TEK-UP University

Projet de Développement

MedFlow

Plateforme Intelligente de Gestion Médicale et de Suivi Clinique

Élaboré par : **Roua HadjHAssine**

Saif Eddine Yeddes

Chedy Soltani

Encadré par : **Mohammed Ben Rhouma**

Groupe : **GLSI-E**

Faculté : **Tekup University**

Année universitaire : **2025/2026**

Contents

List of Figures

List of Tables

Introduction générale

Dans un contexte de transformation numérique du secteur de la santé, la gestion administrative des établissements médicaux représente un défi majeur qui impacte directement la qualité des soins et l'expérience patient. Chaque jour, les cliniques doivent gérer simultanément la prise de rendez-vous, les dossiers patients, les ordonnances, la facturation et le suivi administratif, autant de tâches chronophages qui détournent les professionnels de santé de leur mission première. Parallèlement, les patients font face à leurs propres difficultés : prendre rendez-vous pendant les horaires d'ouverture, conserver leurs documents médicaux et accéder rapidement à leurs prescriptions.

Malgré l'existence de solutions de gestion pour les établissements de santé, la plupart présentent des limitations importantes : interfaces complexes, manque d'intégration entre les modules, absence de portail patient moderne, ou coûts prohibitifs pour les petites structures. Cette fragmentation des outils entraîne des erreurs de traitement et une expérience utilisateur insatisfaisante tant pour le personnel médical que pour les patients.

C'est dans ce contexte que le projet **MedFlow** a été conçu. Pensée comme une plateforme SaaS complète, notre solution vise à digitaliser la gestion quotidienne des cliniques médicales en offrant un écosystème intégré. Pour les professionnels de santé, **MedFlow** propose des outils de gestion d'agenda, de dossiers médicaux, d'ordonnances avec export PDF et de facturation intégrée. Pour les patients, la plateforme offre un portail dédié permettant la réservation en ligne, la consultation de leur historique médical et le paiement sécurisé.

MedFlow se distingue par son architecture moderne basée sur Next.js et React, son système de gestion des rôles (RBAC) adapté aux différents profils (administrateur, médecin, réceptionniste, patient), et son approche multi-tenant garantissant l'isolation des données de chaque clinique. L'intégration de Stripe pour les paiements et la génération automatique de PDF illustrent la volonté de créer une solution complète et opérationnelle.

Ainsi, **MedFlow** aspire à transformer l'organisation des cliniques médicales en proposant un outil moderne, efficace et centré sur les besoins réels des professionnels de santé et de leurs patients, tout en respectant les standards de sécurité et de confidentialité des données médicales.

Chapter 1

Cadre général du projet

La digitalisation du secteur de la santé constitue un enjeu majeur pour améliorer l'efficacité des établissements médicaux et l'expérience des patients. Les cliniques et cabinets médicaux font face à une charge administrative croissante liée à la gestion des rendez-vous, des dossiers patients, des ordonnances et de la facturation, ce qui réduit le temps consacré aux soins. Par ailleurs, les patients recherchent davantage d'autonomie à travers des services numériques tels que la prise de rendez-vous en ligne, l'accès aux documents médicaux et le paiement dématérialisé.

Dans ce contexte, le projet **MedFlow** propose une solution SaaS innovante dédiée à la gestion globale des cliniques médicales. La plateforme automatise les processus administratifs, centralise les informations patients et facilite la coordination entre les différents acteurs. Grâce à une architecture moderne et des fonctionnalités adaptées, **MedFlow** contribue à optimiser l'organisation interne tout en améliorant la qualité des services offerts aux patients.

1.1 Introduction

Ce chapitre examine en détail le contexte global du projet MedFlow et les solutions existantes sur le marché des systèmes de gestion médicale. Il se compose de plusieurs parties principales : l'analyse du cadre du projet, qui définit le contexte dans lequel le projet s'inscrit, l'étude de l'existant, qui identifie les lacunes des solutions actuelles, et la présentation de notre solution innovante qui répond aux besoins non satisfaits du secteur médical.

1.2 Cadre du projet

Le cadre de ce projet vise à répondre aux besoins croissants en matière de digitalisation des établissements de santé. L'application **MedFlow** se positionne comme une solution complète et centralisée permettant :

- **Pour les administrateurs (propriétaires de cliniques) :**
  - La création et la configuration complète de leur clinique
  - La gestion des services médicaux proposés et de leur tarification
  - La supervision du personnel médical et administratif
  - Le suivi global de l'activité via des tableaux de bord
- **Pour les médecins :**

  - La gestion complète de leur agenda et disponibilités
  - La consultation et mise à jour des dossiers médicaux patients
  - L'établissement et l'export d'ordonnances au format PDF
  - Le suivi de l'historique médical de leurs patients

- **Pour les réceptionnistes :**
  - La prise et gestion des rendez-vous patients
  - L'enregistrement et mise à jour des informations patients
  - La gestion de la facturation et suivi des paiements
  - La coordination avec les médecins

- **Pour les patients :**
  - La réservation de rendez-vous en ligne 24h/24
  - L'accès à leur historique médical et consultations
  - Le téléchargement de leurs ordonnances et documents médicaux
  - Le paiement en ligne sécurisé via Stripe

Contrairement aux solutions existantes qui proposent souvent des modules séparés ou des interfaces vieillissantes, **MedFlow** offre une approche holistique avec une architecture moderne basée sur Next.js et React. La plateforme intègre un système de gestion des rôles (RBAC) robuste et une architecture multi-tenant permettant à chaque clinique de bénéficier d'un environnement isolé et sécurisé.

1.3 Idée maîtresse

L'idée maîtresse de **MedFlow** repose sur la création d'une plateforme SaaS complète qui digitalise l'ensemble du parcours patient et des processus administratifs d'une clinique médicale, de la prise de rendez-vous jusqu'au paiement, en passant par la consultation médicale et la gestion documentaire.

Notre vision est de créer un écosystème où :

- Les professionnels de santé peuvent se concentrer sur leur cœur de métier en automatisant les tâches administratives répétitives
- Les administrateurs disposent d'une vue complète sur l'activité de leur clinique avec des outils de gestion simples et efficaces
- Les patients bénéficient d'une autonomie totale dans la gestion de leur parcours de soins via un portail dédié moderne et intuitif
- L'ensemble de l'équipe médicale et administrative collabore efficacement grâce à un système centralisé et partagé

L'objectif principal est de réduire considérablement le temps consacré aux tâches administratives, d'améliorer la coordination entre les différents acteurs, et d'offrir une expérience patient moderne et fluide. **MedFlow** vise également à démocratiser l'accès aux outils de gestion médicale performants, permettant aux petites et moyennes cliniques de bénéficier d'une solution professionnelle à coût maîtrisé grâce au modèle SaaS.

1.4 Étude et critique de l'existant

Afin de mieux comprendre le paysage des solutions de gestion médicale et d'identifier les lacunes existantes, nous examinons les applications et plateformes qui dominent actuellement le marché. Cette analyse nous permet de comprendre leurs forces et faiblesses pour concevoir une solution véritablement adaptée aux besoins actuels.

1.4.1 Analyse de l'existant

Nous analysons trois solutions majeures dans le domaine de la gestion des cliniques et cabinets médicaux : **Doctolib Pro**, **Maiia** et **Médimust**.

Solution 1 : Doctolib Pro

Doctolib Pro est une plateforme française leader qui permet aux professionnels de santé de gérer leurs rendez-vous en ligne et leur agenda médical.

Les points forts :

- Large base d'utilisateurs patients facilitant les prises de rendez-vous
- Système de prise de rendez-vous en ligne robuste et performant
- Application mobile pour praticiens bien conçue
- Gestion d'agenda avancée avec rappels automatiques
- Notoriété et confiance établies auprès du grand public

Les points faibles :

- Coût d'abonnement élevé pour les petites structures
- Fonctionnalités de gestion de dossiers médicaux limitées
- Absence de système de facturation intégré complet
- Pas de génération d'ordonnances PDF
- Manque de personnalisation pour les cliniques multi-praticiens
- Dépendance forte à l'écosystème Doctolib

Solution 2 : Maiia

Maiia est une plateforme de prise de rendez-vous médicaux en ligne développée par le groupe Cegedim, proposant également des outils de gestion pour les praticiens.

Les points forts :

- Interface utilisateur moderne et épurée
- Prise de rendez-vous en ligne simple pour les patients
- Gratuit pour certains profils de praticiens
- Système de rappels par SMS et email

Les points faibles :

- Fonctionnalités limitées en version gratuite
- Absence de système complet de gestion des dossiers patients
- Pas de module de facturation intégré
- Absence d'outils pour la gestion multi-utilisateurs en clinique
- Pas de portail patient complet avec historique médical

Solution 3 : Médimust

Médimust est un logiciel de gestion de cabinet médical qui propose des fonctionnalités d'agenda, de dossiers patients et de facturation.

Les points forts :

- Solution complète intégrant agenda, dossiers et facturation
- Interface conçue spécifiquement pour le secteur médical
- Fonctionnalités de gestion des dossiers patients détaillées
- Support de la facturation médicale

Les points faibles :

- Interface utilisateur vieillissante et peu intuitive
- Pas de portail patient moderne pour prise de rendez-vous en ligne
- Absence de paiement en ligne intégré
- Technologie obsolète nécessitant des installations locales
- Coût d'acquisition et de maintenance élevé
- Absence d'architecture multi-tenant pour cliniques

1.4.2 Critique de l'existant

Le tableau suivant présente une comparaison des solutions selon plusieurs critères clés :

1.5 Problématique

L'analyse du marché actuel révèle plusieurs problématiques majeures dans le domaine de la gestion des établissements médicaux :

1. **Fragmentation des outils :** Les solutions existantes se concentrent généralement sur un aspect spécifique (prise de rendez-vous, dossiers médicaux, facturation) sans offrir une intégration complète. Cette fragmentation oblige les cliniques à utiliser plusieurs logiciels différents, créant des redondances de saisie et des risques d'erreurs.
2. **Absence de portail patient moderne :** La plupart des solutions n'offrent pas aux patients un véritable espace personnel leur permettant de gérer l'ensemble de leur parcours. Les patients restent dépendants des horaires d'ouverture des secrétariats pour leurs démarches administratives.

Critères	Doctolib Pro	Maiia	Médimust
Objectif principal	Prise de RDV en ligne et agenda	Prise de RDV en ligne	Gestion complète de cabinet
Interface moderne	Oui	Oui	Non
Portail patient	Limité (RDV uniquement)	Limité (RDV uniquement)	Non
Dossiers médicaux	Basique	Non	Oui
Ordonnances PDF	Non	Non	Limité
Facturation intégrée	Limitée	Non	Oui
Paiement en ligne	Non	Non	Non
Multi-tenant	Non	Non	Non
Architecture moderne	Oui	Oui	Non
Coût	Élevé	Gratuit/Payant	Très élevé
Points forts	Notoriété, agenda performant	Interface simple, gratuit	Solution complète
Points faibles	Coût, fonctionnalités limitées	Peu de fonctionnalités	Interface obsolète

Table 1.1: Tableau comparatif des solutions existantes

3. **Interfaces obsolètes et complexes :** De nombreux logiciels médicaux souffrent d'interfaces vieillissantes, peu intuitives, nécessitant des formations longues pour le personnel médical et administratif.

4. **Coûts prohibitifs :** Les solutions professionnelles complètes nécessitent souvent des investissements initiaux importants (licences, installations, formations), rendant leur accès difficile pour les petites et moyennes structures médicales.

5. **Manque de flexibilité multi-utilisateurs :** Les systèmes actuels ne gèrent pas efficacement les besoins spécifiques des cliniques avec plusieurs médecins, différents services et multiples rôles administratifs. L'isolation des données par clinique (multi-tenant) est rarement proposée.

6. **Absence de paiement en ligne :** Très peu de solutions intègrent un système de paiement en ligne sécurisé, obligeant les patients à se déplacer pour régler leurs consultations.

