Gestion de Planning

Conception d'un EDT

UNIVERSITÉ ÉVRY
PARIS-SACLAY

RAPPORT DE PROJET

Réalisé par

20231126 EL HATHOUT LINA
20222179 YOUSSEF EDRIS
20246195 BELKACEMI CIRINE
20245625 ADJAZ RYMA

Encadrer par

Mr Beduneau

Année Universitaire

2025/2026

TABLE DES MATIÈRES

Introduction

Présentation du Projet

Ce projet consiste à **concevoir et développer une application** informatique dédiée à la gestion des emplois du temps.

L'application a pour objectif de permettre aux utilisateurs de consulter, organiser et gérer leur **emploi du temps** de manière simple, claire et efficace, en centralisant l'ensemble des informations liées aux cours sur une seule plateforme.

Elle s'adresse principalement aux étudiants, aux enseignants ainsi qu'au personnel administratif ayant besoin d'un outil fiable pour visualiser les cours, les horaires, les salles et les différentes contraintes organisationnelles.

Grâce à cette application, les utilisateurs peuvent accéder rapidement à leur emploi du temps et disposer d'une vue structurée facilitant la planification de leurs activités quotidiennes.

L'application vise à remplacer ou à compléter les supports traditionnels tels que les emplois du temps papier ou les affichages statiques, souvent peu pratiques et difficiles à mettre à jour.
En proposant une solution numérique accessible et intuitive, ce projet permet de réduire les erreurs d'interprétation, d'améliorer la communication des informations et de faciliter la gestion des changements d'horaires

Contexte du projet :

La gestion des emplois du temps constitue une problématique récurrente dans les établissements d'enseignement. La multiplicité des acteurs concernés et la fréquence des modifications rendent l'organisation complexe et parfois source de désorganisation. Les informations sont souvent dispersées, ce qui complique leur consultation et leur mise à jour.

Par ailleurs, les solutions existantes ne répondent pas toujours pleinement aux attentes des utilisateurs. Certaines sont limitées en fonctionnalités, peu flexibles ou ne permettent pas une adaptation rapide aux changements. Cette situation peut entraîner une perte de temps, une mauvaise compréhension des horaires et une diminution de l'efficacité organisationnelle.

Dans ce contexte, **la mise en place** d'un système informatisé dédié à la gestion des emplois du temps apparaît comme une réponse pertinente. Ce projet s'inscrit donc dans une démarche d'amélioration de l'organisation et de la circulation de l'information au sein d'un environnement académique.

Objectifs du projet :

L'objectif principal du projet est de proposer une solution informatique capable de répondre aux contraintes organisationnelles liées à la gestion des emplois du temps.
Les objectifs secondaires sont :

- Structurer et fiabiliser les informations liées aux cours
- Faciliter la gestion et la mise à jour des données
- Offrir une consultation rapide et organisée des plannings
- Améliorer l'expérience utilisateur grâce à une navigation claire
- Concevoir une application évolutive pouvant être enrichie par la suite

Ce projet vise également à appliquer de manière concrète les notions étudiées durant la formation, notamment en analyse des besoins, modélisation et développement d'une application informatique.

Analyse Fonctionnelle des Besoins

Les Acteurs du système

Le système de gestion des emplois du temps implique plusieurs acteurs ayant des rôles et des besoins distincts. Chaque acteur interagit avec la plateforme selon ses responsabilités et ses objectifs.
Étudiant
L'étudiant est un utilisateur du système ayant un rôle principalement orienté vers la consultation des emplois du temps. Il accède aux informations qui lui sont associées en fonction de son inscription académique (année, filière et cohorte).
Fonctionnalités associées :

- Se connecter à la plateforme à l'aide d'un identifiant et d'un mot de passe
- Consulter automatiquement son emploi du temps en fonction de l'année, de la filière et
- de la cohorte auxquelles il est rattaché
- Visualiser son emploi du temps selon différentes vues :

- vue journalière,
- vue hebdomadaire,
- vue mensuelle

- Filtrer l'affichage de l'emploi du temps selon différents critères (type de séance, salle,
- enseignant)
- Consulter les détails d'une séance (horaire, salle, enseignant, type de cours)
- Consulter ses notifications
- Exporter son emploi du temps
- Se déconnecter de la plateforme

Enseignant

L'enseignant est un acteur du système. Il utilise la plateforme pour consulter son emploi du temps, gérer ses disponibilités effectuer des demandes de réservation pour des séances.

Fonctionnalités associées :

- Se connecter à la plateforme à l'aide d'un identifiant et d'un mot de passe
- Consulter son emploi du temps personnel (cours, examens, réunions)
- Visualiser son emploi du temps selon différentes vues :

- vue journalière,
- vue hebdomadaire,
- vue mensuelle

- Filtrer l'affichage de son emploi du temps selon différents critères (type de séance,cohorte, salle)
- Consulter les détails d'une séance (horaire, salle, cohorte, type de cours)
- Gérer ses disponibilités en définissant des créneaux disponibles ou indisponibles
- Consulter l'emploi du temps des cohortes qu'il encadre
- Consulter ses notifications
- Visualiser les salles attribuées aux séances
- Effectuer des demandes de réservation de salles ou de créneaux
- Être informé en cas de modification ou de conflit lié à son emploi du temps
- Se déconnecter de la plateforme

