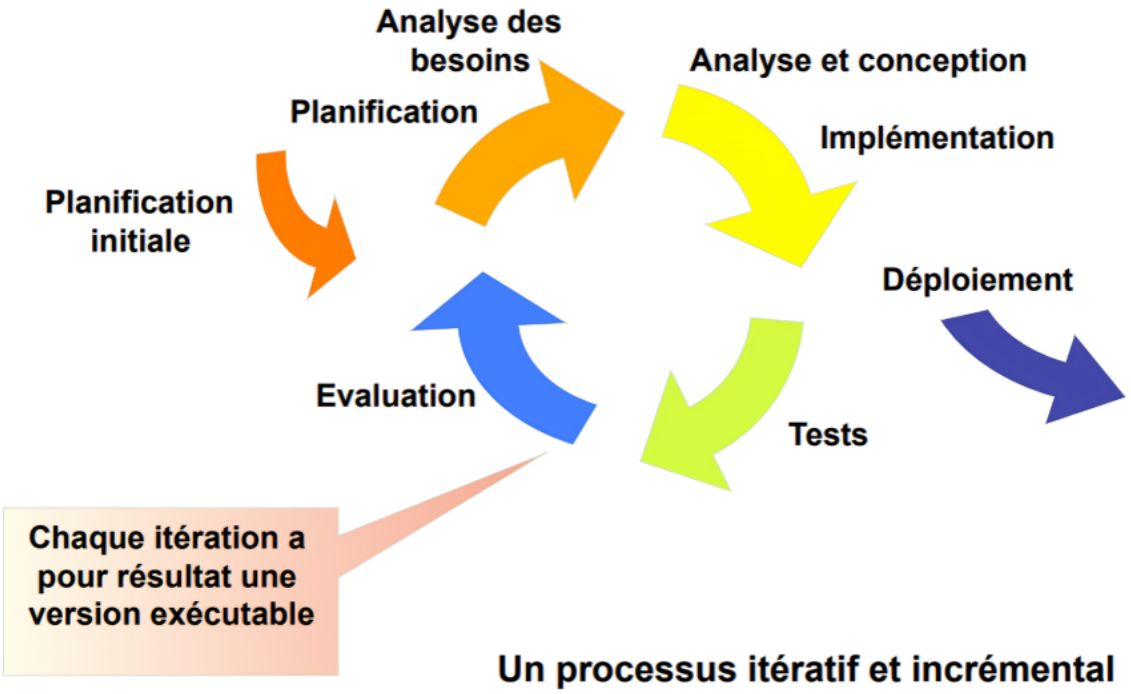


FIGURE 1.1 – Cycle de développement UP. [3]

Le cycle de développement du UP organise les tâches et les itérations en plusieurs étapes, comme illustré dans la Figure I.1 et décrites ci-dessous :

— **Planification initiale** : Cette étape consiste à définir les objectifs principaux du projet, à identifier les acteurs concernés, à estimer les ressources nécessaires et à élaborer une première feuille de route.
— **Analyse des besoins** : Elle permet de préciser la vision du projet en spécifiant les exigences fonctionnelles et techniques, tout en étudiant sa faisabilité.
— **Analyse et conception** : À ce stade, une architecture générale du système est proposée à partir des besoins exprimés. Les modèles de données et les diagrammes de conception sont définis afin de préparer la phase de réalisation.
— **Implémentation** : Cette phase correspond au développement du logiciel. Le code source est produit, les fonctionnalités sont intégrées et des tests unitaires sont réalisés pour assurer la fiabilité de chaque composant.
— **Déploiement** : Le système est installé dans son environnement réel. Cela inclut la configuration des serveurs, la mise en place des paramètres nécessaires et la livraison aux utilisateurs finaux.
— **Vérification et tests** : Des tests approfondis sont effectués pour confirmer que le logiciel répond aux attentes initiales. Ils couvrent les tests unitaires, d'intégration, de système et de performance.

— **Évaluation** : Enfin, une appréciation globale du produit est menée afin de vérifier son comportement dans l'environnement d'utilisation et de dégager d'éventuelles améliorations.

### 1.6.3 Recueil des besoins

Le recueil des besoins est le processus de collecte et d'analyse des exigences d'un projet ou d'un système, afin de comprendre les attentes et les besoins de ce dernier [4].

#### 1.6.3.1 Besoins fonctionnels

Un besoin fonctionnel décrit ce qu'un système est censé faire, autrement dit ses fonctions principales. Il s'agit de déterminer les fonctionnalités spécifiques que ce dernier doit fournir pour répondre aux besoins de ces utilisateurs. Notre système doit répondre aux besoins fonctionnels suivants :

— **Pour les administrateurs :**
  — Génération automatique des emplois du temps.
  — Modification des emplois du temps d'une façon libre.
  — Gestion des absences et impression des justifications.
  — Partage des emplois du temps par Mail.
  — Gestion des ressources (salles, étudiants, enseignants et leçons).
  — Collaboration sur la gestion entre les administrateurs.
— **Pour les enseignants :**
  — Consultation de son emploi du temps personnel.
  — Gestion de ses indisponibilités.
  — Consultation de son profil.
— **Pour les étudiants :**
  — Consultation de son emploi du temps.
  — Gestion de ses absences.
  — Consultation de son profil.

#### 1.6.3.2 Besoins non fonctionnels

Les besoins non fonctionnels caractérisent le système, ce sont des critères de qualité qui permettent de garantir la cohérence du système dans son ensemble et d'assurer sa fiabilité et sa sécurité. Dans le cadre de notre système celui-ci doit répondre aux besoins suivants :

- **Sécurité :**
  - L'accès au système doit être protégé par une authentification forte pour chaque type d'utilisateur.
  - Les mots de passe doivent être cryptés dans la base de données.
  - Chaque utilisateur (étudiant, enseignant, administrateur) doit avoir un compte avec un rôle.
  - Les étudiants ne peuvent consulter que leurs propres emplois du temps et absences.
- **Fiabilité :**
  - En cas de panne serveur, les données ne doivent pas être perdues.
  - Le système doit fonctionner 24h/24 et 7j/7.
- **Performance et efficacité :**
  - Le temps de réponse du système doit être précis et acceptable.
  - Un contrôle doit être effectué sur les champs de saisie pour éviter l'introduction d'informations incohérentes.
- **Utilisabilité et ergonomie :**
  - La plateforme doit être simple à utiliser.
  - Les plateformes des enseignants et des étudiants doivent être responsive (adapté aux mobiles, tablettes et ordinateurs).

## 1.7 Conclusion

Dans ce chapitre, on a présenté les problèmes liés à la gestion du département informatique, en se concentrant sur les caractéristiques qui ne sont pas prises en compte par les outils déjà en place. Après la réalisation de cette analyse, on a aussi pu définir quelques exigences pour optimaliser les solutions possibles et réduire les problèmes identifiés. Ensuite on a fait une comparaison entre les méthodes de développement logiciel pour une meilleure conception. Enfin, un recueil des besoins qui se résume dans les besoins fonctionnels et non fonctionnels de notre système.

Dans le prochain chapitre, nous pouvons passer à l'étape suivante qui sera consacrée à l'analyse approfondie de ces besoins.

# Chapitre 2

# Analyse des besoins

## 2.1 Introduction

Ce chapitre contient l'analyse des besoins qui est basée sur l'identification des différents acteurs du système, le diagramme de contexte. On y définit des cas d'utilisation qui seront modélisés par un diagramme de cas d'utilisation avec la description textuelle de chaque cas d'utilisation et les diagrammes de séquence système.

## 2.2 Identification des Acteurs

Un acteur, est celui qui déclenche l'activité afin de remplir ses obligations métiers, c'est celui qui interagit directement avec le système. Ainsi, dans notre système, nous distinguons les acteurs suivants :

1. **Administrateur :** peut faire la gestion de toutes les données dans l'application comme les étudiants, les enseignants, les salles, les leçons, les absences, les emplois du temps et modifier son mot de passe.
2. **Enseignant :** peut consulter son emploi du temps personnel, peut insérer des contraintes d'indisponibilité, consulter son profil et modifier son mot de passe.
3. **Etudiant :** Peut consulter son emploi du temps et son profil, justifier ses absences et modifier son mot de passe.

## 2.3 Identification des cas d'utilisations

Les cas d'utilisation représentent les différentes interactions entre les acteurs et le système, où ce dernier réagit en accomplissant une série d'actions spécifiques. Dans le cadre de ce projet, on peut identifié les principaux cas d'utilisation associés à chaque acteur, qui seront représentés dans le tableau 2.1.

TABLE 2.1 – Identification des cas d'utilisation

| **Acteur** | **Les cas d'utilisation** | **Description** |
|---|---|---|
| Admin | S'authentifier | |
| | Gérer les années scolaires et les semestres | • Ajouter une nouvelle année scolaire.<br>• Activer une année scolaire, supprimer une année scolaire.<br>• Changer de semestre courant (passer du premier au deuxième semestre). |
| | Gérer les ressources (Étudiant, Enseignant, Leçons, Salles) | • Ajouter et supprimer et rechercher une ressource.<br>• Télécharger les exemplaires de fichier CSV pour importation.<br>• Importer une liste sous forme d'un fichier CSV. |
| | Génération des emplois du temps | Générer des emplois du temps pour toutes les spécialités et les niveaux enregistrés dans la base de données avec un seul clic. |
| | Consulter et modifier les emplois du temps | Consulter les emplois selon la section choisie, avec la possibilité de faire des modifications. |
| | Gérer les absences des étudiants | Accorder ou rejeter une absence, consulter la justification de l'étudiant, et imprimer la fiche d'absence générée par le système. |
| | Profil admin | Consulter ses informations et modifier son mot de passe. |

*Suite à la page suivante...*

| **Acteur** | **Les cas d'utilisation** | **Description** |
|---|---|---|
| Enseignant | S'authentifier | |
| | Emploi du temps Enseignant | Consulter et Exporter en PDF son emploi du temps. |
| | Gérer ses contraintes | Consulter, ajouter et supprimer des contraintes. |
| | Profil Enseignant | Consulter ses informations et modifier son mot de passe. |
| Etudiant | S'authentifier | |
| | Emploi du temps etudiant | Consulter et Exporter en PDF son emploi du temps. |
| | Gérer ses absences | Justifier, supprimer une absence et consulter la réponse du département. |
| | Profil Étudiant | Consulter ses informations et modifier son mot de passe. |

## 2.4 Diagrammes de cas d'utilisation

Un diagramme de cas d'utilisation est un type de diagramme UML utilisé pour modéliser le comportement fonctionnel d'un système en mettant en évidence les interactions entre les utilisateurs (appelés acteurs) et le système lui-même, à travers des scénarios appelés "cas d'utilisation" [5].

### 2.4.1 Diagrammes de cas d'utilisation « Administrateur »

Le diagramme de la Figure 2.1 représente les tâches spécifiées à un administrateur dans le système :

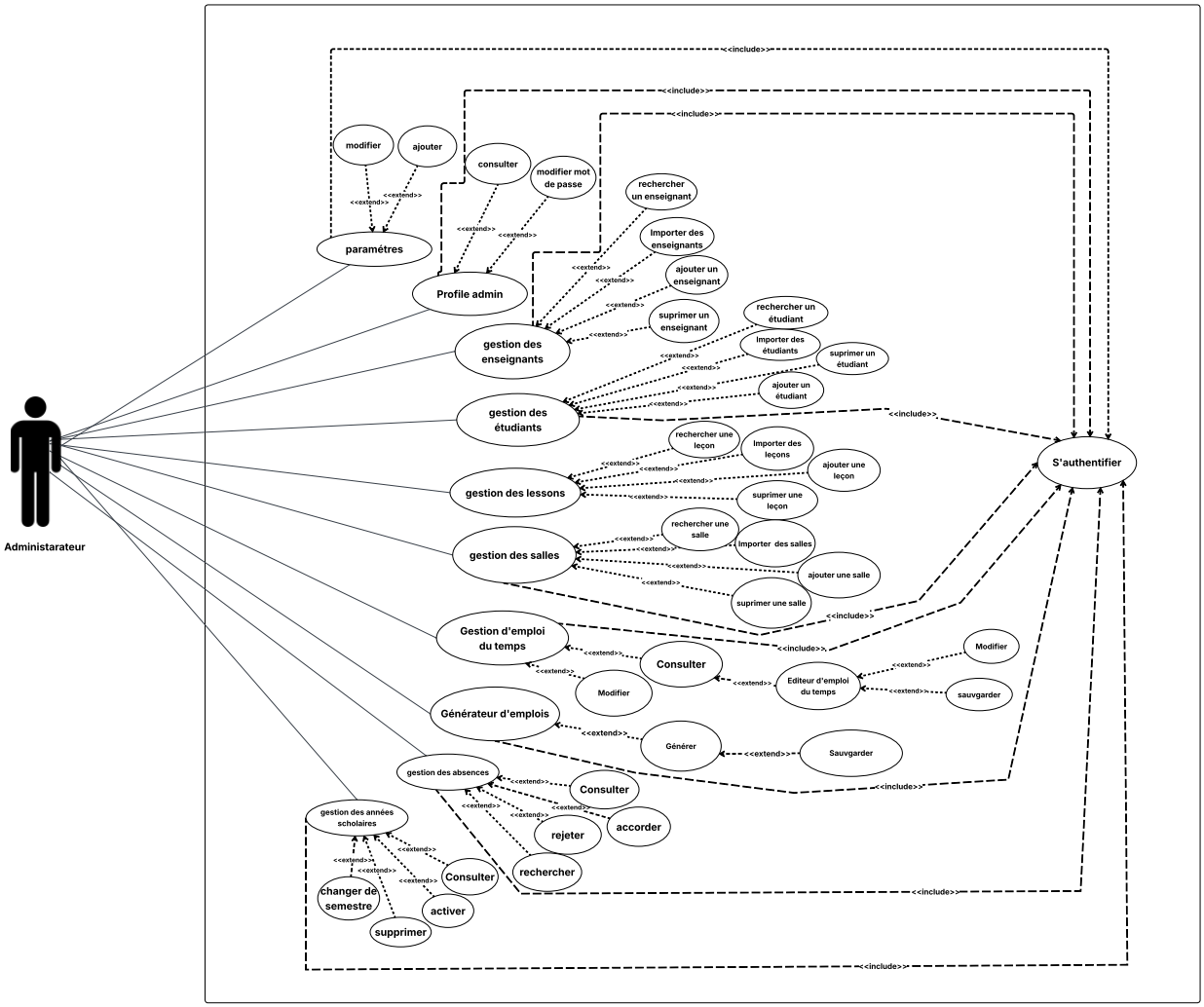


FIGURE 2.1 – Diagramme de cas d'utilisation « Administrateur »

### 2.4.2 Diagrammes de cas d'utilisation « Enseignant »

Le diagramme de la Figure 2.2 représente les tâches spécifiées à un Enseignant dans le système :