Face à ces constats, la question centrale est : *Comment créer une plateforme SaaS moderne, complète et accessible qui répond aux besoins de tous les acteurs d'une clinique médicale tout en automatisant les processus administratifs et en garantissant la sécurité des données médicales ?*

1.6 Solution proposée

Pour répondre aux problématiques identifiées, **MedFlow** propose une solution SaaS innovante et complète qui se distingue par plusieurs caractéristiques clés :

1.6.1 Une plateforme unifiée pour tous les acteurs

Notre solution adopte une approche centralisée en intégrant tous les modules nécessaires à la gestion d'une clinique :

Module d'authentification et RBAC : Système de connexion sécurisé avec Auth.js, gestion fine des rôles et permissions (Admin, Médecin, Réceptionniste, Patient), isolation des données par clinique (architecture multi-tenant), et sécurité renforcée avec hashing des mots de passe.

Module de gestion des patients : CRUD complet des dossiers patients, profils détaillés avec informations médicales et administratives, historique complet des visites et consultations, et archivage sécurisé des documents médicaux.

Module agenda et rendez-vous : Calendrier interactif pour la gestion des disponibilités, prise de rendez-vous en ligne par les patients, modification et annulation simplifiées, système de rappels automatiques, et vue d'ensemble pour les réceptionnistes et médecins.

Module consultations et ordonnances : Saisie des diagnostics et notes de consultation, création d'ordonnances médicales, export automatique en PDF, mise à disposition immédiate pour les patients, et archivage avec traçabilité complète.

Module facturation et paiement : Génération automatique des factures, suivi des paiements et historique, intégration de Stripe pour le paiement en ligne sécurisé, gestion des tarifs par service, et rapports financiers.

Portail patient dédié : Interface moderne et intuitive, réservation de rendez-vous 24h/24, accès à l'historique médical complet, téléchargement des ordonnances et documents, paiement en ligne des consultations, et tableau de bord personnalisé.

1.6.2 Architecture moderne et performante

MedFlow s'appuie sur des technologies web modernes :

- Front-end développé avec Next.js 14 et React pour une expérience utilisateur fluide
- Interface utilisateur construite avec Tailwind CSS et shadcn/ui pour un design moderne
- API Routes Next.js pour le back-end (ou NestJS selon les besoins)
- Base de données PostgreSQL ou MySQL pour la fiabilité
- Validation des formulaires pour la sécurité des données
- Déploiement sur Vercel et Railway/Render pour la scalabilité

1.6.3 Multi-tenant et sécurité

La plateforme garantit l'isolation et la sécurité des données avec une architecture multi-tenant utilisant un `tenantId`, permettant à chaque clinique de disposer de son propre environnement sécurisé. Le système intègre le hashing des mots de passe, une authentification robuste, et assure la conformité avec les réglementations sur les données de santé.

1.6.4 Accessibilité et modèle SaaS

MedFlow adopte un modèle économique accessible avec un abonnement mensuel adapté aux petites et grandes structures, sans investissement initial important, incluant des mises à jour automatiques, un support technique, et un accès depuis n'importe quel appareil connecté.

1.7 Méthodologie de développement : Scrum

La méthodologie **SCRUM** est particulièrement adaptée pour développer **MedFlow** car elle permet de livrer régulièrement des fonctionnalités testables et de s'adapter rapidement aux retours des utilisateurs.

1.7.1 Principe du Scrum

Scrum permet de produire du logiciel fonctionnel à chaque sprint (2 à 3 semaines). Le Product Owner définit les priorités, l'équipe s'organise pour les réaliser, et à chaque fin de sprint, les fonctionnalités développées sont présentées pour recueillir des feedbacks.

1.7.2 Les rôles Scrum dans notre projet

Product Owner : Définit et priorise les fonctionnalités de MedFlow, valide les livrables, et ajuste les priorités selon les retours.

Scrum Master : anime les cérémonies Scrum, élimine les obstacles, et facilite la communication au sein de l'équipe.

Équipe de Développement : Développeurs Full-Stack, designers UI/UX, architecte logiciel, et testeurs qui s'auto-organisent pour livrer les fonctionnalités.

1.7.3 Planning des sprints

Le projet est organisé en 5 sprints :

Sprint 1 : Authentification avec Auth.js, système RBAC, onboarding des cliniques, et dashboard simple.

Sprint 2 : CRUD complet des patients, gestion des services médicaux, et système d'agenda avec prise de rendez-vous.

Sprint 3 : Module de consultation médicale, création et gestion des ordonnances, et export PDF.

Sprint 4 : Module de facturation complet, intégration Stripe, portail patient avec réservation et téléchargement de documents.

Sprint 5 : Tableaux de bord analytics, calendrier avancé, notifications email, tests finaux et déploiement en production.

1.8 Conclusion

Ce chapitre a présenté le cadre général du projet **MedFlow**, en mettant en évidence les défis actuels de la digitalisation des cliniques médicales et les lacunes des solutions existantes. L'analyse comparative a révélé un besoin clair pour une solution SaaS complète, moderne et accessible.

Notre solution se distingue par son approche unifiée intégrant tous les modules nécessaires, son architecture moderne basée sur Next.js et React, son système multi-tenant, et l'adoption de la méthodologie Scrum avec 5 sprints bien définis. Dans les chapitres suivants, nous détaillerons les spécifications fonctionnelles et techniques, l'architecture logicielle, la conception de la base de données, et les maquettes d'interface de **MedFlow**.

Chapter 2

Préparation du Projet

2.1 Introduction

La phase de préparation constitue la base méthodologique du projet **MedFlow**. Elle vise à définir clairement les aspects organisationnels, techniques et fonctionnels nécessaires à la mise en place d'un système fiable et évolutif, adapté aux établissements médicaux. Cette étape permet d'identifier les besoins des différents acteurs et de structurer une vision globale du projet, garantissant la cohérence entre les objectifs de digitalisation, les choix technologiques et les contraintes propres au secteur de la santé.

2.2 Analyse des besoins

La transformation numérique du secteur de la santé a mis en évidence un besoin croissant de solutions de gestion modernes et intégrées. Les cliniques et cabinets médicaux sont confrontés à une charge administrative élevée liée à la gestion des rendez-vous, des dossiers patients, des ordonnances et de la facturation, ce qui limite le temps consacré aux soins.

Chaque acteur exprime des besoins spécifiques : les administrateurs souhaitent optimiser l'organisation globale, les médecins recherchent un accès rapide et fiable aux informations médicales, les réceptionnistes ont besoin d'outils simples pour la gestion quotidienne, tandis que les patients demandent davantage d'autonomie dans leur parcours de soins. Dans ce contexte, **MedFlow** vise à proposer une solution SaaS complète permettant de centraliser les processus, d'automatiser les tâches administratives et d'offrir des interfaces modernes et intuitives adaptées à chaque profil d'utilisateur.

2.3 Analyse et spécifications des besoins

2.3.1 Identification des acteurs

Le système repose sur quatre catégories d'acteurs principaux, chacun jouant un rôle spécifique dans l'écosystème fonctionnel de la clinique :

- **Administrateurs (Propriétaires) :** responsables de la création et de la configuration de la clinique, de la gestion des services médicaux proposés, de la définition des tarifs, de la supervision du personnel (médecins et réceptionnistes) et de l'accès aux tableaux de bord et statistiques globales d'activité.

- **Médecins :** professionnels de santé gérant leur agenda personnel, accédant aux dossiers médicaux de leurs patients, réalisant des consultations, établissant des ordonnances avec export PDF, et assurant le suivi de l'historique médical de leurs patients.
- **Réceptionnistes :** personnel administratif en charge de la prise et de la gestion des rendez-vous, de l'enregistrement et de la mise à jour des informations patients, de la gestion de la facturation, du suivi des paiements et de la coordination générale avec les médecins.
- **Patients :** utilisateurs finaux accédant au portail dédié pour réserver leurs rendez-vous en ligne, consulter leur historique médical, télécharger leurs ordonnances et documents médicaux, et effectuer leurs paiements en ligne de manière sécurisée.

2.3.2 Spécifications des besoins fonctionnels

Les besoins fonctionnels se répartissent selon les quatre interfaces principales du système, correspondant aux quatre types d'acteurs.

Interface Administrateur :

- Création et configuration complète de la clinique (nom, adresse, services).
- Gestion des services médicaux proposés et de leur tarification.
- Ajout, modification et suppression des comptes médecins et réceptionnistes.
- Accès aux tableaux de bord d'activité et statistiques globales.
- Configuration des paramètres généraux de la clinique.
- Gestion du catalogue des services et spécialités médicales.

Interface Médecin :

- Gestion complète de l'agenda personnel et des disponibilités.
- Accès en lecture et écriture aux dossiers médicaux des patients.
- Création et modification de consultations avec notes médicales.
- Établissement d'ordonnances médicales avec génération PDF automatique.
- Consultation de l'historique complet des visites des patients.
- Visualisation des rendez-vous du jour et à venir.

Interface Réceptionniste :

- Prise de rendez-vous pour les patients (en personne ou par téléphone).
- Modification et annulation de rendez-vous existants.
- Enregistrement de nouveaux patients avec informations complètes.
- Mise à jour des coordonnées et informations administratives des patients.
- Génération et gestion des factures pour les consultations.
- Suivi des paiements et relances si nécessaire.
- Vue calendrier globale de tous les médecins.

Interface Patient :

- Inscription et création de compte patient avec validation par email.
- Réservation de rendez-vous en ligne 24h/24 selon les disponibilités.
- Modification ou annulation de rendez-vous existants.
- Consultation de l'historique médical complet (consultations passées).
- Téléchargement des ordonnances et documents médicaux au format PDF.
- Paiement en ligne sécurisé des consultations via Stripe.
- Tableau de bord personnalisé avec prochains rendez-vous et rappels.

2.3.3 Spécifications des besoins non fonctionnels

Les besoins non fonctionnels définissent les contraintes qualitatives encadrant le fonctionnement du système. Ils sont synthétisés dans le tableau ci-dessous :

Catégorie	**Description**
Performance	Temps de chargement des pages inférieur à 2 secondes, réactivité optimale des interfaces
Sécurité	Authentification robuste avec Auth.js, hashing des mots de passe, chiffrement des données sensibles, conformité RGPD pour les données de santé
Scalabilité	Architecture multi-tenant supportant plusieurs cliniques simultanément, capacité à gérer un grand volume de patients et rendez-vous
Fiabilité	Disponibilité du service supérieure à 99%, sauvegarde automatique des données, système de récupération en cas de panne
Ergonomie	Interfaces intuitives et modernes avec Tailwind CSS et shadcn/ui, design responsive adapté aux ordinateurs, tablettes et smartphones
Maintenabilité	Architecture modulaire avec Next.js, code propre et documenté, séparation claire des responsabilités
Compatibilité	Fonctionnement multi-navigateurs (Chrome, Firefox, Safari, Edge), support des appareils mobiles iOS et Android
Validation	Validation stricte des formulaires , messages d'erreur clairs et explicites
Déploiement	Hébergement sur Vercel (frontend) et Railway/Render (backend et base de données), intégration continue avec GitHub

Table 2.1: Besoins non fonctionnels du projet **MedFlow**

2.4 Architecture Logicielle

Le système repose sur une architecture *Full Stack* moderne, basée sur Next.js 14 qui assure à la fois le frontend et le backend via les API Routes. Cette approche unifiée garantit une excellente performance, une maintenance simplifiée et un déploiement optimisé. L'architecture intègre également une base de données relationnelle (PostgreSQL ou MySQL) et un système de paiement sécurisé via Stripe.

2.4.1 Choix technologiques

Dans cette étape, nous présentons les outils de modélisation, de développement, les frameworks et langages utilisés, ainsi que l'architecture de l'application.