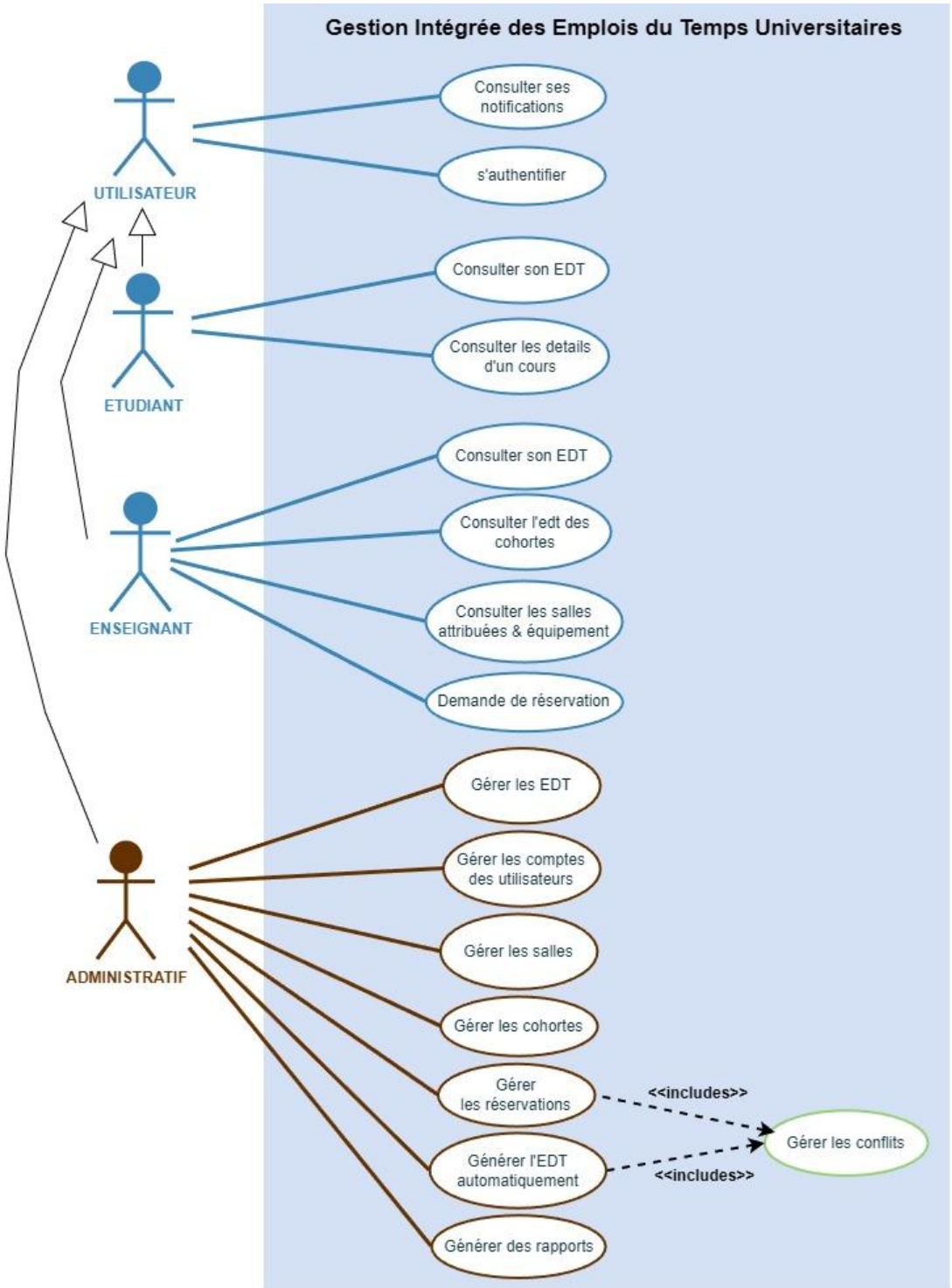
Administratif

L'administratif est l'acteur central du système. Il est responsable de la gestion globale des ressources, de la planification des emplois du temps et de la résolution des conflits afin d'assurer la cohérence du planning universitaire.

- Se connecter à la plateforme à l'aide d'un identifiant et d'un mot de passe
- Gérer les comptes utilisateurs (création et affectation des rôles étudiants et enseignants)
- Gérer les salles et leurs caractéristiques (capacité, type, équipements)
- Créer et modifier les emplois du temps
- Planifier les séances (CM, TD, TP, examens, événements)
- Gérer et valider les réservations de salles et de créneaux
- Détecter automatiquement les conflits d'emploi du temps (double réservation, indisponibilité d'un enseignant, dépassement de capacité)
- Résoudre les conflits de planification de manière manuelle
- Valider les emplois du temps avant leur mise à disposition aux utilisateurs
- Disposer d'un journal des modifications (historique des versions des emplois du temps)
- Visualiser les emplois du temps globaux avec des filtres (par cohorte, salle, enseignant, type de séance)
- Importer / exporter des utilisateurs (étudiants/enseignants)
- Se déconnecter de la plateforme

Cas d'Utilisation (Diagramme)

La figure suivante présente le diagramme de cas d'utilisation du système de gestion intégrée des emplois du temps universitaires.



Ce diagramme met en évidence trois acteurs principaux : l'étudiant, l'enseignant et l'administratif. L'étudiant utilise le système essentiellement pour consulter son emploi du temps et les informations associées. L'enseignant dispose de fonctionnalités supplémentaires liées à la gestion de ses disponibilités et aux demandes de réservation. L'administratif, acteur central du système, assure la gestion globale des ressources, la planification des emplois du temps et la gestion des conflits, garantissant ainsi la cohérence du planning universitaire.