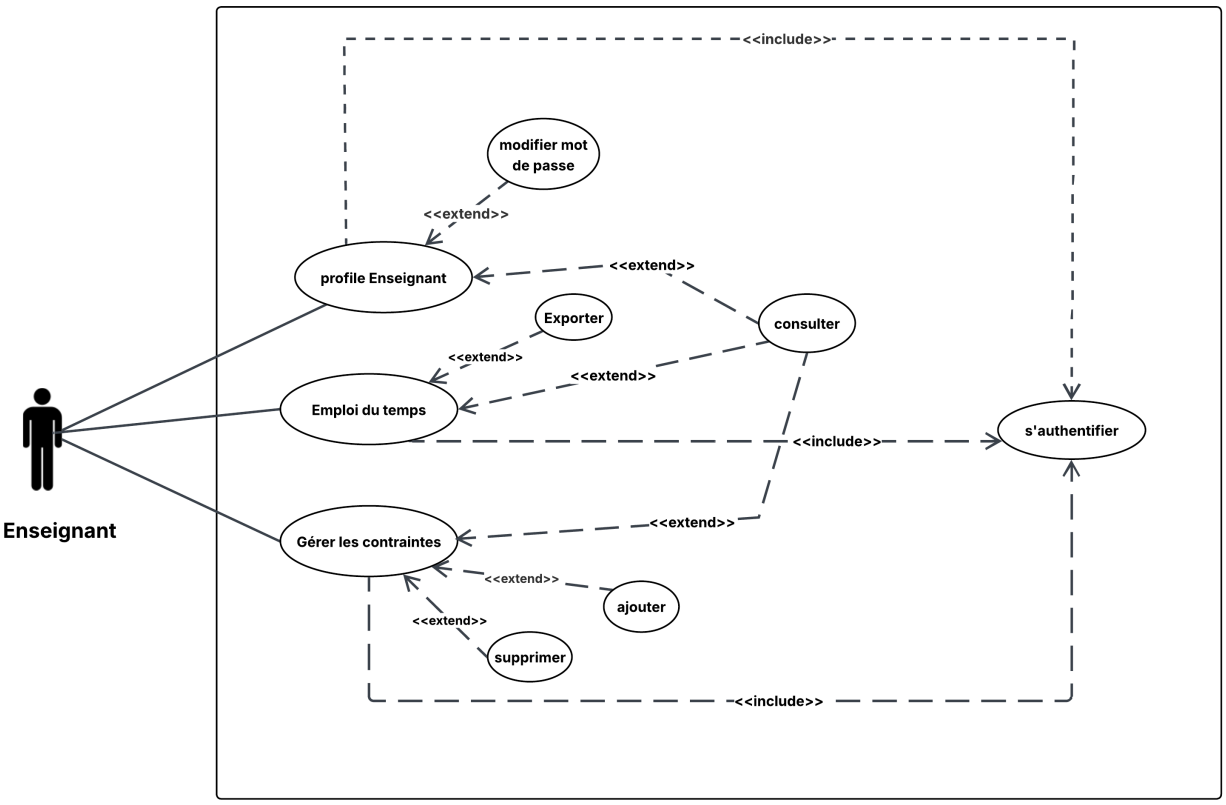


FIGURE 2.2 – Diagramme de cas d'utilisation « Enseignant »

### 2.4.3 Diagrammes de cas d'utilisation « Étudiant »

Le diagramme de la Figure 2.3 représente les tâches spécifiées à un Étudiant dans le système

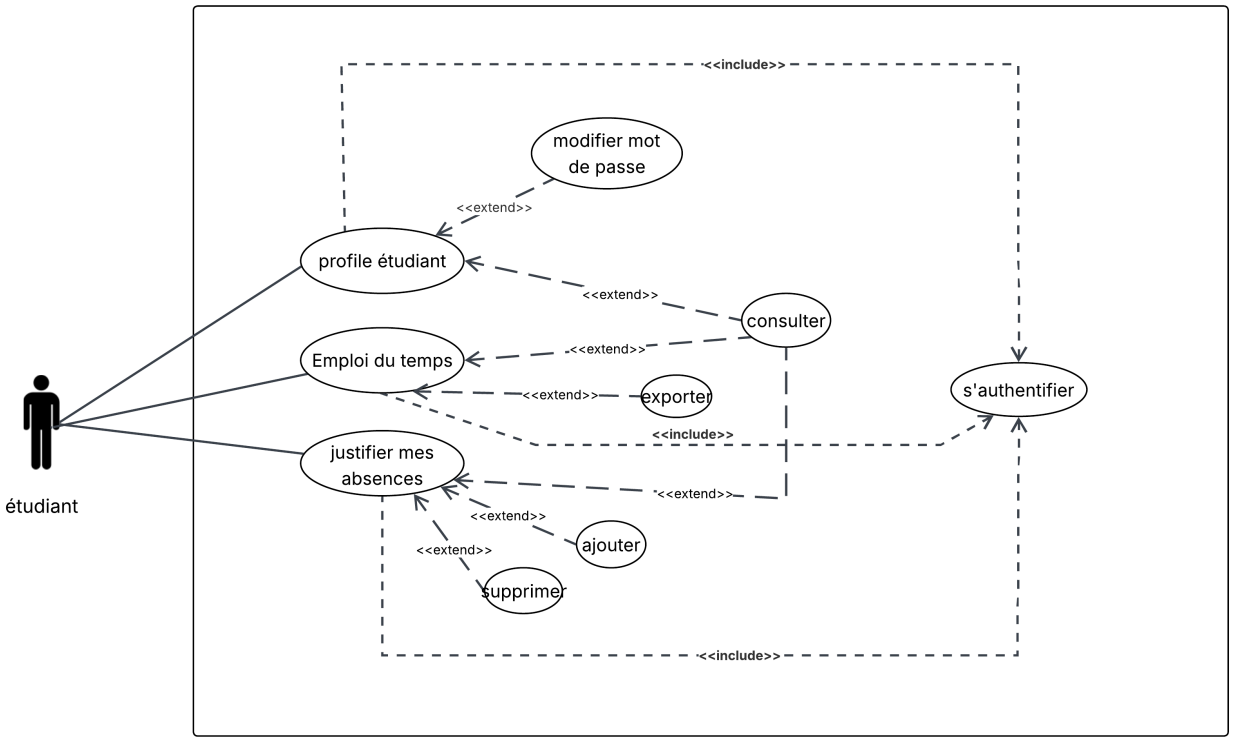


FIGURE 2.3 – Diagramme de cas d'utilisation « Étudiant »

## 2.5 Description textuelle des cas d'utilisation

La description textuelle consiste en un ensemble de scénarios décrivant des situations de succès ou d'échec qui représentent une séquence spécifique d'actions se déroulant du début à la fin d'un cas d'utilisation. Elle détaille comment un acteur spécifique utilise le système pour atteindre un objectif donné [6]. Dans cette section, nous allons donner une description textuelle pour chaque cas d'utilisation identifié dans notre système.

### 2.5.1 Description textuelle des cas d'utilisation « s'authentifier »

La description textuelle de l'un des principaux cas d'utilisation qui concerne tous les agents du système « s'authentifier » est présenté dans le Tableau 2.2.

TABLE 2.2 – Description textuelle du cas d'utilisation « s'authentifier »

| | |
|---|---|
| **Cas d'utilisation** | S'authentifier |
| **Acteur** | Administrateur, Enseignant, Étudiant |
| **Objectif** | Permet à chaque acteur d'accéder à sa propre interface. |
| **Pré condition** | L'acteur possède un compte. |
| **Post condition** | Permet à l'acteur d'accéder à sa propre interface. |
| **Scénario nominal** | 1. Le système affiche la page d'authentification.<br>2. L'acteur saisie ses identifiants (Email, mot de passe).<br>3. Le système vérifie la validité des identifiants.<br>4. L'acteur se redirige vers sa propre interface. |
| **Scénario alternatif** | Identifiants erronés ou inexistants, le système renvoie un message d'erreur. |

### 2.5.2 Description textuelle des cas d'utilisation « Gestion des enseignants »

La description textuelle du cas d'utilisation « Gestion des enseignant » est présenté dans le Tableau 2.3.

TABLE 2.3 – Description textuelle du cas d'utilisation « Gestion des enseignants »

| | |
|---|---|
| **Cas d'utilisation** | Gestion des enseignants |
| **Acteur** | Administrateur |
| **Objectif** | Permet à l'acteur consulter la liste des enseignants, d'ajouter ou de supprimer un enseignant, télécharger le fichier modèle d'importation au format CSV et d'importer une liste d'enseignants à partir de ce dernier. |
| **Pré condition** | Accéder à l'espace admin. |
| **Post condition** | Permet à l'acteur de gérer les enseignants. |
| **Scénario nominal** | 1. L'acteur demande d'accéder à la page Gestion d'enseignant.<br>2. Le système le redirige vers cette page.<br>3. L'acteur choisit « ajouter un enseignant ».<br>4. Le système lui affiche le formulaire d'ajout.<br>5. L'acteur remplit le formulaire et valide.<br>6. L'acteur choisit « supprimer un enseignant ».<br>7. Le système affiche la confirmation de suppression.<br>8. L'acteur confirme la suppression.<br>9. L'acteur choisit « importer un fichier ».<br>10. Le système lui affiche ses documents pour choisir le fichier.<br>11. L'acteur confirme son choix et lance l'importation.<br>12. L'acteur choisit « Télécharger le modèle ».<br>13. Le système lance le téléchargement.<br>14. L'acteur saisit un texte sur le champ de recherche.<br>15. Le système lance une recherche et lui rend le résultat. |
| **Scénario alternatif 1** | Si le fichier ne respecte pas le format de données :<br>Fichier incompatible, le système affiche un message d'erreur « erreur lors de l'importation ». |
| **Scénario alternatif 2** | Si aucun résultat n'a été trouvé lors de la recherche :<br>Le système affiche un message « aucun résultat trouvé ». |
| **Scénario alternatif 3** | Si l'acteur oublie de remplir un champ :<br>Le système affiche un message d'erreur « Veuillez remplir tous les champs ». |

### 2.5.3 Description textuelle des cas d'utilisation « Génération d'emploi du temps »

La description textuelle du cas d'utilisation « Générateur d'emploi du temps » est présenté dans le Tableau 2.4.

TABLE 2.4 – Description textuelle du cas d'utilisation « Génération d'emploi du temps »

| | |
|---|---|
| **Cas d'utilisation** | Génération d'emploi du temps |
| **Acteur** | Administrateur |
| **Objectif** | Permet à l'acteur de générer des emplois du temps avec un seul clic. |
| **Pré condition** | Accéder à l'espace admin. |
| **Post condition** | Les emplois du temps sont générés et la base de données est mise à jour. |
| **Scénario nominal** | 1. L'acteur demande d'accéder à la page Générateur d'emploi du temps.<br>2. Le système le redirige vers cette page.<br>3. L'acteur clique sur « générer ».<br>4. Le système lance la génération.<br>5. Le système affiche un Pop-Up de confirmation contenant les conflits rencontrés par l'algorithme, lui propose deux choix « sauvegarder », « relancer la génération ».<br>6. L'utilisateur choisit « sauvegarder ».<br>7. Le système enregistre les emplois du temps et les conflits dans la base de données.<br>8. L'utilisateur choisit « relancer la génération ».<br>9. Le système refait le même scénario. |
| **Scénario alternatif** | Si la génération échoue :<br>Le système renvoie un message d'erreur « erreur lors de la génération ». |

## 2.6 Diagrammes de séquence

Un diagramme de séquence Est un type de diagramme UML qui permet de visualiser les scénarios de chaque cas d'utilisation en se concentrant sur l'ordre chronologique des interactions entre les objets [2].

### 2.6.1 Diagrammes de séquence « s'authentifier »

La figure 2.4 présente le diagramme de séquence du cas d'utilisation "s'authentifier".

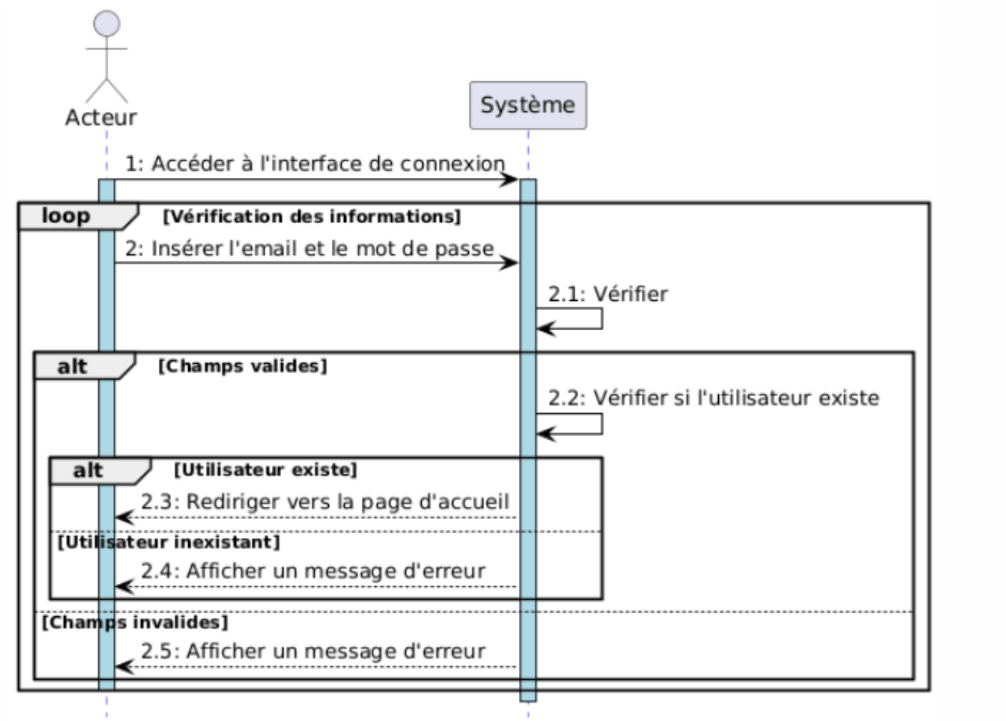


FIGURE 2.4 – Diagramme de séquence « s'authentifier »

### 2.6.2 Diagrammes de séquence « ajouter un enseignant »

La figure 2.5 présente le diagramme de séquence du cas d'utilisation "ajouter un enseignant".

