Université Polytechnique Sousse

Département d'Informatique

Année Universitaire 2026–2027

Conception et Réalisation d'une Application Web de Gestion de Cabinet Médical

Projet Semestriel - Génie Logiciel

Réalisé par :

Intissar BENATIA

Yasmina JAMELI

5ème Année - INFO - Génie Logiciel

Encadré par :

Mr. Hayethem SAOUDI

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Contexte et Problématique

La digitalisation du secteur médical constitue un enjeu stratégique pour optimiser la gestion des cabinets médicaux. Les systèmes manuels actuels présentent des limitations en efficacité, traçabilité et accessibilité des données patients, nécessitant une solution moderne et automatisée.

1.2 Objectifs du Projet

Ce projet vise cinq objectifs principaux : automatiser la gestion des rendez-vous avec prévention des conflits, centraliser les dossiers médicaux de manière sécurisée, générer automatiquement les documents PDF, offrir des interfaces adaptées par rôle, et garantir la sécurité des données conformément aux standards médicaux.

1.3 Méthodologie

Le développement de ce projet suit une méthodologie classique , où les différentes phases sont réalisées de manière séquentielle et structurée. Cette approche permet une meilleure maîtrise du projet, une documentation complète et une validation progressive des résultats à chaque étape.

Chapitre 2

Analyse et Spécification des Besoins

2.1 Identification des Acteurs

Le système repose sur quatre acteurs : le **Patient** qui gère ses rendez-vous et consulte son dossier, le **Médecin** qui crée consultations et génère documents, la **Secrétaire** qui coordonne patients et paiements.

2.2 Besoins Fonctionnels

Le système couvre **23 besoins fonctionnels** répartis par acteur :

2.2.1 Pour les Patients

L'interface patient offre l'authentification sécurisée (email ou Google OAuth), la prise de rendez-vous avec disponibilités temps réel, la consultation du dossier médical complet, le téléchargement des ordonnances PDF et la gestion des données personnelles.

2.2.2 Pour les Médecins

L'espace médecin permet la visualisation de l'agenda complet, la création de consultations avec diagnostic détaillé, la génération automatique d'ordonnances et certificats PDF, et la consultation de l'historique patient avec graphiques d'évolution.

2.2.3 Pour les Secrétaires

Le poste secrétariat regroupe la gestion complète des patients (CRUD), l'administration des rendez-vous avec notifications, l'enregistrement des paiements multi-modes et la génération de reçus officiels.

2.3 Besoins Non-Fonctionnels

Propriété	**Exigence**
Performance	Temps de réponse $<$ 2s pour 95% des requêtes
Disponibilité	Uptime 99.5% avec backups quotidiens
Sécurité	JWT + HTTPS + Validation entrées + Hachage BCrypt
Ergonomie	Interface intuitive, responsive (mobile/tablette/desktop)
Scalabilité	Support 1000+ patients et 100+ RDV simultanés
Maintenabilité	Code modulaire, documenté, architecture 3 couches

Chapitre 3

Modélisation et Conception

3.1 Diagramme de Cas d'Utilisation

Le diagramme ci-dessous représente les interactions entre les 4 acteurs et les 23 cas d'utilisation du système.

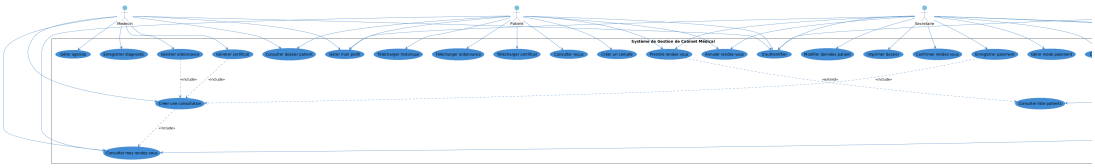

FIGURE 3.1 – Diagramme de cas d'utilisation global du système

Acteurs et Cas Principaux : Le **Patient** gère son parcours médical en autonomie : authentification, prise de RDV avec sélection médecin, consultation dossier avec allergies et antécédents, téléchargement des documents médicaux. Le **Médecin** dispose d'outils professionnels avancés : création consultations avec diagnostic détaillé, génération automatique ordonnances et certificats, historique patient avec graphiques. La **Secrétaire** coordonne l'administratif : création patients, gestion agenda avec planification RDV, enregistrement paiements multi-modes, impression dossiers.

3.2 Diagramme de Classes

Le modèle de données repose sur **9 entités principales** avec relations et héritage.