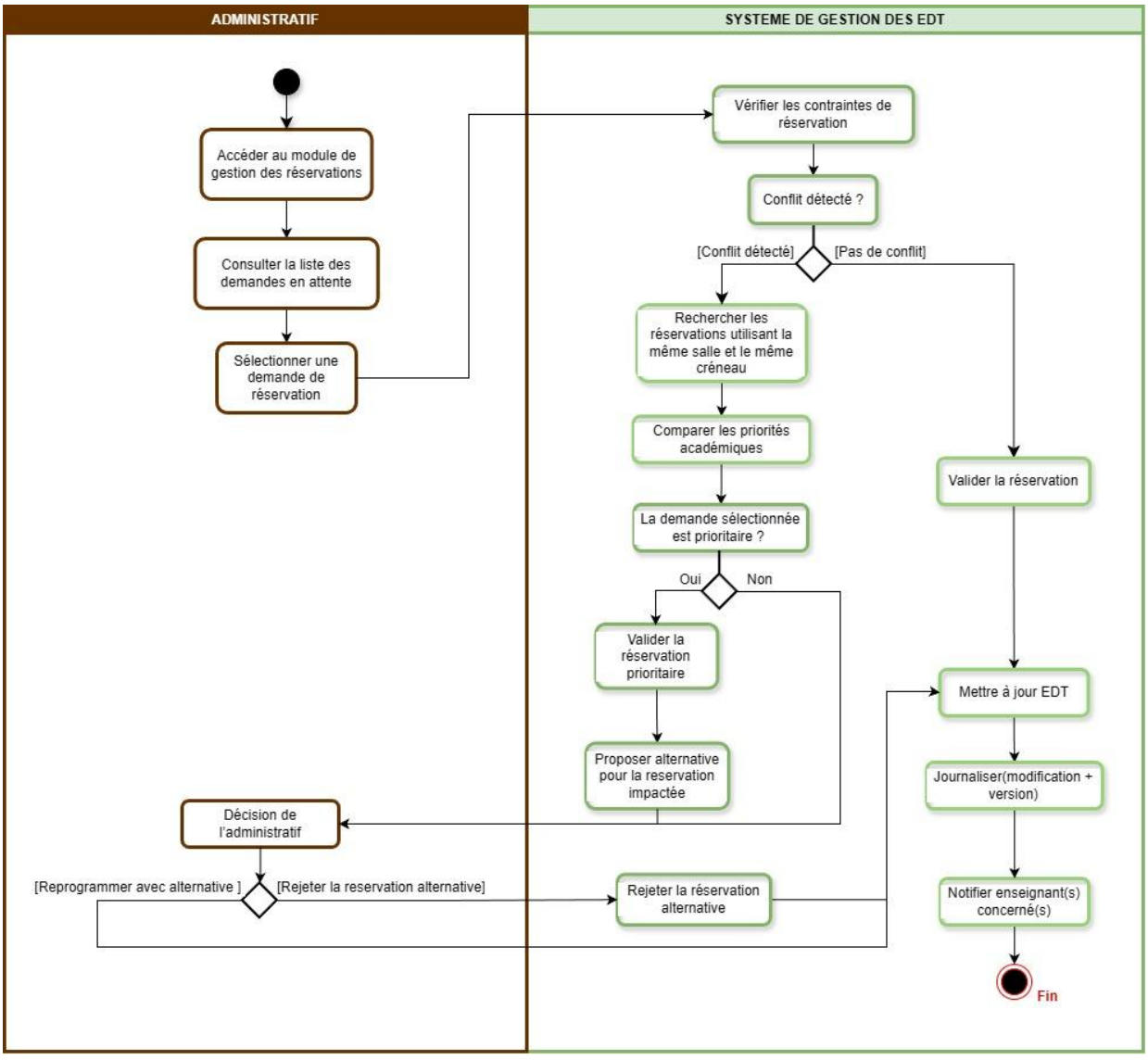
Règles de Gestion

Le fonctionnement de la plateforme repose sur un ensemble de règles de gestion visant à garantir la cohérence, la fiabilité et la validité des emplois du temps générés.

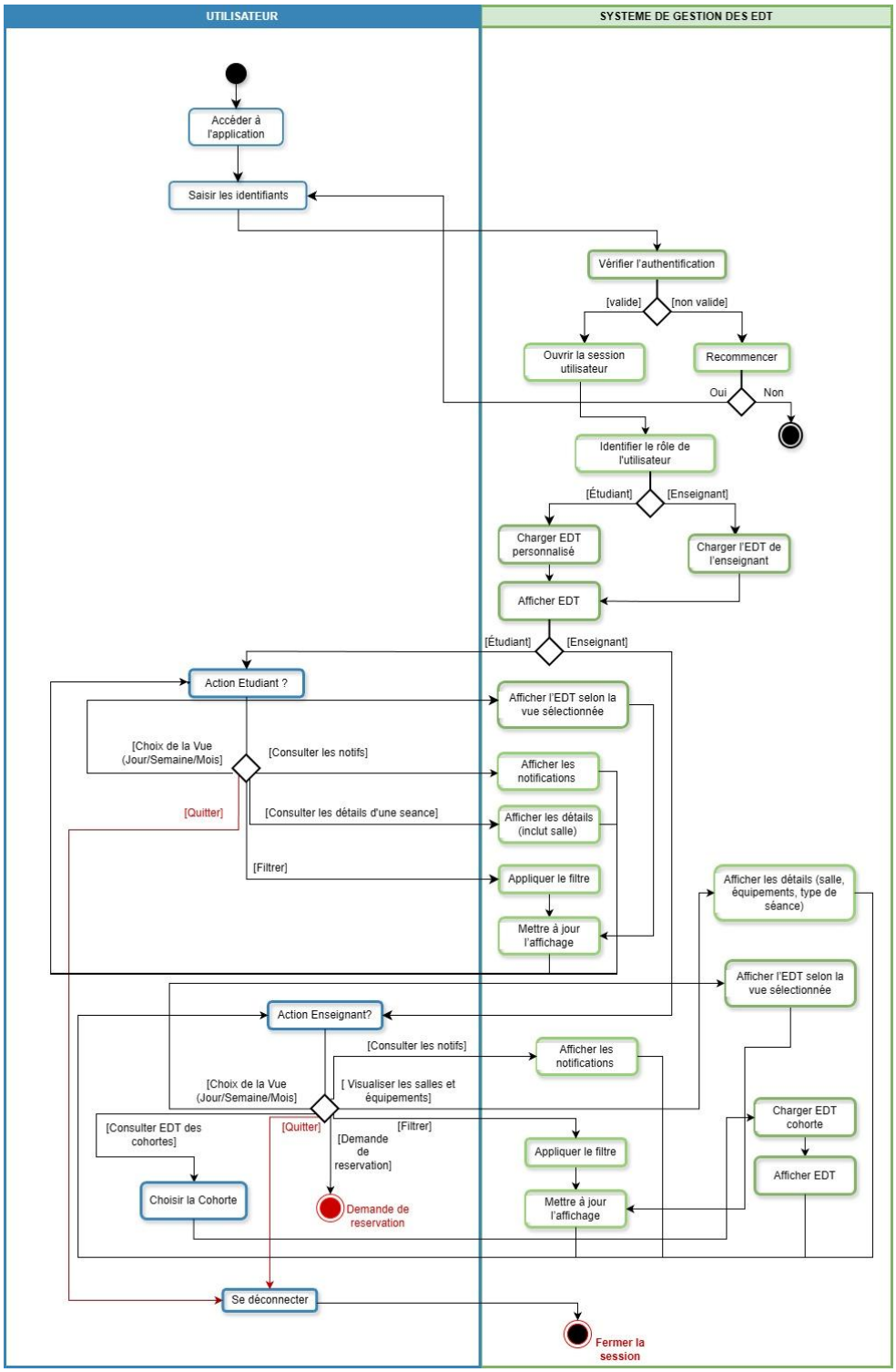
- Une salle ne peut être affectée qu'à une seule séance pour un même créneau horaire.
- Une cohorte (groupe d'étudiants) ne peut pas avoir deux séances qui se chevauchent sur un même créneau horaire.
- Un enseignant ne peut être planifié pour plusieurs séances se déroulant simultanément.
- La capacité d'une salle doit être suffisante pour accueillir l'effectif de la cohorte associée à la séance.
- Toute séance doit obligatoirement être associée à une salle, une cohorte et un enseignant.
- Toute création ou modification d'un emploi du temps entraîne une vérification automatique des conflits potentiels.
- Seuls les emplois du temps validés par l'administratif sont accessibles en consultation par les étudiants.
- En cas de conflit de réservation(cours/examens/événements), le système applique un ordre de priorité (par exemple : examens > cours réguliers > événements ponctuels), afin d'aider l'administratif à arbitrer.
- Lorsqu'un emploi du temps est modifié/validé, les utilisateurs concernés reçoivent une notification.
- Toute modification d'une séance (horaire, salle, enseignant) doit être historisée dans un journal des changements (date, auteur, action).