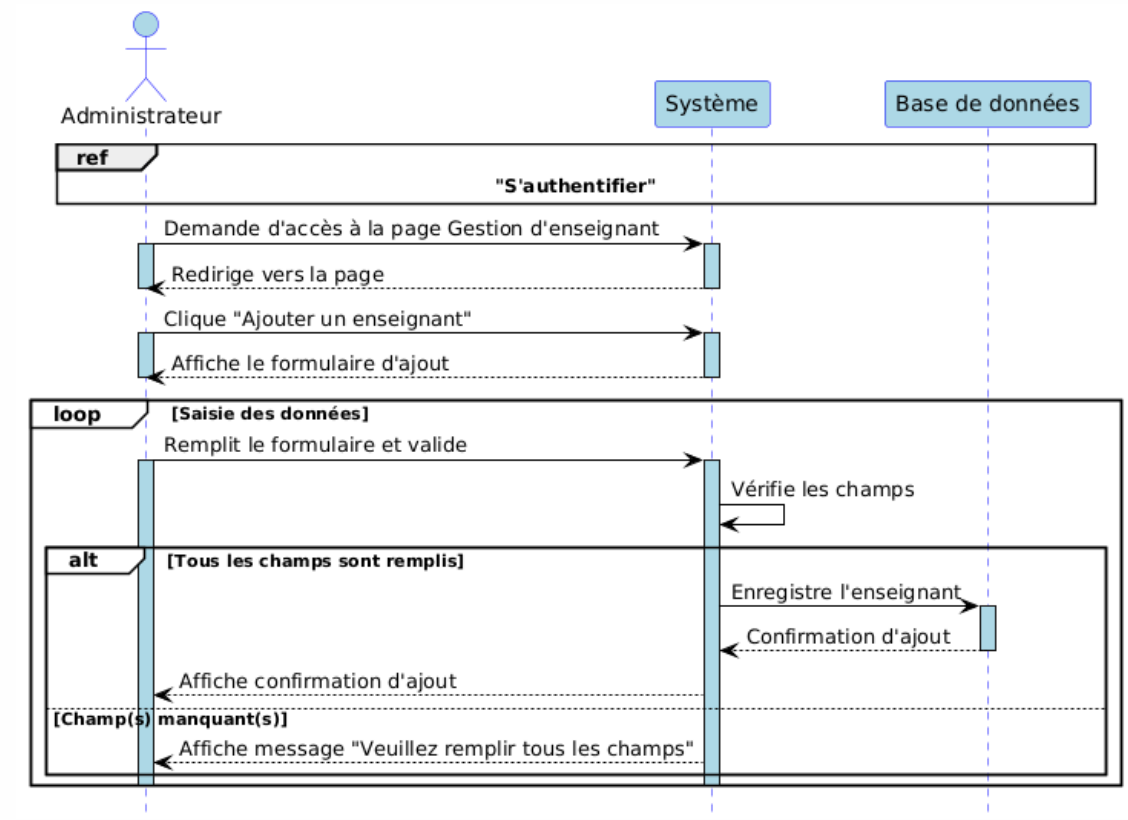


FIGURE 2.5 – Diagramme de séquence « ajouter un enseignant »

### 2.6.3 Diagrammes de séquence « Importer des enseignants »

La figure 2.6 présente le diagramme de séquence du cas d'utilisation "importer une liste d'enseignants".

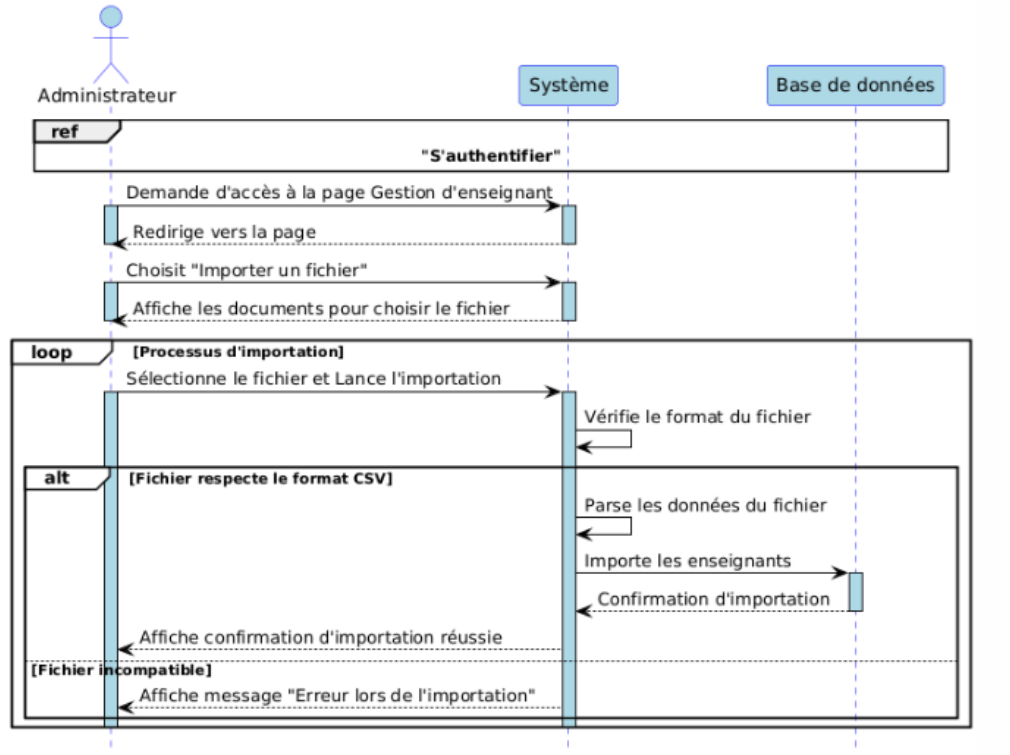


FIGURE 2.6 – Diagramme de séquence « importer des enseignants »

### 2.6.4 Diagrammes de séquence « Génération des emplois du temps »

La figure 2.7 présente le diagramme de séquence du cas d'utilisation génération des emplois du temps.

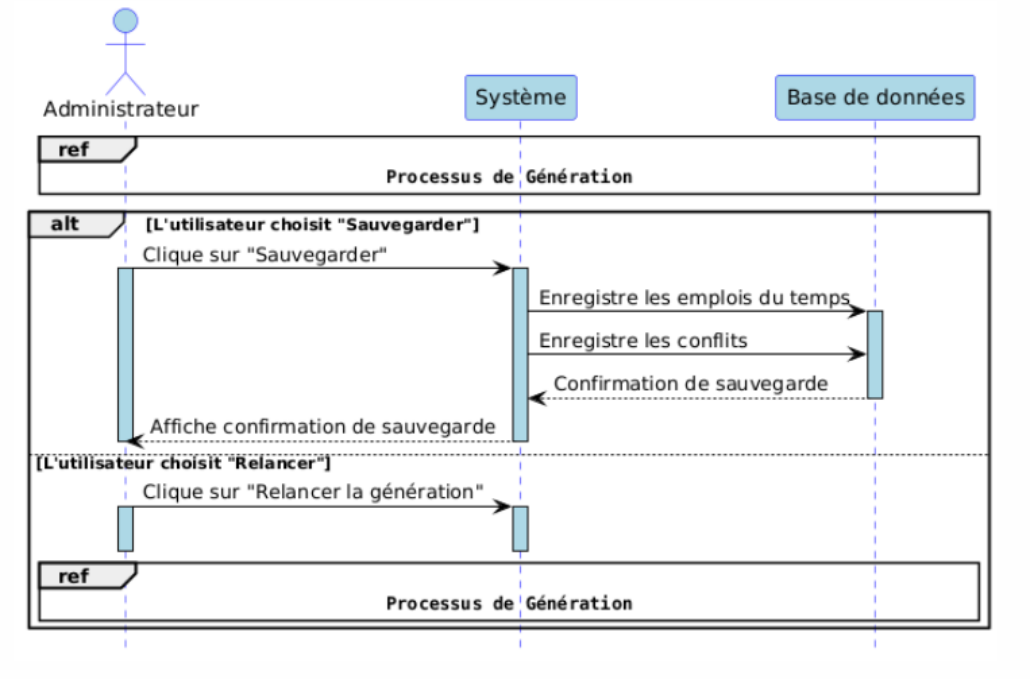


FIGURE 2.7 – Diagramme de séquence « Génération d'emplois du temps »

### 2.6.5 Diagrammes de séquence « Processus génération »

La figure 2.8 présente le diagramme de séquence du cas d'utilisation "processus de génération".
Le Générateur représente un service qui implémente un algorithme génétique pour optimiser la génération des emplois du temps.

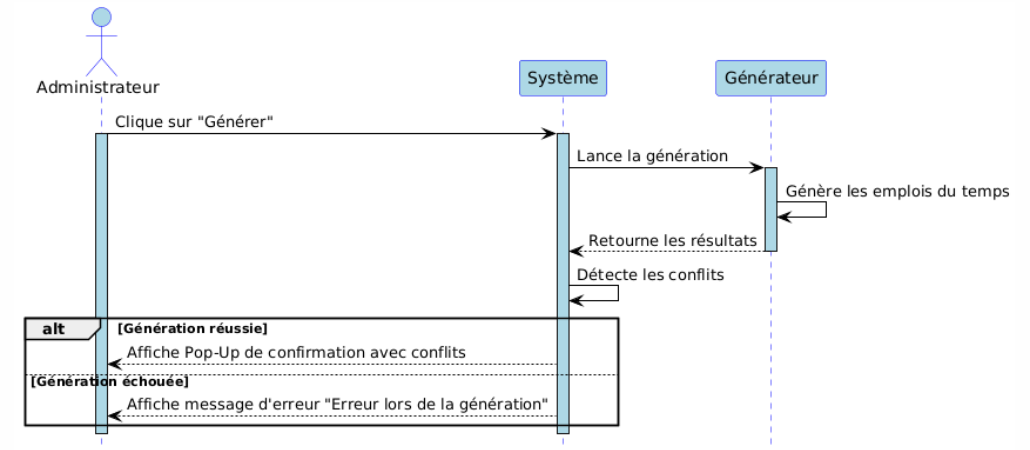


FIGURE 2.8 – Diagramme de séquence « Processus de génération »

## 2.7 Conclusion

Dans le chapitre actuel, j'ai pris le temps d'analyser les besoins de notre système, en m'appuyant sur les diagrammes UML, où j'ai présenté quelques diagrammes du cas d'utilisation et leurs description textuelle avec leurs diagrammes de séquence aussi.

Dans le chapitre suivant, j'ai pour but de décrire notre système de manière plus détaillés.

# Chapitre 3

# Conception

## 3.1 Introduction

Dans ce chapitre, on va entamer la phase de la conception, cette dernière permet de structurer et d'organiser le projet. On commence par la création du diagramme de classes où on vas décrire les déférentes classes, leurs attributs et leurs relations. Après, On passe à la conception du modèle relationnel et on définit la structure de la base de données, les relations entre les tables, ainsi que la description détaillée des attributs des tables de la base de données. Enfin, nous donnons une représentation des algorithmes génétique et leurs utilisation dans notre système.

## 3.2 Diagramme de classe

Un diagramme de classe est un outil de modélisation utilisé pour représenter les composants statiques d'un système logiciel, tels que les classes, les interfaces et les relations entre elles. Il fournit une représentation visuelle des entités du système et de leurs interactions, ce qui permet aux développeurs de comprendre la structure et l'organisation du système avant sa mise en œuvre [7].

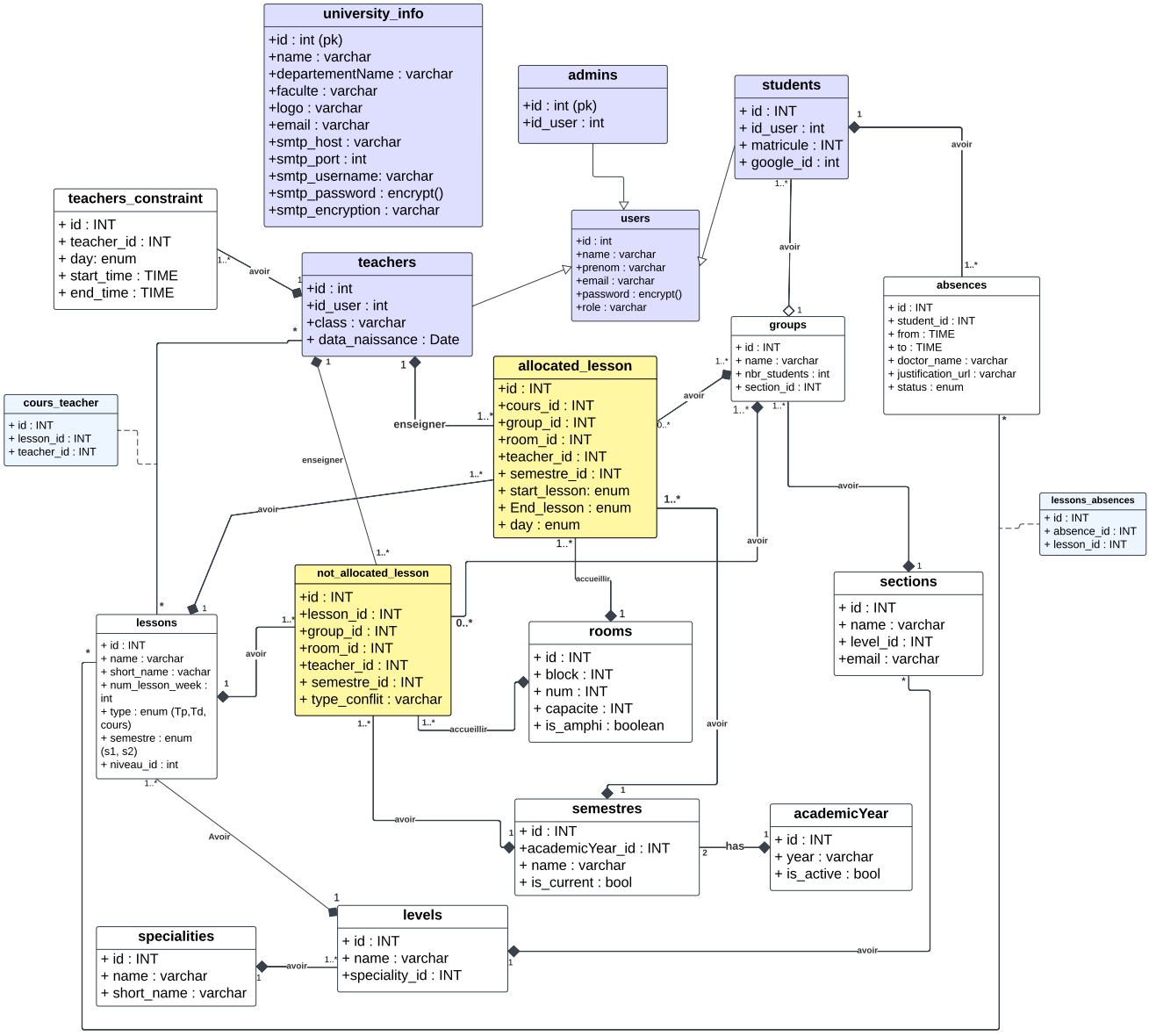


FIGURE 3.1 – Diagramme de classes de l'application

## 3.3 Le modèle relationnel

Le modèle relationnel est une méthode de représentation des données sous forme de relations (tables), introduite par E.F. Codd en 1970. Il repose sur des concepts mathématiques d'ensembles et sur l'unicité des données dans des structures tabulaires [**?**].