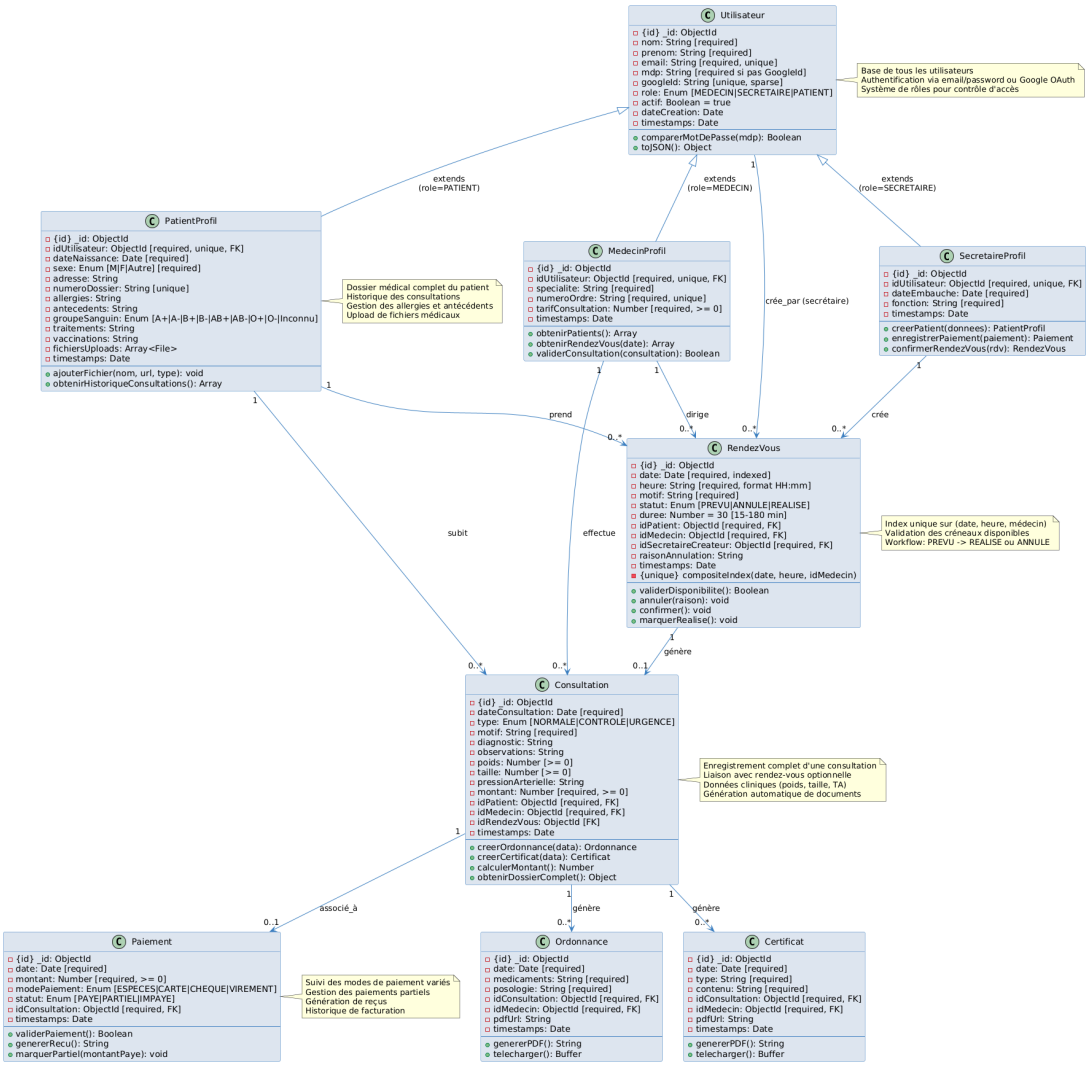

FIGURE 3.2 – Diagramme de classes - Architecture des données

Relations clés : L'architecture suit un modèle orienté objet avec héritage : la classe **Utilisateur** constitue l'entité mère dont héritent PatientProfil, MedecinProfil et SecretaireProfil. Les relations métier sont cohérentes : un PatientProfil possède plusieurs RendezVous (1..*), un MedecinProfil réalise plusieurs Consultations (1..*), et chaque Consultation génère une Ordonnance, un Certificat et un Paiement (1..1 ou 0..*).

