DEVELOPPEMENT D'UNE PLATEFORME INTELLIGENTE DE PLANIFICATION, DE GESTION ET RESERVATION DE SALLES

Filière : GENIE INFORMATIQUE

Réalisé par :

- LUSAMOTE KIMFUTA Claudia
- MOUPIGA TOMBE Elisia
- Issa D DEMBELE
- SOKPOH Kimberly Viwoassi

Encadrant(e) :

- Mme HAIDRAR
- M. HOUEKPO DANNON

Année universitaire : 2025 - 2026

HESTIM
ENGINEERING & BUSINESS SCHOOL

Table des matières

INTRODUCTION GÉNÉRALE 6

Chapitre 1 : Cahier des Charges 7

1.1- Introduction 7

1.1- Présentation du Projet 7

1.2- Objectifs du Projet 7

1.2.1- Objectif général 7

1.2.2- Objectif spécifique 8

1.3- Conduite du projet 8

1.3.1- Méthodologie 8

1.3.2- Planning du projet 9

1.4- Public cible 11

1.4.1- Utilisateurs principaux 11

1.4.2- Utilisateurs secondaires 12

1.4.3- Enjeux pour le public cible 12

1.5- Problématique 12

1.6- Conclusion 13

Chapitre 2 : Analyse et Conception 14

2.1- Introduction 14

2.2- Analyse des besoins 14

2.2.1- Les besoins fonctionnels 14

2.2.2- Les besoins non fonctionnels 16

2.3- Conception hybride avec MERISE et UML 17

2.3.1- Identification des acteurs 17

2.3.2- Modèle conceptuel des données (MCD) 19

2.3.3- Modèle logique des données (MLD) 21

2.3.4- Diagramme de cas d'utilisation 24

2.3.5- Diagramme de classes 25

2.3.6- Diagramme de séquence 25

2.3.7- Diagramme d'activités 27

2.3.8- Diagramme d'état 28

2.4- Conclusion 29
Chapitre 3 : Architecture et étude technique 30
3.1- Introduction 30
3.2- Architecture du projet 30
3.2.1- Diagramme de l'architecture globale 30
3.2.2- Composants principaux 32
3.3 – Les langages et les Frameworks de développement 33
3.3.1- Langages 33
3.3.2- Frameworks 34
3.4- Les outils de développement 37
3.5- Conclusion 40
Chapitre 4 : Maquettes et Design de l'Interface 41
4.1- Introduction 41
4.2- Vue d'ensemble de la maquette 41
4.3- Descriptions détaillées des différentes pages 42
4.4- Conclusion 43
Chapitre 5 : Conception technique & développement Backend 44
5.1- Introduction 44
5.2- Architecture générale 44
5.3- Explication des choix technologiques 47
5.4- Développement du Backend 47
5.5- Gestion des sessions utilisateurs 48
5.6- Tests et Débogage 50
5.7- Perspectives d'amélioration du Backend 50
5.8- Conclusion 51
CONCLUSION GÉNÉRALE 52

Liste des figures

Figure 1 : Cycle de développement en V 9
Figure 2 : Diagramme de Gantt 9
Figure 3: Modèle conceptuel de données (MCD) 21
Figure 4: Modèle logique de données (MLD) 23
Figure 5: Diagramme de cas d'utilisation 24
Figure 6: Diagramme de classes 25
Figure 7: Diagramme de séquence 26
Figure 8: Diagramme d'activité - Détection et Résolution de Conflits 27
Figure 9: Diagramme d'état - Cycle de vie d'une réservation 29
Figure 10: Architecture global 31
Figure 11: Diagramme de composants 32
Figure 12 : Logo de JavaScript 33
Figure 13: Logo de HTML5 33
Figure 14: : Logo CSS3 34
Figure 15: SQL 34
Figure 16: Logo de React JS 35
Figure 17: Logo de Material UI 35
Figure 18: Logo de Node JS 35
Figure 19: Logo d'Express Js 35
Figure 20: Logo Sequelize 36
Figure 21: Logo de Visual Studio Code 37
Figure 22: : Logo Git & GitHub 38
Figure 23: Logo MAMP 38
Figure 24: Logo Postman 38
Figure 25: Logo Figma 39
Figure 26: Logo Vercel 39
Figure 27: Logo Draw.io & Lucidchart 40
Figure 28: Logo PlantUML 40
Figure 29: Vue d'ensemble de la maquette 41
Figure 30: Page de Bienvenue 42
Figure 31: Page de Connexion 42
Figure 32: Différentes Fonctions 42
Figure 33: Interface Admin 42
Figure 34: Interface professeur 43
Figure 35: Interface Etudiant 43
Figure 36: Vue d'ensemble de l'architecture 46
Figure 37: Liste des tables de base de données 46
Figure 38: Chaîne logique de traitement d'une requête HTTP dans une API REST 48

Liste des tableaux

Tableau 1: Résumé des acteurs et leurs rôles 18
Tableau 2: Rôle utilisateur/Droits principaux 49

INTRODUCTION GÉNÉRALE

Dans un contexte où la transformation numérique occupe une place centrale dans l'amélioration de l'efficacité organisationnelle, la digitalisation des processus pédagogique représente un enjeu majeur pour les établissements d'enseignement supérieur.

C'est dans cette perspective que s'inscrit le **projet PACTE**, dont l'objectif est de concevoir et de développer une application web de gestion de planning universitaire. Cette solution vise à optimiser la planification des cours, la réservation des salles ainsi que la coordination entre enseignants, étudiants et administration au sein de l'école **HESTIM**.

Ce projet mobilise un ensemble de compétences acquises au cours de la formation, notamment en développement web, en modélisation des données et en conception d'architectures logicielles. Il met en pratique les méthodologies de conception (**MCD, MLD, UML**), les principes de gestion des bases de données relationnelles et l'intégration d'un backend connecté à une base de données.

L'architecture choisie repose sur une approche moderne et découplée, articulée autour d'une **API REST** développée avec **Node.js** et d'un frontend réalisé en **React** sous forme de Single Page Application (**SPA**). Ce modèle, souvent qualifié d'architecture Client–Serveur Moderne (**CSM**), permet une séparation claire des responsabilités, favorise la maintenabilité du code et autorise une évolution indépendante des couches frontend et backend. Il garantit également une meilleure performance et une expérience utilisateur plus fluide.

Au-delà de sa dimension technique, le **projet PACTE** répond à une problématique réelle de gestion efficace et transparente des ressources pédagogiques. Il contribue à améliorer la communication entre les différents acteurs de l'établissement et illustre la manière dont les technologies web contemporaines peuvent être mises au service de la gestion académique et institutionnelle.

Chapitre 1 : Cahier des Charges

1.1- Introduction

Avant toute conception technique, il est essentiel de définir le cadre général du projet, ses objectifs, les besoins à satisfaire et la méthode à suivre. Le cahier des charges constitue la base du processus de développement en précisant les aspects fonctionnels, organisationnels et techniques qui orienteront la suite du travail. Dans le cadre du **projet PACTE**, il permet d'identifier les attentes des utilisateurs, de les traduire en exigences concrètes et de déterminer les contraintes ainsi que la méthodologie de conception et de réalisation. Ce chapitre présente donc la description générale du projet, ses objectifs, la démarche adoptée, le planning de mise en œuvre et la problématique à laquelle cette solution entend répondre.

1.1- Présentation du Projet

Ce projet a pour objectif de concevoir et de développer une plateforme web intelligente dédiée à la planification automatique des cours, à la gestion des réservations de salles et à la synchronisation des emplois du temps des enseignants et des étudiants. La solution s'inscrit dans une démarche d'optimisation des ressources pédagogiques notamment les salles, les enseignants et les créneaux horaires tout en garantissant une expérience utilisateur fluide, moderne et intuitive.

Le projet place l'équipe de développement dans la position d'une cellule de conception logicielle chargée de mettre en œuvre un système intelligent de gestion et de réservation de salles. Ce système devra centraliser l'ensemble des données liées à la planification, prévenir les conflits d'horaires et proposer une interface ergonomique adaptée aux besoins de chaque utilisateur.

1.2- Objectifs du Projet

1.2.1- Objectif général

L'objectif principal du projet PACTE est de concevoir et développer une application web intelligente permettant la **planification automatisée des cours** et la **gestion des réservations de salles** au sein d'un établissement académique.

Cette plateforme doit permettre de centraliser toutes les informations liées aux enseignants, aux étudiants, aux groupes, aux créneaux horaires et aux ressources disponibles, afin de garantir une organisation optimale et sans conflits des emplois du temps.

1.2.2- Objectif spécifique

Pour atteindre cet objectif global, plusieurs objectifs spécifiques ont été définis :

- Automatiser la **planification des cours** en fonction des disponibilités des enseignants, des salles et des groupes.
- Éviter les **conflits d'horaires** (même salle, même enseignant, même groupe sur un créneau donné).
- Offrir une **interface conviviale et réactive** adaptée aux différents profils d'utilisateurs (administrateur, enseignant, étudiant).
- Mettre en place une **base de données centralisée** assurant la cohérence et la traçabilité des informations.
- Intégrer un **système de notification** pour informer les utilisateurs des modifications ou annulations de cours.
- Garantir la **sécurité et la fiabilité** des données grâce à une authentification sécurisée et une bonne gestion des droits d'accès.
- Faciliter la **consultation en temps réel** des emplois du temps actualisés.

1.3- Conduite du projet

1.3.1- Méthodologie

Pour la conduite de ce projet, nous avons retenu le **cycle de développement en V**, un modèle structuré qui met en relation directe les étapes de **conception** et celles de **validation**. Cette approche repose sur une démarche séquentielle où chaque phase de conception possède son équivalent en phase de test, garantissant ainsi une **traçabilité complète** des exigences et une **qualité maîtrisée** tout au long du processus de développement.

Le projet **HESTIM Planning** s'y prête particulièrement bien, car il requiert une **analyse rigoureuse des besoins fonctionnels et techniques**, suivie d'une **implémentation méthodique** et de **vérifications continues**.
Ainsi, après chaque étape clé (spécification, conception, implémentation), des **tests de validation** sont réalisés afin d'assurer la conformité du produit final avec les attentes initiales.

Cette méthode permet donc de **minimiser les erreurs** et de **sécuriser le processus de développement**, tout en assurant une **meilleure communication** entre les membres de l'équipe projet. Elle s'adapte parfaitement au projet **HESTIM Planning**, qui nécessite à la fois une **modélisation précise** (via MERISE et UML) et une **implémentation rigoureuse** à travers un backend Node.js et un frontend **React**.

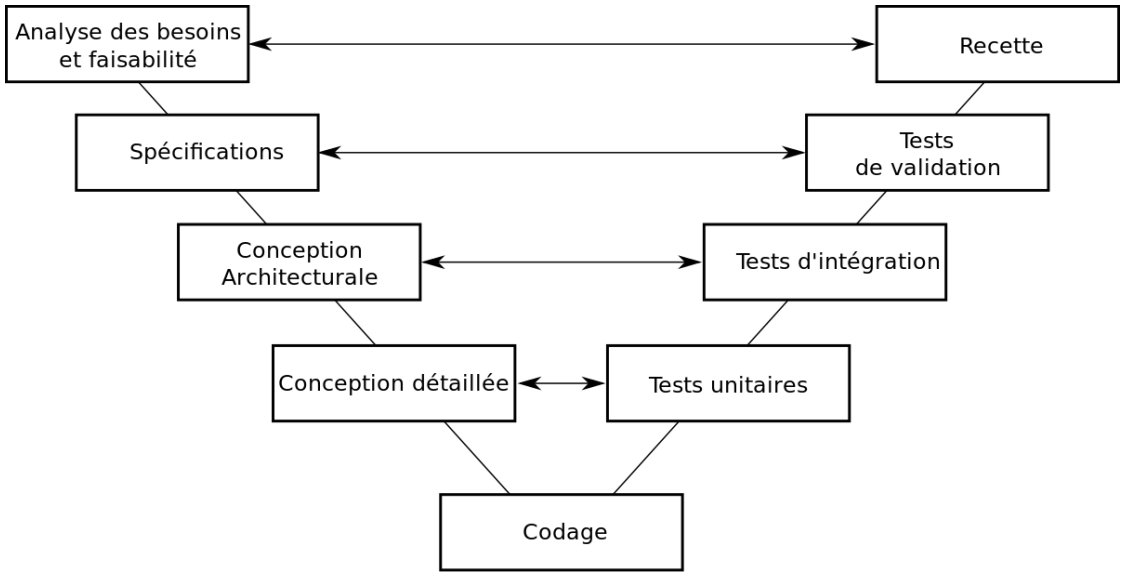

Figure 1 : Cycle de développement en V

1.3.2- Planning du projet

La planification du projet *HESTIM Planning* repose sur une organisation rigoureuse des différentes étapes de conception, de développement et de validation.

Le **diagramme de Gantt** ci-dessous présente la répartition chronologique des tâches réalisées tout au long du projet.