### 3.3.1 Les règles fondamentales du modèle relationnel

— **Unicité :**
Chaque donnée doit être représentée une seule fois dans une seule relation (ou table). Cela permet d'éviter les redondances et d'assurer la normalisation des données [8].

— **Garantie d'accès :**
Il faut qu'il existe un accès unique et non ambigu à chaque donnée. À partir du modèle relationnel, une valeur précise peut être atteinte en désignant la table, la colonne (attribut) et la clé primaire de la ligne qui contient la donnée à extraire [8].

— **Traitement des valeurs nulles :**
Le modèle relationnel accepte l'utilisation de valeurs nulles pour représenter un manque d'informations, des données inconnues ou une valeur non applicable. Cependant, traiter ces valeurs nécessite de la prudence pour prévenir les erreurs logiques [8].

### 3.3.2 Schéma relationnel

Pour obtenir le modèle relationnel à partir du diagramme de classes, nous avons appliqué les règles générales de transformation suivantes :

— **Transformation des classes en tables :** Chaque classe du diagramme est transformée en une table relationnelle. Exemple : la classe *User* devient la table `USERS`.

— **Transformation des attributs en colonnes :** Chaque attribut d'une classe est transformé en une colonne de la table correspondante. Exemple : l'attribut *name* de la classe *User* devient la colonne `name` dans la table `USERS`.

— **Transformation des associations en clés étrangères :** Chaque relation entre classes est représentée par une clé étrangère. Exemple : la relation entre *Teacher* et *User* est représentée par la clé étrangère `#id_user` dans la table `TEACHERS`.

— **Gestion des cardinalités :** Les associations de type « un-à-plusieurs » sont représentées par une clé étrangère. Les associations de type « plusieurs-à-plusieurs » donnent lieu à une table liaison. Exemple : la relation entre *Teacher* et *Lesson*, elle est donc représentée par la table associative `COURS_TEACHER`.

— **Héritage :** Les spécialisations (héritage) du diagramme de classes sont transformées en tables séparées reliées à la classe mère par une clé étrangère. Exemple : les classes *Teacher*, *Student*, et *Admin* héritant de *User* deviennent des tables séparées avec la clé étrangère `#id_user`.

— **Valeurs nulles :** Conformément au modèle relationnel, certains attributs peuvent accepter la valeur `NULL` pour représenter une information manquante ou inapplicable.

En appliquant ces règles, nous obtenons le schéma relationnel suivant :

```
USERS (id, name, last_name, email, password, role)

TEACHERS (id, #id_user, matricule, education_rank,
          date_naissance)

ADMINS (id, #id_user, level)

STUDENTS (id, #id_user, #group_id, date_naissance,
          matricule, google_id)

SPECIALITIES (id, name, short_name)

LEVELS (id, name, #speciality_id)

SECTIONS (id, name, email, #level_id)

GROUPS (id, name, nbr_students, #section_id)

LESSONS (id, name, short_name, num_lesson_week, type, semestre,
         #niveau_id)

COURS_TEACHER (id, #lesson_id, #teacher_id)

TEACHERS_CONSTRAINT (id, #teacher_id, day,
                     start_time, end_time)

ROOMS (id, block, num, capacite, is_amphi)
```

```
SEMESTERS (id, #academicYear_id, name, is_current)

ACADEMICYEAR (id, year, is_active)

ALLOCATED_LESSON (id, #lesson_id, #group_id, #room_id,
                  #teacher_id, #semestre_id,
                  start_lesson, end_lesson, day)

NOT_ALLOCATED_LESSON (id, #lesson_id, #group_id, #room_id,
                      #teacher_id, #semestre_id, type_conflit)

ABSENCES (id, #student_id, date, from, to, doctor_name,
          justification_url, status)

LESSONS_ABSENCES (id, #absence_id, #lesson_id)

UNIVERSITY_INFO (id, name, departement\_ame, faculte, logo,
                 email, smtp_host, smtp_port, smtp_username,
                 smtp_password, smtp_encryption)
```

**Légende :**

— id : Clé primaire

— #attribut : Clé étrangère

## 3.4 Description détaillée des attributs des tables de la base de données

Dans cette section, on se consacre à la description détaillée de toutes les tables et des attributs représentés dans le modèle relationnel. En d'autres termes, nous passons du **(MLD)** au **(MPD)**.

— **MLD (Modèle Logique de Données)** : correspond à la traduction du modèle conceptuel (MCD) en un modèle relationnel. Les entités deviennent des tables, les attributs deviennent des colonnes, et les associations sont représentées par des clés étrangères. Le MLD reste indépendant du système de gestion de bases de données (SGBD) choisi [9].

— **MPD (Modèle Physique de Données)** : consiste à adapter le MLD à un SGBD concret (par exemple PostgreSQL, MySQL, Oracle). Le MPD précise les **types de données**, les **contraintes** (NOT NULL, UNIQUE, DEFAULT), ainsi que les mécanismes spécifiques au SGBD (auto-incrémentation, séquences, index, etc.). C'est à ce stade que le schéma relationnel devient directement implémentable dans une base de données réelle [9].

### 3.4.1 Table « Users »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de l'utilisateur (clé primaire) |
| **name** | varchar | Nom de l'utilisateur |
| **prenom** | varchar | Prénom de l'utilisateur |
| **email** | varchar | Adresse email de l'utilisateur (pour connexion et contact) |
| **password** | encrypt() | Mot de passe chiffré de l'utilisateur |
| **role** | varchar | Rôle de l'utilisateur dans le système (étudiant, enseignant, admin) |

### 3.4.2 Table « Teachers »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de l'enseignant (clé primaire) |
| **user_id** | int | Référence à l'identifiant de l'utilisateur (clé étrangère vers users) |
| **education_Rank** | varchar | Rang éducatif de l'enseignant |
| **matricule** | varchar | Numéro matricule de l'enseignant |
| **date_naissance** | date | Date de naissance de l'enseignant |

### 3.4.3 Table « Admins »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de l'administrateur (clé primaire) |
| **user_id** | int | Référence à l'identifiant de l'utilisateur (clé étrangère vers users) |

### 3.4.4 Table « Students »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de l'étudiant (clé primaire) |
| **user_id** | int | Référence à l'identifiant de l'utilisateur (clé étrangère vers users) |
| **groupe_id** | int | Référence au groupe de l'étudiant (clé étrangère vers groups) |
| **matricule** | int | Numéro matricule de l'étudiant |
| **google_id** | int | Identifiant Google pour intégration avec services Google |
| **date_naissance** | date | Date de naissance de l'étudiant |

### 3.4.5 Table « Specialities »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de la spécialité (clé primaire) |
| **name** | varchar | Nom complet de la spécialité |
| **short_name** | varchar | Nom abrégé ou acronyme de la spécialité |

### 3.4.6 Table « Levels »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique du niveau d'études (clé primaire) |
| **name** | varchar | Nom du niveau (ex : Licence 1, Master 2) |
| **speciality_id** | int | Référence à la spécialité (clé étrangère vers specialities) |

### 3.4.7 Table « Sections »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de la section (clé primaire) |
| **name** | varchar | Nom de la section |
| **level_id** | int | Référence au niveau (clé étrangère vers levels) |

### 3.4.8 Table « Groups »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique du groupe (clé primaire) |
| **name** | varchar | Nom du groupe |
| **number_students** | int | Nombre d'étudiants dans le groupe |
| **section_id** | int | Référence à la section (clé étrangère vers sections) |

### 3.4.9 Table « Lessons »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique du cours (clé primaire) |
| **name** | varchar | Nom complet du cours |
| **short_name** | varchar | Nom abrégé ou code du cours |
| **nbr_lesson_week** | int | Nombre de séances par semaine |
| **type** | varchar | Type de cours (TP, TD ou cours magistral) |
| **level_id** | int | Référence au niveau d'études (clé étrangère vers levels) |

### 3.4.10 Table « Cours_teacher »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de l'association (clé primaire) |
| **lesson_id** | int | Référence au cours (clé étrangère vers lessons) |
| **teacher_id** | int | Référence à l'enseignant (clé étrangère vers teachers) |

### 3.4.11 Table « Teachers_constraint »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de la contrainte (clé primaire) |
| **teacher_id** | int | Référence à l'enseignant (clé étrangère vers teachers) |
| **day** | enum | Jour de la semaine concerné par la contrainte |
| **start_time** | time | Heure de début de la contrainte |
| **end_time** | time | Heure de fin de la contrainte |

### 3.4.12 Table « Rooms »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de la salle (clé primaire) |
| **block** | int | Bloc ou bâtiment où se trouve la salle |
| **num** | int | Numéro de la salle |
| **capacite** | int | Capacité d'accueil (nombre de places) |
| **is_amphi** | boolean | Indique si la salle est un amphithéâtre |

### 3.4.13 Table « Semestres »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique du semestre (clé primaire) |
| **name** | varchar | Nom du semestre |
| **niveau_id** | int | Référence au niveau (clé étrangère vers levels) |

#### 3.4.13.1 Table « Allocated_lessons »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de l'allocation (clé primaire) |
| **lesson_id** | int | Référence au cours (clé étrangère vers lessons) |
| **group_id** | int | Référence au groupe (clé étrangère vers groups) |
| **room_id** | int | Référence à la salle (clé étrangère vers rooms) |
| **teacher_id** | int | Référence à l'enseignant (clé étrangère vers teachers) |
| **semestre_id** | int | Référence au semestre (clé étrangère vers semestres) |
| **start_lesson** | time | Créneau de début du cours |
| **end_lesson** | time | Créneau de fin du cours |
| **day** | enum | Jour de la semaine |

#### 3.4.13.2 Table « Not_allocated_lessons »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique (clé primaire) |
| **lesson_id** | int | Référence au cours non alloué (clé étrangère vers lessons) |
| **group_id** | int | Référence au groupe concerné (clé étrangère vers groups) |
| **room _id** | int | Référence à la salle potentielle (clé étrangère vers rooms) |
| **teacher_id** | int | Référence à l'enseignant potentiel (clé étrangère vers teachers) |
| **semestre_id** | int | Référence au semestre (clé étrangère vers semestres) |

#### 3.4.13.3 Table « Absences »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de l'absence (clé primaire) |
| **student_id** | int | Référence à l'étudiant absent (clé étrangère vers students) |
| **from** | date | Date de début de l'absence |
| **to** | date | Date de fin de l'absence |
| **doctor_name** | varchar | Nom du médecin ayant fourni un justificatif |
| **justification_url** | varchar | Lien vers le document justificatif |
| **status** | enum | Statut de l'absence (justifiée, non justifiée, en attente) |

#### 3.4.13.4 Table « Absence_lessons »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique de l'association (clé primaire) |
| **absence_id** | int | Référence à l'absence (clé étrangère vers absences) |
| **lesson_id** | int | Référence au cours manqué (clé étrangère vers lessons) |

#### 3.4.13.5 Table « University_Infos »

| **Attribut** | Type | Description |
|---|---|---|
| **id** | int | Identifiant unique des paramétres (clé primaire) |
| **name** | varchar | Nom de l'université |
| **departement_name** | varchar | Nom du département concerné |
| **faculte** | varchar | Nom de la faculté |
| **logo** | varchar | Chemin vers le logo de l'université |
| **email** | varchar | Adresse email utilisée pour les notifications |
| **smtp_host** | varchar | Serveur SMTP utilisé pour l'envoi des mails |
| **smtp_port** | int | Port utilisé pour la connexion SMTP |
| **smtp_username** | varchar | Nom d'utilisateur SMTP |
| **smtp_password** | varchar | Mot de passe pour le SMTP |
| **smtp_encryption** | varchar | Type de chiffrement utilisé (ex : TLS, SSL) |

## 3.5 Algorithmes génétiques

Les Algorithmes génétiques (AGs) ont été initialement développés par John Holland en 1975. Ce sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Leur fonctionnement est relativement simple : on commence avec une population de solutions potentielles (appelées chromosomes), choisies arbitrairement. Ensuite, on évalue leurs performances (appelée *fitness*). Sur la base de ces performances, une nouvelle population est générée par application de trois opérateurs : la sélection, le croisement (*crossover*), et la mutation. Le processus est répété jusqu'à obtention d'une solution satisfaisante [23].

Un algorithme génétique suit généralement les étapes fondamentales suivantes :

— **Initialisation de la population** : génération aléatoire d'un ensemble initial de solutions (chromosomes).
— **Évaluation (Fitness)** : chaque individu est évalué à l'aide d'une fonction de fitness qui mesure sa qualité.
— **Sélection** : les meilleurs individus sont sélectionnés en fonction de leur performance pour être reproduits.
— **Croisement (Crossover)** : combiner des paires d'individus sélectionnés, pour créer ainsi de nouveaux individus.
— **Mutation** : modification aléatoire de certains gènes afin d'introduire de la diversité et d'éviter la stagnation.
— **Remplacement** : la nouvelle génération remplace soit partiellement ou totalement l'ancienne.
— **Critère d'arrêt** : le processus est répété jusqu'à ce qu'un certain critère soit atteint.

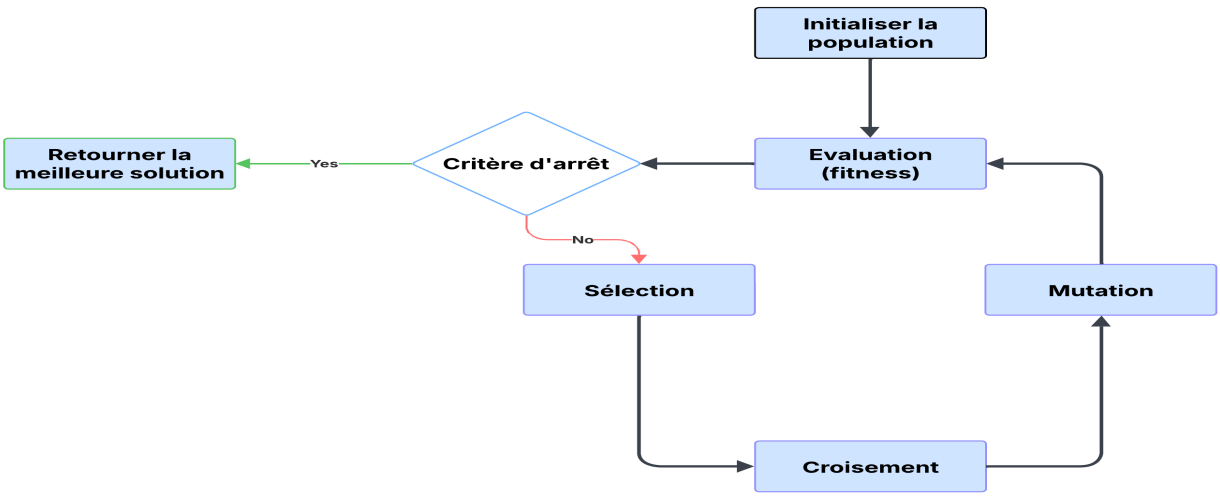


FIGURE 3.2 – Schéma général du fonctionnement de l'algorithme [23]

### 3.5.1 Principe appliqué à notre système

Les champs d'application des AGs sont vastes : optimisation de fonctions, finance, théorie des jeux, planification, etc. Dans notre cas, l'algorithme génétique est utilisé pour résoudre le problème de génération d'emplois du temps.