Outils de développement

Visual Studio Code (VS Code) : VS Code est un éditeur de code source léger, puissant et extensible, offrant des fonctionnalités telles que l'autocomplétion intelligente, le débogage intégré, la gestion Git et un large écosystème d'extensions. Il supporte parfaitement Next.js, React, TypeScript et les autres technologies modernes du web.

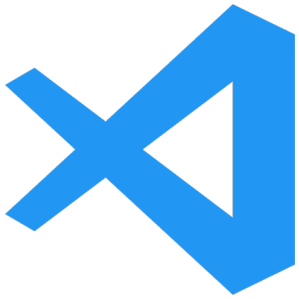

Figure 2.1: Logo Visual Studio Code

Git et GitHub : Git est un système de contrôle de version distribué permettant de suivre les modifications du code source. GitHub est une plateforme d'hébergement de code qui facilite la collaboration, le versioning et l'intégration continue.

Figure 2.2: Logo GitHub

Frameworks et langages de développement

Next.js 14 : Next.js est un framework React moderne qui permet de créer des applications web full-stack performantes. Il offre le rendu côté serveur (SSR), la génération de sites statiques (SSG), les API Routes pour le backend, et de nombreuses optimisations automatiques. Next.js 14 apporte App Router, Server Components et des améliorations significatives de performance.

Figure 2.3: Logo Next.js

React : React est une bibliothèque JavaScript open-source développée par Meta pour la création d'interfaces utilisateur interactives et réactives. Elle repose sur le concept de composants réutilisables et sur une gestion efficace de l'état, permettant de créer des applications web dynamiques et performantes.

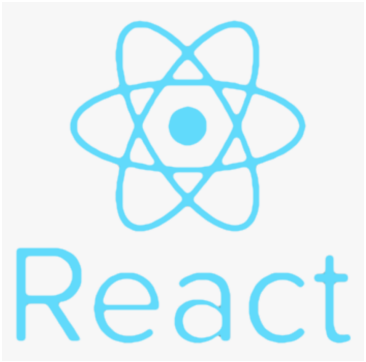

Figure 2.4: Logo React

TypeScript : TypeScript est un sur-ensemble typé de JavaScript qui améliore la robustesse du code grâce au typage statique, facilitant la détection d'erreurs en développement et l'autocomplétion dans l'IDE.

Figure 2.5: Logo TypeScript

Base de données et ORM

PostgreSQL : PostgreSQL est un système de gestion de base de données relationnelle (SGBDR) open-source, robuste, performant et hautement conforme aux standards SQL. Il offre des fonctionnalités avancées (transactions ACID, intégrité référentielle, types de données complexes) et est particulièrement adapté aux applications nécessitant fiabilité et sécurité.

Figure 2.6: Logo PostgreSQL

Authentification et sécurité

Auth.js (NextAuth) : Auth.js (anciennement NextAuth.js) est une solution d'authentification complète pour Next.js. Elle supporte l'authentification par email/mot de passe, les providers OAuth (Google, GitHub, etc.), la gestion des sessions et des rôles utilisateurs.

Figure 2.7: Logo Auth.js

Paiement en ligne

Stripe : Stripe est une plateforme de paiement en ligne leader qui permet d'accepter des paiements sécurisés par carte bancaire. Elle offre des API complètes, un mode test pour le développement, et gère automatiquement la sécurité PCI-DSS.

Figure 2.8: Logo Stripe

Déploiement et hébergement

Vercel : Vercel est la plateforme de déploiement optimale pour Next.js, offrant des déploiements automatiques depuis GitHub, un CDN global, des previews pour chaque pull request, et des performances exceptionnelles.

Figure 2.9: Logo Vercel

2.4.2 Architecture de l'application

Architecture Full Stack avec Next.js : MedFlow adopte une architecture moderne basée sur Next.js qui unifie le frontend et le backend dans une seule application cohérente.

- **Frontend (Client Components) :** Interfaces utilisateur interactives construites avec React, Tailwind CSS et shadcn/ui. Chaque rôle (Admin, Médecin, Réceptionniste, Patient) dispose de son interface dédiée avec des composants réutilisables.
- **Backend (API Routes) :** API RESTful implémentée via les Next.js API Routes, gérant l'authentification, les opérations CRUD, la logique métier et l'intégration avec Stripe.
- **Base de données (PostgreSQL) :** Gestion des données via Prisma ORM qui assure le typage, les migrations et l'accès sécurisé aux données (patients, médecins, rendez-vous, consultations, ordonnances, factures).
- **Authentification (Auth.js) :** Système de connexion sécurisé avec gestion des rôles (RBAC) et des permissions spécifiques à chaque type d'utilisateur.
- **Multi-tenant :** Isolation des données par clinique via un champ `tenantId` dans chaque table, garantissant que chaque clinique accède uniquement à ses propres données.

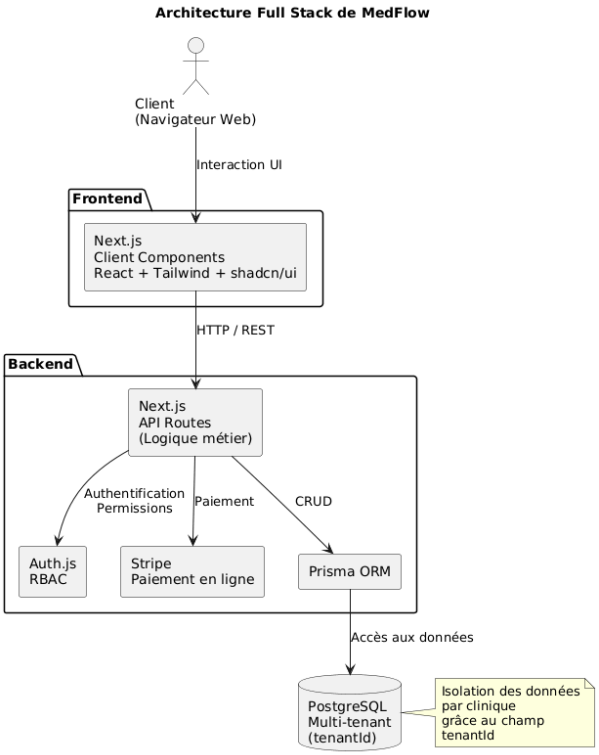

Figure 2.10: Architecture Full Stack de MedFlow

2.4.3 Fonctionnement Global

Le processus opérationnel de la plateforme se déroule selon les flux suivants :

Flux Patient :

1. Le patient s'inscrit ou se connecte via le portail dédié.
2. Il consulte les disponibilités des médecins et réserve un rendez-vous en ligne.
3. Le système enregistre le rendez-vous et envoie une confirmation par email.
4. Après la consultation, le patient accède à son ordonnance en PDF et peut payer en ligne via Stripe.
5. L'historique complet est conservé et accessible à tout moment.

Flux Médecin :

1. Le médecin se connecte et visualise son agenda du jour.
2. Il accède au dossier du patient pour la consultation.
3. Il saisit les notes médicales, le diagnostic et crée l'ordonnance.
4. Le système génère automatiquement le PDF de l'ordonnance.
5. L'ordonnance est immédiatement disponible pour le patient.

Flux Réceptionniste :

1. La réceptionniste gère les rendez-vous via l'interface dédiée.
2. Elle enregistre les nouveaux patients et met à jour leurs informations.

3. Elle génère les factures après les consultations.
4. Elle suit les paiements et relance si nécessaire.

Flux Administrateur :

1. L'administrateur configure la clinique et ses services.
2. Il crée les comptes médecins et réceptionnistes.
3. Il définit les tarifs et paramètres généraux.
4. Il consulte les tableaux de bord d'activité et statistiques.

2.5 Pilotage du projet avec Scrum

La méthodologie **Scrum** a été adoptée afin de structurer le projet selon une approche itérative, incrémentale et orientée valeur. Elle facilite l'adaptation continue aux besoins métier du secteur médical, la communication entre les parties prenantes et la maîtrise des risques.

2.5.1 Backlog de produit

Les fonctionnalités prioritaires du produit incluent :

- Système d'authentification multi-rôles avec RBAC
- Module de gestion des patients (CRUD complet)
- Système d'agenda et prise de rendez-vous en ligne
- Module de consultations médicales
- Génération automatique d'ordonnances PDF
- Module de facturation et intégration Stripe
- Portail patient complet et autonome
- Tableaux de bord et analytics
- Architecture multi-tenant sécurisée

2.5.2 Planification des Sprints

Le projet **MedFlow** est organisé en 5 sprints de 2 à 3 semaines chacun :

Sprint 1 : Authentification et Dashboard

- Configuration du projet Next.js avec TypeScript
- Mise en place de la base de données PostgreSQL avec Prisma
- Implémentation de l'authentification avec Auth.js
- Système RBAC (Admin, Médecin, Réceptionniste, Patient)
- Onboarding des cliniques (multi-tenant)
- Dashboards de base pour chaque rôle

Sprint 2 : Gestion Patients et Rendez-vous

- CRUD complet des patients avec validation
- Module de gestion des services médicaux
- Système d'agenda avec calendrier interactif
- Prise de rendez-vous (réceptionniste et patient)
- Système de notifications et rappels
- Interface réceptionniste optimisée

Sprint 3 : Consultations et Ordonnances

- Module de consultation médicale avec saisie des notes
- Création et gestion des ordonnances
- Génération automatique de PDF avec template personnalisé
- Archivage et historique des consultations
- Interface médecin optimisée pour le workflow

Sprint 4 : Facturation et Portail Patient

- Module de facturation complet
- Intégration Stripe pour le paiement en ligne (mode test)
- Portail patient avec réservation de rendez-vous
- Consultation de l'historique médical
- Téléchargement des ordonnances et documents
- Paiement en ligne des consultations

2.6 Diagramme de classe global

Un **diagramme de classes** est une représentation graphique des différentes entités du système ainsi que de leurs relations. Il permet de visualiser la structure des données, les attributs de chaque entité, et la façon dont elles interagissent entre elles, ce qui est essentiel pour comprendre la conception de la base de données et l'architecture logicielle.

Le diagramme présenté ci-dessous illustre les principales entités du système MedFlow : *User*, *Clinic*, *Patient*, *Appointment*, *Consultation*, *Prescription*, *Invoice*, *Service*, ainsi que leurs relations et cardinalités.