Cette planification a permis de suivre l'avancement des travaux, de coordonner les activités entre les membres de l'équipe, et de garantir le respect des délais fixés. 1.3.2.1- Diagramme de Gantt

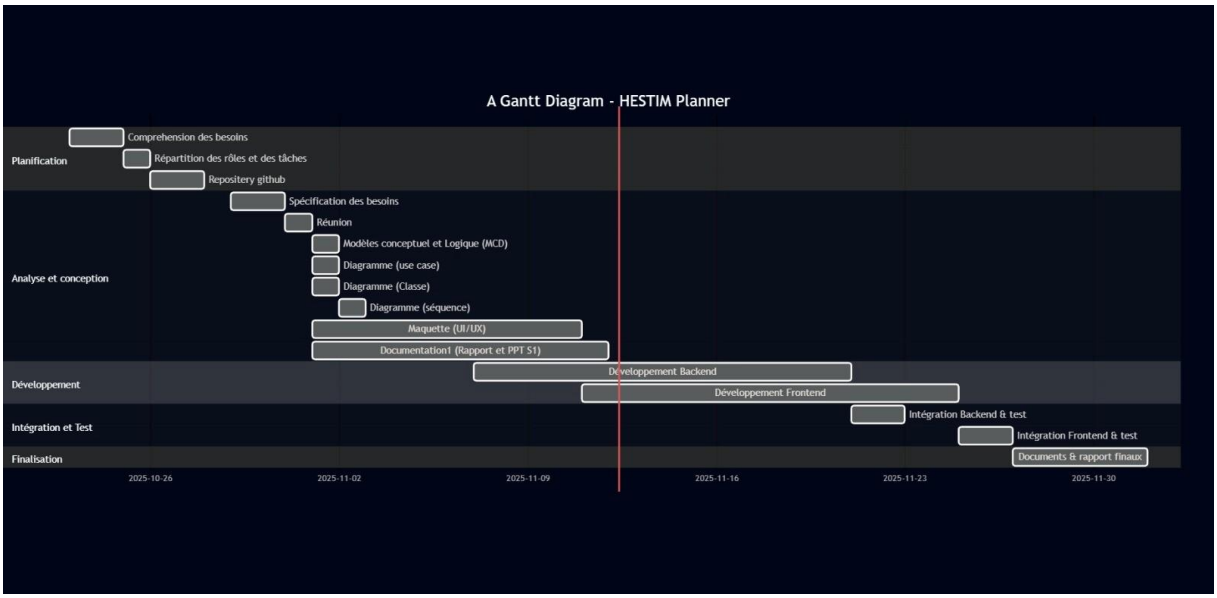

Figure 2 : Diagramme de Gantt

1.3.2.2- Explication des tâches du diagramme de Gantt

Le diagramme de Gantt ci-dessus présente la planification globale du projet HESTIM Planner. Il met en évidence les différentes phases, leurs dépendances et leur chronologie. Le projet est organisé en cinq grandes étapes : planification, analyse et conception, développement, intégration et test, et finalisation.

1. **Planification**

Cette phase prépare la base du projet :

- Compréhension des besoins (2 jours) : analyse du problème à résoudre, recueil des besoins fonctionnels et techniques.
- Répartition des rôles et des tâches (1 jour) : définition claire des responsabilités et tâches de chaque membre de l'équipe.
- Création du repository GitHub (2 jours) : mise en place de l'environnement collaboratif pour le code source et le suivi du projet.

2. **Analyse et conception**

Étape essentielle pour structurer le projet avant le développement :

- Spécification des besoins (2 jours) : rédaction du cahier des charges détaillé.
- Réunion (1 jour) : validation des choix fonctionnels et techniques avec l'équipe.
- Modèles conceptuel et logique (1 jour) : conception du modèle de données (MCD).
- Diagrammes UML (use case, classe, séquence) : modélisation du comportement et de la structure du système.
- Maquette (UI/UX) (10 jours) : conception des interfaces graphiques et expérience utilisateur.
- Documentation 1 (Rapport et présentation S1) (11 jours) : production du premier livrable du projet, incluant le rapport et la présentation intermédiaire.

3. **Développement**

Cette phase consiste à concrétiser la conception :

- Développement backend (14 jours) : mise en place du serveur, de la base de données et des API nécessaires.
- Développement frontend (14 jours) : intégration des maquettes, création de l'interface et connexion au backend.

4. **Intégration et test**

Une étape cruciale pour garantir la qualité du système :

Intégration backend & test (2 jours) : vérification du bon fonctionnement du serveur et des API.

Intégration frontend & test (2 jours) : tests des interfaces utilisateur et validation de la communication entre les deux parties du système.

5. **Finalisation**

Clôture du projet et préparation du rendu :

Documents & rapport finaux (5 jours) : rédaction du rapport final, synthèse des résultats, préparation de la soutenance et dépôt du livrable complet.

1.4- Public cible

Le projet **HESTIM Planning** s'adresse principalement aux acteurs de l'établissement **HESTIM**, mais sa conception lui permet également d'être adaptée à d'autres structures d'enseignement supérieur. L'objectif est de répondre aux besoins spécifiques des différents profils d'utilisateurs impliqués dans la gestion des emplois du temps et des ressources pédagogiques.

1.4.1- Utilisateurs principaux

- **Les administrateurs** :
  Ils sont responsables de la gestion globale du système. Leur rôle consiste à créer, modifier et supprimer les comptes utilisateurs (enseignants, étudiants), à gérer les filières, groupes, salles, et à superviser l'ensemble des réservations. Ils assurent également la résolution des conflits d'emplois du temps et le suivi des activités.

- **Les enseignants** :
  Ils peuvent consulter leurs emplois du temps, déclarer leurs disponibilités, effectuer ou modifier des réservations de salles pour leurs cours et être notifiés en cas de modification ou de conflit horaire.

- **Les étudiants** :
  Ils ont accès à un espace personnel leur permettant de consulter leurs cours, leurs emplois du temps, et les éventuelles modifications ou annulations. Le système leur garantit une meilleure visibilité sur la planification académique.

1.4.2- Utilisateurs secondaires

- **Le personnel administratif et technique** :
  Chargé du support et de la maintenance du système, il veille au bon fonctionnement de la plateforme, à la mise à jour des données et à la sécurité du système.

- **Les responsables pédagogiques** :
  Ils peuvent analyser l'utilisation des ressources (taux d'occupation des salles, disponibilités des enseignants, répartition horaire des cours) afin d'optimiser la planification académique.

1.4.3- Enjeux pour le public cible

L'application vise à simplifier la **gestion des emplois du temps**, à **réduire les erreurs et les conflits** dans la planification, et à **améliorer la communication** entre les acteurs pédagogiques. Elle constitue ainsi un outil moderne d'organisation, de coordination et de pilotage des activités académiques au sein de l'établissement.

1.5- Problématique

Dans la plupart des établissements d'enseignement supérieur, la **planification des cours et la gestion des réservations de salles** constituent une tâche complexe et chronophage. Cette opération est souvent réalisée **manuellement** ou à l'aide de **fichiers Excel**, ce qui entraîne de nombreuses **difficultés organisationnelles** : chevauchements de cours, **conflits d'emplois du temps**, erreurs de saisie, ou encore **mauvaise répartition des ressources** (salles, enseignants, créneaux horaires).

De plus, la coordination entre les différents acteurs **administrateurs**, **enseignants** et **étudiants** devient difficile à mesure que le volume d'informations augmente. L'absence d'un système centralisé et automatisé conduit à une **perte de temps considérable** et à un manque **de visibilité globale** sur la planification.

Le projet **HESTIM Planning** vise à répondre à ces problématiques en proposant une **solution web intelligente et intégrée**, capable d'**automatiser la planification**, de **gérer efficacement les réservations de salles**, et d'**éviter les conflits d'horaires** grâce à une logique de détection automatisée.
L'enjeu principal est donc de concevoir un système **fiable, ergonomique et évolutif**, qui facilite le travail des administrateurs tout en offrant aux enseignants et étudiants un accès simple et instantané à leurs emplois du temps actualisés.

1.6- Conclusion

Ce premier chapitre a permis de définir le **cadre général du projet HESTIM Planning** en précisant ses objectifs, sa méthodologie, son public cible ainsi que la problématique à laquelle il répond. L'élaboration du **cahier des charges** a constitué une étape essentielle pour cerner les besoins réels des utilisateurs et poser les fondations du futur système.

En choisissant une **méthodologie de développement en V**, le projet s'inscrit dans une approche rigoureuse et structurée, garantissant une bonne traçabilité entre les phases de conception, de développement et de validation. Cette analyse préliminaire servira de base à la **phase suivante**, consacrée à l'**analyse fonctionnelle et à la conception technique** du système. Le prochain chapitre détaillera ainsi les besoins identifiés, les modèles conceptuels et logiques, ainsi que les différents diagrammes UML utilisés pour formaliser la solution envisagée.

Chapitre 2 : Analyse et Conception

2.1- Introduction

Après avoir défini les objectifs, la méthodologie et les contraintes du projet dans le cahier des charges, cette deuxième étape vise à **analyser en profondeur les besoins fonctionnels et techniques** afin de poser les bases de la conception du système *HESTIM Planning*.
L'analyse et la conception constituent en effet des phases déterminantes du processus de développement, car elles assurent la **cohérence entre les attentes des utilisateurs et la structure technique de la solution**.

Dans un premier temps, nous identifierons les **besoins fonctionnels et non fonctionnels** du système, avant de passer à la **modélisation conceptuelle et logique** des données à travers les approches **MERISE** et **UML**.
Cette double démarche permettra de représenter à la fois la structure des informations (MCD, MLD) et le comportement du système (cas d'utilisation, diagrammes de classes et de séquence).
Ces modèles serviront de **référentiel de conception** pour le développement du projet et garantiront une meilleure compréhension entre les différents intervenants.

Ainsi, ce chapitre débutera par une **analyse détaillée des besoins** du système, qui constituera la base de la conception à venir. Cette analyse permettra de dégager les fonctionnalités essentielles, les contraintes à respecter, et les interactions principales entre les utilisateurs et l'application.

2.2- Analyse des besoins

2.2.1- Les besoins fonctionnels

Les besoins fonctionnels regroupent l'ensemble des fonctionnalités que le système doit offrir pour répondre aux attentes des utilisateurs. Ils traduisent de manière concrète le comportement attendu de l'application et constituent le cœur même du développement du projet *HESTIM Planning*.
L'objectif est de permettre une gestion efficace, fluide et automatisée des emplois du temps et des réservations de salles au sein de l'établissement.

Ainsi, les principaux besoins fonctionnels identifiés sont les suivants :

- **Authentification et gestion des rôles**
  Le système doit permettre la connexion sécurisée des utilisateurs selon leur profil (administrateur, enseignant ou étudiant). Chaque rôle disposera d'un niveau d'accès et de fonctionnalités spécifiques.
- **Gestion des utilisateurs**

L'administrateur doit pouvoir créer, modifier, supprimer et consulter les comptes utilisateurs. Les informations personnelles telles que le nom, le prénom, l'adresse Email et le rôle doivent être correctement gérées.

- **Gestion des filières et groupes**
  L'application doit offrir la possibilité de créer, modifier et supprimer des filières ainsi que des groupes d'étudiants. Chaque groupe est associé à une filière et à un niveau d'étude.
- **Gestion des cours**
  Le système doit permettre la gestion complète des cours, incluant la création, la modification et la suppression, ainsi que l'association des cours à une filière, un niveau et un semestre précis.
- **Gestion des salles**
  L'administrateur doit pouvoir gérer les salles (ajouter, modifier, désactiver) en renseignant leur type (amphi, laboratoire, salle informatique, etc.), leur capacité, leur localisation et leurs équipements.
- **Gestion des créneaux horaires**
  Le système doit permettre la définition des créneaux horaires disponibles pour la planification des cours, en précisant les jours, heures de début et de fin, ainsi que la période (matin, après-midi, soir).
- **Réservation automatique**
  L'administrateur doit pouvoir effectuer ou valider des réservations de cours en s'assurant qu'aucun conflit d'horaire n'existe entre les salles, les enseignants ou les groupes.
- **Détection des conflits**
  Le système doit être capable d'identifier automatiquement les conflits liés à une double réservation de salle, d'enseignant ou de groupe, et d'en informer l'utilisateur concerné.
- **Notification des utilisateurs**
  Chaque modification, création ou annulation de cours doit générer une notification à destination des utilisateurs concernés (enseignants et étudiants).
- **Consultation des emplois du temps**
  Les enseignants et les étudiants doivent pouvoir consulter leur emploi du temps personnel de manière claire.
- **Gestion des disponibilités**
  Les enseignants doivent pouvoir renseigner leurs créneaux de disponibilité ou d'indisponibilité, afin de faciliter la planification automatique des séances.
- **Historique des réservations**

Le système doit conserver un historique complet de toutes les réservations, incluant les créations, modifications et suppressions, afin d'assurer une traçabilité et une transparence dans la gestion des emplois du temps.

En somme, ces fonctionnalités assurent une interaction fluide entre les différents utilisateurs tout en garantissant une organisation optimale des ressources pédagogiques et matérielles. Elles constituent le fondement fonctionnel du système *HESTIM Planning*.

Ces fonctionnalités constituent le cœur du système **HESTIM Planning**. Elles garantissent une gestion efficace des emplois du temps, tout en assurant la fluidité des interactions entre les différents acteurs du projet.

2.2.2- Les besoins non fonctionnels

Les besoins non fonctionnels définissent les critères de qualité et les contraintes techniques que le système HESTIM Planning doit respecter pour garantir son bon fonctionnement, sa performance et sa maintenabilité. Ils ne concernent pas directement les fonctionnalités visibles par les utilisateurs, mais influencent fortement la fiabilité et la convivialité de l'application.