Chaque emploi du temps est représenté par un chromosome, composé d'un ensemble de gènes. Chaque gène représente une leçon allouée, c'est-à-dire une combinaison de : **cours, enseignant, salle, groupe et créneau horaire**.

L'objectif est de générer des emplois du temps pour tout le département sans conflits, tout en respectant un certain nombre de contraintes obligatoires et d'autres optionnelles qui seront satisfaites selon la disponibilité des ressources. Voici quelques contraintes que le système respecte actuellement :

#### 3.5.1.1 Contraintes obligatoires :

— Un enseignant ne peut pas enseigner deux cours en même temps ;
— Une salle ou un amphithéâtre ne peut accueillir qu'un seul créneau à la fois ;
— Un groupe ne peut pas avoir deux leçons en même temps ;
— La leçon doit être affectée à une salle de son type (cours $\rightarrow$ amphi, TD $\rightarrow$ salle TD, TP $\rightarrow$ salle TP).

#### 3.5.1.2 Contraintes optionnelles :

— Éviter certains créneaux (ex. : jeudi après-midi) ;
— Respecter la disponibilité des enseignants ;
— Faire appel au même enseignant pour tous les types de leçons d'un même module et groupe.

### 3.5.2 Pseudo-code de l'algorithme utilisé

**Algorithm 1:** Algorithme génétique pour la génération d'un emploi du temps

**Result:** Emploi du temps optimisé sans conflits

```
Générer aléatoirement une population initiale d'emplois du temps
while le critère d'arrêt n'est pas atteint do
    Évaluer la fitness de chaque emploi du temps
    Sélectionner les meilleurs individus (fitness élevée)
    Appliquer le croisement pour créer de nouveaux individus
    Appliquer une mutation aléatoire sur certains individus
    Remplacer l'ancienne population par la nouvelle
Retourner le meilleur emploi du temps trouvé
```

### 3.5.3 Fonctionnement de la fonction "fitness"

La fonction "fitness" évalue la qualité d'un emploi du temps en analysant chaque leçon et en identifiant les conflits ou violations de contraintes. Elle fonctionne de manière systématique :

#### 3.5.3.1 Analyse des conflits et contraintes

Pour chaque leçon de l'emploi du temps, la fonction vérifie : - les conflits d'enseignants (un enseignant ne peut pas être à deux cours simultanément), - les conflits de salles (une salle ne peut accueillir qu'un seul cours à la fois), - les conflits de groupes ou sections (un même groupe ou section ne peut suivre deux cours en même temps), - la compatibilité entre le type de salle et le type de leçon (cours, TD, TP).

De plus, elle prend en compte les **préférences facultatives**, telles que : - éviter certains créneaux (jeudi après-midi ou dernière heure), - respecter la disponibilité des enseignants, - regrouper les cours consécutifs d'un même module pour un même groupe.

Chaque violation de ces règles augmente un compteur de pénalités, tandis que certaines bonnes configurations (cours consécutifs correctement placés) peuvent donner un bonus.

#### 3.5.3.2 Calcul du score et résultat

Après avoir évalué toutes les leçons, la fonction de fitness calcule un score unique pour l'emploi du temps. Le score final est calculé de la manière suivante :

$$\text{Fitness} = \frac{1}{1 + \text{total\_conflicts}}$$

où `total_conflicts` correspond à la somme de toutes les pénalités moins les bonus éventuels.

- Un emploi du temps sans conflit obtient une fitness proche de 1.

- Un emploi du temps avec de nombreux conflits obtient une fitness proche de 0.

Ce score permet à l'algorithme de comparer les emplois du temps, sélectionner les meilleurs, et l'évolution vers des solutions optimales.

## 3.6 Conclusion

Dans ce chapitre, on a présenté le diagramme de classes de l'application. Ensuite, élaboré le modèle relationnel pour le projet et fournir une description détaillée des attributs de chaque table de la base de données. Enfin, On a présenté l'algorithme utilisé pour résoudre le problème de gestion des emplois du temps.

Le prochain chapitre sera consacré à la réalisation de l'application, où j'utiliserai les outils et les langages appropriés et illustrerai l'architecture globale et quelques interfaces de l'application.

# Chapitre 4

# Réalisation

## 4.1 Introduction

Dans ce chapitre, nous entamons la partie pratique, et présentons les outils et l'environnement de développement utilisés pour réaliser la plateforme. Enfin, on va conclure ce chapitre en exposant quelques interfaces de notre plateforme.

## 4.2 Langages et environnement de développement

Dans les sections suivantes, On va lister les différentes technologies utilisées pour réaliser notre plateforme.

### 4.2.1 Les langages et bibliothèques utilisés

Cette section va présenter les langages et les bibliothèques utilisé pour la réalisation de ce projet.

#### 4.2.1.1 HTML

HyperText Markup Language (HTML) est un langage de balisage utilisé pour structurer et formater le contenu des pages web. Il permet de décrire la structure d'un document en utilisant des balises qui définissent la fonction de chaque élément, tels que des titres, des paragraphes, des images, des liens hypertexte, des formulaires de saisie, et bien plus encore [10].

#### 4.2.1.2 CSS

Cascading Style Sheets (CSS) est un langage informatique côté client utilisé pour appliquer des styles et des mises en forme aux fichiers HTML ou XML. Il permet de séparer la présentation visuelle d'une page web de sa structure, en définissant des règles qui spécifient la manière dont les éléments HTML doivent être affichés à l'écran [11].

#### 4.2.1.3 JavaScript

JavaScript est un langage de programmation orienté objet utilisé pour rendre les pages web interactives. Il permet de créer des animations, des effets visuels, et des comportements dynamiques. Son utilisation est devenue incontournable dans le développement web moderne [12].

#### 4.2.1.4 JSX

JavaScript XML (JSX) est une extension de syntaxe pour JavaScript, utilisée principalement avec la bibliothèque React. Elle permet d'écrire du code qui ressemble à du HTML directement dans du JavaScript, facilitant ainsi la création d'interfaces utilisateur. JSX est ensuite transformé en appels JavaScript standards via un compilateur [13].

#### 4.2.1.5 React

React est une bibliothèque JavaScript open-source développée par Meta (anciennement Facebook). Elle permet de créer des interfaces utilisateur interactives et dynamiques. Elle repose sur une architecture déclarative et component-based (basée sur les composants), ce qui facilite la construction d'applications web dynamiques, notamment les applications web à page unique (SPA) [14]. J'ai utilisé la version 18 de React pour la réalisation de ce projet.

#### 4.2.1.6 Tailwind CSS

Tailwind CSS est un framework CSS utilitaire open-source qui permet de concevoir rapidement des interfaces web en utilisant des classes prédéfinies. Il adopte une approche "utility-first", ce qui signifie que la mise en forme se fait directement dans le HTML via des classes utilitaires, favorisant ainsi la rapidité de développement et la personnalisation des composants [15].

#### 4.2.1.7 Material UI

Material UI est une bibliothèque de composants d'interface utilisateur pour React. Elle permet aux développeurs de créer des interfaces utilisateur élégantes et cohérentes [16].

#### 4.2.1.8 PHP

Hypertext Preprocessor (PHP) est un langage de script polyvalent populaire, particulièrement adapté au développement web. Rapide, flexible et pragmatique, PHP optimise tout, de votre blog aux sites web les plus populaires au monde[17]. J'ai utilisé la version 8 de PHP pour la réalisation de ce projet.

#### 4.2.1.9 Laravel

Laravel est un framework PHP open-source basé sur le modèle MVC (Modèle-Vue-Contrôleur), conçu pour le développement d'applications web modernes. Il fournit une multitude de fonctionnalités et d'outils pour accélérer le processus de développement [18]. J'ai utilisé la version 11 de Laravel pour la réalisation de ce projet.

### 4.2.2 Le système de gestion des bases de données

Dans le cadre de ce projet le système de gestion de bases de données choisis est (MySQL), pour sa stabilité, sa fiabilité, ses performances, sa sécurité et sa flexibilité. De plus, il est utilisé par de nombreuses grandes entreprises et organisations.

### 4.2.3 Outils de développement utilisés

Dans cette sections, On présente les outils utilisé pour développé cette application.

#### 4.2.3.1 WAMP Server

WAMP Server est un ensemble de logiciels libres, populaires et faciles à installer, utilisé pour créer un environnement de développement web local. Cela facilite aux développeurs la création, le test et la démonstration d'applications web avant de les déployer sur un serveur distant [19].

#### 4.2.3.2 PhpMyAdmin

PhpMyAdmin est une application web libre, écrite en PHP, qui permet de manipuler des bases de données MySQL ou Maria DB via une interface graphique accessible depuis un navigateur. Elle offre de nombreuses fonctionnalités telles que l'exécution de requêtes SQL, une simple gestion des tables et bien plus encore. Cet outil est largement utilisé pour sa simplicité et son efficacité dans la gestion de bases de données relationnelles [20].

#### 4.2.3.3 Visual Studio Code

Visual Studio Code est un éditeur de code open source, léger et puissant, qui dispose d'un écosystème riche en extensions [21].

#### 4.2.3.4 Lucidchart

Lucidchart est un outil de création de diagrammes en ligne, permettant de modéliser visuellement des processus, des structures logiques, des diagrammes UML, des schémas de base de données et bien plus encore. Il est largement utilisé dans le domaine du développement logiciel, de la gestion de projets et de la modélisation de systèmes pour sa facilité d'utilisation [22].

## 4.3 Architecture logicielle de l'application

La **Figure 4.1** représente les échanges entre le client et le serveur selon notre stack **Laravel** pour le back-end et **React** pour le front-end. L'application est conçue selon une architecture de type *client-serveur*, où l'interface utilisateur (client) interagit avec le serveur via des requêtes HTTP, principalement au format JSON.

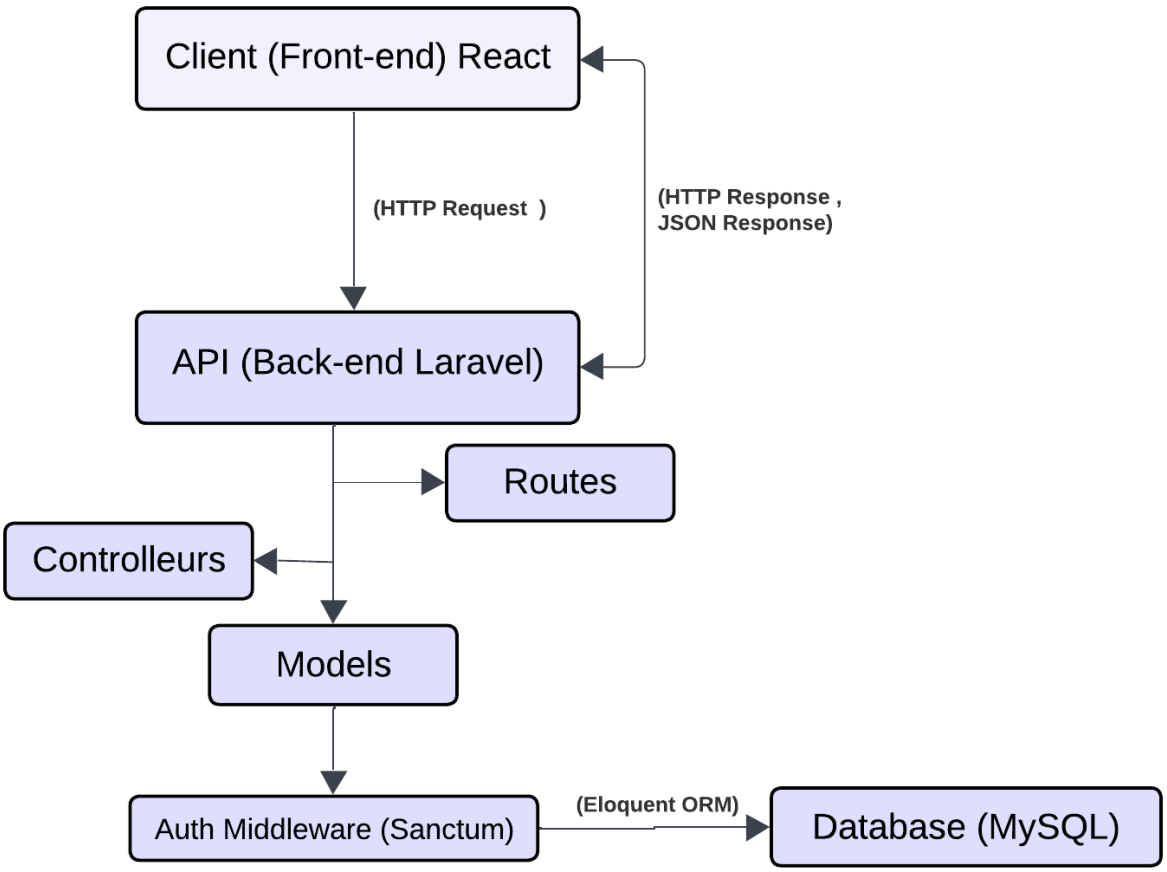


FIGURE 4.1 – Architecture logicielle de l'application

## 4.4 L'architecture globale de l'application

La **figure 4.2** illustre l'architecture globale de notre application de manière visuelle et simplifiée.