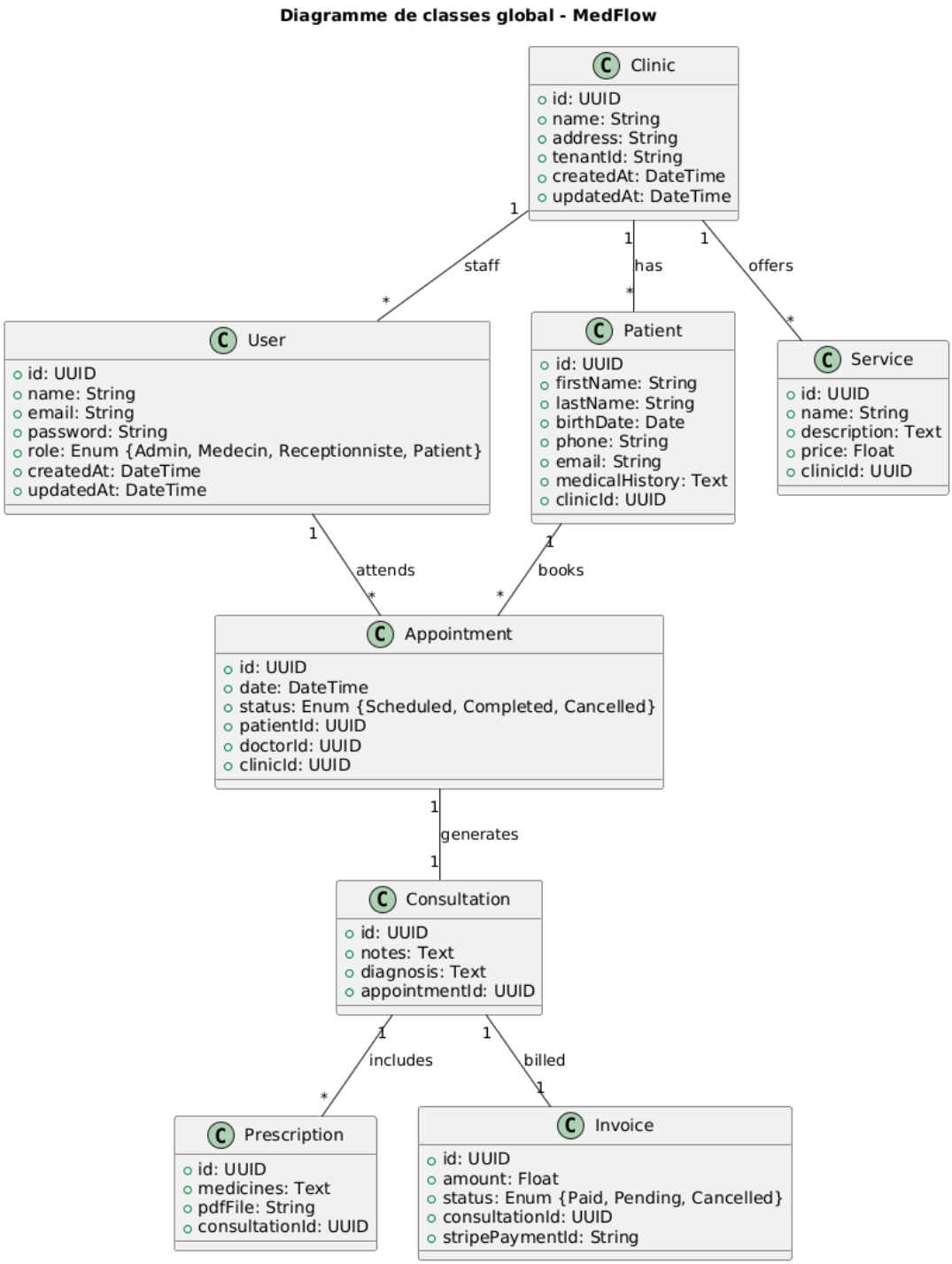

Figure 2.11: Diagramme de classes global du système MedFlow

Description des principales entités :

- **User :** Représente tous les utilisateurs du système (Admin, Médecin, Réceptionniste, Patient) avec leur rôle et leurs informations d'authentification.
- **Clinic :** Représente une clinique dans le système multi-tenant, avec ses services et son personnel.
- **Patient :** Contient les informations détaillées des patients, reliées à leurs rendez-vous et consultations.
- **Appointment :** Représente un rendez-vous entre un patient et un médecin, avec date, heure et statut.

- **Consultation :** Contient les notes médicales, diagnostics et ordonnances associées à une visite.
- **Prescription :** Représente une ordonnance médicale avec les médicaments prescrits et le fichier PDF généré.
- **Invoice :** Facture associée à une consultation avec montant, statut de paiement et lien Stripe.
- **Service :** Services médicaux proposés par la clinique avec leurs tarifs.

2.7 Diagramme de cas d'utilisation global

Un **diagramme de cas d'utilisation** représente les interactions entre les différents acteurs du système et les fonctionnalités qu'ils utilisent. Il permet de visualiser de manière synthétique les services offerts par MedFlow et les rôles des différents utilisateurs.

Les principaux cas d'utilisation du système s'articulent autour des actions suivantes :

Administrateur :

- Créer et configurer la clinique
- Gérer les services médicaux et tarifs
- Gérer le personnel (médecins, réceptionnistes)
- Consulter les tableaux de bord et statistiques

Médecin :

- Gérer son agenda et disponibilités
- Consulter les dossiers patients
- Réaliser des consultations
- Créer et gérer des ordonnances
- Consulter l'historique médical

Réceptionniste :

- Prendre et gérer les rendez-vous
- Enregistrer et mettre à jour les patients
- Gérer la facturation
- Suivre les paiements

Patient :

- S'inscrire et se connecter
- Réserver un rendez-vous en ligne
- Consulter son historique médical
- Télécharger ses ordonnances
- Payer en ligne

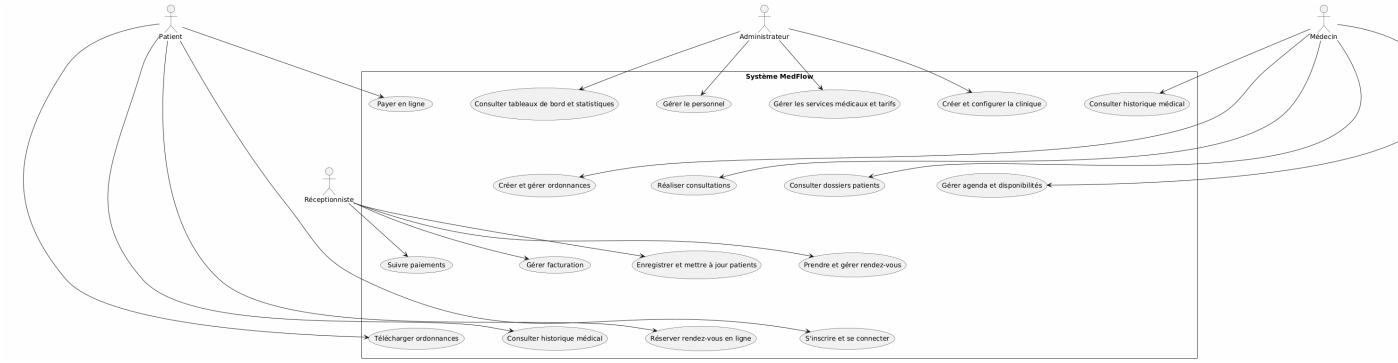

Figure 2.12: Diagramme de cas d'utilisation global du système MedFlow

2.8 Conclusion

La phase de préparation a permis d'établir les bases méthodologiques et techniques du projet **MedFlow**. L'analyse des besoins a identifié les attentes des différents acteurs, tandis que les spécifications fonctionnelles et non fonctionnelles ont défini un cadre de conception adapté au secteur médical.

Le choix d'une architecture Full Stack basée sur des technologies modernes garantit la robustesse, la scalabilité et la maintenabilité de la plateforme. Les modèles de conception, appuyés par les diagrammes de classes et de cas d'utilisation, offrent une vision claire du fonctionnement global du système. Enfin, l'adoption de la méthodologie Scrum, structurée en sprints, permet une réalisation progressive et maîtrisée du projet.

Chapter 3

Release 1 : Mise en place du socle technique et gestion de base

Introduction de la Release

La première release, désignée sous l'appellation **Release 1 – MVP Fondations**, constitue le noyau initial de la plateforme *MedFlow*. Elle incarne la version minimale viable permettant de mettre en place l'infrastructure technique de base, le système d'authentification multi-rôles, ainsi que les premières fonctionnalités essentielles de gestion des patients et des rendez-vous.

Dans l'optique d'assurer une montée en puissance progressive et contrôlée du produit, cette release adopte une segmentation agile fondée sur la méthodologie Scrum. Elle se compose de deux sprints successifs, structurés de manière à introduire graduellement les fonctionnalités essentielles : la mise en place de l'authentification et des dashboards, puis l'implémentation de la gestion complète des patients, des services médicaux et du système de rendez-vous. Chacun de ces sprints aboutit à un incrément du produit présentant une valeur opérationnelle immédiate, conformément aux principes de l'agilité et de l'amélioration continue.

Cette première phase a pour vocation de matérialiser les fondements techniques, architecturaux et fonctionnels du système, tout en garantissant une extensibilité optimale pour les futures releases. Elle pose les bases d'une plateforme SaaS multi-tenant sécurisée et performante, capable de gérer efficacement les besoins quotidiens d'une clinique médicale.

3.1 Sprint 1 : Authentification, RBAC et Dashboards

3.1.1 Introduction

Le premier sprint se concentre exclusivement sur l'infrastructure d'authentification et la mise en place du système de gestion des rôles et permissions (RBAC). En tant que pierre angulaire de l'ensemble de la plateforme, ce sprint vise à définir la structure de sécurité du système, à configurer l'environnement de développement avec Next.js 14, et à établir les premières interfaces utilisateur adaptées à chaque rôle.

3.1.2 Objectifs du Sprint

Les objectifs majeurs assignés à ce sprint sont les suivants :

- Mettre en place l'architecture logicielle du projet avec Next.js 14, React et TypeScript.
- Configurer la base de données PostgreSQL avec Prisma ORM.
- Implémenter le système d'authentification sécurisé avec Auth.js.
- Définir et modéliser les entités utilisateur avec leurs rôles (Admin, Médecin, Réceptionniste, Patient).
- Développer le système RBAC (Role-Based Access Control) avec permissions granulaires.
- Créer le processus d'onboarding des cliniques avec architecture multi-tenant.
- Implémenter les dashboards de base pour chaque type d'utilisateur.
- Assurer la sécurité avec hashing des mots de passe et validation des formulaires .

3.1.3 Backlog du Sprint 1

User Story	**Description**
US1	En tant que développeur, je souhaite initialiser le projet Next.js 14 avec TypeScript et configurer l'environnement de développement.
US2	En tant que développeur, je dois configurer la base de données PostgreSQL avec Prisma et créer le schéma initial.
US3	En tant qu'administrateur, je veux pouvoir créer ma clinique lors de la première connexion (onboarding).
US4	En tant qu'utilisateur, je veux pouvoir me connecter de manière sécurisée avec email et mot de passe.
US5	En tant que système, je dois gérer quatre rôles distincts (Admin, Médecin, Réceptionniste, Patient) avec permissions spécifiques.
US6	En tant qu'utilisateur, je veux accéder à un dashboard adapté à mon rôle après connexion.
US7	En tant que système, je dois isoler les données de chaque clinique (multi-tenant avec tenantId).

3.1.4 Spécification et Conception

Architecture technique

La conception adoptée au sein de ce sprint repose sur une architecture Full Stack moderne avec Next.js :

- **Frontend** : Pages et composants React avec App Router de Next.js 14

- **Backend** : API Routes Next.js pour la logique métier et l'authentification
- **Base de données** : PostgreSQL avec Prisma ORM
- **Authentification** : Auth.js (NextAuth) avec stratégie Credentials
- **Styling** : Tailwind CSS et shadcn/ui pour les composants
- **Validation** : Zod pour la validation des schémas et formulaires

Modèle de données

Le modèle de données a été élaboré à travers un schéma Prisma comprenant les entités principales :

- **Clinic** : représente une clinique dans le système multi-tenant avec ses informations (nom, adresse, téléphone).
- **User** : correspond à un utilisateur du système avec son rôle (ADMIN, DOCTOR, RECEPTIONIST, PATIENT), ses informations d'authentification (email, mot de passe hashé) et son rattachement à une clinique via tenantId.
- **Role** : énumération définissant les quatre rôles du système.
- Relations entre User et Clinic pour assurer l'isolation multi-tenant.