Les principaux besoins non fonctionnels du projet sont les suivants :

- **Performance et réactivité**
  L'application doit assurer un temps de réponse rapide lors de la consultation des emplois du temps, de la création ou de la mise à jour d'une réservation. Le chargement des données doit être optimisé pour offrir une expérience fluide, même en cas de forte utilisation.
- **Sécurité et confidentialité**
  Les données des utilisateurs doivent être protégées contre tout accès non autorisé. Les mots de passe doivent être stockés sous forme chiffrée, les sessions sécurisées, et les échanges entre le client et le serveur protégés (HTTPS).
  De plus, la gestion des rôles et permissions doit garantir un accès restreint aux seules fonctionnalités autorisées selon le profil (administrateur, enseignant, étudiant).
- **Fiabilité et disponibilité**
  Le système doit être stable et fonctionnel à tout moment. Il doit être capable de gérer les erreurs sans interruption du service et de sauvegarder régulièrement les données pour éviter toute perte en cas d'incident.
- **Ergonomie et accessibilité**
  L'interface utilisateur doit être claire, intuitive et facile à prendre en main pour tous les profils d'utilisateurs. Une attention particulière doit être portée à la lisibilité des informations, à la cohérence graphique et à la simplicité des interactions.
- **Compatibilité et portabilité**
  L'application doit être compatible avec les principaux navigateurs web (Chrome, Edge,

  Firefox, etc.) et s'adapter aux différents supports (ordinateur, tablette, smartphone) grâce à un design responsive.
- **Évolutivité et maintenabilité**

Le code doit être structuré et documenté de manière à faciliter les futures évolutions du système. Il doit être possible d'ajouter de nouvelles fonctionnalités ou d'effectuer des corrections sans impacter la stabilité générale du projet.

- **Traçabilité et journalisation**
  Toutes les actions critiques (créations, modifications, suppressions, réservations, détections de conflits, etc.) doivent être enregistrées dans une base d'historique, afin de permettre un suivi et un audit complet des opérations réalisées.
- **Sauvegarde et restauration**
  Des mécanismes de sauvegarde automatique de la base de données doivent être prévus pour assurer la récupération rapide des informations en cas de défaillance du système.
- **Respect des normes et standards**
  Le développement du projet doit suivre les bonnes pratiques de conception logicielle, notamment en matière d'architecture (**MVC**), de normalisation de la base de données et de structuration du code selon les conventions des Frameworks utilisés.

En résumé, ces besoins non fonctionnels garantissent la robustesse, la sécurité et la pérennité du système HESTIM Planning. Ils assurent non seulement une utilisation agréable pour les utilisateurs finaux, mais aussi une maintenance aisée pour les développeurs et administrateurs du projet.

2.3- Conception hybride avec MERISE et UML

2.3.1- Identification des acteurs

L'identification des acteurs constitue une étape essentielle de la phase d'analyse fonctionnelle. Elle permet de déterminer les différentes parties prenantes qui interagiront avec la plateforme de planification, de gestion et de réservation de salles. Chaque acteur possède des rôles, des droits et des responsabilités spécifiques dans le système.

1. Administrateur

Rôle : L'administrateur est le principal gestionnaire du système. **Responsabilités :**

- Créer, modifier ou supprimer les comptes utilisateurs (enseignants, étudiants).
- Gérer les salles (ajout, suppression, mise à jour de la capacité et du type).
- Superviser la planification globale et résoudre les conflits éventuels.
- Gérer les paramètres du système (semestre, modules, groupes, horaires).
- Générer des rapports et statistiques sur l'utilisation des salles.

Interactions avec le système :

- Accès complet à toutes les fonctionnalités.
- Visualisation globale des emplois du temps.
- Accès au tableau de bord analytique.

2. Enseignant

Rôle : L'enseignant est un utilisateur qui interagit avec la plateforme pour planifier ou consulter ses cours. **Responsabilités :**

- Consulter son emploi du temps personnel.
- Effectuer des demandes de réservation de salles selon sa disponibilité.
- Modifier ou annuler un cours si nécessaire.

Interactions avec le système :

- Consultation des créneaux disponibles.
- Réservation de salles pour un module donné.
- Accès restreint à ses propres cours et plannings.

3. Étudiant

Rôle : L'étudiant est un utilisateur qui consulte les emplois du temps pour suivre ses cours. **Responsabilités :**

- Consulter les emplois du temps de son groupe.
- Recevoir des notifications en cas de modification de planning.
- Accéder aux informations sur les salles et les enseignants.

Interactions avec le système :

- Lecture seule du planning.
- Accès personnalisé selon le groupe ou la promotion.

Tableau 1: Résumé des acteurs et leurs rôles

Acteur	Rôle principal	Accès/Droits
Administrateur	Supervision et gestion globale du système	Lecture / Écriture / Suppression
Enseignant	Réservation et gestion de ses propres cours	Lecture / Écriture limitée
Étudiant	Consultation des plannings	Lecture seule

2.3.2- Modèle conceptuel des données (MCD)

Le **Modèle Conception de données (MCD)** du projet **HESTIM Planning** définit la structure fondamentale de l'information du système ainsi que les relations existantes entre ses différentes composantes métier. Il repose sur la représentation des principaux auteurs et ressources impliqués dans la gestion des emplois du temps universitaires : **utilisateurs (administrateurs, enseignants, étudiants), filières, groupes, cours, salles, créneaux horaires et séances planifiées.**

Dans la nouvelle organisation fonctionnelle validée par l'endrament, la notion de *réservation* est remplacée par celle d'**affectation**, qui reflète fidèlement le processus réel :

- **Seuls les membres de l'administration réalisent les affectations** de salles et créneaux pour les séances, après réception des besoins exprimés par les enseignants et tenant compte des disponibilités déclarées ainsi que des contraintes de ressources.

L'entité **Affectation** constitue désormais le **noyau central du système**. Elle matérialise une séance programmée reliant obligatoirement :

- Un **Cours**,
- Un **Groupe d'étudiants**,
- Un **Enseignant**,
- Une **Salle**,
- Un **Créneau horaire**,
- Et un **Utilisateur administrateur responsable** de l'affectation.

Ce lien multi-ressources est représenté à traves l'association **AFFECTER** (association **6-aire**), qui génère une entité forte *Affectation*. Chaque ressource peut participer à plusieurs affectations au fil du temps, tandis qu'une affectation se rapporte toujours à **une instance unique de chacune des entités citées.**

Fonctionnalités métier intégrées dans le MCD

Le modèle intègre également plusieurs fonctionnalités essentielles au fonctionnement opérationnel du système :

Gestion des disponibilités

L'association *DISPONIBILITE* permet aux enseignants de déclarer leurs périodes de disponibilité ou d'indisponibilité pour chaque créneau horaire. Ces données sont exploitées lors de la phase d'affectation afin de limiter les conflits de planning et d'assurer une planification cohérente.

Demandes de report

L'entité *DemandeReport* permet aux enseignants de solliciter un report de séance en cas d'empêchement ponctuel.

Chaque demande :

- Est initiée par un enseignant,
- Concerne une affectation spécifique,
- Précise un **motif**, une **nouvelle date proposée**, et un **statut de validation**,
- Est ensuite traitée par l'administration.

Détection et gestion des conflits

Les conflits de planification (double réservation de salle, indisponibilité d'un enseignant, chevauchement de créneaux pour un même groupe, etc.) sont représentés par l'entité **Conflit**, reliée aux séances concernées par l'association **CONFLIT_ENTRE**.

Un conflit peut impliquer deux ou plusieurs affectations simultanées et faire l'objet d'un traitement administratif.

Système de notifications

L'entité Notification assure la diffusion des informations aux utilisateurs :

- **Validation ou modification d'affectations,**
- **Annulations,**
- **Conflits détectés,**
- **Réponses aux demandes de report.**

Chaque notification est liée à un utilisateur unique, tandis qu'un utilisateur peut recevoir plusieurs notifications.

Traçabilité et audit

La traçabilité complète du système est assurée par l'entité HistoriqueAffectation, couplée à l'association HISTORISER, qui enregistre :

- **Chaque création,**
- **Modification,**
- **Suppression ou annulation d'une affectation,**
- **La date de l'action,**
- **Les anciennes et nouvelles valeurs,**
- **Le commentaire associé,**
- **Et éventuellement l'utilisateur responsable.**

Ce mécanisme garantit la transparence du système et permet un audit complet des décisions administratives.

Héritage des utilisateurs

Le modèle adopte une structure d'héritage IS-A à partir de l'entité générique User, spécialisée en :

- **Enseignant (dispose d'attributs métiers : spécialité, grade, département, etc.),**
- **Étudiant (identification académique et niveau de formation).**

L'administrateur ne possède pas de sous-classe spécifique : il est distingué par la valeur de l'attribut *role* au sein de l'entité User.

Représentation graphique

Le MCD a été modélisé à l'aide de **PlantUML**, suivant une convention visuelle claire :

- **Entités** : fond vert clair #E8F5E9,
- **Associations** : fond orange clair #FFE4B5,
- **Relations d'héritage** : flèche de spécialisation IS-A,
- **Cardinalités explicites dans les deux sens**, permettant une lecture complète de chaque relation selon la norme MERISE.

L'ensemble du diagramme décrit avec précision l'organisation informationnelle du système, sert de référence fonctionnelle et constitue la base conceptuelle du Modèle

Logique de Données (MLD) destiné à l'implémentation relationnelle du projet sous MySQL.

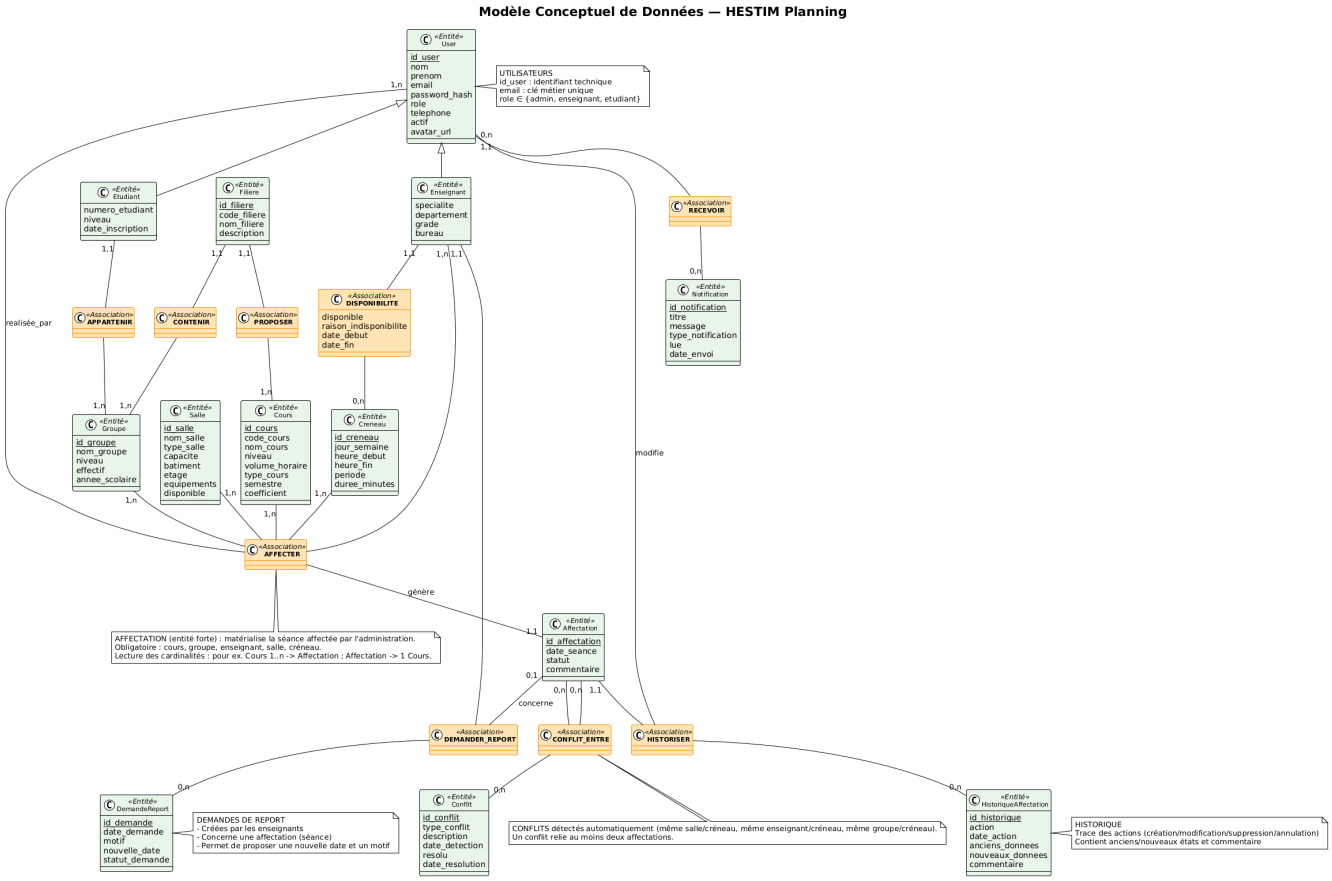

Figure 3: Modèle conceptuel de données (MCD)

2.3.3- Modèle logique des données (MLD)

Le **Modèle Logique des Données (MLD)** traduit le MCD en une structure relationnelle exploitable par le SGBD **MySQL** via l'ORM **Sequelize**. Chaque entité conceptuelle devient une table dotée d'une **clé primaire**, tandis que les relations sont implémentées par **des clés étrangères** assurant l'intégrité référentielle.

L'entité centrale du système est **Affectations**, qui représente une séance planifiée et relie les principales ressources : **Cours, Groupe, Enseignant, Salle, Créneau**, ainsi que l'**Administrateur** ayant validé la programmation.

La gestion des utilisateurs repose sur la table **Users**, spécialisée en **Enseignants** et **Etudiants** par héritage. L'organisation pédagogique est structurée autour des tables **Filières**, **Groupes** et **Cours**. Les ressources matérielles sont gérées via les tables **Salles** et **Créneaux**, tandis que les disponibilités des enseignants sont représentées par la table **Disponibilités**.