Diagramme d'activité

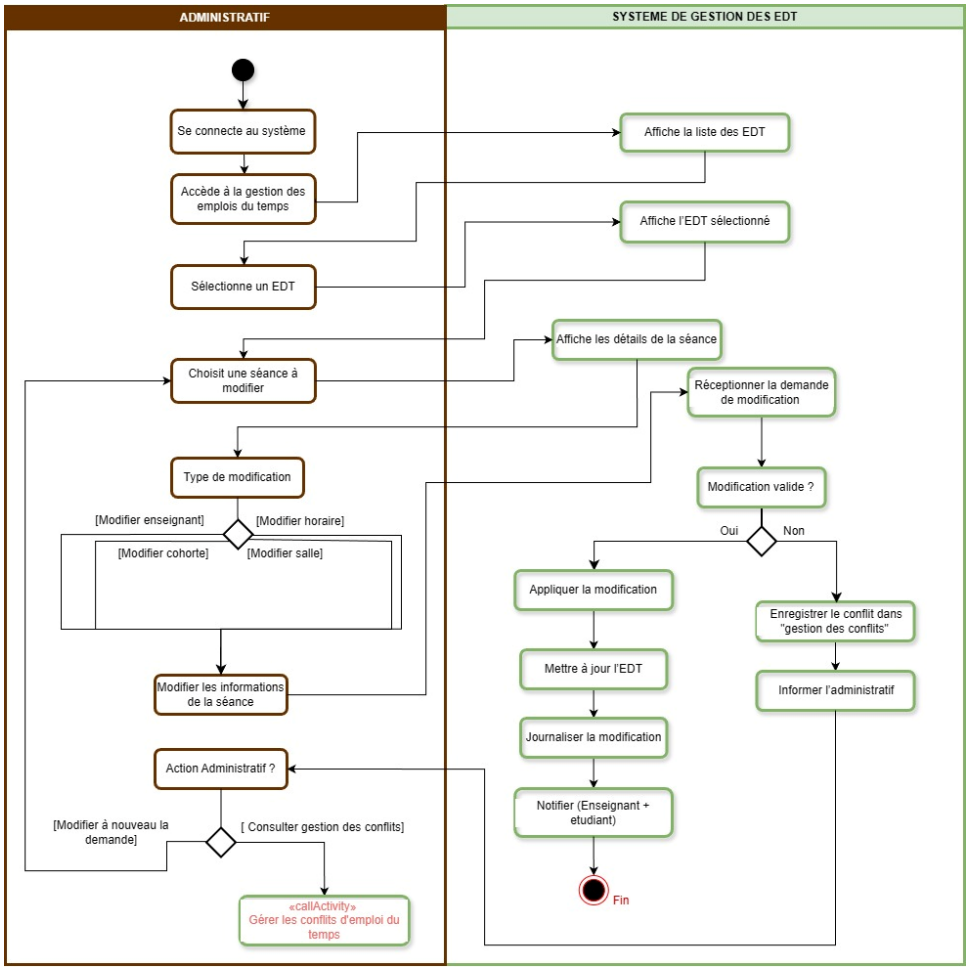
Gérer les réservations



Consulter l’emploi du temps



Modifier un emploi du temps