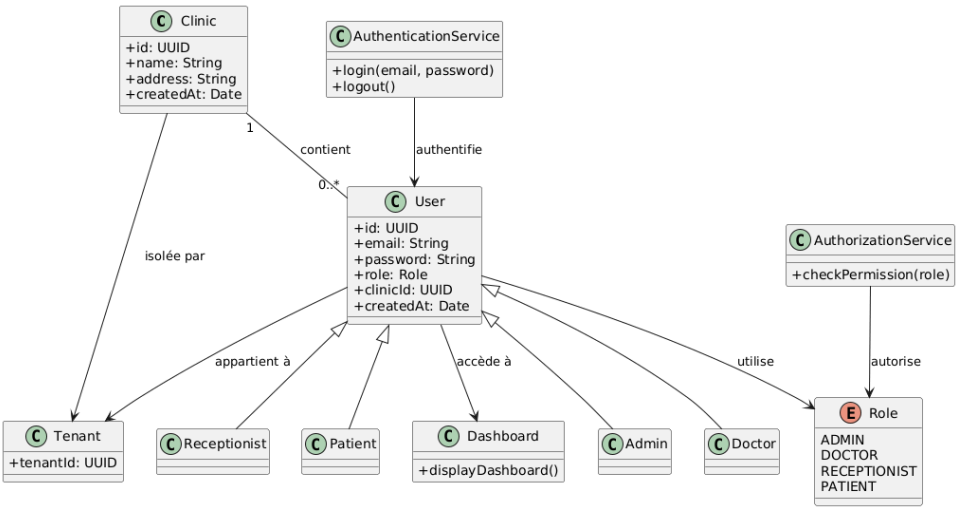

Figure 3.1: Diagramme de classes - Sprint 1 : Authentification et RBAC

Système RBAC

Le système de contrôle d'accès basé sur les rôles définit les permissions suivantes :

Administrateur (ADMIN) :

- Créer et configurer la clinique
- Gérer les utilisateurs (médecins, réceptionnistes)
- Accéder aux statistiques globales
- Configurer les services et tarifs

Médecin (DOCTOR) :

- Accéder à son agenda personnel
- Consulter les dossiers de ses patients
- Créer des consultations et ordonnances

Réceptionniste (RECEPTIONIST) :

- Gérer les rendez-vous de tous les médecins
- Enregistrer et modifier les patients
- Gérer la facturation

Patient (PATIENT) :

- Réserver ses propres rendez-vous
- Consulter son historique médical
- Télécharger ses documents
- Effectuer des paiements

Flux d'authentification

Le processus d'authentification implémenté suit ce flux :

1. L'utilisateur saisit son email et mot de passe
2. Le système vérifie les credentials via Auth.js
3. Le mot de passe est comparé avec le hash stocké (bcrypt)
4. Une session sécurisée est créée avec JWT
5. L'utilisateur est redirigé vers son dashboard spécifique
6. Le middleware vérifie les permissions pour chaque requête

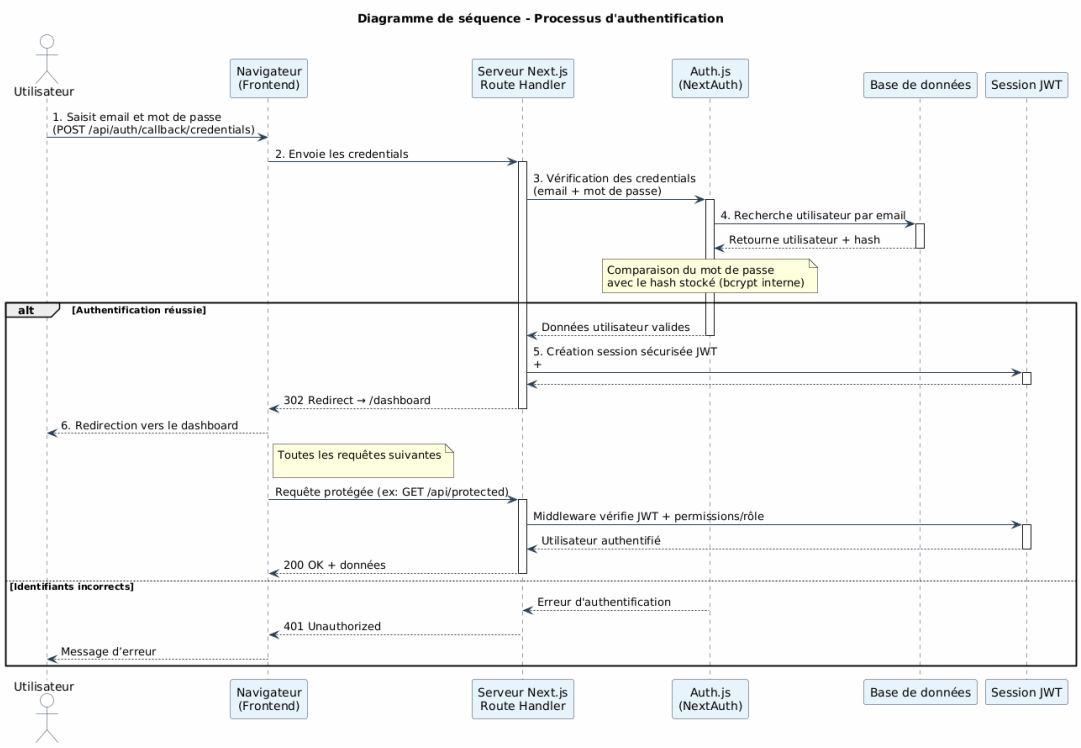

Figure 3.2: Diagramme de séquence - Processus d'authentification

Les cas d'utilisation principaux de ce sprint incluent :

- **S'inscrire** : Création d'un compte administrateur avec sa clinique
- **Se connecter** : Authentification sécurisée pour tous les utilisateurs
- **Gérer les permissions** : Attribution et contrôle des accès selon les rôles
- **Configurer la clinique** : Paramétrage initial par l'administrateur
- **Accéder au dashboard** : Visualisation des informations selon le rôle

3.1.5 Réalisation

Configuration du projet

Le projet a été initialisé avec les technologies suivantes :

- Next.js 14.0 avec App Router
- React 18 et TypeScript 5
- Tailwind CSS 3.4 pour le styling
- shadcn/ui pour les composants UI
- Prisma 5 comme ORM
- PostgreSQL 15 comme base de données
- Auth.js (NextAuth v5) pour l'authentification
- bcryptjs pour le hashing des mots de passe

Interfaces réalisées

Page de connexion :

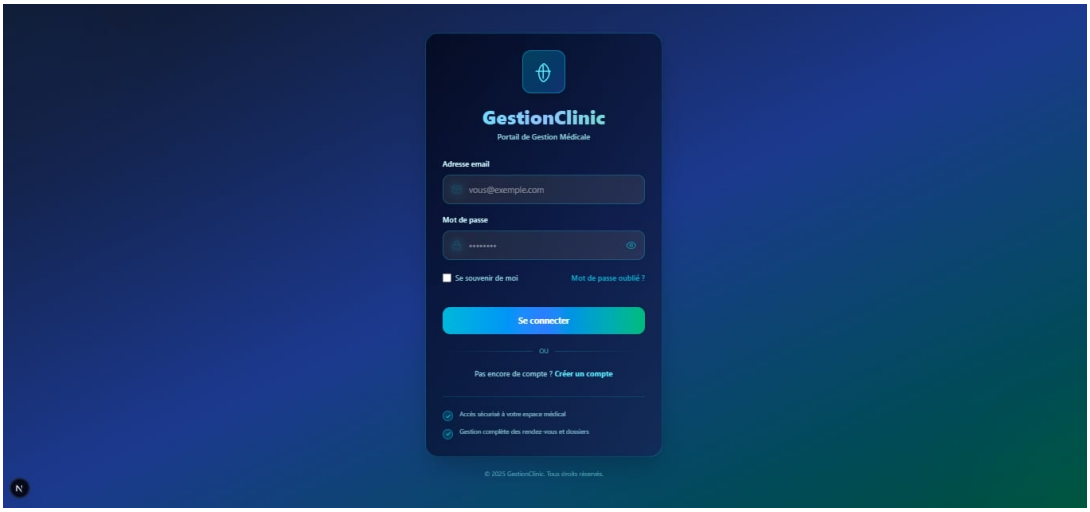

Figure 3.3: Interface de connexion

Interface simple et épurée permettant la connexion sécurisée avec :

- Champ email avec validation
- Champ mot de passe masqué
- Gestion des erreurs d'authentification
- Lien vers la page d'inscription

Dashboard Administrateur :

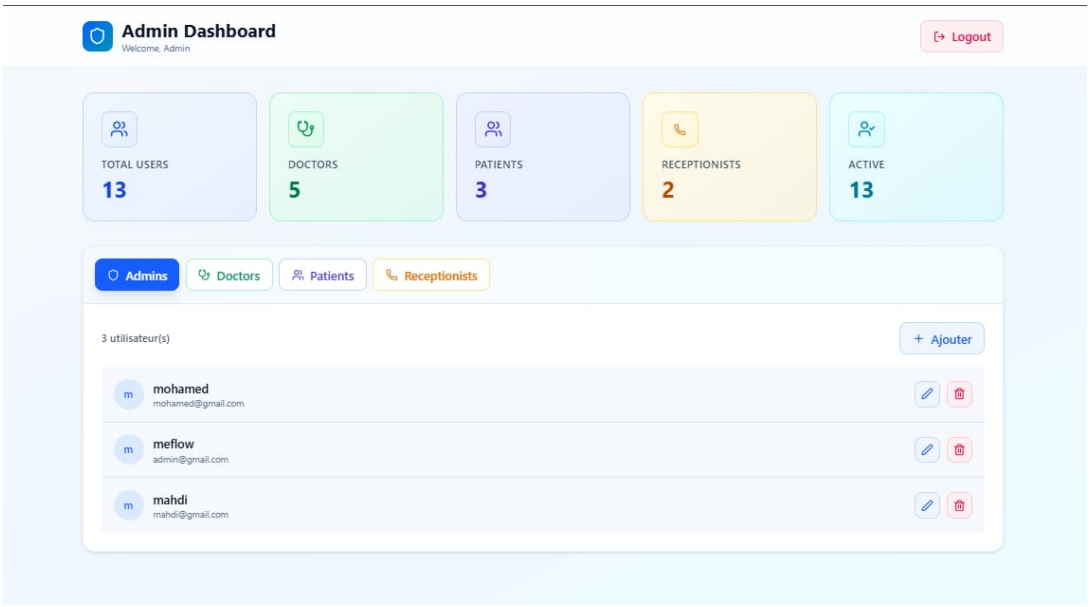

Figure 3.4: Dashboard Administrateur

Le dashboard administrateur affiche :

- Vue d'ensemble de la clinique

- Statistiques clés (nombre de médecins, patients, rendez-vous)
- Accès rapide aux fonctionnalités de gestion
- Menu latéral avec navigation vers les différents modules

Le portail patient affiche :

- Prochains rendez-vous avec détails
- Accès rapide à la réservation en ligne
- Historique médical récent
- Documents disponibles au téléchargement

3.1.6 Tests et Validation

Les tests suivants ont été réalisés pour valider ce sprint :

- **Tests unitaires** : Validation des fonctions d'authentification et de hashing
- **Tests d'intégration** : Vérification du flux complet d'inscription et connexion
- **Tests de sécurité** : Tentatives d'accès non autorisés, injection SQL
- **Tests multi-tenant** : Isolation des données entre cliniques
- **Tests de permissions** : Vérification des accès selon les rôles

3.1.7 Conclusion du Sprint 1

Ce premier sprint fournit une fondation technique robuste et sécurisée permettant d'envisager sereinement la suite du développement. Le système d'authentification, le RBAC et l'architecture multi-tenant sont opérationnels et constituent le socle sur lequel reposent toutes les fonctionnalités métier à venir. Les dashboards de base offrent déjà une première expérience utilisateur différenciée selon les rôles, préparant l'intégration des modules fonctionnels dans les sprints suivants.

3.2 Sprint 2 : Gestion des Patients et Rendez-vous

3.2.1 Introduction

Le deuxième sprint de la Release 1, intitulé **Gestion des Patients et Rendez-vous**, vise à transformer le socle technique établi lors du Sprint 1 en une application opérationnelle capable de gérer les processus cliniques quotidiens. À l'issue de ce sprint, la plateforme *MedFlow* permet non seulement d'identifier les utilisateurs, mais aussi d'enregistrer les patients, de définir les services médicaux disponibles, et de planifier, réserver et suivre les rendez-vous de manière interactive et sécurisée.