Les fonctionnalités avancées sont prises en charge par plusieurs tables complémentaires :

- **Demandes_Report** : gestion des demandes de report de séances,
- **Conflits** et **Conflits_Affectations** : détection et suivi des conflits de planning,
- **Notifications** : information des utilisateurs,
- **Historique_Affectations** : traçabilité complète des opérations effectuées sur les séances.

Ce MLD garantit une **base de données normalisée**, cohérente avec les besoins du projet **HESTIM Planning**, permettant une gestion fiable des plannings universitaires et une implémentation efficace du backend.

Modèle Logique de Données — HESTIM Planning

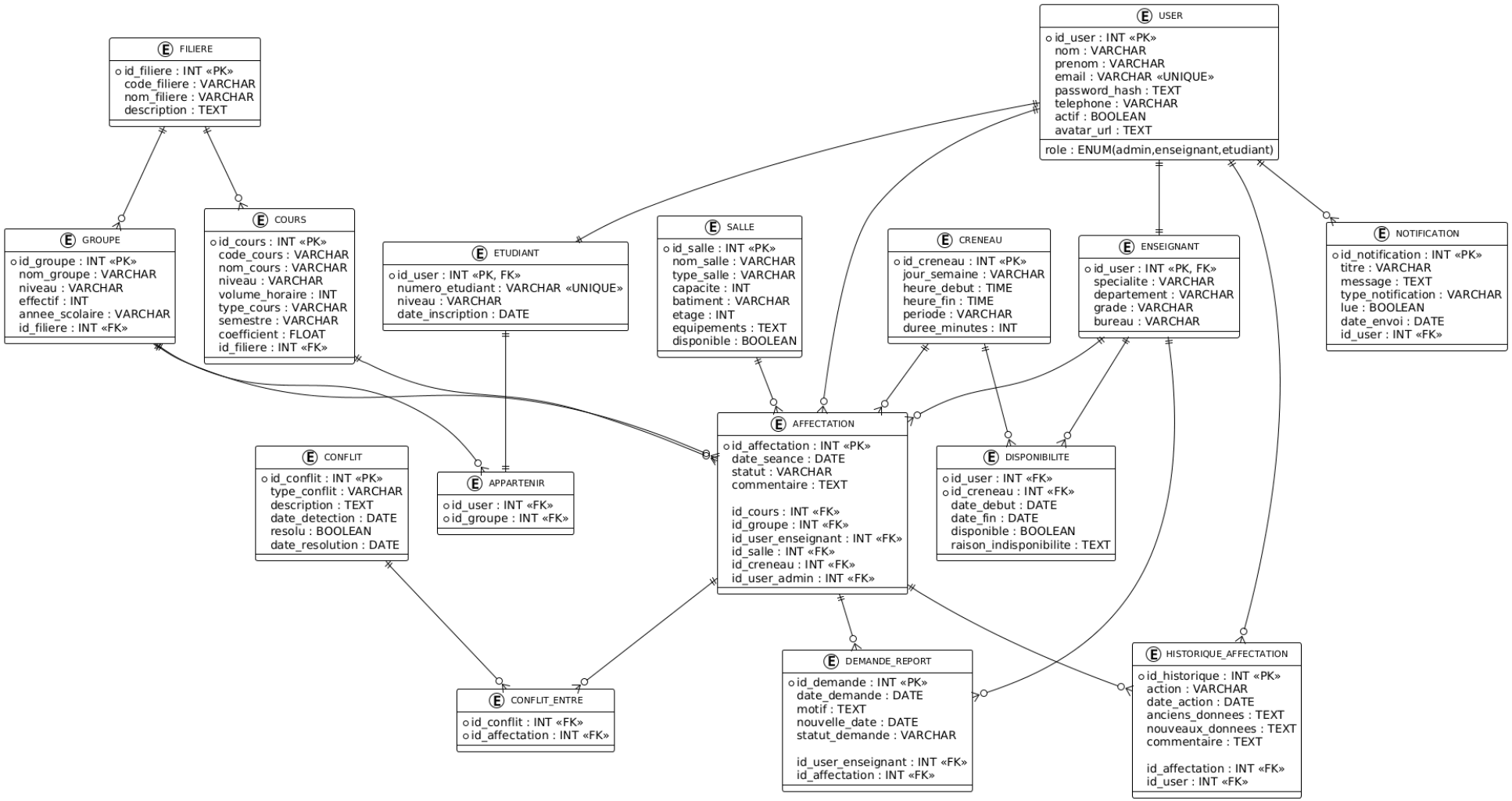

Figure 4: Modèle logique de données (MLD)

2.3.4- Diagramme de cas d'utilisation

Le **diagramme de cas d'utilisation** du projet *HESTIM Planning* illustre les différentes interactions entre les acteurs du système et les fonctionnalités principales de la plateforme. Trois acteurs interagissent avec le système : **l'administrateur**, **l'enseignant** et **l'étudiant**. L'administrateur dispose d'un accès complet lui permettant de gérer les utilisateurs, les cours, les groupes, les salles et la planification automatique. L'enseignant peut gérer ses disponibilités, créer ou modifier ses réservations, et consulter son emploi du temps. L'étudiant, quant à lui, peut consulter les emplois du temps, rechercher des salles disponibles et recevoir des notifications.
Le diagramme met également en évidence les liens d'**inclusion** (pour la détection automatique des conflits) et d'**extension** (pour les fonctions de filtrage et d'exportation de l'emploi du temps).

Ce diagramme offre une vision globale et claire des fonctionnalités offertes par la plateforme *HESTIM Planning* selon les rôles des utilisateurs.

Il montre comment le système centralise la gestion des ressources pédagogiques tout en automatisant les processus de planification et de résolution des conflits.
Ce modèle fonctionnel constitue la base du développement de l'interface et de la logique métier, garantissant une répartition cohérente des droits et une expérience utilisateur fluide pour chaque acteur du système.

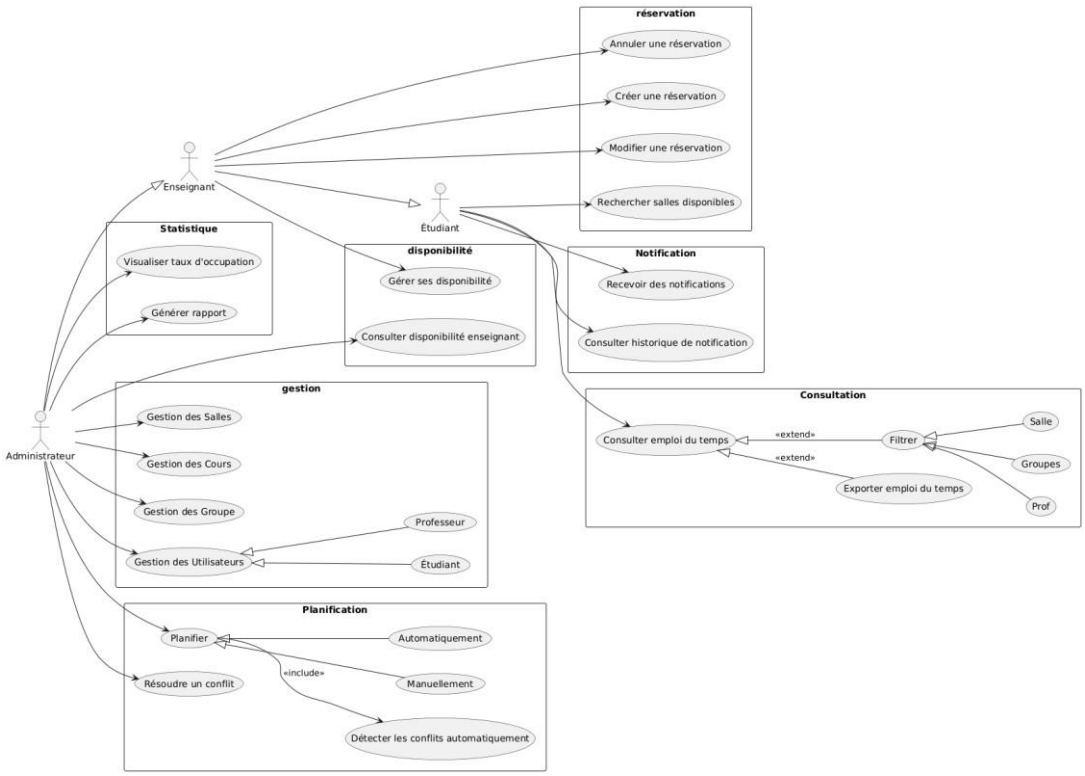

Figure 5: Diagramme de cas d'utilisation

2.3.5- Diagramme de classes

Le **diagramme de classes** du projet *HESTIM Planning* représente la structure statique du système en modélisant les entités logicielles, leurs attributs et leurs relations. Chaque classe correspond à une composante fonctionnelle de la plateforme, comme les utilisateurs, les cours, les groupes, les salles, ou encore les réservations. L'héritage entre les classes User, Enseignant et Etudiant permet de mutualiser les caractéristiques communes tout en conservant les spécificités de chaque profil.
La classe *Reservation* occupe une position centrale : elle relie les éléments académiques et logistiques du système (cours, enseignant, groupe, salle et créneau).
Des classes supplémentaires, comme *Conflit, Notification* et *DisponibiliteEnseignant*, assurent la détection automatique des conflits, la communication entre utilisateurs et la gestion des disponibilités.

Ce diagramme illustre l'architecture objet du système et sert de fondation à l'implémentation orientée objet du projet.
Il met en évidence la cohérence entre les entités du domaine et les classes applicatives, garantissant une conception claire, modulaire et maintenable.
Le diagramme de classes facilite ainsi le passage vers la phase de développement, en assurant la traçabilité entre les modèles conceptuels (MCD/MLD) et la structure logicielle de la plateforme *HESTIM Planning*.

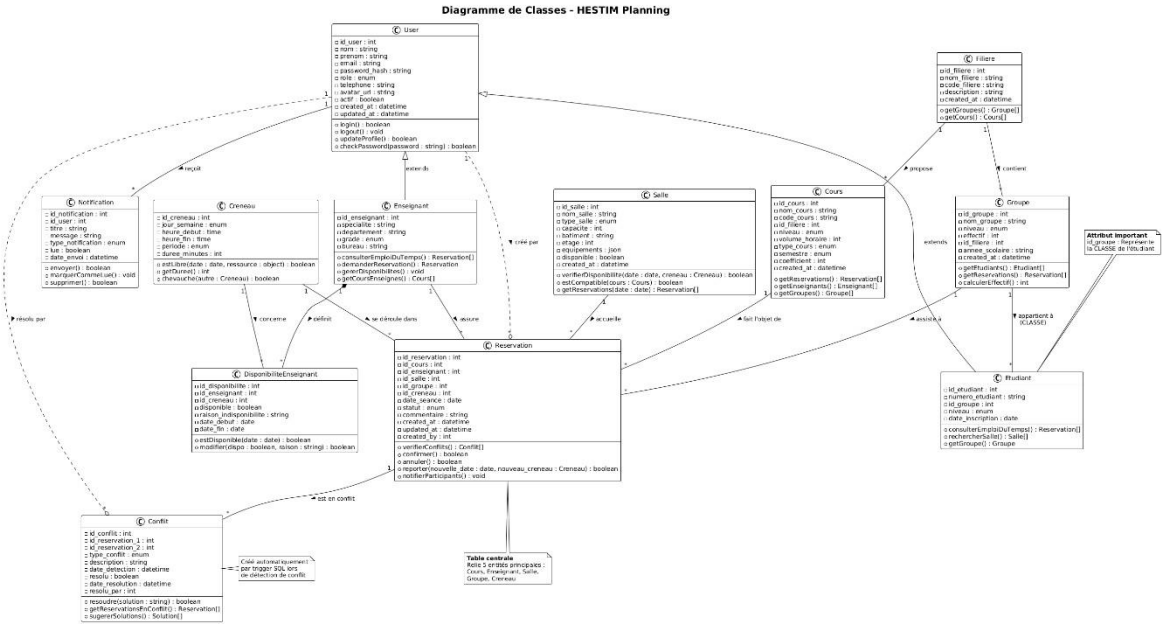

Figure 6: Diagramme de classes

2.3.6- Diagramme de séquence

Le **diagramme de séquence** du projet *HESTIM Planning* décrit le déroulement complet du processus de **création d'une réservation** par un administrateur.
Le scénario débute par la saisie des informations nécessaires via l'interface utilisateur, suivie de l'envoi d'une requête HTTP sécurisée à l'API.

L'authentification est ensuite vérifiée par un middleware avant la validation des données par le contrôleur de réservation. Un service spécialisé, **ConflictService**, analyse simultanément les éventuels conflits de salle, d'enseignant ou de groupe.
En l'absence de conflit, la réservation est enregistrée dans la base de données, les notifications correspondantes sont générées et envoyées aux utilisateurs concernés. Le processus se termine par un retour de succès affiché sur l'interface et la mise à jour du calendrier des réservations.

Ce diagramme met en évidence la **séquence d'interactions entre les différentes couches du système** (frontend, backend, services et base de données), tout en intégrant des mécanismes de **sécurité**, de **vérification des conflits** et de **notification automatique**. Il illustre la fluidité et la cohérence de la communication entre les composants, garantissant la fiabilité de la planification et la réactivité de la plateforme.
Ce modèle fonctionnel constitue ainsi une base solide pour l'implémentation du module de réservation et la synchronisation en temps réel des emplois du temps dans *HESTIM Planning*.