3.3 Diagrammes de Séquence

3.3.1 Scénario 1 : Prise de Rendez-vous

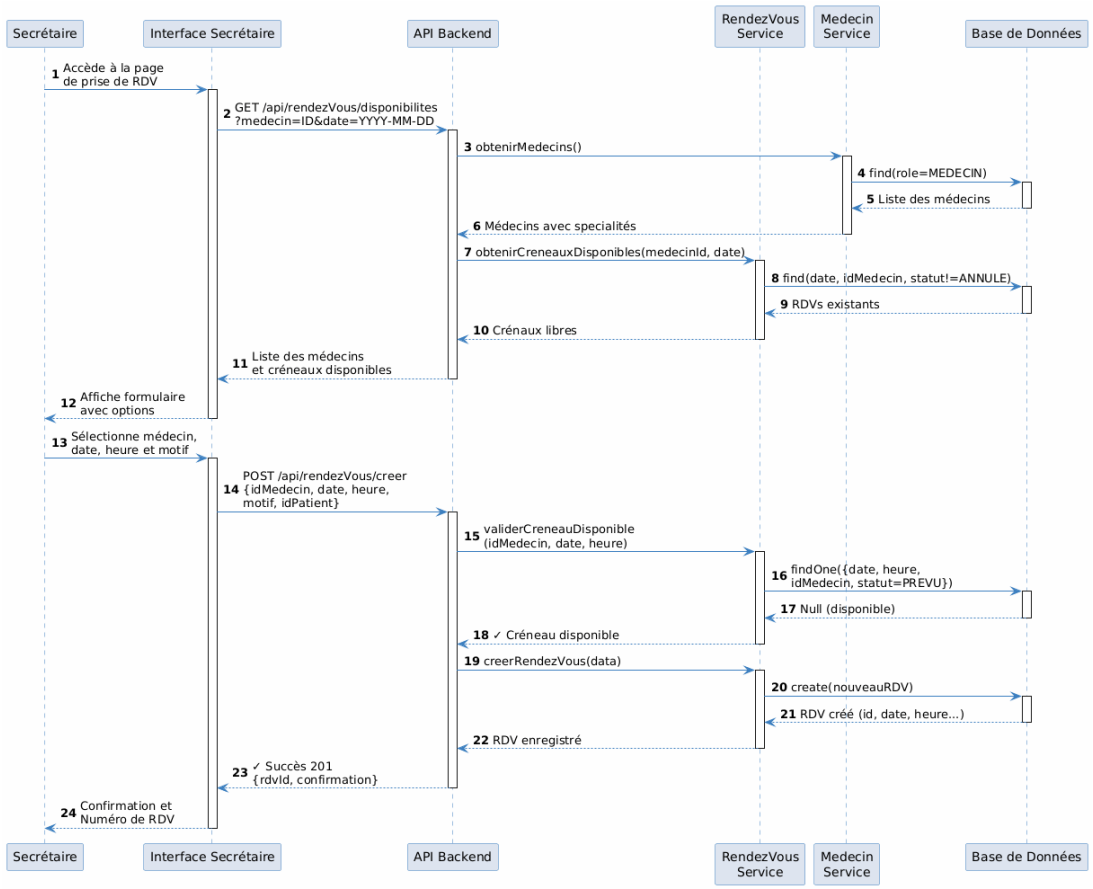

FIGURE 3.3 – Séquence : Processus complet de prise de rendez-vous

Flux principal : Le processus suit six étapes clés : la secrétaire accède l'interface de planification, le système charge médecins et créneaux disponibles, sélection des paramètres (patient, médecin, date, heure, motif), vérification automatique de disponibilité pour prévenir conflits, création RDV avec statut PREVU en base, et confirmation avec numéro unique et notification email au patient.