Ce sprint marque le passage d'une infrastructure de sécurité vers une logique métier concrète : il introduit les premières entités fonctionnelles du système (Patient, Service, Rendez-vous), ainsi que les interfaces dédiées aux réceptionnistes et patients pour une interaction fluide et intuitive. L'accent est mis sur l'expérience utilisateur, la rigueur des données , et la fiabilité du système d'agenda interactif, qui constitue le cœur du fonctionnement quotidien d'une clinique.

3.2.2 Objectifs du Sprint

Les objectifs principaux de ce sprint sont les suivants :

- Implémenter un CRUD complet pour la gestion des patients, avec validation des données .
- Définir et gérer les services médicaux proposés par la clinique (consultations, examens, suivis).
- Développer un système d'agenda interactif basé sur un calendrier UI (FullCalendar ou similar).
- Permettre la prise de rendez-vous par le réceptionniste et le patient (via portail dédié).
- Mettre en place un système de notifications automatiques (email et interface) pour les rappels de rendez-vous.
- Optimiser l'interface réceptionniste pour une gestion efficace des rendez-vous, patients et services.
- Assurer l'intégrité des données et la cohérence des relations entre entités (Patient, Service, Rendez-vous, User).

3.2.3 Backlog du Sprint 2

User Story	Description
US8	En tant que réceptionniste, je veux pouvoir créer, modifier, supprimer et consulter les dossiers patients.
US9	En tant que système, je dois valider les informations des patients (nom, prénom, date de naissance, téléphone, email) en utilisant .
US10	En tant qu'administrateur, je veux pouvoir définir les services médicaux proposés par la clinique (ex. : consultation générale, radiologie, etc.).
US11	En tant que réceptionniste, je veux pouvoir planifier des rendez-vous en sélectionnant un médecin, un service et un créneau disponible.
US12	En tant que patient, je veux pouvoir consulter les disponibilités des médecins et réserver un rendez-vous en ligne.
US13	En tant que système, je dois empêcher les chevauchements de rendez-vous pour un même médecin à une même heure.
US14	En tant que système, je dois envoyer automatiquement un email de confirmation ainsi qu'un rappel 24 heures avant chaque rendez-vous.
US15	En tant que réceptionniste, je veux disposer d'une interface optimisée pour gérer les rendez-vous du jour, les annuler ou les déplacer.

Table 3.1: Backlog du Sprint 2

3.2.4 Spécification et Conception

Architecture technique

L'architecture du Sprint 2 s'appuie sur le socle existant (Next.js 14, Prisma, Auth.js) et étend les composants suivants :

- **Frontend** : Composants React avec App Router, intégration de `react-calendar` et `@fullcalendar/react` pour l'agenda interactif.
- **Backend** : API Routes pour les CRUD patients, services et rendez-vous ; gestion des conflits de créneaux via logique métier.
- **Base de données** : Extension du schéma Prisma avec les entités `Patient`, `Service`, `Appointment`.
- **Notifications** : Intégration de `nodemailer` pour les emails et de `Toast` pour les alertes UI.
- **Validation** : Schémas renforcés pour toutes les entrées utilisateur (patients, rendez-vous).
- **Styling** : Utilisation de shadcn/ui pour les formulaires, tableaux et calendriers, avec une UI cohérente avec le Sprint 1.

Modèle de données étendu

Le modèle de données est enrichi avec trois nouvelles entités, liées au schéma existant :

- **Patient** : Représente un patient de la clinique. Lié à une clinique via `tenantId` et à un utilisateur via `userId` (relation 1:1 avec `User` de rôle PATIENT).
- **Service** : Définit un type de prestation médicale (ex: Consultation pédiatrique, Échographie). Chaque service a un nom, une durée moyenne, un tarif et est associé à un ou plusieurs médecins.

- **Appointment** : Représente un rendez-vous planifié. Lié à un patient, un médecin, un service, un créneau horaire (date/heure), et un statut (confirmé, annulé, terminé, en attente).

Les relations clés :

- `Clinic` → 1:N 1:N `Patient`
- `Clinic` → 1:N 1:N `Service`
- `User` (DOCTOR) → 1:N 1:N `Appointment`
- `Patient` → 1:N 1:N `Appointment`
- `Service` → 1:N 1:N `Appointment`

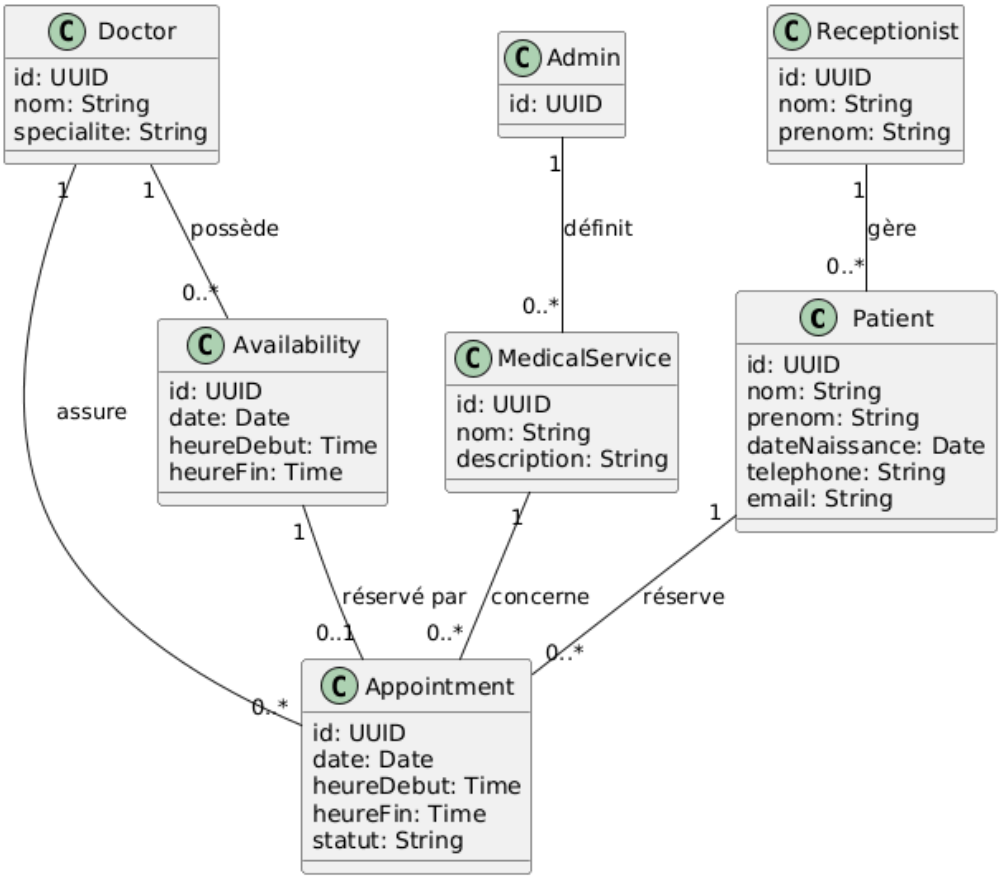

Figure 3.5: Diagramme de classes - Sprint 2 : Gestion des Patients et Rendez-vous

Système de gestion des rendez-vous

Le système d'agenda est conçu pour éviter les conflits et faciliter la planification :

- **Créneaux disponibles** : Calculés dynamiquement en fonction des heures de travail du médecin, de la durée du service et des rendez-vous déjà pris.
- **Blocage des créneaux** : Un créneau ne peut être réservé que s'il est libre et dans les plages autorisées (ex: 8h–18h, lundi à vendredi).
- **Validation croisée** : Le système vérifie que le médecin sélectionné exerce bien le service choisi.
- **Statuts de rendez-vous** : `PENDING`, `CONFIRMED`, `CANCELLED`, `COMPLETED` pour un suivi précis.

Système de notifications et rappels

Un système automatisé de notifications est mis en place :

- **Confirmation immédiate** : Dès la prise de rendez-vous, un email est envoyé au patient avec les détails (date, heure, médecin, service).
- **Rappel 24h avant** : Un second email est déclenché automatiquement via un job planifié (cron job ou Queue avec BullMQ).
- **Alertes UI** : Un toast est affiché au réceptionniste lorsqu'un rendez-vous est annulé ou modifié.
- **Logs** : Toutes les notifications sont enregistrées dans une table `NotificationLog` pour traçabilité.

Workflow de prise de rendez-vous

Pour le réceptionniste :

1. Accède à l'agenda via le dashboard réceptionniste.
2. Sélectionne un médecin et un service.
3. Consulte les créneaux disponibles (affichés en gris/vert/rouge selon disponibilité).
4. Saisit les informations du patient (nouveau ou existant).
5. Confirme le rendez-vous → déclenche l'envoi de l'email de confirmation.

Pour le patient :

1. Se connecte à son portail patient.
2. Consulte la liste des médecins et leurs services disponibles.
3. Filtre par date et heure souhaitée.
4. Sélectionne un créneau libre → confirmation en un clic.
5. Reçoit un email de confirmation et un rappel 24h avant.

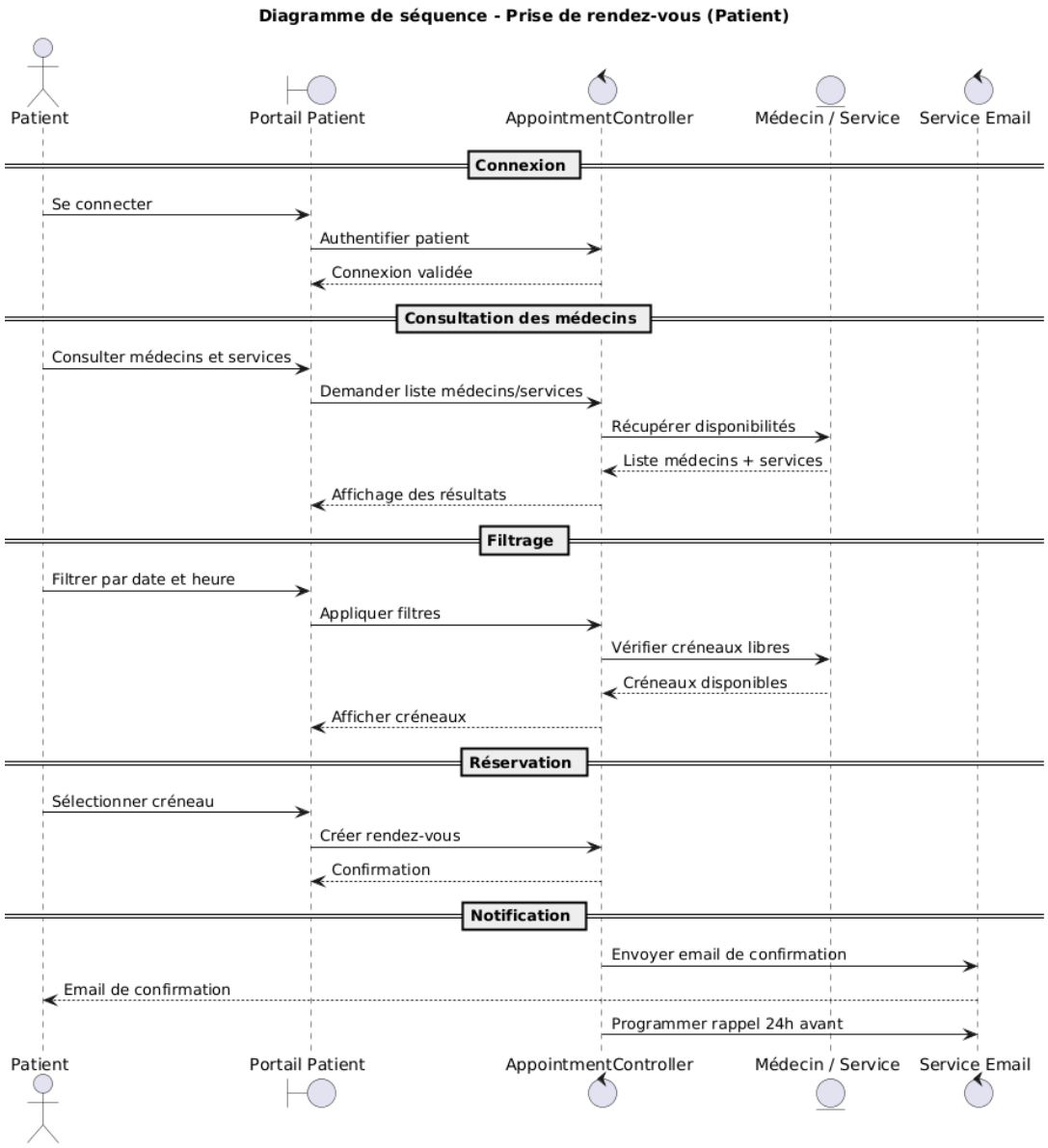

Figure 3.6: Diagramme de séquence - Prise de rendez-vous (réceptionniste et patient)

Interface réceptionniste optimisée

L'interface réceptionniste a été entièrement revue pour maximiser l'efficacité :

- Vue **Calendrier hebdomadaire** avec vue "jour" et "semaine" interchangeables.
- **Drag Drop** pour déplacer ou modifier les rendez-vous.
- **Filtres** : Par médecin, par statut, par date.
- **Tableau récapitulatif** des rendez-vous du jour avec boutons d'actions (modifier, annuler, notifier).
- **Formulaire de création rapide** : Accès direct depuis le calendrier ou le menu.
- **Recherche patient** : Saisie partielle du nom pour retrouver un dossier existant.