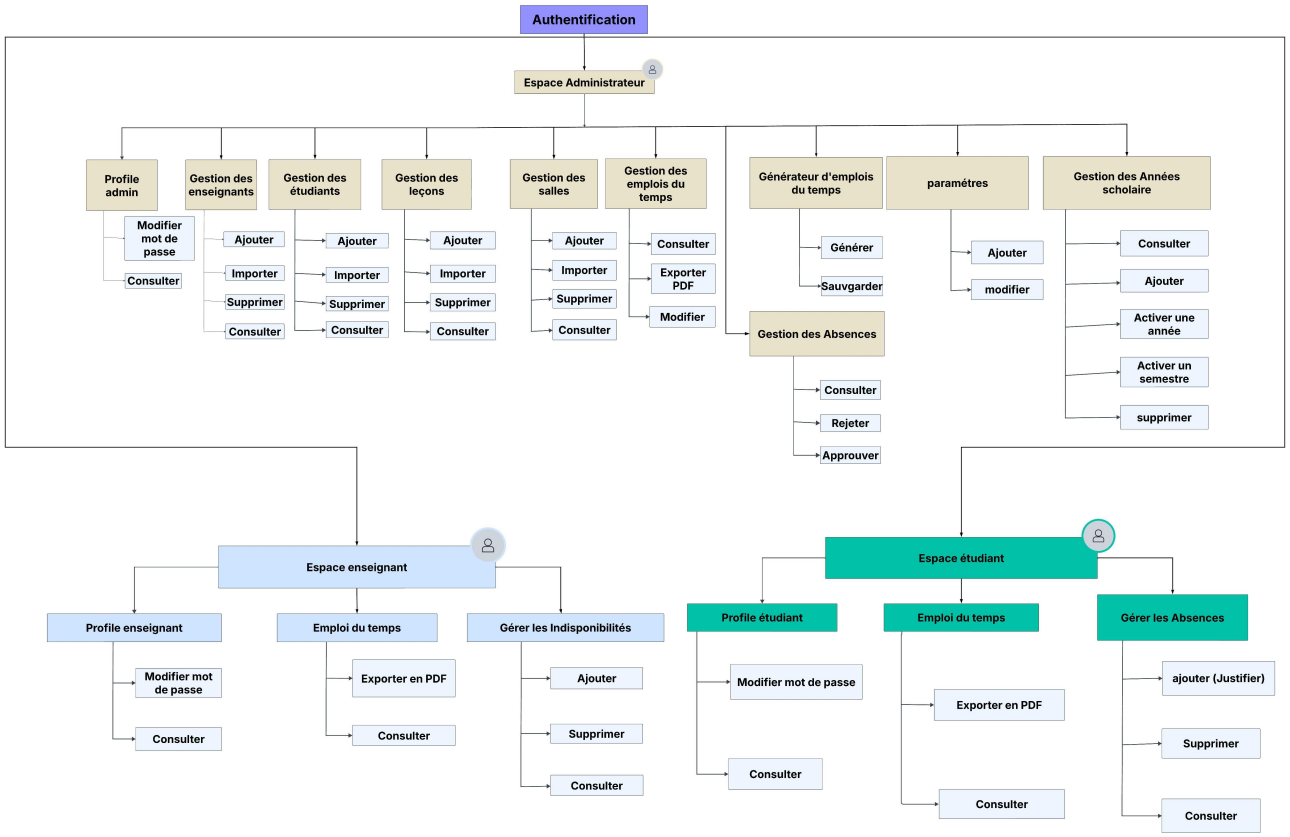


FIGURE 4.2 – Architecture globale de l'application

## 4.5 Présentation des interfaces de l'application

Notre application web, **"Scholaris"**, combine le mot anglais *"school"* et le mot grec *"aris"*, qui signifie "Excellence". Ce nom reflète l'objectif principal de la plateforme : proposer un outil moderne, performant et adapté aux besoins du secteur éducatif.

Pour lui donner une identité claire et facilement reconnaissable, nous avons conçu un logo spécifique pour **"Scholaris"**, présenté à la figure 4.3, illustrant ainsi l'identité visuelle et l'ergonomie de l'application.

Dans la suite, nous présenterons les principales interfaces de l'application.


FIGURE 4.3 – Logo « Scholaris »

### 4.5.1 Interface « Connexion »

Dans la **figure 4.4** nous pouvons observer l'interface d'authentification qui apparaît lors du lancement de l'application. Les acteurs peuvent s'authentifier pour avoir accès à leurs espaces.

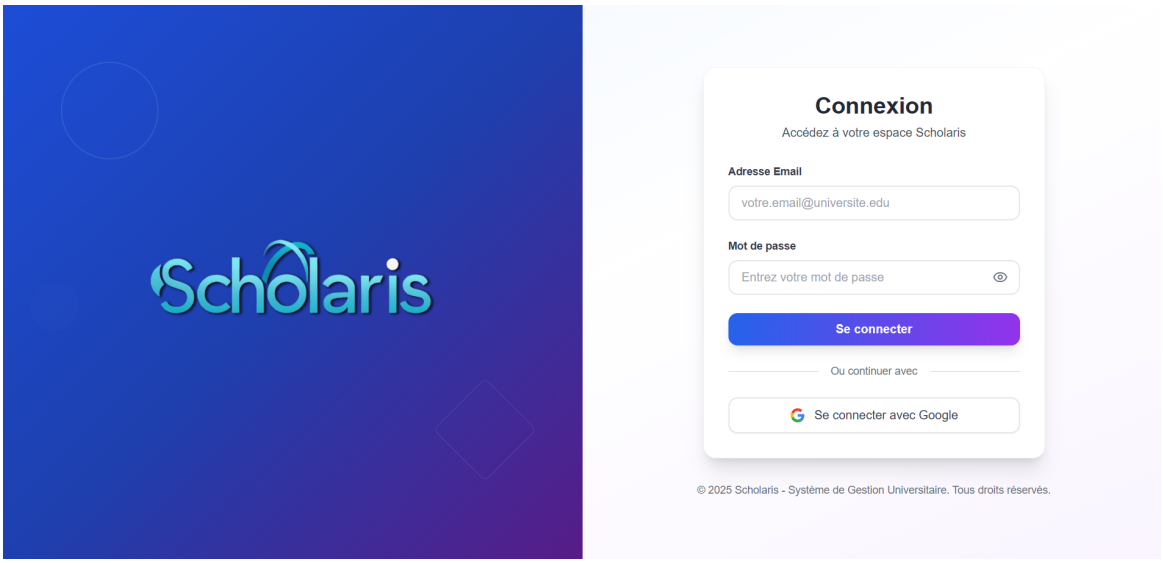


FIGURE 4.4 – Interface « Connexion »

### 4.5.2 Interface « Tableau de bord administrateur »

Dans la **figure 4.5** nous pouvons observer l'interface du tableau de bord des admins, où ils ont la possibilité de consulter les statistiques de la base de données et vérifier les conflits .

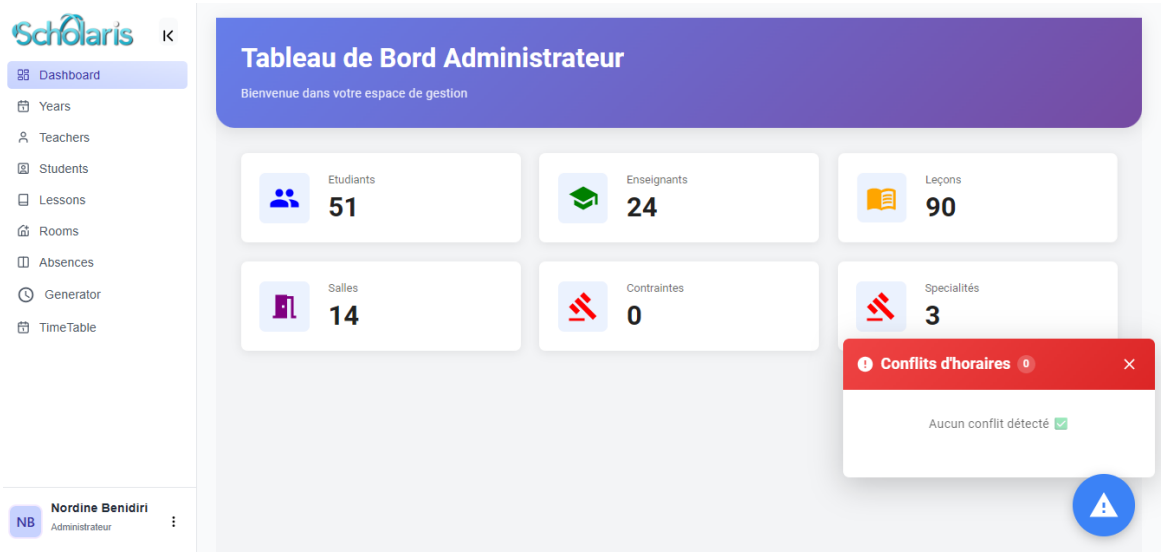


FIGURE 4.5 – Interface « Tableau de bord administrateur »

### 4.5.3 Interface « Gestion des enseignants »

Les **figures 4.6 et 4.7** présentent l'interface de gestion des enseignants. Cette interface permet à l'administrateur d'ajouter, de supprimer, ou de rechercher un enseignant, ainsi que de consulter la liste des enseignants enregistrés dans la base de données. L'administrateur peut également télécharger un fichier Excel d'exemple destiné à l'importation, et importer des enseignants à partir de ce fichier.

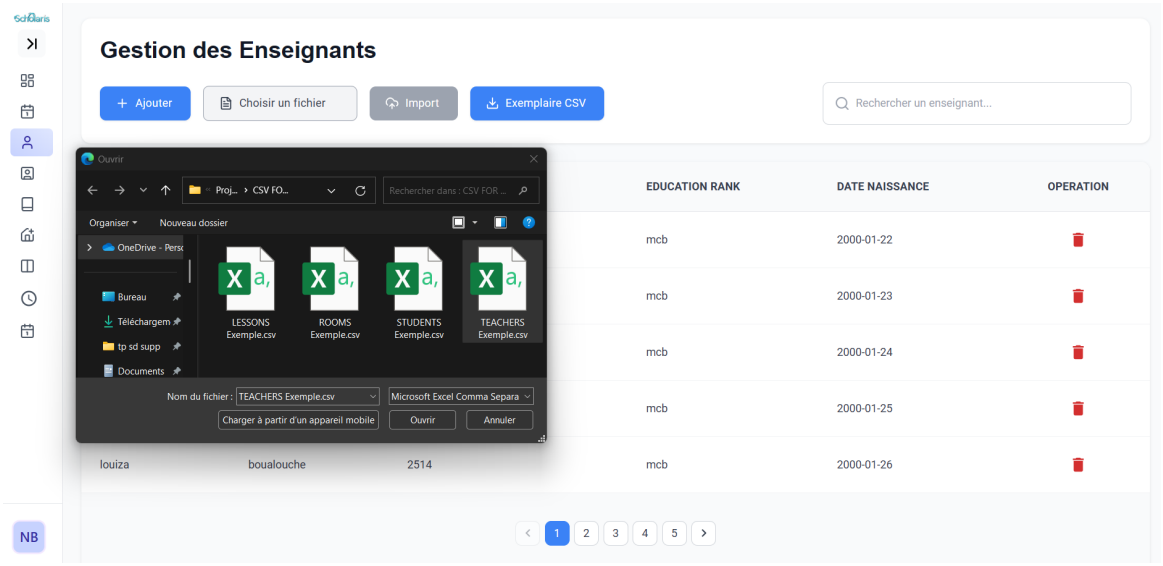


FIGURE 4.6 – Interface « Gestion des enseignants »

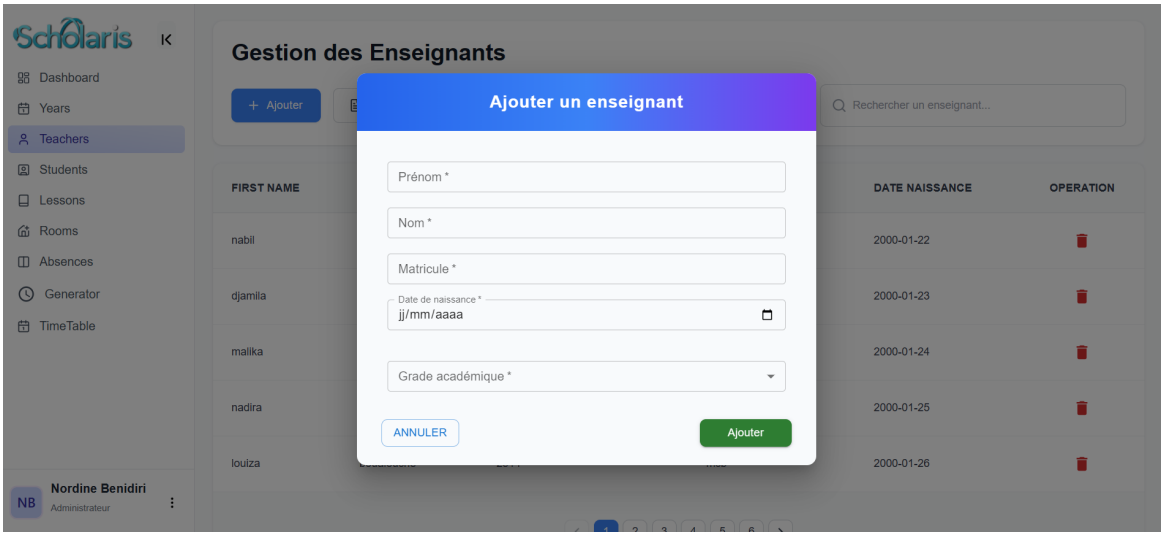


FIGURE 4.7 – Formulaire « Ajouter un enseignant »

### 4.5.4 Interface « Générateur des emplois du temps »

Les**figure 4.8 et figure 4.9** illustrent L'interface de génération des emplois du temps qui permet à l'administrateur de lancer automatiquement la création des emplois du temps pour l'ensemble des spécialités enregistrées dans le système.

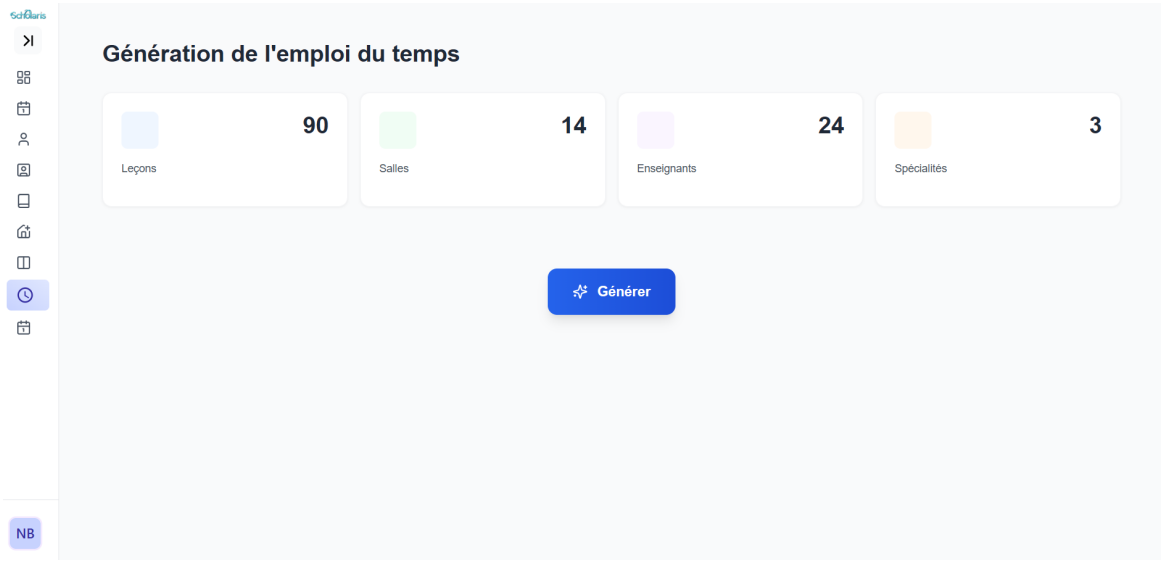


FIGURE 4.8 – Interface « Générateur des emplois du temps »