Gérer les conflits

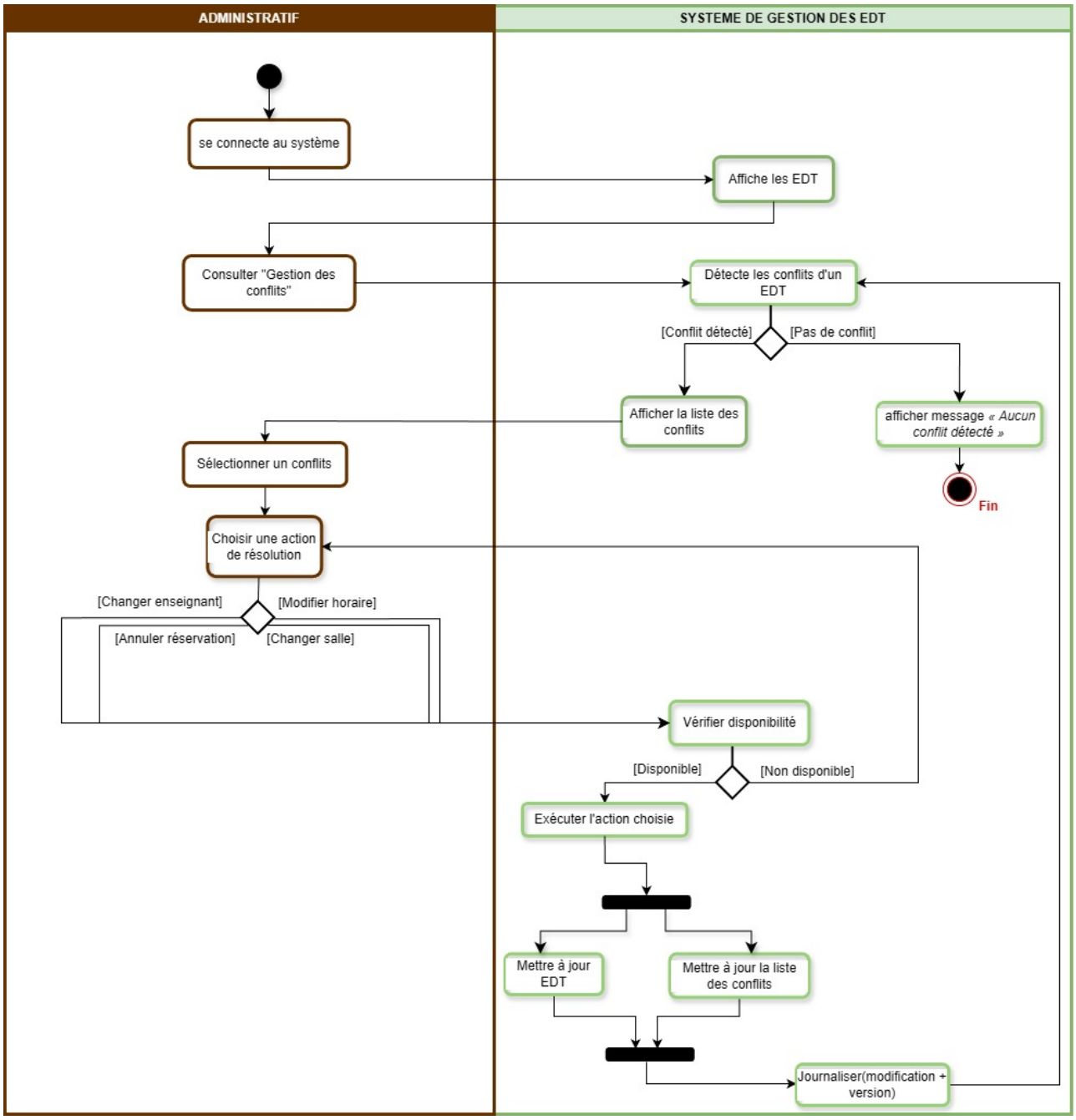
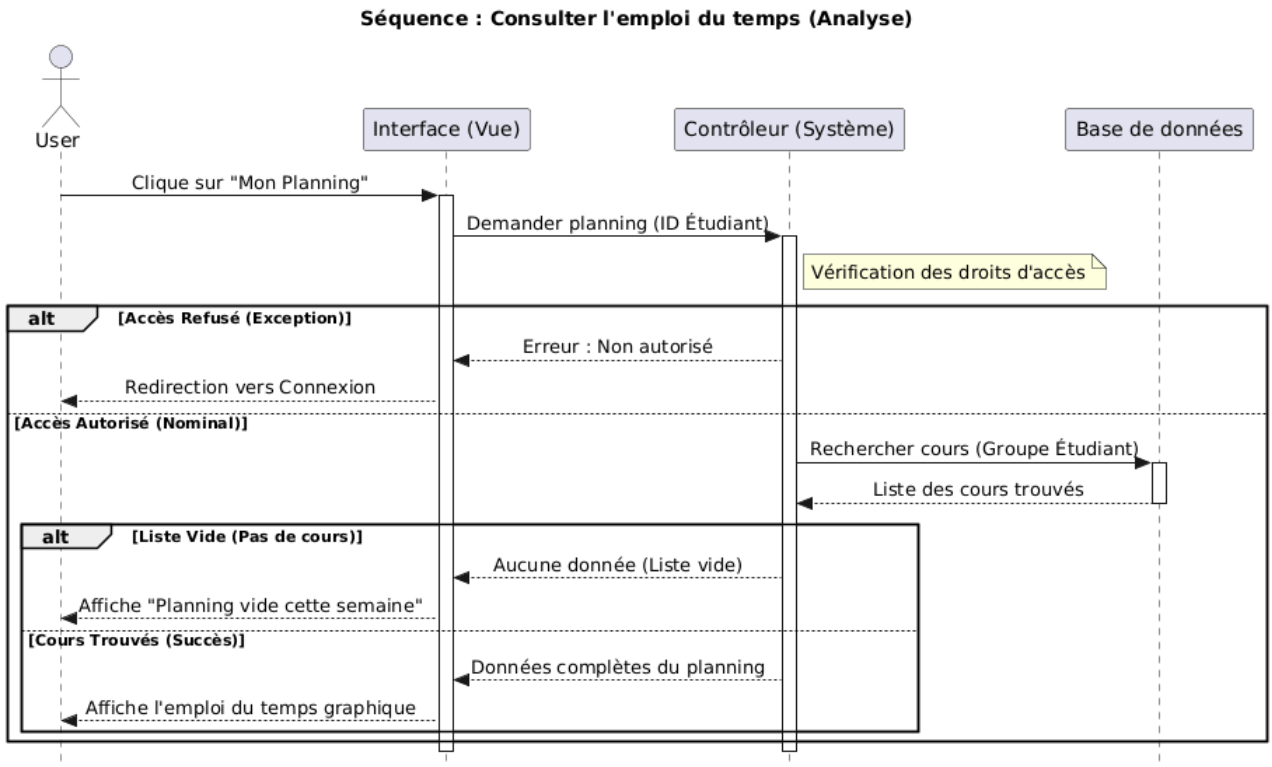
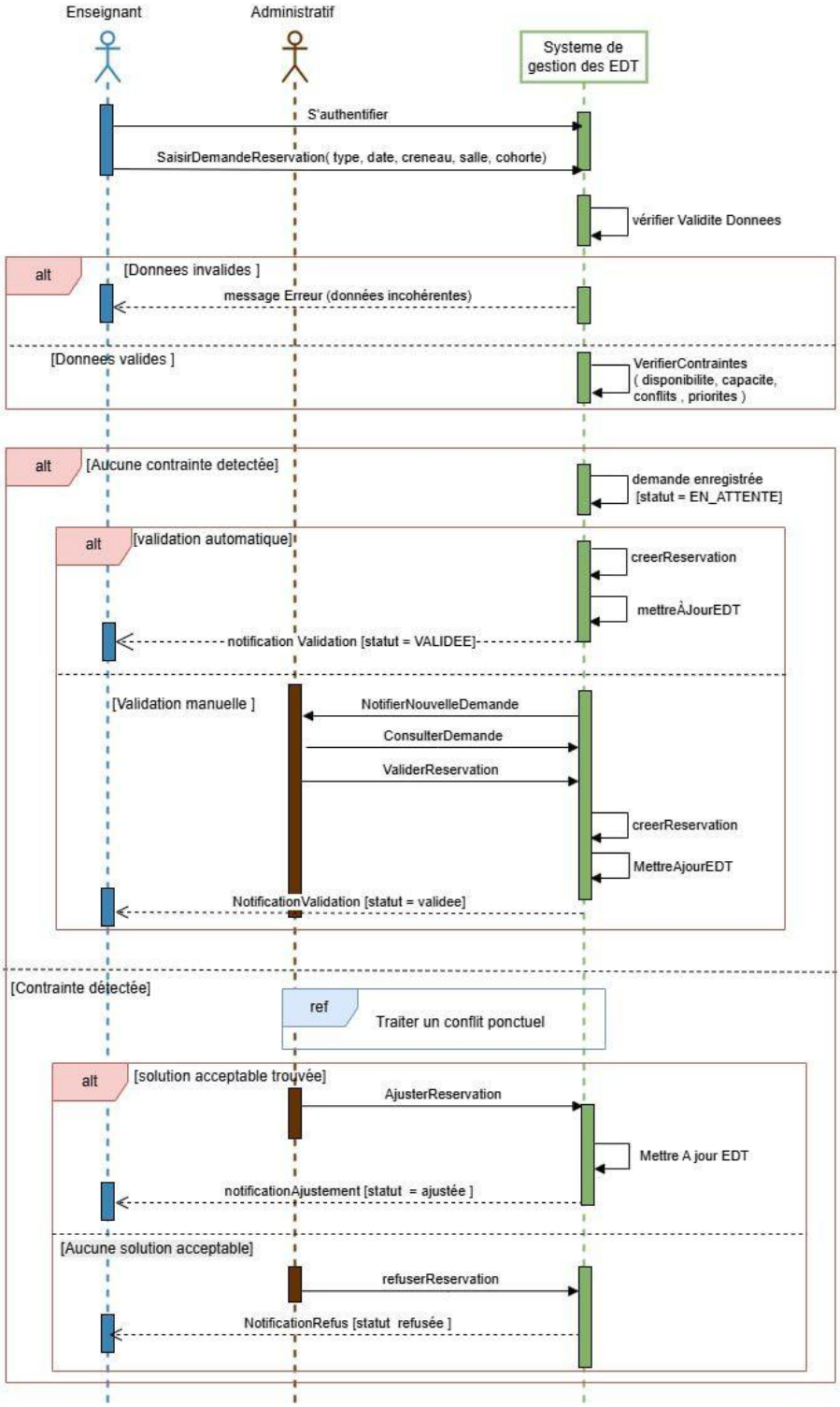