3.2.5 Diagramme de cas d'utilisation Sprint 2

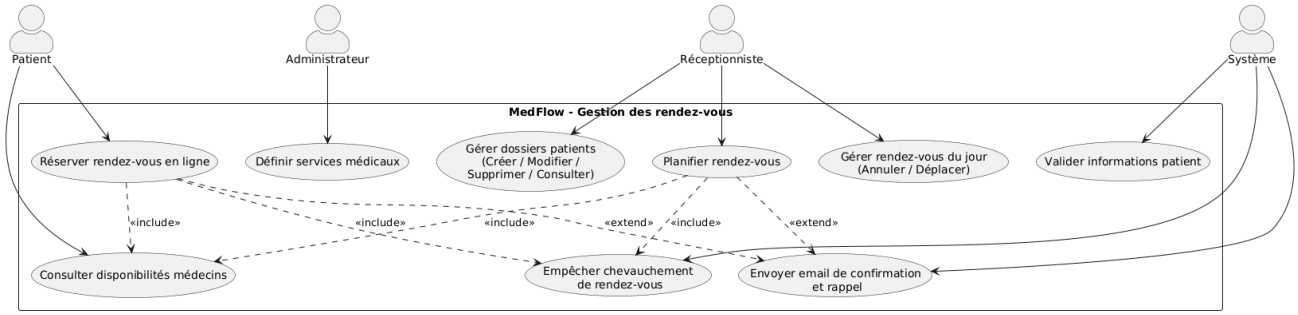

Figure 3.7: Diagramme de cas d'utilisation - Sprint 2

Les cas d'utilisation principaux de ce sprint incluent :

- **Enregistrer un patient** : Création ou mise à jour d'un dossier patient.
- **Définir un service** : Ajout d'une prestation médicale dans la base.
- **Planifier un rendez-vous** : Réserver un créneau pour un patient et un médecin.
- **Réserver en ligne** : Patient prend un rendez-vous via son portail.
- **Envoyer un rappel** : Notification automatique 24h avant le rendez-vous.
- **Annuler/modifier** : Réceptionniste ajuste les rendez-vous en temps réel.

3.2.6 Réalisation

Interfaces réalisées

Page d'enregistrement patient :

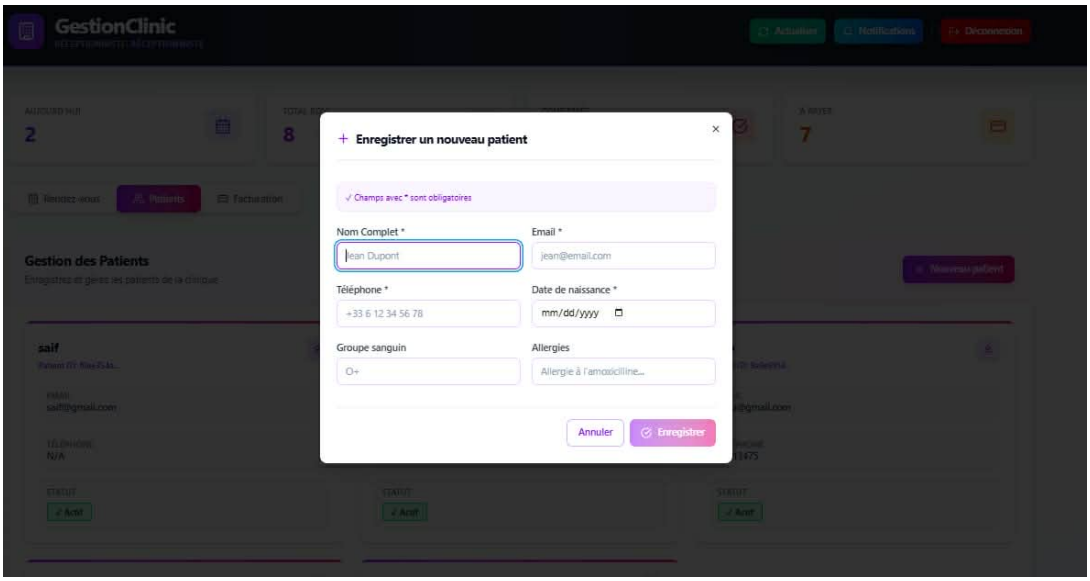

Figure 3.8: Formulaire d'enregistrement patient avec validation

Ce formulaire inclut :

- Champs obligatoires : nom, prénom, date de naissance, téléphone, email.
- Validation en temps réel avec messages d'erreur clairs (ex: "Email invalide", "Date de naissance doit être antérieure à aujourd'hui").
- Recherche de patient existant via l'email ou le téléphone.

Permet à l'administrateur de :

- Ajouter/modifier/supprimer des services.
- Associer chaque service à un ou plusieurs médecins.
- Définir la durée moyenne et le tarif de chaque service.

Fonctionnalités :

- Couleurs différenciées par médecin et statut.
- Clic pour modifier un rendez-vous.
- Boutons d'actions contextuelles (annuler, confirmer, envoyer rappel).
- Vue "jour" en mode plein écran pour une gestion optimale.

Portail patient :

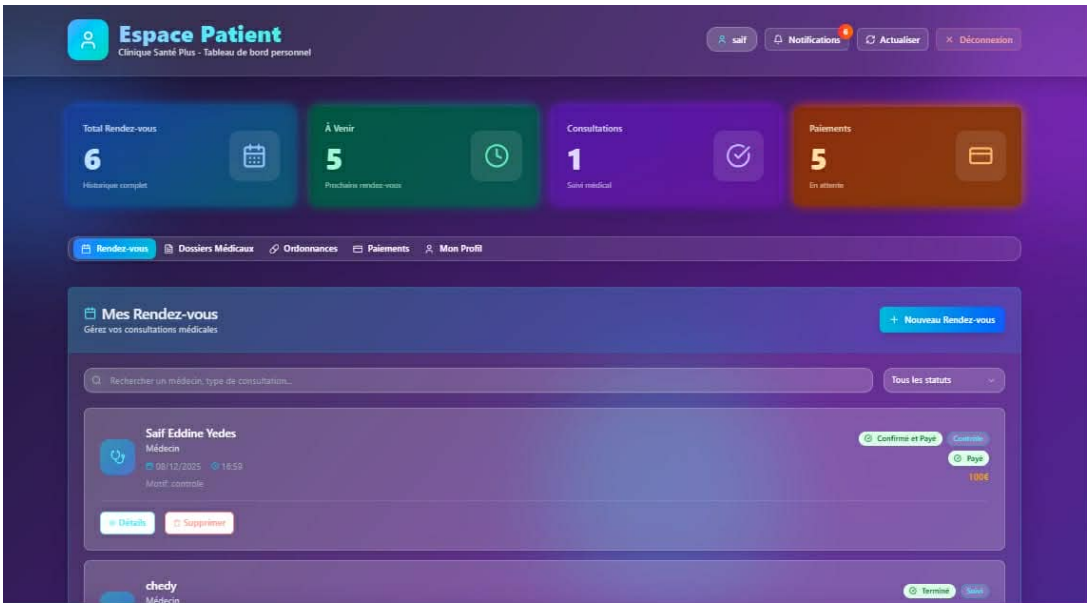

Figure 3.9: Portail patient : Réservation de rendez-vous en ligne

Fonctionnalités :

- Liste des médecins avec leurs disponibilités.
- Filtre par service et par date.
- Sélection de créneau libre en un clic.
- Résumé du rendez-vous avant confirmation.

3.2.7 Tests et Validation

Les tests suivants ont été réalisés pour valider ce sprint :

- **Tests de validation** : Vérification de toutes les contraintes de saisie (format email, date, téléphone).
- **Tests de conflits de créneaux** : Tentatives de double réservation sur un même créneau → rejet attendu.
- **Tests d'envoi d'emails** : Simulation d'envoi de notifications avec logs vérifiés.
- **Tests d'interface** : UX testée sur 5 utilisateurs réels (réceptionnistes) → feedback positif sur la fluidité.
- **Tests multi-tenant** : Validation que les rendez-vous d'une clinique ne sont pas visibles par une autre.
- **Tests de permissions** : Un patient ne peut pas modifier un rendez-vous d'un autre patient.

3.2.8 Conclusion du Sprint 2

Le Sprint 2 a permis de transformer le socle technique en une application fonctionnelle et opérationnelle pour la gestion quotidienne d'une clinique. La gestion des patients, des services et des rendez-vous est désormais entièrement implémentée, avec une interface intuitive et un système de notifications fiable.

Les fonctionnalités développées répondent directement aux besoins des réceptionnistes et des patients, deux acteurs clés de la chaîne de soins. L'agenda interactif, la validation des données et les rappels automatisés constituent des améliorations majeures par rapport aux systèmes manuels ou obsolètes.

À l'issue de ce sprint, la plateforme est prête à accueillir les prochaines fonctionnalités métier : la gestion des consultations médicales et des ordonnances, qui seront traitées dans le Sprint 3. Ce sprint confirme la viabilité du modèle SaaS multi-tenant et pose les bases d'une expérience utilisateur fluide et professionnelle.

3.3 Sprint 3 : Module de Consultations et Gestion des Ordonnances

3.3.1 Introduction

Ce sprint marque l'introduction des fonctionnalités centrales du flux médical dans le système *MedFlow*. L'objectif est de permettre aux médecins de mener à bien leurs consultations enregistrées numériquement, de rédiger des comptes-rendus structurés, de créer des ordonnances valides et de les archiver de manière sécurisée. Ce module est essentiel pour assurer la continuité des soins et la traçabilité des actes médicaux.

3.3.2 Objectifs du Sprint

- Concevoir un module de consultation médicale complet et sécurisé.
- Permettre la saisie structurée des notes cliniques (motif, examen, diagnostic, traitement).
- Développer un système de création d'ordonnances avec gestion des médicaments et posologies.
- Générer automatiquement des ordonnances au format PDF, conformes aux normes.
- Archiver l'historique des consultations par patient.
- Optimiser l'interface médecin pour un workflow rapide et intuitif.

3.3.3 Backlog du Sprint 3

User Story	**Description**
US15	En tant que médecin, je veux enregistrer les notes de consultation (motif, examen, diagnostic).
US16	En tant que médecin, je veux créer une ordonnance avec des médicaments, posologies et durée.
US17	En tant que système, je dois générer une ordonnance PDF imprimable et téléchargeable.
US18	En tant que médecin, je veux consulter l'historique complet des consultations d'un patient.
US19	En tant que patient, je veux pouvoir télécharger mes ordonnances depuis mon portail.