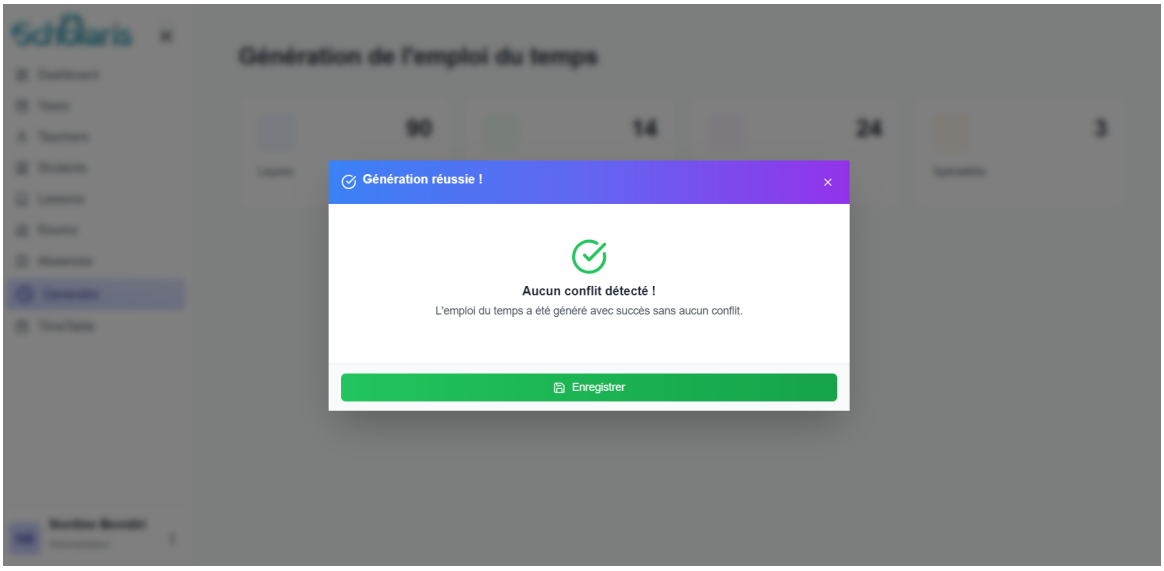


FIGURE 4.9 – Formulaire « Fin Génération des emplois du temps »

### 4.5.5 Interface « Gestion des emplois du temps »

Les **figures 4.10 et 4.11** illustrent l'interface permettant de consulter tous les emplois du temps des spécialités par section, exporter en pdf l'emploi du temps de la section choisis, envoyer par email à tout les étudiants de la section, et aussi la possibilité de modifier une leçon alloué selon un formulaire intelligent capable de vérifier automatiquement les contraintes et les conflits horaires en fonction du créneau sélectionné.

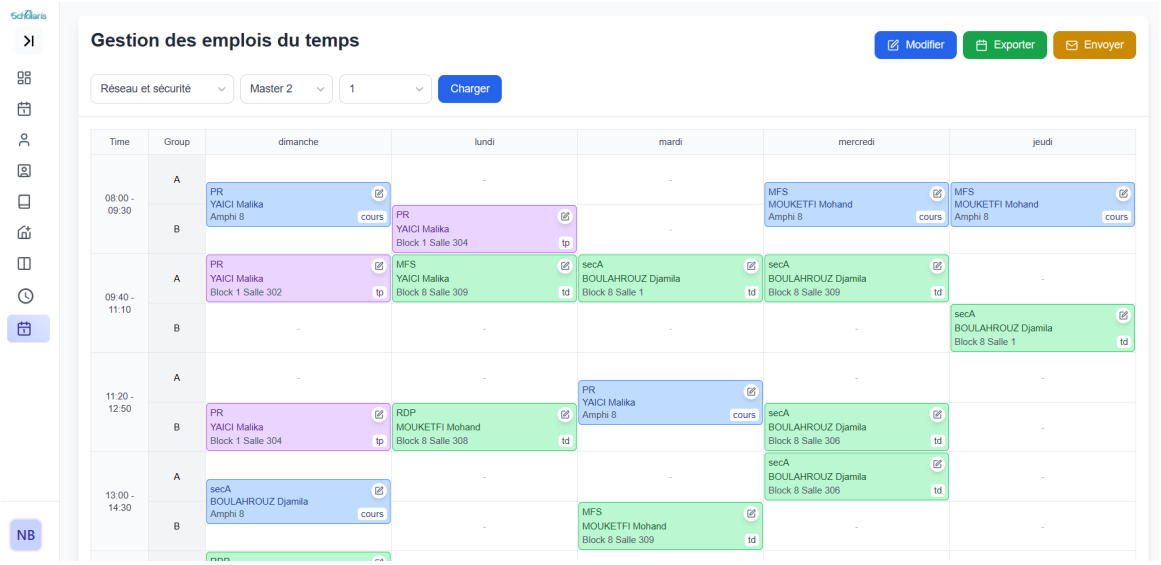


FIGURE 4.10 – Interface « Gestion des emplois du temps »

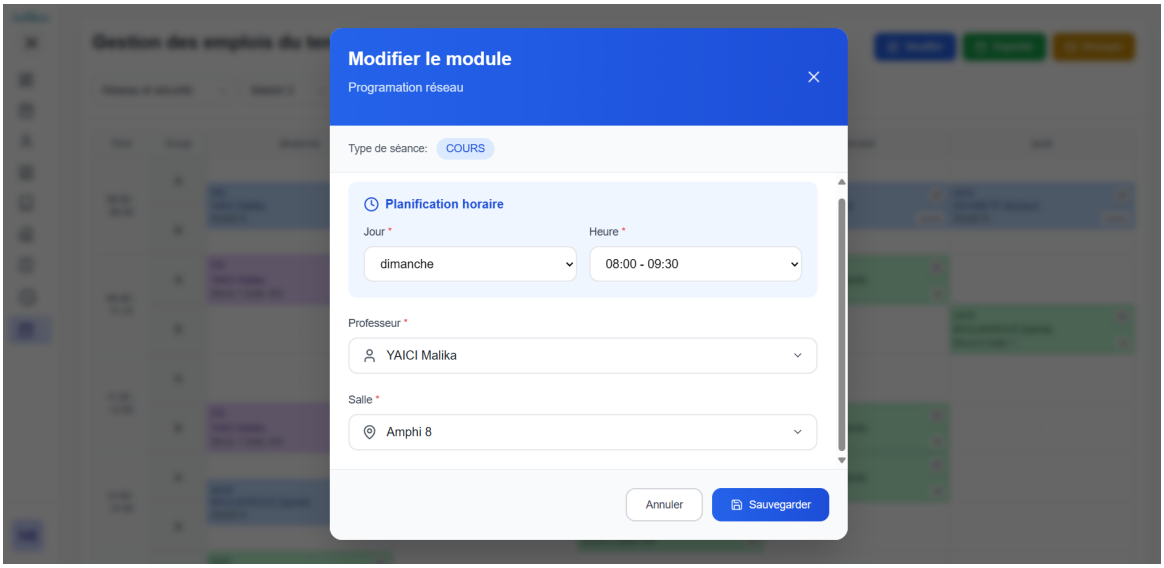


FIGURE 4.11 – Formulaire « Modifier une leçon »

### 4.5.6 Interface « Emploi du temps étudiant »

Dans les **figures 4.12 et 4.13**, on peut voir l'interface permettant à un étudiant de consulter son emploi du temps en fonction de son groupe. Elle est conçue de manière responsive, offre une expérience utilisateur fluide et intuitive pour un meilleur confort de consultation.

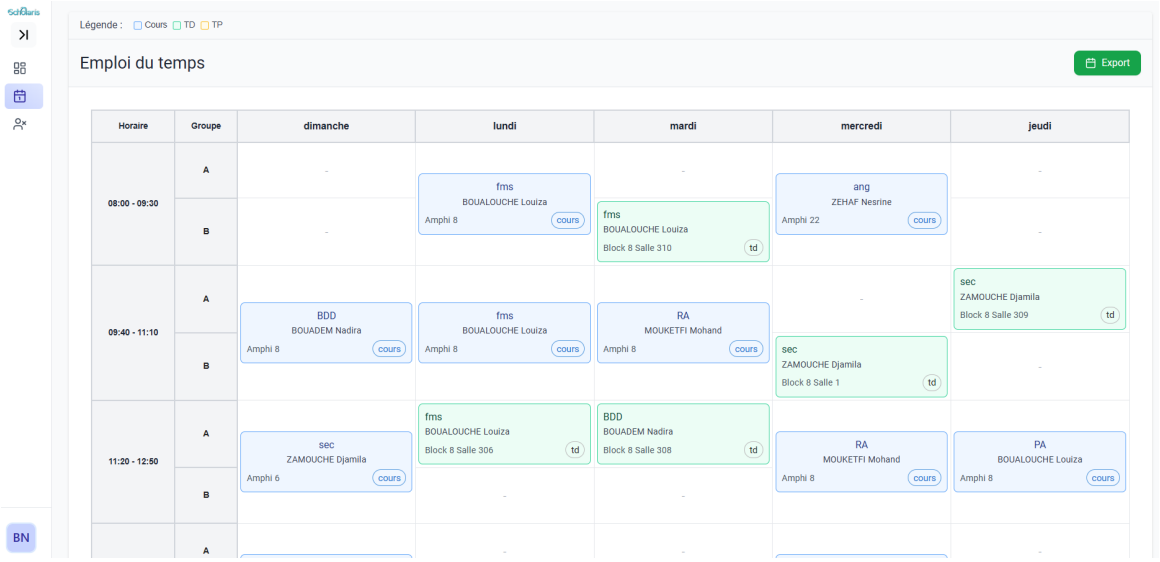


FIGURE 4.12 – Interface « Emploi du temps Étudiant »

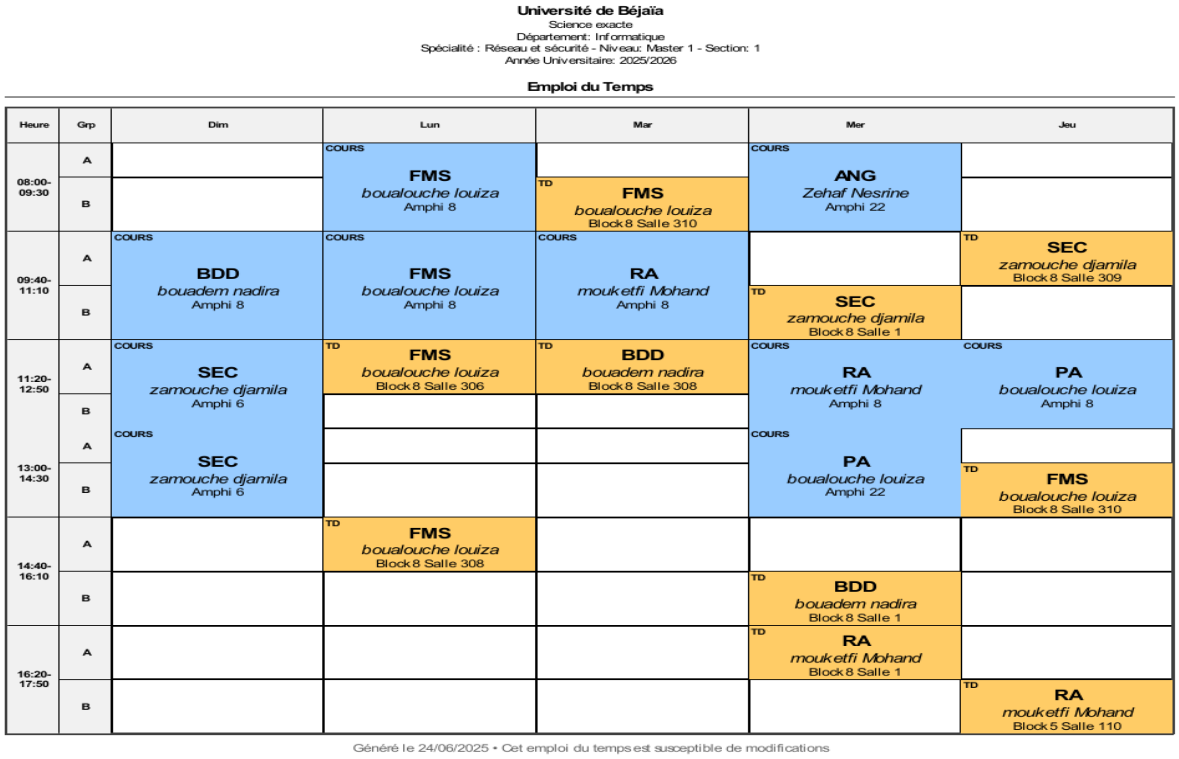


FIGURE 4.13 – Le PDF exporter

### 4.5.7 Interface « Absences étudiant »

Les **figures 4.14 et 4.15**, montrent l'interface des absences pour l'étudiant. Elle lui permet de consulter sa liste d'absences, de voir la réponse du département après le traitement, et de justifier ou de supprimer une absence.