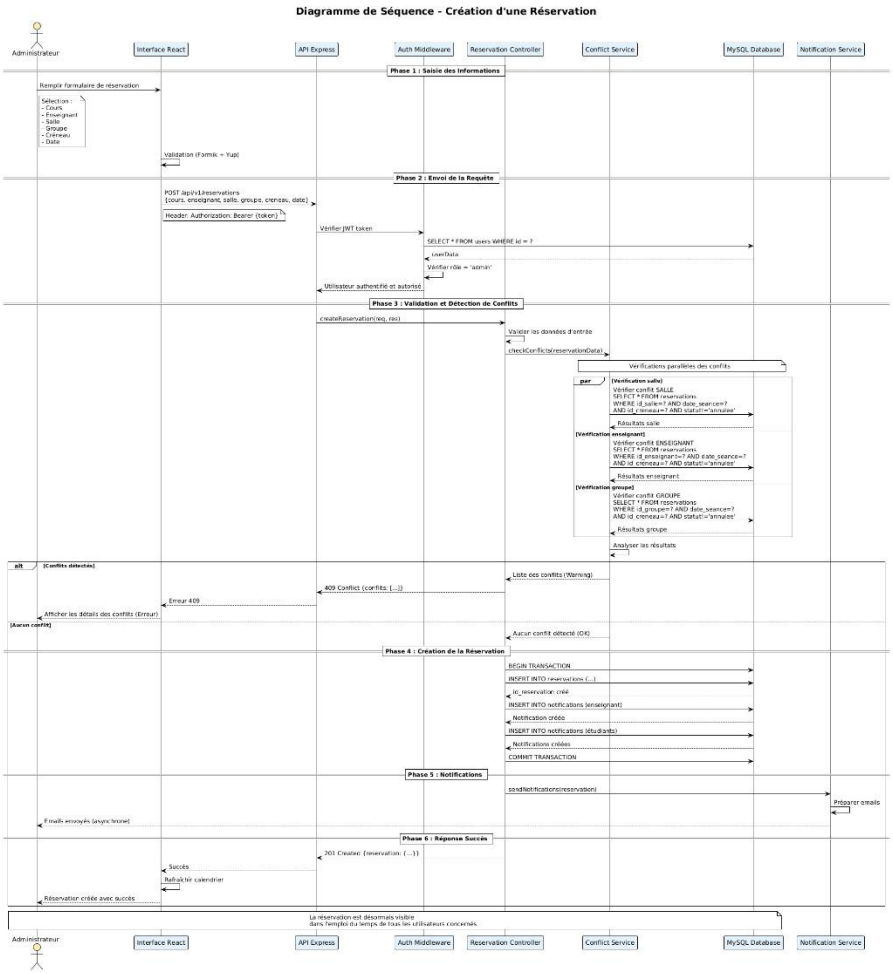

Figure 7: Diagramme de séquence

2.3.7- Diagramme d'activités

Le **diagramme d'activités** du projet *HESTIM Planning* illustre le déroulement du processus de **détection et de résolution des conflits** lors de la création d'une réservation. Le flux débute par la validation des données saisies, suivie d'une vérification parallèle des disponibilités de la salle, de l'enseignant et du groupe.
En cas de conflit détecté, le système enregistre les anomalies et notifie l'administrateur, qui peut soit modifier la réservation, soit la forcer, soit l'annuler.
S'il n'existe aucun conflit, le système procède à des validations complémentaires (comme la vérification de la capacité de la salle), puis crée la réservation et envoie les notifications correspondantes aux utilisateurs concernés.

Ce diagramme met en évidence la **logique métier complète** de la plateforme, intégrant les vérifications de cohérence, la gestion des erreurs et les actions correctives possibles. Il montre la coordination entre les processus automatiques et les interventions humaines dans la résolution des conflits, tout en assurant la traçabilité et la fiabilité des opérations.

Le modèle d'activité garantit ainsi une gestion fluide des réservations et contribue à la robustesse globale du système *HESTIM Planning*.

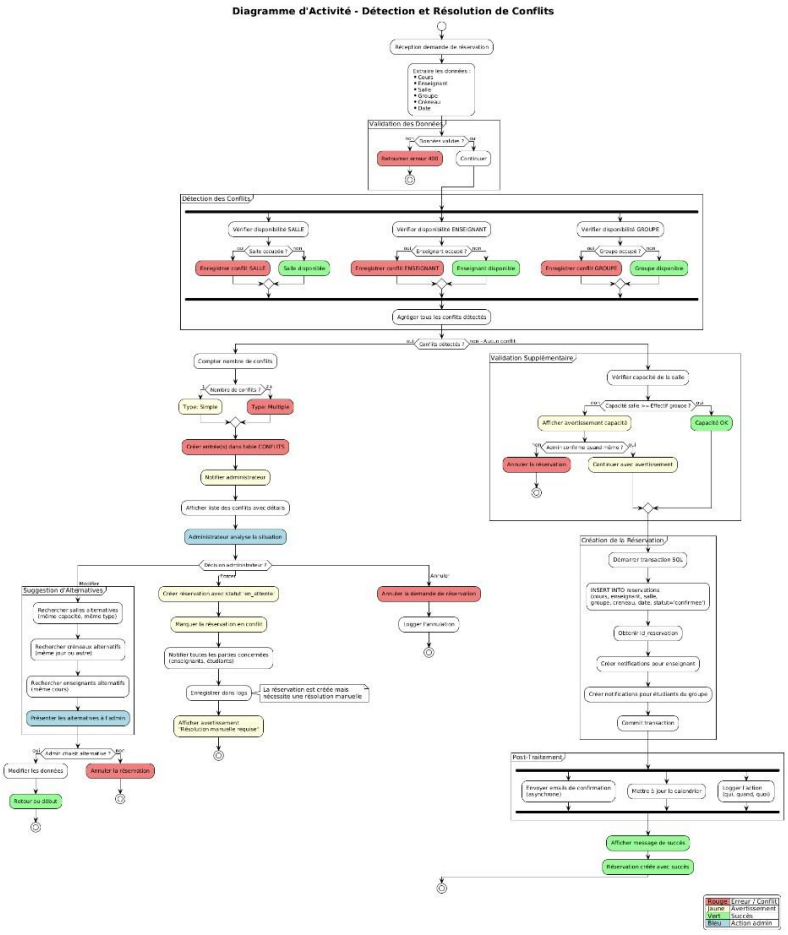

Figure 8: Diagramme d'activité - Détection et Résolution de Conflits

2.3.8- Diagramme d'état

Le diagramme d'état présenté ci-dessous modélise le **cycle de vie d'une réservation** au sein du système *HESTIM Planning*. Il permet de visualiser les différents **états successifs** qu'une réservation peut prendre, ainsi que les **transitions** déclenchées par des événements internes (validation, détection de conflits, modification, etc.) ou externes (actions de l'administrateur ou passage du temps).

Le processus débute par l'état **Brouillon**, correspondant à la création initiale de la réservation.
Durant cette phase, l'administrateur ou l'utilisateur autorisé remplit le formulaire de saisie. Une fois les données soumises et validées, la réservation passe à l'état **En attente**, où elle fait l'objet d'une vérification automatique des conflits (salle, enseignant, groupe, créneau, etc.) et éventuellement d'une approbation manuelle.

Si aucun conflit n'est détecté, ou après résolution des conflits, la réservation est alors **confirmée**.

L'état **Confirmée** constitue l'état principal du système : la réservation devient visible dans les emplois du temps des enseignants et des étudiants, et déclenche l'envoi automatique des **notifications** ainsi que la **mise à jour des calendriers**.

À partir de cet état, plusieurs transitions sont possibles :

- En cas de modification (salle, créneau, date), la réservation repasse temporairement par l'état **En attente** pour une nouvelle vérification.
- En cas de report, la réservation bascule vers l'état **Reportée**, avant d'être revalidée.
- Si elle est annulée, elle passe dans l'état **Annulée**, accompagné d'un motif enregistré.
- Enfin, lorsque la date de la séance est passée, elle devient **Terminée** et ses données sont transférées dans l'historique.

Les états **Terminée** et **Annulée** mènent tous deux à l'état **Archivée**, garantissant la **traçabilité** et la **conservation** des informations pour d'éventuelles analyses ou audits ultérieurs.

Ainsi, ce diagramme offre une vision dynamique du fonctionnement interne du module de réservation et met en évidence la **gestion intelligente des transitions**, assurant la cohérence et la fiabilité des données tout au long du processus.

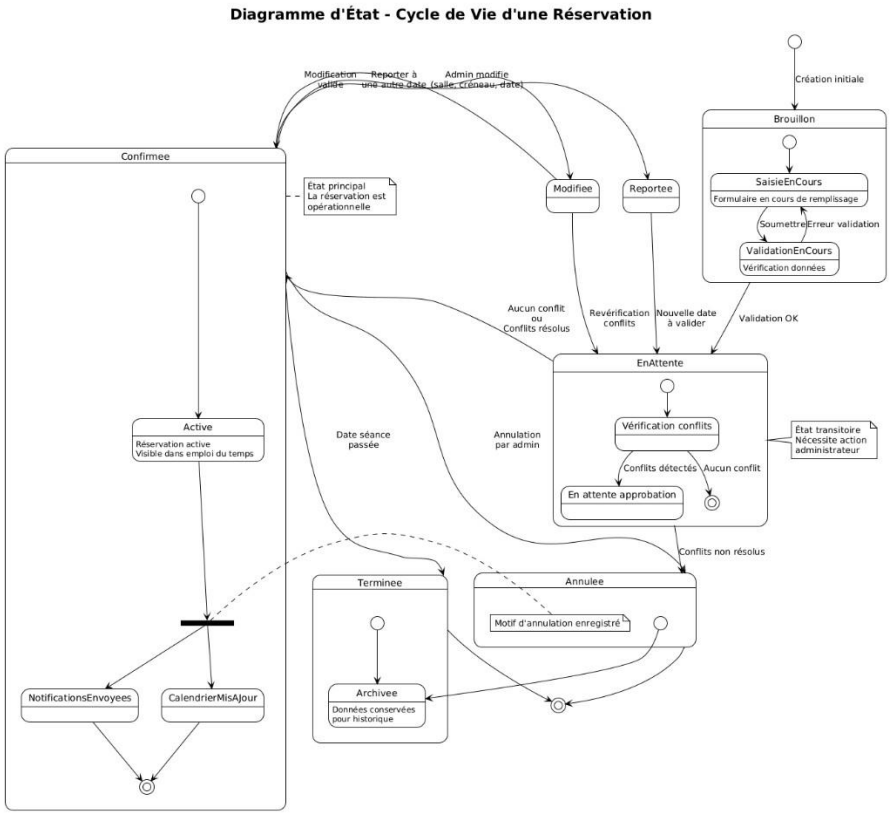

Figure 9: Diagramme d'état - Cycle de vie d'une réservation

2.4- Conclusion

Ce chapitre a permis de poser les fondations conceptuelles du projet *HESTIM Planning* à travers une analyse approfondie des besoins et une conception méthodique basée sur les approches hybride avec **MERISE** et **UML**.
L'étude des besoins fonctionnels et non fonctionnels a permis de cerner précisément les attentes des différents utilisateurs, tout en identifiant les contraintes techniques et organisationnelles à respecter.

La phase de conception a ensuite traduit ces besoins en modèles concrets et cohérents : le **Modèle Conceptuel de Données (MCD)** et le **Modèle Logique de Données (MLD)** ont défini la structure de la base de données, tandis que les **diagrammes UML** (cas d'utilisation, classes, séquence, états) ont décrit le comportement dynamique du système et les interactions entre ses composants.

Ces différentes représentations ont contribué à clarifier l'architecture fonctionnelle du projet, en assurant la cohérence entre les données, les traitements et les acteurs.
Elles constituent désormais une **base solide** pour la phase suivante, dédiée à la **mise en œuvre technique et à l'étude de l'architecture logicielle**, qui sera présentée dans le chapitre suivant.

Chapitre 3 : Architecture et étude technique

3.1- Introduction

Après avoir défini les besoins fonctionnels et non fonctionnels du système ainsi que les modèles conceptuels et logiques lors de la phase d'analyse et de conception, il est désormais essentiel d'aborder l'aspect technique du projet.
Ce chapitre a pour objectif de présenter l'**architecture globale** du système *HESTIM Planning*, en détaillant les différentes **couches logicielles**, leurs **interactions**, ainsi que les **technologies utilisées** pour la mise en œuvre.

L'étude de l'architecture permettra d'expliquer la répartition des rôles entre les composants du **frontend**, du **backend** et de la **base de données**, tout en justifiant les choix techniques adoptés.
Une attention particulière sera accordée à la **cohérence**, la **sécurité**, la **performance** et la **maintenabilité** du système, afin de garantir un fonctionnement fiable et évolutif.

Ce chapitre présentera également les **langages**, **Frameworks** et **outils de développement** utilisés pour construire la plateforme, ainsi que les **principes d'intégration** et de **communication** entre les différentes parties de l'application.
L'objectif est de démontrer comment les choix technologiques soutiennent les exigences fonctionnelles et assurent une architecture robuste, modulable et conforme aux standards du développement moderne.

3.2- Architecture du projet

Le projet *HESTIM Planning* repose sur une architecture **modulaire** et **n-tiers**, pensée pour assurer une bonne séparation des responsabilités, une évolutivité du système et une maintenance simplifiée.
Cette architecture s'articule principalement autour de trois couches : **le frontend**, **le backend**, et **la base de données**, interconnectées à travers une **API REST** sécurisée.

3.2.1- Diagramme de l'architecture globale

L'architecture globale du système peut être représentée de la manière suivante :

- **Frontend (Client Web)** : développé avec **React.js** (via Vite), il constitue l'interface utilisateur. Il gère la présentation, la navigation, l'interaction avec l'utilisateur et la communication avec le serveur via des requêtes HTTP (Axios / Fetch API).
  Le frontend adopte une approche **SPA (Single Page Application)**, ce qui permet une navigation fluide et rapide.
  Il intègre un système d'**authentification**, un **mode sombre/clair**, ainsi qu'un design moderne et ergonomique basé sur **Material UI (MUI)**.