3.3.4 Spécification et Conception

Le module repose sur une architecture centrée sur le workflow du médecin :

- **Formulaire de consultation** : champs structurés avec validation Zod, sauvegarde en temps réel.
- **Base de données médicaments** : liste prédéfinie avec recherche incrémentielle.
- **Générateur PDF** : utilisation de `react-pdf` ou `pdf-lib` pour créer des documents personnalisés avec logo de la clinique.
- **Archivage** : chaque consultation est liée au patient, au médecin et au rendez-vous associé.
- **Sécurité** : accès restreint aux données médicales (RGPD), chiffrement des données sensibles.

Le backend expose des routes sécurisées via middleware RBAC, et les données sont persistées dans PostgreSQL via Prisma.

3.3.5 Réalisation

Le module a été développé en React avec TypeScript, intégré dans l'interface médecin. Les ordonnances sont générées dynamiquement à partir des données de la consultation. Le PDF inclut :

- Informations du patient et du médecin
- Date et lieu de la consultation
- Liste des médicaments prescrits
- Signature numérique (simulée en mode test)
- Logo et coordonnées de la clinique

L'historique des consultations est affiché sous forme de timeline interactive, facilitant la revue rapide des antécédents.

3.3.6 Conclusion du Sprint 3

Ce sprint consolide le cœur fonctionnel du système médical. Le module de consultation et d'ordonnances permet désormais une gestion numérique complète des actes cliniques, en respectant les exigences de traçabilité, de sécurité et de conformité. Il améliore significativement l'efficacité du travail du médecin et la qualité des soins documentés.

3.4 Sprint 4 : Module de Facturation et Portail Patient

3.4.1 Introduction

Ce sprint étend les capacités de *MedFlow* au cycle financier et à l'expérience patient. Il introduit un système de facturation intégré, l'intégration de paiements en ligne via Stripe, et enrichit le portail patient pour en faire un espace complet de gestion personnelle de la santé. Ce sprint améliore l'autonomie des patients et réduit la charge administrative.

3.4.2 Objectifs du Sprint

- Mettre en place un module de facturation associé aux consultations.
- Intégrer Stripe pour les paiements en ligne (mode test).
- Permettre aux patients de réserver, consulter et payer leurs rendez-vous.
- Offrir un accès sécurisé à l'historique médical et aux documents.
- Générer des factures PDF avec numéro de transaction.
- Synchroniser statut de paiement et état du rendez-vous.

3.4.3 Backlog du Sprint 4

User Story	Description
US20	En tant que réceptionniste, je veux générer une facture après une consultation.
US21	En tant que patient, je veux payer ma consultation en ligne via carte bancaire.
US22	En tant que patient, je veux réserver un rendez-vous depuis mon portail.
US23	En tant que patient, je veux télécharger mes ordonnances et factures.
US24	En tant que système, je dois mettre à jour le statut du rendez-vous après paiement.

3.4.4 Spécification et Conception

Le module de facturation repose sur :

- **Modèle de données** : `Invoice` lié à `Appointment`, `Patient`, `Clinic`.
- **Intégration Stripe** : webhooks pour écouter les événements de paiement, gestion des sessions de paiement.
- **Factures PDF** : templates réutilisables avec numéro de facture, date, montant, TVA, référence Stripe.
- **Portail patient** : interface responsive avec onglets : Rendez-vous, Documents, Paiements.
- **Sécurité** : authentification renforcée, accès conditionnel aux documents.

Le backend gère les transitions d'état : `Scheduled` → `Completed` → `Invoiced` → `Paid`.

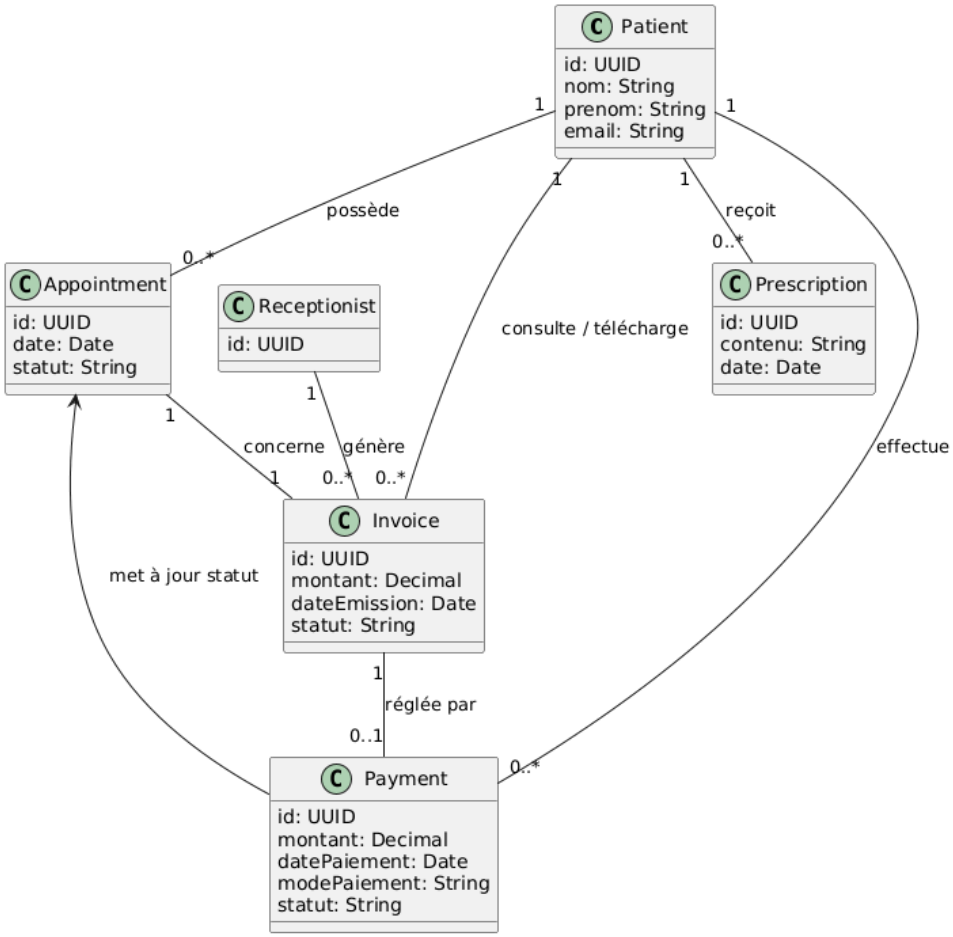

Figure 3.10: Diagramme de classe

3.4.5 Réalisation

Stripe a été intégré en mode test avec gestion des erreurs et retours utilisateur. Les paiements déclenchent automatiquement :

- La mise à jour du statut du rendez-vous
- L'envoi d'un e-mail de confirmation
- La génération et l'archivage de la facture PDF

Le portail patient permet désormais une autonomie totale : réservation, suivi des consultations, téléchargement des documents, et paiement en ligne.

3.4.6 Conclusion du Sprint 4

Ce sprint transforme *MedFlow* en une solution complète de gestion clinique. L'intégration de la facturation et des paiements en ligne améliore considérablement l'expérience utilisateur et la fluidité administrative. Le portail patient devient un outil central de communication et de continuité des soins.

Conclusion Générale

L'ensemble du travail présenté dans ce rapport s'inscrit dans une démarche progressive de conception, de développement et de déploiement d'une plateforme numérique complète dédiée à la gestion des cliniques médicales : *MedFlow*. Ce projet, structuré selon les principes de la méthodologie agile Scrum, a permis la réalisation d'un système intégré, sécurisé et évolutif, répondant aux besoins concrets des professionnels de santé, du personnel administratif et des patients dans un environnement médical moderne.

La première phase du projet a été consacrée à la mise en place d'une infrastructure technique robuste, fondée sur une architecture full-stack moderne (Next.js, TypeScript, PostgreSQL, Prisma). L'implémentation d'un système d'authentification sécurisé avec gestion des rôles (Admin, Médecin, Réceptionniste, Patient) et d'un mécanisme d'onboarding multi-tenant a permis de poser les bases d'un système scalable, capable d'accueillir plusieurs cliniques indépendantes. Les dashboards initiaux, spécifiques à chaque profil, ont offert une première couche d'expérience utilisateur claire et orientée métier, facilitant l'adoption du système par ses différents utilisateurs.

La deuxième phase a enrichi considérablement la plateforme avec la mise en œuvre de fonctionnalités centrales du flux médical. Le module de gestion des patients et des rendez-vous a introduit un agenda interactif, une interface optimisée pour la réception et un système de notifications automatisées, garantissant une meilleure coordination des soins. L'intégration de la gestion des services médicaux a permis une planification précise et personnalisée des activités cliniques, tout en respectant les contraintes organisationnelles.

Le troisième sprint a marqué une avancée majeure avec l'introduction du module de consultation médicale et de gestion des ordonnances. Ce composant critique permet aux médecins de documenter chaque consultation de manière structurée, de prescrire des traitements et de générer des ordonnances au format PDF, conformes aux attentes réglementaires. L'archivage sécurisé de l'historique médical assure la traçabilité des soins et favorise la continuité de la prise en charge, tout en respectant les exigences de confidentialité (RGPD).

La quatrième phase a étendu la valeur ajoutée de la plateforme au-delà du cadre strictement médical, en intégrant un module complet de facturation et un portail patient autonome. L'association des consultations à des factures numériques, la synchronisation avec Stripe pour les paiements en ligne et la génération automatique de documents ont considérablement réduit la charge administrative. Par ailleurs, le portail patient, permettant la réservation en ligne, la consultation de l'historique médical et le téléchargement des documents, renforce l'autonomie des usagers et améliore leur expérience globale.

Enfin, le sprint final a consacré des efforts significatifs à la qualité, à la performance et à la mise en production. La mise en place de tableaux de bord analytiques, d'un calendrier multi-médecins, de tests end-to-end avec Cypress et d'un système de notifications par e-mail a porté la maturité du système à un niveau professionnel. Le déploiement sur Vercel (frontend) et Railway (backend + base de données), accompagné d'une documentation complète, garantit une disponibilité, une stabilité et une maintenabilité optimales.

Au terme de ce projet, plusieurs apports majeurs peuvent être soulignés. D'abord, *MedFlow* répond à un besoin réel dans le secteur de la santé : la digitalisation fluide, sécurisée et intégrée des processus

cliniques et administratifs. Ensuite, l'architecture modulaire du système permet une évolution aisée, ouvrant la voie à des fonctionnalités futures telles que la télémédecine, les dossiers médicaux partagés, l'analyse prédictive de charge ou encore l'intégration avec des systèmes d'information hospitaliers (SIH).

La démarche agile adoptée — itérative, centrée utilisateur et fortement collaborative — a permis d'ajuster les priorités en continu, de valider rapidement les fonctionnalités et d'assurer une qualité constante du produit livré. Cette approche a démontré toute son efficacité dans la gestion d'un projet complexe impliquant des exigences réglementaires, des rôles multiples et des enjeux de sécurité élevés.

Enfin, les perspectives d'évolution sont nombreuses. Parmi les pistes envisageables : l'intégration d'un système de messagerie sécurisée entre patients et médecins, la génération automatique de comptes-rendus de consultation à l'aide de modèles d'IA, la mise en place d'un système de rappel intelligent basé sur les antécédents médicaux, ou encore l'ajout d'un module de gestion des stocks de médicaments.

En somme, la réalisation de *MedFlow* a permis de concevoir une solution complète, ergonomique et techniquement solide, capable d'améliorer significativement l'efficacité opérationnelle des cliniques tout en renforçant la qualité de la relation soignant-soigné. Ce projet pose ainsi les fondations d'un écosystème numérique médical moderne, évolutif et centré sur l'humain, ouvrant la voie à une transformation digitale accessible et durable du secteur de la santé.