Diagramme de Séquence

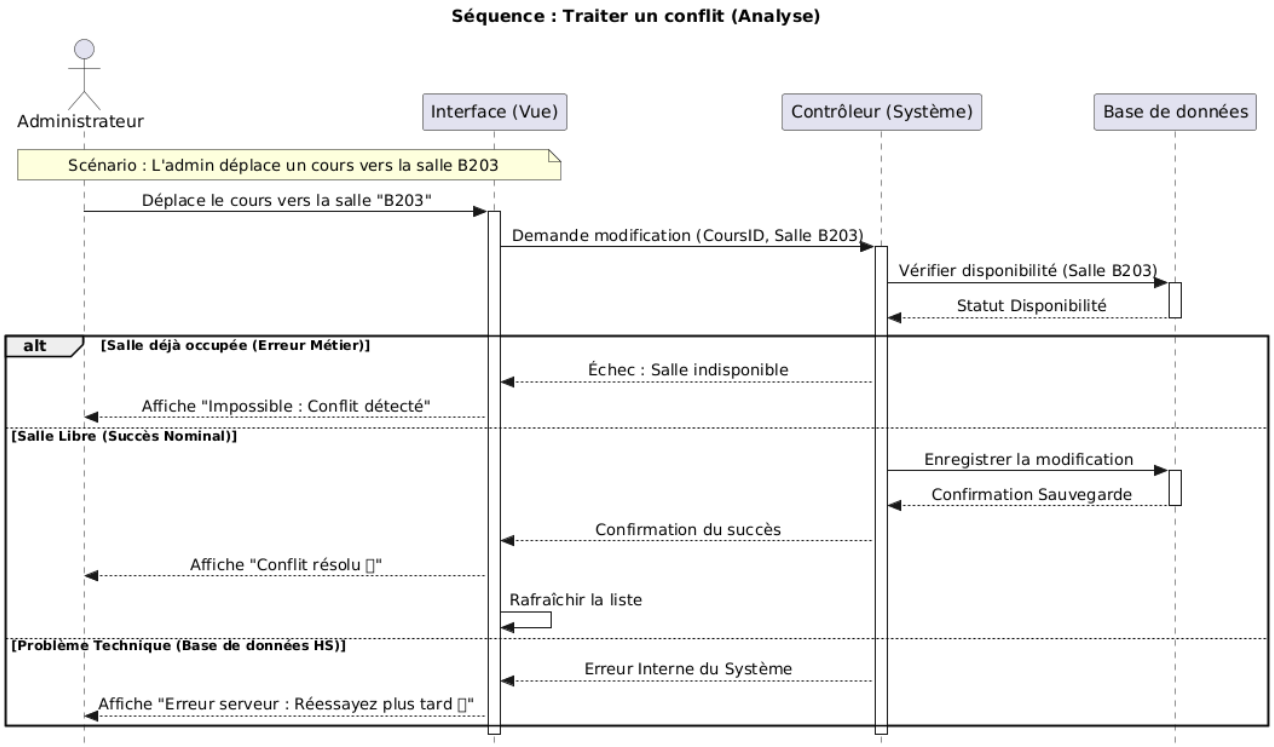
Consulter l'emploi du temps



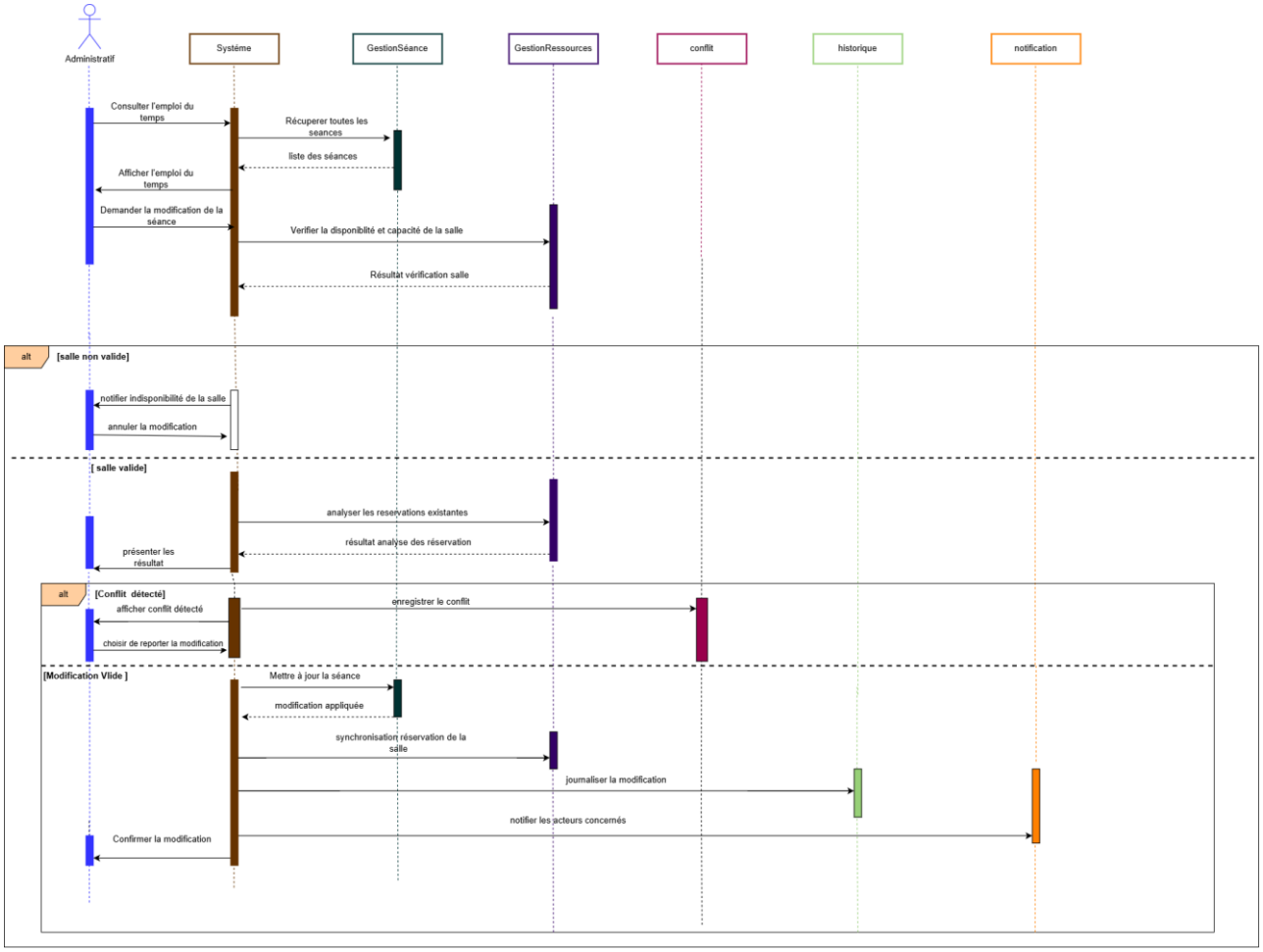
Demande de réservation


Enseignant
Administratif
Systeme de gestion des EDT
S'authentifier
SaisirDemandeReservation(type, date, creneau, salle, cohorte)
vérifier Validite Donnees
alt
[Donnees invalides]
message Erreur (données incohérentes)
[Donnees valides]
VerifierContraintes (disponibilite, capacite, conflits , priorites)
alt
[Aucune contrainte detectée]
demande enregistrée [statut = EN_ATTENTE]
alt
[validation automatique]
creerReservation
mettreÀJourEDT
notification Validation [statut = VALIDEE]
[Validation manuelle]
NotifierNouvelleDemande
ConsulterDemande
ValiderReservation
creerReservation
MettreAjourEDT
NotificationValidation [statut = validee]
[Contrainte détectée]
ref
Traiter un conflit ponctuel
alt
[solution acceptable trouvée]
AjusterReservation
Mettre A jour EDT
notificationAjustement [statut = ajustée]
[Aucune solution acceptable]
refuserReservation
NotificationRefus [statut refusée]

Traiter un conflit ponctuel



Modifier un emploi du temps



Architecture Technique

Pour ce projet, nous avons choisi des technologies Web modernes basées sur JavaScript. L'objectif est d'utiliser un langage unique (JS) à la fois côté client et côté serveur afin de faciliter la collaboration et la maintenance du code au sein de l'équipe.

Choix de la Stack (MERN)

Nous avons opté pour une architecture Client-Serveur classique de type MERN (SQLite, Express, React, Node) / SERN (SQL, Express, React, Node) :

Frontend (Interface Utilisateur) :

React.js : Nous avons choisi la bibliothèque React.js pour le développement de l'interface client.

- Justification : React est un standard de l'industrie qui permet une approche par composants. Nous pourrons ainsi créer des briques réutilisables (ex : un composant Calendar, une CourseCard) pour accélérer le développement.