- **Backend (Serveur d'application)** : conçu avec **Node.js** et **Express.js**, il joue le rôle d'intermédiaire entre la base de données et le client.

Le backend expose des **endpoints RESTful** permettant la gestion des utilisateurs, des enseignants, des étudiants, des salles, des réservations, des notifications et des conflits.
Il intègre également des **middlewares** pour la gestion de la sécurité (authentification JWT, validation des entrées, gestion des erreurs, CORS, etc.).

- **Base de données (Persistance)** : le stockage des données est assuré par **MySQL**, choisi pour sa fiabilité, sa compatibilité avec les environnements web et son intégration fluide avec Sequelize ORM.
Le modèle logique des données (MLD) issu de la conception MERISE y est implémenté, garantissant la cohérence et l'intégrité des informations (avec clés primaires, étrangères et contraintes de relations).

- **API REST** : l'API assure la communication entre les différentes couches du système. Elle transporte les données au format **JSON** et repose sur les méthodes standard **HTTP (GET, POST, PUT, DELETE)**.
Cette approche facilite la scalabilité du système et rend possible l'intégration future d'autres interfaces (mobile, tableau de bord admin, etc.).

- **Sécurité et Hébergement** :
Le système prévoit une gestion des accès basée sur les **rôles** (administrateur, enseignant, étudiant), ainsi qu'une sécurisation des échanges via **HTTPS** et des **tokens JWT**.
L'application est conçue pour être **déployée sur Vercel** (frontend) et **hébergée sur un serveur Node ou service cloud** pour le backend (ex. Render, Railway ou AWS).
La base de données peut être hébergée sur **phpMyAdmin / MAMP** en local ou sur **PlanetScale / MySQL Cloud** en production.

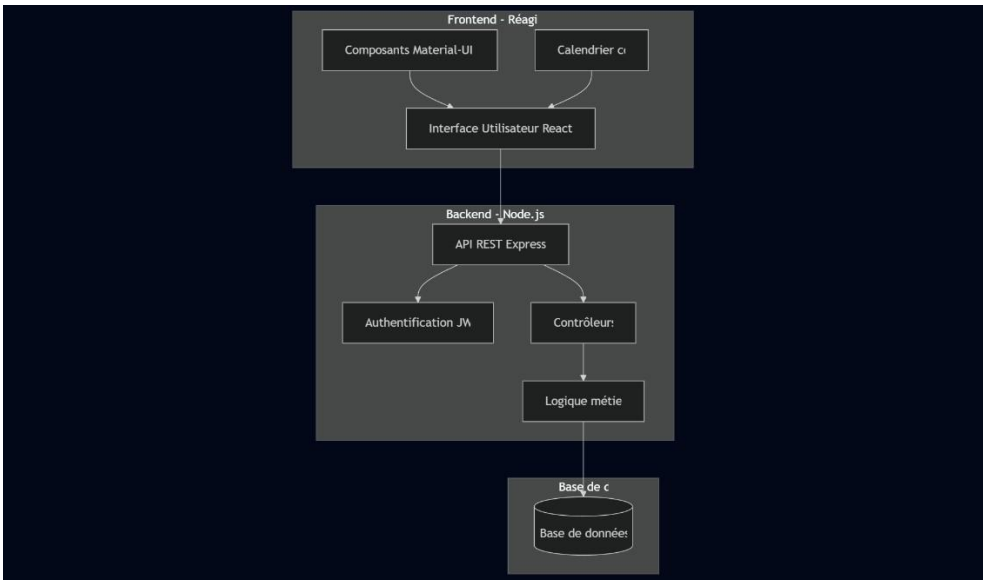

Figure 10: Architecture global

3.2.2- Composants principaux

L'architecture du projet s'organise autour de plusieurs **modules fonctionnels** interdépendants :

- **Module Utilisateurs** :

Gère l'authentification, la création de comptes, la gestion des rôles et les informations personnelles.

- **Module Enseignants et Étudiants** :

Permet la gestion des profils spécifiques à chaque catégorie, avec la spécialité, le groupe, le niveau, etc.

- **Module Cours et Filières** :

Centralise les informations relatives aux matières, filières, groupes et niveaux académiques.

- **Module Réservations** :

Cœur du système, il gère la planification des cours, l'affectation des salles, des créneaux et des enseignants.
Il inclut la détection de conflits, la gestion de l'historique et le suivi du cycle de vie des réservations.

- **Module Notifications** :

Informe automatiquement les utilisateurs des modifications, annulations ou nouveaux cours via des alertes personnalisées.

- **Module Conflits et Disponibilités** :

Assure la détection automatique des conflits d'emploi du temps et la gestion des indisponibilités des enseignants.

- **Module Historique** :

Archive les anciennes réservations et conserve les traces de modifications pour assurer la traçabilité.

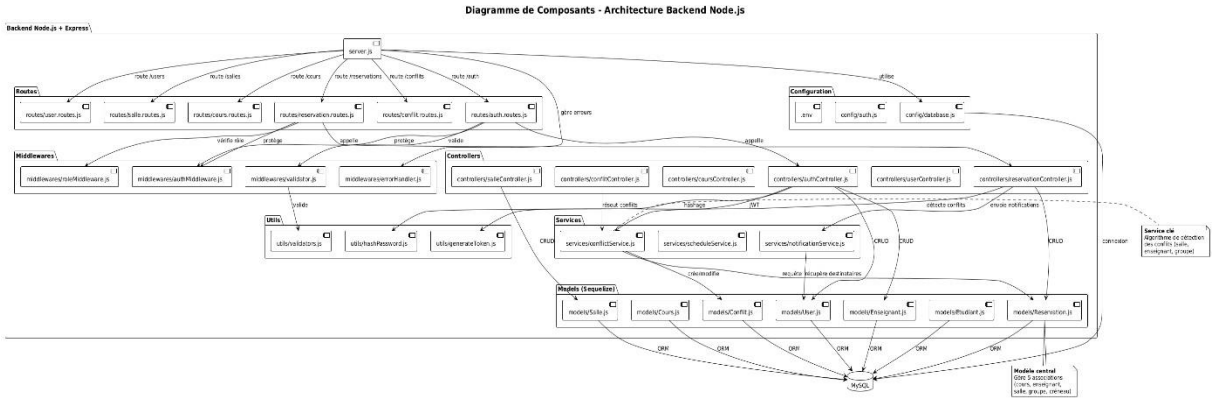

Figure 11: Diagramme de composants

3.3 – Les langages et les Frameworks de développement

Le choix des langages et Framework utilisés pour le développement du projet *HESTIM Planning* repose sur des critères de **performance**, de **flexibilité**, et de **maintenabilité**. L'objectif est d'utiliser un ensemble d'outils modernes, cohérents entre eux et largement adoptés dans le monde professionnel du développement web.

3.3.1- Langages

Le projet repose sur plusieurs langages, chacun répondant à un besoin spécifique :

- **JavaScript (ES6+)** :
  Langage principal du projet, utilisé à la fois côté frontend (avec React.js) et côté backend (avec Node.js).
  Il permet une meilleure homogénéité du code sur l'ensemble de la stack, favorisant la productivité et la cohérence du développement.

Figure 12 : Logo de JavaScript

- **HTML5** :
  Utilisé pour la structure et le contenu de l'interface utilisateur. Il sert de base au rendu des composants React et assure une compatibilité optimale avec les navigateurs modernes.

Figure 13: Logo de HTML5

- **CSS3** :

Langage de style utilisé pour la mise en forme de l'interface. Il permet d'assurer la cohérence visuelle, l'adaptabilité (responsive design) et l'expérience utilisateur. L'utilisation de **Material UI** et du **Dark/Light Mode** facilite la personnalisation et l'ergonomie de l'interface.

Figure 14: : Logo CSS3

- **SQL** :

Utilisé pour l'interaction avec la base de données **MySQL**, à travers l'ORM Sequelize. Il assure la manipulation structurée des données, la gestion des contraintes et la cohérence des relations entre tables.

Figure 15: SQL

3.3.2- Frameworks

Les Framework et bibliothèques sélectionnés ont été choisis pour leur stabilité, leur documentation complète et leur large communauté d'utilisateurs.

React.js (Frontend) :

Framework JavaScript moderne permettant la création d'interfaces utilisateur dynamiques et réactives.

Il repose sur le concept de **composants réutilisables**, ce qui facilite la maintenance du code.

Le projet utilise **Vite** comme outil de build pour optimiser la vitesse de développement et le temps de rechargement.

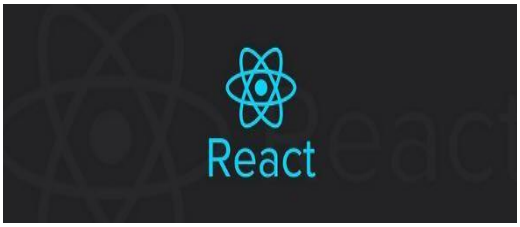

Figure 16: Logo de React JS

- **Material UI (MUI)** :

Bibliothèque de composants React basée sur les principes du Material Design de Google.

Elle permet de concevoir une interface moderne, harmonieuse et professionnelle tout en réduisant le temps de développement.

Figure 17: Logo de Material UI

- **Node.js (Backend)** :

Environnement d'exécution JavaScript côté serveur, performant et non bloquant. Il permet la gestion simultanée de plusieurs requêtes grâce à son modèle asynchrone basé sur des événements.

Figure 18: Logo de Node JS

Express.js (Framework serveur) :

Framework minimaliste pour Node.js qui facilite la création d'API REST.

Il gère efficacement les routes, les middlewares et les requêtes HTTP, tout en offrant une grande flexibilité.

Figure 19: Logo d'Express Js

- **Sequelize (ORM pour MySQL)** :

Outil de mapping objet-relationnel (ORM) permettant d'interagir avec la base de données sans écrire directement des requêtes SQL.
Il simplifie la gestion des modèles, des associations et des migrations tout en garantissant la portabilité du code.

Figure 20: Logo Sequelize

Avantages du choix technologique

- Uniformisation du langage avec **JavaScript** sur toute la stack (frontend et backend).
- **Performance élevée** grâce à Vite et Node.js.
- **Productivité accrue** via React et MUI.
- **Séparation claire** entre la logique métier, la présentation et la persistance des données.
- **Évolutivité** du projet facilitée par Sequelize et l'architecture modulaire.
- **Interopérabilité** avec d'autres systèmes via l'API REST.

3.4- Les outils de développement

La réussite du projet *HESTIM* planning repose non seulement sur le choix des langages et Framework, mais également sur l'utilisation d'outils de développement performants et adaptés aux besoins du cycle de vie du projet.

Ces outils ont été sélectionnés pour **faciliter la collaboration**, **améliorer la qualité du code**, et **accélérer le processus de développement et de déploiement**.

1. Visual Studio Code (VS Code)

C'est l'environnement de développement intégré (IDE) principal du projet. Léger, puissant et hautement personnalisable, **VS Code** offre de nombreuses extensions qui optimisent la productivité, telles que :

- *Prettier* pour le formatage automatique du code,
- *ESLint* pour le contrôle de la qualité et la détection d'erreurs JavaScript,
- *Material Icon Theme* pour une meilleure lisibilité des fichiers,
- *GitLens* pour la gestion du versionnement.

Son intégration native avec Git et sa compatibilité avec Node.js et React en font un outil idéal pour le développement full-stack JavaScript.

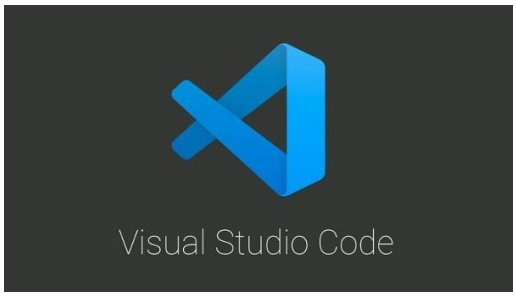

Figure 21: Logo de Visual Studio Code

2. Git & GitHub

Le projet est géré sous **Git**, un système de contrôle de version distribué, et hébergé sur **GitHub** pour permettre la collaboration entre les membres de l'équipe. Ces outils facilitent :

- Le suivi des modifications du code,
- La gestion des branches pour le développement parallèle,
- Les pull requests pour la revue de code,
- Le stockage sécurisé et centralisé du projet.

GitHub offre également des fonctionnalités comme les **Issues** et les **Projects**, utiles pour la gestion des tâches et le suivi de l'avancement.

Figure 22: : Logo Git & GitHub

3. MAMP

MAMP est utilisé pour la gestion du serveur local, notamment pour la base de données **MySQL** et l'administration via **phpMyAdmin**.
Il permet de tester les fonctionnalités du backend et d'assurer la communication entre l'API Node.js et la base de données pendant la phase de développement.

Figure 23: Logo MAMP

4. Postman

Postman est un outil essentiel pour tester et documenter les **API REST** développées avec Express.js.
Il permet d'envoyer des requêtes HTTP, de visualiser les réponses du serveur, de tester les endpoints, et de simuler différents scénarios (authentification, erreurs, etc.).
Son interface simple aide à vérifier rapidement le bon fonctionnement du backend avant l'intégration au frontend.

Figure 24: Logo Postman

5. Figma

Figma a été utilisé pour la conception des maquettes et l'élaboration du design de l'interface utilisateur.
Cet outil collaboratif permet de visualiser l'apparence finale de l'application, de valider les choix ergonomiques et de préparer l'intégration front-end de manière efficace.
Il joue un rôle central dans la **phase de prototypage** et de **validation du design**.