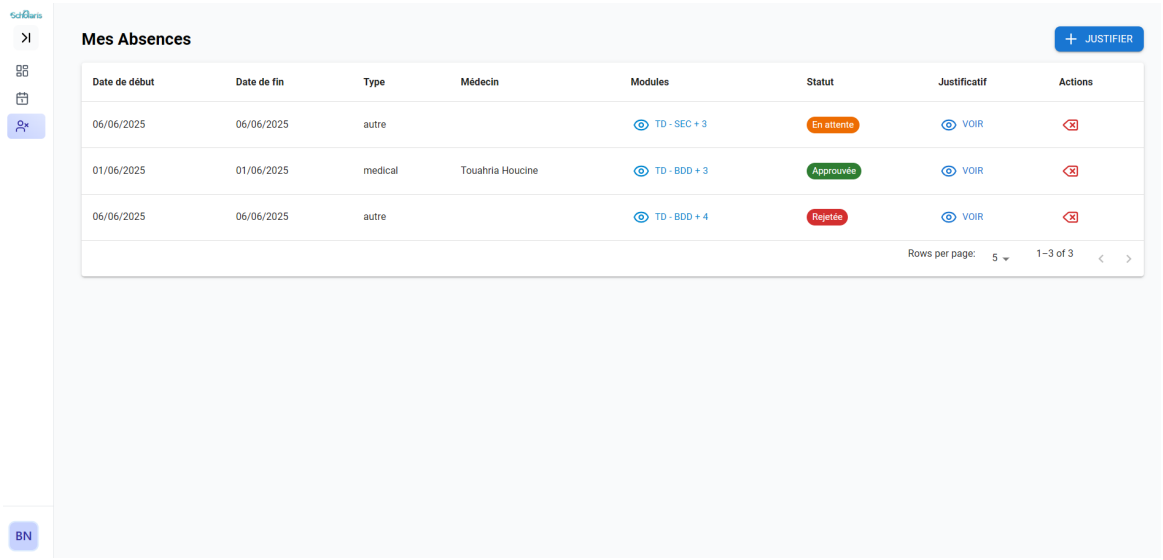


FIGURE 4.14 – Interface « Liste d'absences Étudiant »

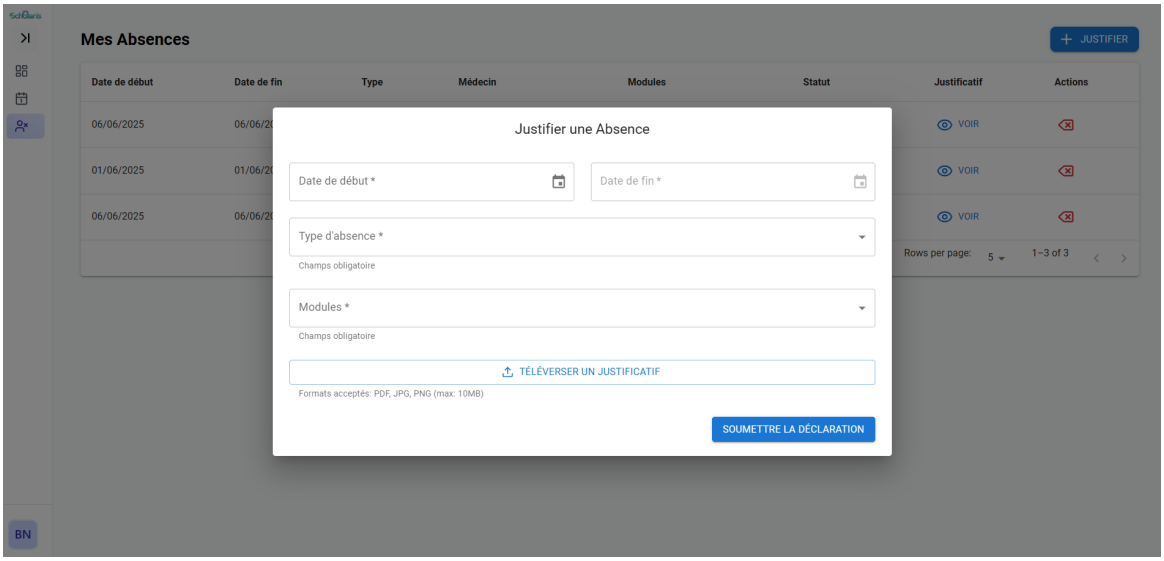


FIGURE 4.15 – Formulaire « Justifier une absence »

### 4.5.8 Interface « Contraintes enseignant »

Les **figures 4.16 et 4.17**, illustrent l'interface des contraintes enseignant. Elle permet à ce dernier de voir sa liste de contraintes enregistrer, d'ajouter ou de supprimer une contrainte.

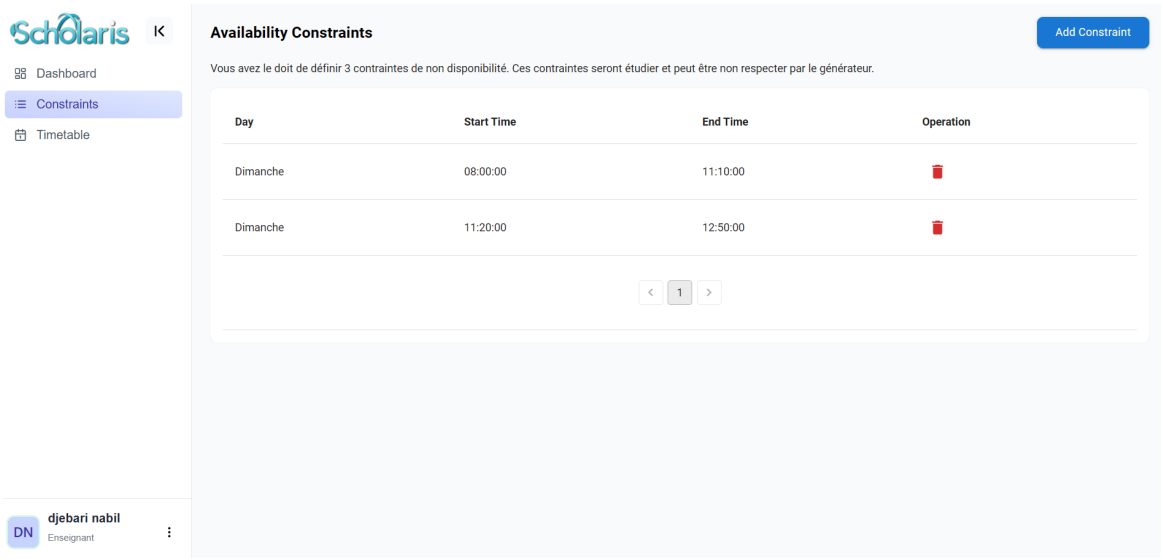


FIGURE 4.16 – Interface « Liste des contraintes enseignant »

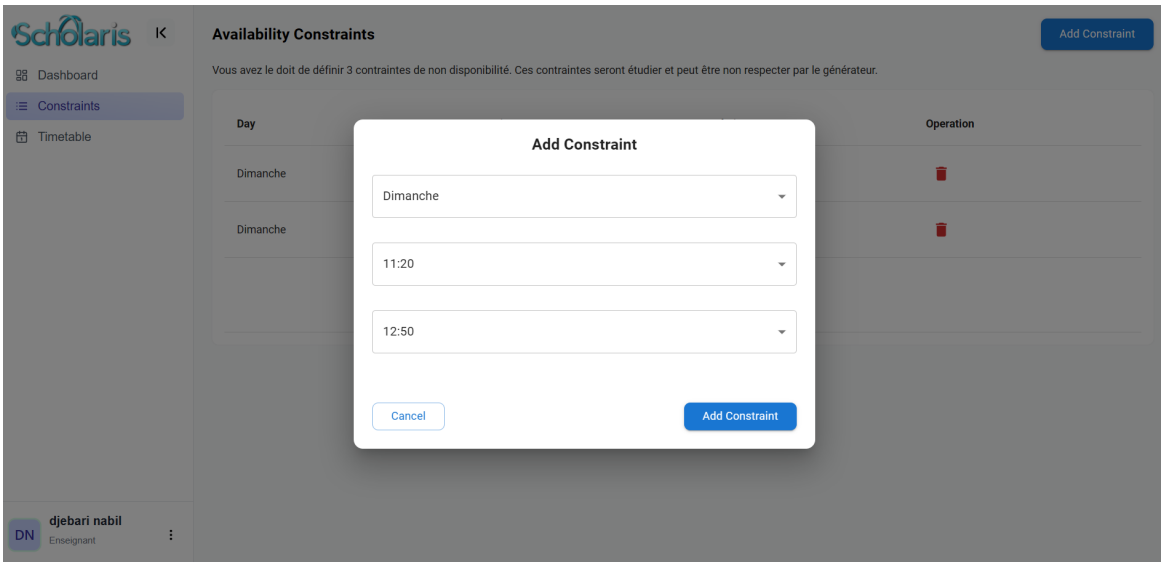


FIGURE 4.17 – Formulaire « Ajouter une contrainte »

# Conclusion Générale

Pendant la réalisation de ce mémoire, on a pu acquérir des compétences en développement et conception web, ainsi qu'en modélisation UML. on a également amélioré notre capacité à analyser un besoin réel et à proposer une solution adaptée, tout en respectant les contraintes techniques et fonctionnelles du domaine.

Ce projet nous a permis aussi de mieux comprendre le fonctionnement d'un département universitaire, les difficultés rencontrées dans la gestion des emplois du temps et des absences. Grâce aux outils et aux méthodes utilisés, on a pu concevoir une solution qui simplifie et automatise plusieurs tâches administratives.

Au cours de ce travail, on a pu atteindre plusieurs objectifs importants, tels que :

— La génération automatique des emplois du temps à l'aide d'un algorithme génétique.
— La gestion centralisée des ressources pédagogiques du département.
— La mise en place d'un système de suivi et de justification des absences.
— L'organisation de l'accès à la plateforme selon les profils (administrateur, enseignant, étudiant).
— La consultation personnalisée des emplois du temps selon l'utilisateur.
— La liberté de modifier et de manipuler chaque emploi du temps généré.
— Un système de détection de conflits pour éviter toute erreur de l'algorithme.
— Notifications par mail par l'administrateur lors de la génération ou lors d'une modification sur un planning.
— Exportation en PDF selon l'utilisateur.

Bien que ce projet ne soit pas encore finalisé, le développement d'une solution informatique demeure un processus évolutif, qui s'améliore avec le temps et en fonction des besoins. La solution proposée constitue ainsi une base fonctionnelle et évolutive pour la modernisation des processus de gestion au sein du département.

Bien que l'application soit fonctionnelle, plusieurs pistes d'amélioration peuvent être envisagées pour améliorer et enrichir la plateforme :

— Développer une application mobile pour les étudiants et les enseignants.
— Intégrer un système d'authentification renforcé pour plus de sécurité, notamment par la mise en place d'une authentification à deux facteurs (2FA), et l'ajout de mécanismes de détection d'anomalies lors des connexions.
— Mettre en place un système de gestion de comptes et de permissions.
— Étendre la plateforme à d'autres départements ou services universitaires.

Par ailleurs, il serait pertinent d'intégrer des fonctionnalités complémentaires telles que la génération automatique des plannings d'examens ou l'établissement d'un canal de communication direct entre étudiants et enseignants pour la gestion des absences, dans le but d'enrichir davantage la plateforme et d'offrir des services plus complets et pratiques.

En conclusion, ce projet a constitué une expérience enrichissante tant sur le plan technique que personnel, en apportant une contribution concrète à la modernisation du fonctionnement du département à travers une solution pratique, efficace, et adaptée aux besoins actuels.

# Bibliographie


[1] Pressman, R. S. *Software Engineering : A Practitioner's Approach.* 7th Edition, McGraw-Hill, 2010.

[2] Audibert, Laurent. *UML 2 - De l'apprentissage à la pratique.* paris : Eyrolles, 2006. Chapitre 7 : Diagrammes d'interaction.

[3] Booch, G. *The unified modeling language user guide.* amérique : Addison-Wesley, 2005.

[4] Easterbrook, S. (2004). *What is Requirements Engineering ?* University of Toronto. [En ligne] [Consulté le : 12 juin 2025.] https ://www.cs.toronto.edu/ sme/papers/2004/FoRE-chapter01-v7.pdf

[5] Use-case diagrams in UML modeling. *IBM.* [En ligne] [Consulté le : 22 février 2025.] https ://www.ibm.com/docs/en/rational-soft-arch/9.6.1 ?topic=mapping-use-case-diagram-model-elements.

[6] UML Description textuelle des cas d'utilisation. *clicours.* [En ligne] [Consulté le : 2 mars 2025.] https ://www.clicours.com/uml-description-textuelle-des-cas-dutilisation/.

[7] Audibert, Laurent. *UMl 2 - De l'apprentissage à la pratique.* paris : Eyrolles, 2006. Chapitre 3 : Diagrame de classe.

[8] A relational model of data for large shared data banks. *ACM Digital Library home.* [En ligne] [Consulté le : 2 mars 2025.] https ://dl.acm.org/doi/10.1145/362384.362685.

[9] Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.

[10] W3Schools. *HTML Tutorial.* [En ligne] [Consulté le : 2 avril 2025.] https ://www.w3schools.com/html/.

[11] World Wide Web Consortium. *Cascading Style Sheets (CSS).* [En ligne] [Consulté le : 2 avril 2025.] https ://www.w3.org/Style/CSS/specs.fr.html.

[12] Mozilla Developer Network (MDN). (2023). *JavaScript – Introduction.* [En ligne] [Consulté le : 3 avril 2025.] https ://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Introduction.

[13] React. (2023). *Introducing JSX.* [En ligne] [Consulté le : 3 avril 2025.] https ://reactjs.org/docs/introducing-jsx.html.

[14] React. (2023). *React – A JavaScript library for building user interfaces.* [En ligne] [Consulté le : 5 avril 2025.] https ://reactjs.org.

[15] Tailwind CSS. (2023). *Documentation officielle.* [En ligne] [Consulté le : 10 avril 2025.] https ://tailwindcss.com/docs.

[16] MUI. (s.d.). *Material-UI Documentation.* [En ligne] [Consulté le : 10 avril 2025.] https ://mui.com/getting-started/usage.

[17] PHP. (2023). *Introduction – What is PHP ?* [En ligne] [Consulté le : 12 avril 2025.] https ://www.php.net/manual/en/intro-whatis.php.

[18] Laravel. (2023). *The PHP Framework for Web Artisans.* [En ligne] [Consulté le : 12 avril 2025.] https ://laravel.com.

[19] wamp server. *documentation-en.* [En ligne] [Consulté le : 12 avril 2025.] https ://www.wampserver.com/en/category/documentation-en/.

[20] phpMyAdmin. (2024). *phpMyAdmin - Official website.* [En ligne] [Consulté le : 15 avril 2025.] https ://www.phpmyadmin.net/.

[21] Microsoft. (s.d.). *Visual Studio Code Documentation.* [En ligne] [Consulté le : 15 avril 2025.] https ://code.visualstudio.com/docs.

[22] Lucid Software. (2024). *Lucidchart : Visual Collaboration Suite.* [En ligne] [Consulté le : 10 juin 2025.] https ://www.lucidchart.com.

[23] hal.science. (s.d). *Présentation des algorithmes génétiques et de leurs applications en économie.* [En ligne] [Consulté le : 19 juin 2025.] https ://hal.science/hal-00125103v1/document.

# RÉSUMÉ

L'informatique joue un rôle essentiel dans tous les domaines, car elle facilite le traitement, la gestion et l'accès à l'information, tout en automatisant de nombreuses tâches, ce qui améliore considérablement l'efficacité et la productivité. Dans ce mémoire, une application web de gestion a été conçue et réalisée pour le département informatique de l'université de Béjaïa, dans le but de simplifier certaines tâches administratives. Pour ce faire, le processus de développement logiciel UP a été suivi, et le langage UML a été utilisé pour la modélisation. L'application, nommée **SCHOLARIS**, a été développée à l'aide de l'éditeur VS Code, en utilisant la bibliothèque JavaScript React pour le front-end, ainsi que le framework Laravel pour le back-end, reposant sur le langage PHP et le SGBD MySQL.

**Mots-clés :** Algorithme génétique, Génération des emplois du temps, React, laravel, UML.

# ABSTRACT

Information technology plays a vital role in all fields, as it facilitates data processing, management, and access to information, while automating numerous tasks, thereby significantly improving efficiency and productivity. In this thesis, a web-based management application was designed and developed for the Computer Science Department of the University of Béjaïa, with the aim of simplifying certain administrative tasks. To achieve this, the Unified Process (UP) development methodology was followed, and UML was used for modeling. The application, named **SCHOLARIS**, was developed using the VS Code editor, with the React JavaScript library for the front-end, and the Laravel framework for the back-end, based on the PHP language and the MySQL DBMS.

**Keywords :** Genetic Algorithm, Timetable Generation, React, Laravel, UML.