- Expérience Utilisateur : Cela nous permettra de construire une Single Page Application (SPA), offrant une navigation fluide et instantanée sans rechargement de page, ce qui est indispensable pour manipuler un emploi du temps complexe.

Backend (Serveur API) :

Node.js avec Express : Le serveur sera développé en Node.js, couplé au framework Express.

- Justification : Node.js nous permet de mettre en place rapidement une API REST performante.
- Architecture : L'utilisation d'Express facilitera le routage des requêtes (ex : /api/courses, /api/users) et nous permettra de respecter le pattern architectural MVC (Modèle-Vue-Contrôleur), assurant une séparation claire entre la logique métier et les données.

Base de Données :

SQLite (SQL / Relationnel) :

- Justification : Nous avons choisi une base relationnelle pour garantir la cohérence des liens entre cours, salles et professeurs. Le choix spécifique de SQLite s'explique par sa simplicité : c'est une base "sans serveur" (stockée dans un simple fichier) qui facilite le travail d'équipe sans installation complexe.

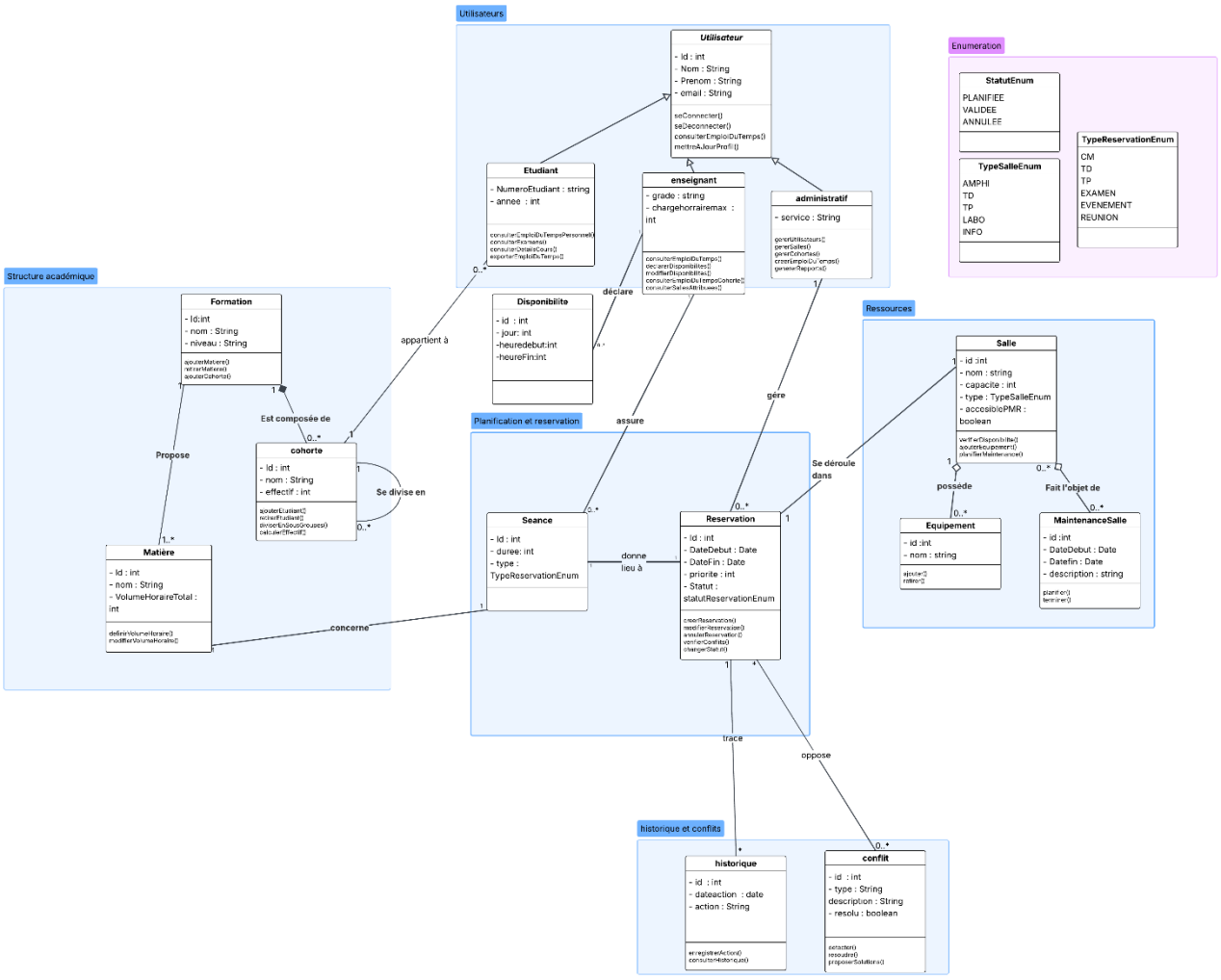
Outils de Développement

Afin d'harmoniser le travail de l'équipe, nous utilisons les outils suivants :

- **Éditeur de code :** Visual Studio Code (VS Code) avec configuration partagée (Prettier) pour maintenir un style de code uniforme au sein de l'équipe.
- **Versionning :** Git et GitHub pour la gestion des versions, le travail collaboratif et le suivi des évolutions du projet.
- **Tests et validation de l'API :** Postman, utilisé pour tester, vérifier et valider les endpoints de l'API REST avant leur intégration dans le frontend, garantissant ainsi le bon fonctionnement des échanges entre le client et le serveur.