Figure 25: Logo Figma

6. Vercel *(Déploiement Frontend)*

Vercel est la plateforme choisie pour le déploiement du frontend React.js.
Elle offre une intégration fluide avec GitHub, un déploiement automatique à chaque push, et des performances optimisées grâce à son réseau de distribution (CDN).

Figure 26: Logo Vercel

7. Draw.io & Lucidchart

Ces outils ont été utilisés pour créer certains schémas complémentaires (architecture globale, organisation des composants, etc.) afin d'assurer une compréhension claire du système.

L'ensemble de ces outils constitue un environnement de travail complet et cohérent, favorisant la collaboration, la rigueur et la productivité tout au long du projet *HESTIM Planning*. Ils assurent une bonne communication entre les différentes phases du développement, depuis la conception jusqu'au déploiement final.

draw.io Lucidchart

Figure 27: Logo Draw.io & Lucidchart

8.PlantUML *(Conception)* :

Utilisé pour la modélisation des diagrammes UML et MERISE (MCD, MLD, séquences, classes, états...).
Cet outil facilite la documentation technique et la visualisation des structures logiques du système.

Figure 28: Logo PlantUML

3.5- Conclusion

Ce chapitre a permis de présenter l'architecture technique globale du projet *HESTIM Planning*, ainsi que les principaux outils et technologies utilisés pour sa conception et son développement.
L'approche retenue repose sur une **architecture web moderne**, articulée autour d'un **frontend en React.js**, d'un **backend en Node.js/Express**, et d'une **base de données MySQL** gérée via Sequelize. Cette structure modulaire garantit à la fois **performance**, **maintenabilité** et **évolutivité** de l'application.

Les différents composants de l'architecture interagissent de manière cohérente pour assurer la gestion des réservations, la planification automatique, la détection des conflits et la centralisation des informations.
De plus, les outils de développement tels que **VS Code**, **GitHub**, **Postman**, **Figma** ou encore **MAMP** ont permis d'assurer une organisation rigoureuse, une collaboration efficace entre les membres de l'équipe, et un suivi fluide du cycle de vie du projet.

Ainsi, cette étude technique jette les bases solides nécessaires à la mise en œuvre concrète de l'application.
Le chapitre suivant sera consacré à la **réalisation graphique et ergonomique**, à travers la présentation des **maquettes et du design d'interface**, qui matérialisent visuellement les choix techniques et fonctionnels détaillés jusqu'ici.

Chapitre 4 : Maquettes et Design de l'Interface

4.1- Introduction

Après avoir défini l'architecture technique et les choix technologiques du projet, cette étape se concentre sur la **conception visuelle et ergonomique** de l'application *HESTIM Planning*.

L'objectif principal de ce chapitre est de **transformer les spécifications fonctionnelles et techniques** établies précédemment en une **interface utilisateur intuitive, esthétique et cohérente**, adaptée aux besoins des différents profils d'utilisateurs : administrateurs, enseignants et étudiants.

La phase de maquettage joue un rôle essentiel dans la réussite du projet, car elle permet de **visualiser la navigation**, **structurer les pages** et **optimiser l'expérience utilisateur (UX)** avant la phase de développement.

Les maquettes ont été réalisées à l'aide de l'outil **Figma**, en respectant les principes de design moderne : **simplicité, cohérence visuelle, accessibilité et réactivité (responsive design)**.

Ce chapitre présentera dans un premier temps **une vue d'ensemble des maquettes** de l'application, puis **une description détaillée de chaque page**, afin d'illustrer la manière dont les besoins fonctionnels se traduisent concrètement dans l'interface.

4.2- Vue d'ensemble de la maquette

Cette image représente, l'ensemble des pages de notre application web.

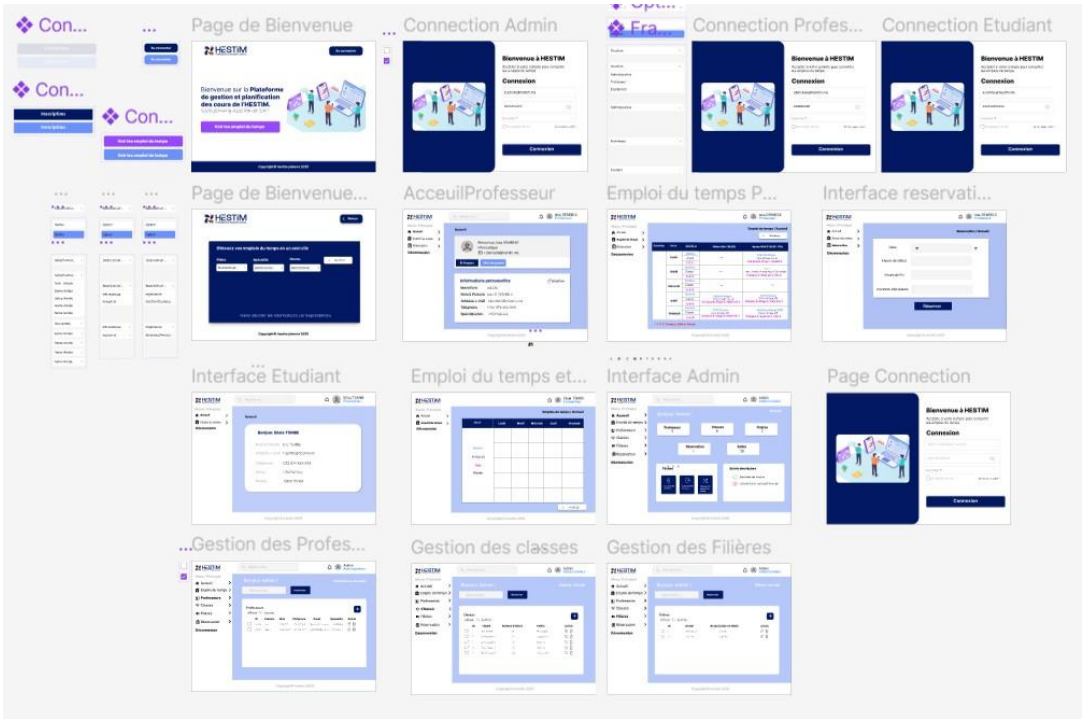

Figure 29: Vue d'ensemble de la maquette

4.3- Descriptions détaillées des différentes pages

Pour une meilleure compréhension de notre maquette, voici quelques détails sur certaines interfaces dont disposeront les utilisateurs :

Figure 30: Page de Bienvenue

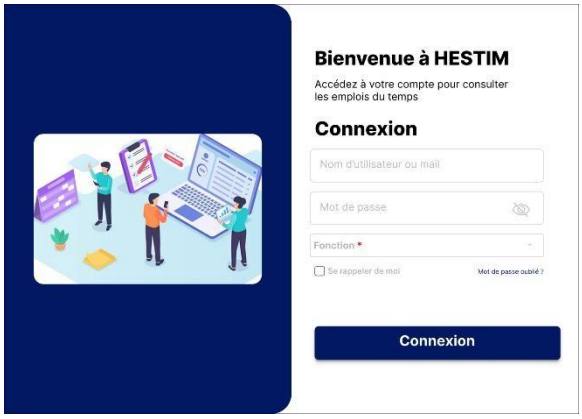

Figure 31: Page de Connexion

1. **Page de Bienvenue**
Constituée d'un message de « **Bienvenue** » et de 2 boutons principaux « **se connecter** » et « **Voir les emplois du temps** », elle est la vitrine de notre application.

2. **Page de Connexion**
Constituée **d'un formulaire** de connexion, elle permet **l'authentification** des utilisateurs.

Figure 32: Différentes Fonctions

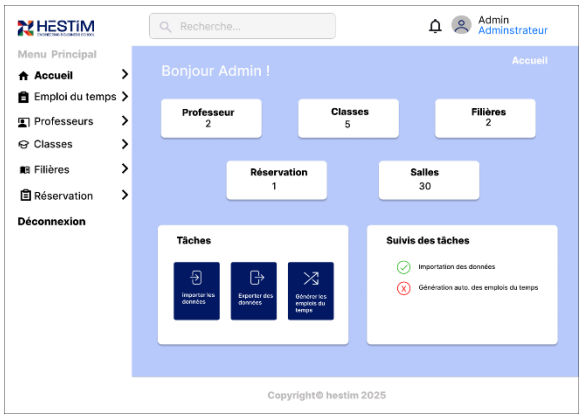

Figure 33: Interface Admin

3. **Liste des fonctions**
En plus du **formulaire**, la page de Connexion est aussi constituée d'une **liste déroulante** contenant les différentes fonctions qui pourront être utilisée par les utilisateurs

4. **Interface Admin** Cette page est constituée d'un **tableau de bord**, avec les différentes, tâches à effectuer par l'administrateur, ainsi qu'un **menu**.

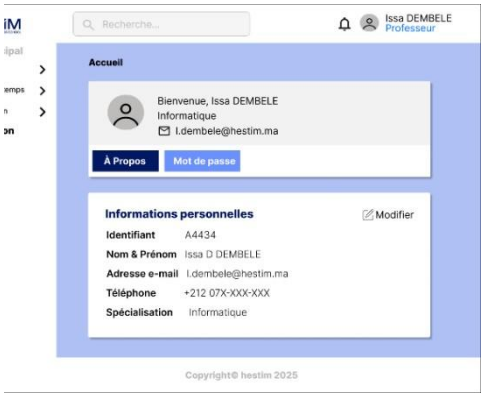

Figure 34: Interface professeur

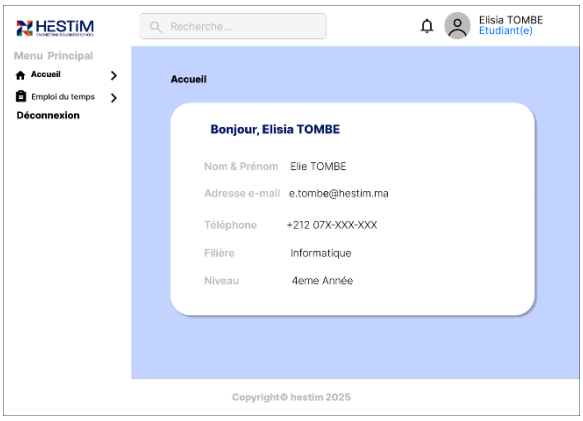

Figure 35: Interface Etudiant

- **Interface Professeur**
  Constituée d'une **fiche** regroupant toutes les informations du professeur, d'un **menu** et de deux boutons **« A propos »**, **« mot de passe »**.

- **Interface Etudiante**
  Constituée d'une **fiche** regroupant toutes les informations de l'étudiant et d'un **menu**.

4.4- Conclusion

Ce chapitre a permis de présenter le travail réalisé sur la conception visuelle de l'application *HESTIM Planning*. À travers les maquettes et le design de l'interface, l'objectif a été de proposer une expérience utilisateur fluide, intuitive et moderne, tout en assurant une cohérence graphique avec l'identité du projet.
L'approche adoptée s'est appuyée sur les principes de l'ergonomie et du design fonctionnel, afin de garantir une navigation simple et un accès rapide aux fonctionnalités essentielles, telles que la gestion des réservations, la consultation des emplois du temps et la détection des conflits.

Les maquettes vont constituer un guide visuel déterminant pour la phase de développement frontend, en assurant une transition efficace entre la conception et l'implémentation. Elles traduisent concrètement la logique fonctionnelle du système en éléments graphiques exploitables par l'utilisateur final.

Ainsi, la conception graphique vient compléter l'architecture technique et la modélisation fonctionnelle précédemment étudiées, en offrant une vision globale, esthétique et opérationnelle du produit final.
La section suivante, dédiée à la **conclusion générale**, viendra récapituler les acquis du projet et proposer des perspectives d'amélioration pour les versions futures du système.

Chapitre 5 : Conception technique & développement Backend

5.1- Introduction

Dans ce chapitre, nous présenterons la conception technique et la mise en œuvre du **backend** de l'application **HESTIM Planning**. Le backend constitue le cœur fonctionnel du système puisqu'il assure la gestion des données, la sécurité des accès, le traitement des règles métiers et la communication entre l'interface utilisateur et la base de données.

L'architecture retenue repose sur une **API REST** développée avec **Node.js** et **Express** combinée à l'ORM **Sequelize** pour la gestion de la persistance des données dans une base MySQL. Ce choix permet de structurer le code selon une organisation modulaire, favorisant la maintenabilité, la réutilisabilité et l'évolutivité de l'application.

Ce chapitre détaille successivement **l'architecture générale** du backend, les choix technologiques opérés, la mise en œuvre des différents modules, la gestion de l'authentification des utilisateurs, les aspects liés à la sécurité, ainsi que la phase de tests et de débogage. Il se conclut par une présentation des perspectives d'amélioration futures du système.

5.2- Architecture générale

Le backend de l'application **HESTIM Planning** est construit selon une architecture **modulaire orienté services**, basée sur le modèle **API REST**. Cette organisation permet de séparer clairement les différentes responsabilités du système afin d'assurer une meilleure lisibilité du code et une facilité de maintenance.

L'architecture repose sur les principaux composants suivants :

1. **Serveur principal**

   Le fichier ***server.js*** constitue le point d'entrée de l'application. Il initialise le serveur Express, configure les middlewares globaux, établit la connexion à la base de données et enregistre l'ensemble des routes exposées par l'API REST.

2. **Couche « Routes »**

   Le dossier ***routes*** définit l'ensemble des **endpoints** de l'API. Chaque fichier de route correspond à un module fonctionnel spécifique (utilisateurs, enseignants, salles, affectations, etc.) et transmet les requêtes http aux contrôleurs adéquats.

Rôle :

- Définir les URL de l'API
- Associer chaque route à une méthode de contrôleur

- Appliquer les middlewares si nécessaire (authentification, validation)

3. **Couche « Controllers »**

Les **controllers** contiennent la logique applicative du projet. Ils traitent les requêtes reçues depuis les routes, appliquent les règles métier et communiquent avec les modèles Sequelize.

Rôle :

- Validation des données
- Traitements métier (création d'affectation, gestion des conflits, reporting)
- Gestion des réponses HTTP

4. **Couche « Models »**

Les **models Sequelize** représentent l'ensemble des entités issues du Modèle Logique de Données (MLD) :

User, Enseignant, Salle, Cours, Creneau, Affectation, HistoriqueAffectations, etc.

Rôle :

- Définition des structures de données
- Gestion des relations entre tables
- Interrogation de la base MySQL via Sequelize

5. **Dossier « Middleware »**

Le dossier **middleware** regroupe les composants transversaux du backend :

- Vérification des tokens JWT
- Gestion des rôles utilisateurs (admin, enseignant, étudiant)
- Validation des entrées utilisateur

Ces mécanismes garantissent la sécurité et la cohérence des données échangées entre client et le serveur.

6. **Dossier « Config »**

Il contient les fichiers de configuration globale :

- Paramètres de connexion à la base de données
- Variables d'environnement
- Configuration Sequelize

7. **Dossier « Utils »**

Les fonctions utilitaires partagées (gestion des dates, formatage, traitement logique annexe) sont regroupées dans ce dossier afin de favoriser la réutilisation du code.

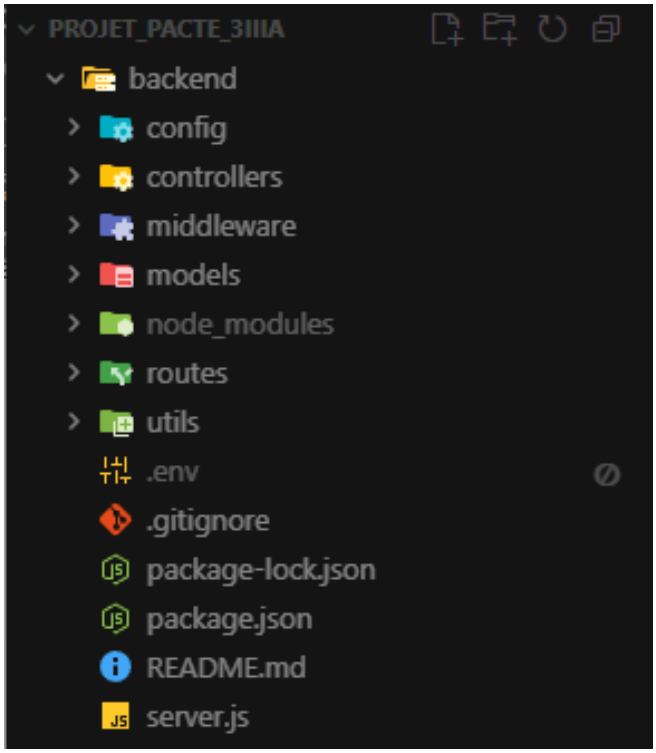

Figure 36: Vue d'ensemble de l'architecture

8. Base de données

La persistance est assurée par **MySQL**, interfacée avec Sequelize. La base repose directement sur la structure du MLD validé lors de la phase de conception (chapitre 2).

Table	Action	Lignes	Type	Interclassement	Taille	Perte
affectations	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	112,0 kio	-
appartenir	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
conflitaffectations	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	48,0 kio	-
conflits	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	16,0 kio	-
cours	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	48,0 kio	-
creneaux	Parcourir Structure Rechercher Insérer Vider Supprimer	14	InnoDB	utf8mb4_0900_ai_ci	16,0 kio	-
demandereports	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	48,0 kio	-
disponibilites	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	48,0 kio	-
enseignants	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	16,0 kio	-
etudiants	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
filiere	Parcourir Structure Rechercher Insérer Vider Supprimer	5	InnoDB	utf8mb4_0900_ai_ci	272,0 kio	-
groupes	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
historiqueaffectations	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	48,0 kio	-
notifications	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
salles	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	16,0 kio	-
users	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
16 tables	Somme	19	MyISAM	utf8mb4_0900_ai_ci	848,0 kio	0 o

Figure 37: Liste des tables de base de données

5.3- Explication des choix technologiques

Le développement du backend de l'application **HESTIM Planning** s'appuie sur un ensemble de technologies modernes, reconnues pour leur fiabilité, leur performance et leur large adoption dans le domaine du développement web.

Le choix du moteur **Node.js** s'est imposé en raison de son architecture événementielle asynchrone, parfaitement adaptée à la gestion de multiples requêtes simultanées. Il permet d'obtenir un serveur rapide et léger, idéal pour les applications de type API REST.

Le framework **Express.js** a été utilisé afin de structurer efficacement les routes HTTP et de simplifier la gestion des requêtes ainsi que des réponses du serveur. Sa simplicité d'intégration avec différents middlewares le rend particulièrement adapté aux projets nécessitant une organisation modulaire.

Pour la gestion de la base de données relationnelle, le choix s'est porté sur **MySQL** en raison de sa stabilité, de sa performance et de sa compatibilité avec les systèmes d'information institutionnels.

L'ORM **Sequelize** a été adopté afin de faciliter l'interaction entre le backend et la base de données. Il permet de représenter les tables sous forme d'objets JavaScript, tout en prenant en charge :

- La définition des relations entre entités,
- La gestion des migrations,
- La validation des données,
- Et l'abstraction des requêtes SQL complexes.

Enfin, l'outil **JSON Web Token (JWT)** a été mis en place pour gérer l'authentification sécurisée des utilisateurs, tandis que **bcrypt** est utilisé pour le hachage des mots de passe afin de garantir la confidentialité des informations sensibles.

5.4- Développement du Backend

Le développement du backend a consisté à implémenter l'ensemble des fonctionnalités nécessaires à la gestion du système de planification et d'affectation des ressources pédagogiques.

L'API REST expose des endpoints organisés par domaine fonctionnel :

- **Gestion des utilisateurs** : création, authentification, attribution de rôles (admin, enseignant, étudiant).
- **Gestion des enseignants et étudiants** : consultation et mise à jour des profils.

- **Gestion des salles** : ajout, consultation, modification des caractéristiques des salles.
- **Gestion des cours et créneaux** : organisation du contenu pédagogique et des plages horaires.
- **Gestion des affectations** : attribution des salles et créneaux aux cours proposés par les enseignants.
- **Gestion des conflits** : détection automatique des collisions de planning.
- **Gestion de l'historique** : traçabilité des modifications effectuées par les administrateurs.
- **Gestion des notifications** : information des utilisateurs en cas de création, modification ou annulation de séances.

Chaque module suit la structure suivante :

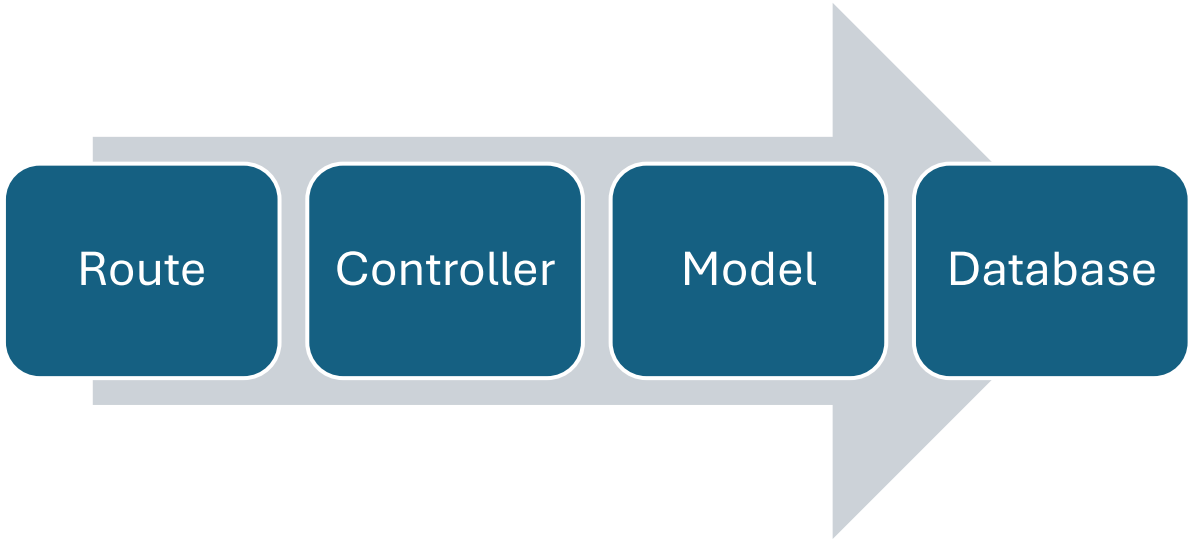

Figure 38: Chaîne logique de traitement d'une requête HTTP dans une API REST

Les règles métiers sont principalement gérées au niveau des contrôleurs afin de garantir la cohérence des données et le respect des contraintes fonctionnelles définies lors de la phase d'analyse.

5.5- Gestion des sessions utilisateurs

La gestion des sessions repose sur un système d'authentification basé sur **JSON Web Tokens (JWT)**.

Fonctionnement :

1. L'utilisateur s'authentifie à l'aide de son **adresse email et mot de passe**.
2. Le backend vérifie les identifiants :
   - Mot de passe comparé via **bcrypt**
   - Rôle extrait depuis la base de données
3. Si l'authentification est validée, le serveur génère un **JWT signé**.
4. Le token est transmis au client et stocké de manière sécurisée.
5. Chaque requête suivante vers l'API inclut ce token dans l'header Authorization.
6. Un middleware vérifie la validité du token avant d'autoriser l'accès aux ressources demandées.

Sécurisation par rôles :

Des contrôles d'autorisation ont été mis en place afin de restreindre certaines fonctionnalités :

Tableau 2: Rôle utilisateur/Droits principaux

Rôle utilisateur	Droits principaux
Administrateur	Création et affectation des séances et gestion complète
Enseignant	Transmission des disponibilités et demandes de report
Étudiant	Consultation des emplois du temps

Sécurité & Validation

La sécurité de l'API repose sur plusieurs mécanismes complémentaires :

Authentification sécurisée :

- JWT pour les sessions
- bcrypt pour le hachage des mots de passe

Validation des entrées :

- Vérification des champs obligatoires
- Vérification des formats (email, date, identifiants numériques)
- Protection contre l'injection SQL via Sequelize

Gestion des rôles :

- Middleware d'autorisation par type d'utilisateur

Gestion des erreurs :

- Messages contrôlés
- Codes HTTP normalisés (401, 403, 404, 500)

5.6- Tests et Débogage

La phase de test et de débogage a permis d'assurer la qualité et la fiabilité de l'API développée.

Les principaux outils utilisés sont :

- **Postman** pour la simulation des appels API,
- Console Node.js pour le débogage des traitements,
- Logs serveur pour l'analyse des erreurs.

Les tests ont principalement porté sur :

- Les scénarios d'authentification,
- La création des affectations,
- La détection des conflits de créneaux,
- Le bon fonctionnement de l'historique,
- La sécurisation des routes protégées.

Chaque module a été validé de manière indépendante avant l'intégration dans le système global.

5.7- Perspectives d'amélioration du Backend

Plusieurs pistes d'évolution peuvent être envisagées pour renforcer le backend :

- Mise en place de **tests automatisés** (Jest, Mocha).
- Ajout d'un module d'**intelligence artificielle** pour la génération automatique d'emplois du temps.
- Intégration d'un système de **cache** (Redis) pour améliorer les performances.
- Déploiement d'une **API versionnée** afin de garantir la compatibilité future.
- Ajout d'un module de **journal d'audit avancé** pour la traçabilité administrative.

5.8- Conclusion

Ce chapitre a présenté la conception et le développement du backend de l'application **HESTIM Planning**. L'architecture retenue, basée sur une API REST en Node.js et Express combinée à Sequelize et MySQL, offre une solution modulaire, sécurisée et évolutive, adaptée aux besoins de gestion académique de l'établissement.

Les mécanismes d'authentification, de gestion des rôles et de validation garantissent la protection des données et la conformité aux exigences fonctionnelles identifiées lors de l'analyse. Le backend constitue ainsi un socle robuste permettant l'implémentation des fonctionnalités principales de planification, de gestion des affectations et de traçabilité.

CONCLUSION GÉNÉRALE

Le projet **HESTIM Planning** s'inscrit dans une démarche de conception et de développement d'une plateforme web destinée à optimiser la gestion des emplois du temps et des réservations de salles au sein d'un établissement d'enseignement.
À travers les différentes étapes déjà amorcées, ce travail a permis de poser les bases solides du système, en identifiant clairement les besoins fonctionnels, en définissant les objectifs du projet, et en établissant les modèles conceptuels et logiques nécessaires à sa mise en œuvre.

L'approche adoptée, fondée sur le **cycle de développement en V**, va permettre d'organiser méthodiquement la progression du projet, en distinguant les phases d'analyse, de conception et de préparation technique. Cette structuration garantit une meilleure maîtrise du processus et facilitera les prochaines étapes de développement et d'intégration.

Pour la suite, le travail portera sur la **réalisation concrète de la plateforme**, à travers le développement du backend (API Node.js, base de données MySQL) et du frontend (interface web avec React.js).
Une attention particulière sera également accordée à l'ergonomie, à la fiabilité du système et à la détection automatique des conflits d'emploi du temps.

En somme, le projet HESTIM Planning représente aujourd'hui **une phase cruciale de préparation et de structuration** avant la mise en œuvre technique. Il constitue un socle solide pour la création future d'une solution moderne, intuitive et performante au service de la gestion académique.