Modélisation des Données

Diagramme de Classes (UML)



Algorithmique et Planification

Logique de Détection des Conflits

La détection des conflits permet d'identifier les situations où plusieurs événements utilisent en même temps la même ressource (salle, enseignant, cohorte) ou se chevauchent sur un même créneau horaire.

- **Principe :**

Chaque événement est caractérisé par ses attributs : date, heure de début, heure de fin, salle, enseignant et cohorte. Pour chaque nouvel événement à planifier, le système vérifie l'absence de chevauchement avec les événements déjà existants.

- **Étapes de la logique :**

Récupération des événements existants correspondant à la même salle, le même enseignant ou la même cohorte.
Comparaison des plages horaires :
Si l'heure de début d'un événement est comprise entre le début et la fin d'un autre événement → conflit détecté.
Si l'heure de fin d'un événement est comprise entre le début et la fin d'un autre événement → conflit détecté.

- **Gestion des conflits détectés :**

En cas de conflit, le système remonte la liste des séances concernées à l'administrateur, qui peut alors modifier l'horaire, changer de ressource ou annuler la réservation afin de garantir un emploi du temps cohérent et sans double réservation.

Algorithme de Planification

En effectuant nos recherches, nous avons rapidement compris que générer un emploi du temps parfait est un problème mathématique très difficile. Plus on a de cours et de profs, plus le temps de calcul pour trouver la "meilleure" solution est long. Comme nous n'avons pas les ressources pour coder une intelligence artificielle complexe, nous avons choisi une approche plus directe : l'Algorithme Glouton.

Le principe de notre algorithme est simple : on essaie de placer les cours un par un, au premier endroit qui fonctionne, sans revenir en arrière. Pour optimiser les chances que ça marche, on ne prend pas les cours au hasard. On les trie d'abord par priorité

Gestion de Projet et Organisation

Répartition des Rôles

Étant donné la nature du projet, une grande partie de la complexité repose sur la logique métier côté backend.
Pour cette raison, l'équipe a fait le choix d'une organisation où tous les membres participent au backend et à la base de données, afin d'assurer une bonne compréhension globale du système et une cohérence dans l'implémentation.

Le développement du frontend, bien que essentiel pour l'expérience utilisateur, nécessite moins de ressources humaines et repose principalement sur la conception d'interfaces et de composants réutilisables. Deux membres ont donc été désignés comme référents frontend.
La répartition des rôles est la suivante :

Youssef Edris – Chef de projet / Développement Fullstack :
Edris intervient sur l'ensemble du projet (backend, frontend et base de données). En tant que chef de projet, il assure la coordination de l'équipe, la répartition des tâches et la cohérence technique globale. Il participe au développement backend, à l'intégration frontend et à la mise en place des choix techniques structurants du système.

Adjaz Ryma – Développement Backend:
Ryma est principalement en charge du développement backend du projet. Elle participe à la mise en œuvre des fonctionnalités côté serveur et à l'implémentation de la logique métier nécessaire au bon fonctionnement du système de gestion des emplois du temps.

Belkacemi Cirine – Développement Backend & Frontend :
Cirine intervient à la fois sur le backend et le frontend du projet. Elle a notamment participé à la conception de la maquette de l'application, réalisée avec l'aide de l'équipe. Cette maquette sert de référence pour le développement du frontend, tant au niveau de la structure que de l'organisation des interfaces.

El Hathout Lina – Développement Backend & Frontend:
Lina est impliquée dans le développement backend et frontend. Elle contribue à l'implémentation des fonctionnalités de l'application et à l'intégration des interfaces utilisateur, en veillant à la cohérence globale et à la facilité d'utilisation du système.

Base de données (travail commun) :
La conception et la gestion de la base de données constituent un travail commun à l'ensemble de l'équipe. Tous les membres participent à la modélisation des données, à la définition des entités et de leurs relations, ainsi qu'à l'adaptation du schéma de données aux besoins fonctionnels du backend et du frontend, afin de garantir une base de données cohérente, évolutive et adaptée aux exigences du système.

Méthodologie

Nous appliquons un workflow Git strict pour éviter les pertes de code et les conflits :

1. Branches par fonctionnalité : Le travail ne se fait jamais directement sur la branche main. Chaque tâche fait l'objet d'une branche dédiée (ex: feat/login-screen, feat/algo-conflict).
2. Revue de Code : L'intégration du code se fait via des Pull Requests sur GitHub. Chaque fonctionnalité doit être validée par un autre membre de l'équipe avant d'être fusionnée.

Planning Prévisionnel (Gantt)

Le développement est découpé en trois grandes phases :

- Phase 1 : Initialisation et Conception (Semaines 1-2)
    - Mise en place de l'architecture technique (Client/Serveur).
    - Modélisation de la base de données (Diagrammes de Classes/MCD).

- Phase 2 : Fonctionnalités Essentielles (Semaines 3-5)
    - Développement du CRUD complet (Gestion des ressources).
    - Mise en place de l'authentification et des rôles.
    - Affichage de l'emploi du temps (Vue lecture).
- Phase 3 : Logique Métier Avancée (Semaines 6-8)
    - Implémentation de l'algorithme de détection et gestion des conflits.
    - Finalisation des tableaux de bord statistiques.
    - Tests globaux et corrections.

Conclusion et Perspectives

Cette phase de modélisation a été une étape charnière pour notre projet. Avant de nous lancer dans le développement, nous avons dû structurer notre pensée et, honnêtement, l'exercice s'est révélé plus complexe que prévu.

La principale difficulté n'était pas technique, mais logique. En réalisant les diagrammes de séquence, nous nous sommes rendu compte que nos premières idées étaient parfois trop vagues. Il a fallu trancher des débats au sein du groupe : jusqu'où doit-on aller dans le détail ? Comment représenter l'algorithme de génération automatique.

Nous avons aussi appris à faire la distinction entre l'analyse fonctionnelle (ce que le système doit faire) et l'architecture technique (comment il va le faire). Les retours sur nos diagrammes nous ont forcés à rester neutres dans la modélisation. Au final, ce dossier de conception nous offre une feuille de route claire : nous savons exactement quelles données circulent et comment gérer les cas d'erreurs avant même d'avoir écrit la première ligne de code.