3.3.2 Scénario 2 : Création de Consultation

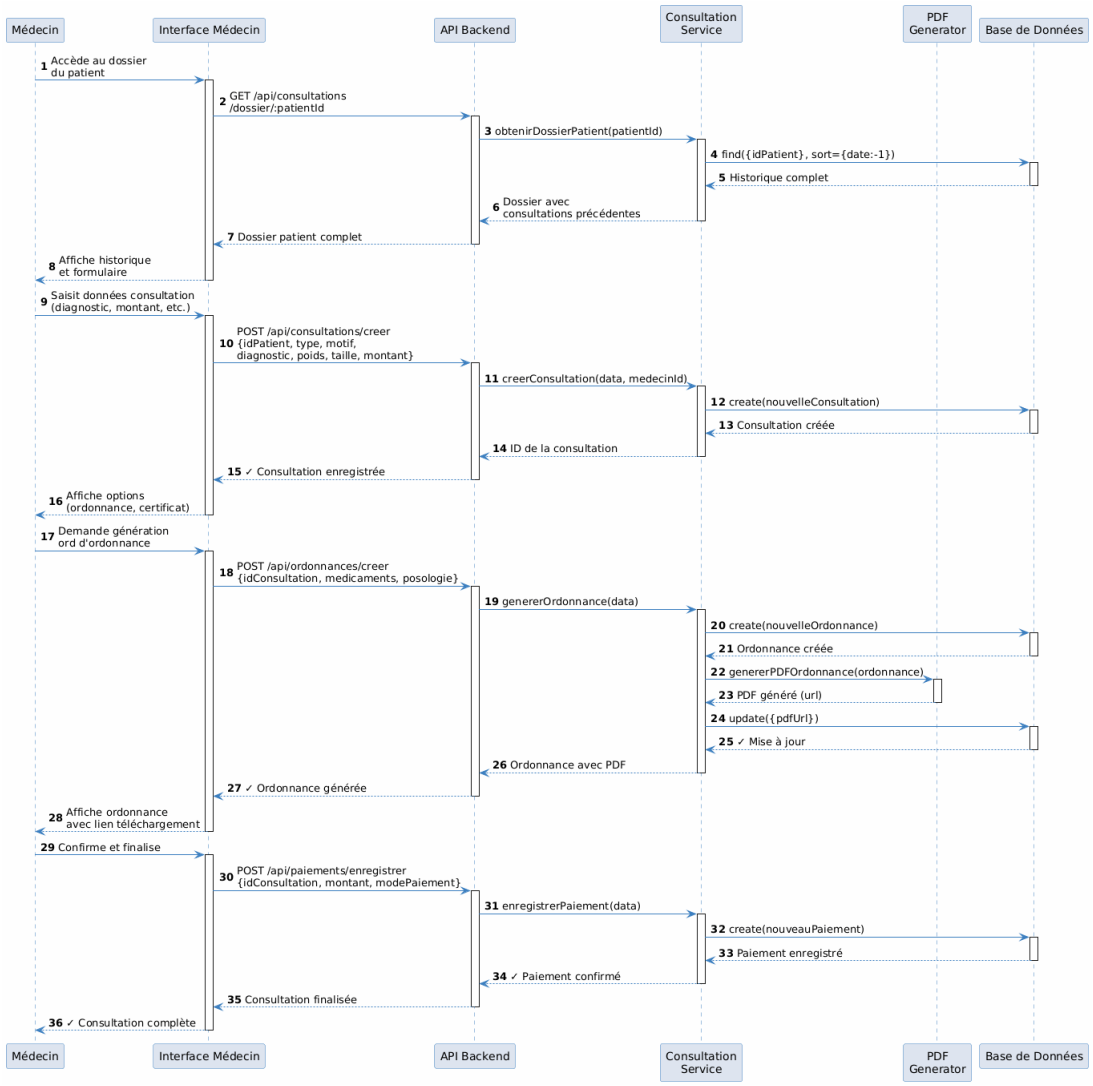

FIGURE 3.4 – Séquence : Création consultation et génération d'ordonnance

Flux principal : Le processus s'articule autour de six phases : accès au dossier patient complet avec historique médical, saisie détaillée de la consultation (diagnostic, observations, constantes vitales), création de l'enregistrement en base avec horodatage, génération automatique de l'ordonnance PDF avec PDFKit et mentions légales, et envoi notification email au patient avec lien de téléchargement sécurisé.

Chapitre 4

Architecture Logicielle et Physique

4.1 Architecture Logique

L'application adopte une **architecture 3 couches** pour séparer responsabilités :

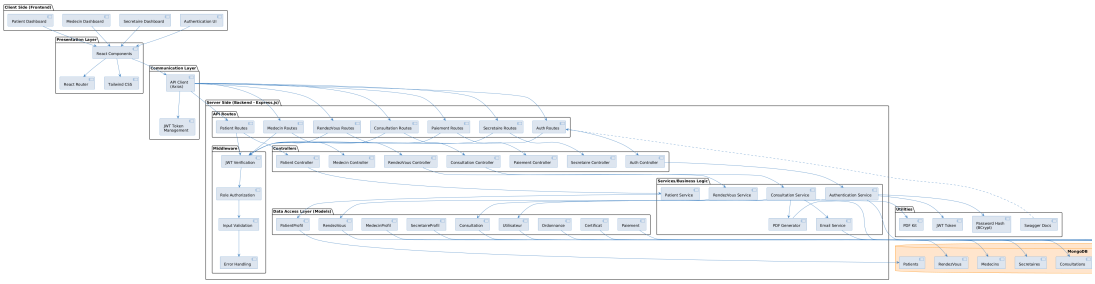

FIGURE 4.1 – Architecture logique du système - Vue 3 couches

Couche	Responsabilités
Présentation	Interface utilisateur (React 19 + Tailwind CSS), Routing (React Router), Appels API (Axios)
Métier	API REST (Express.js), Logique métier (Services), Authentification (JWT + Passport), Génération PDF (PDFKit)
Données	Base de données (MongoDB), ODM (Mongoose), Schémas et validations

4.2 Architecture Physique

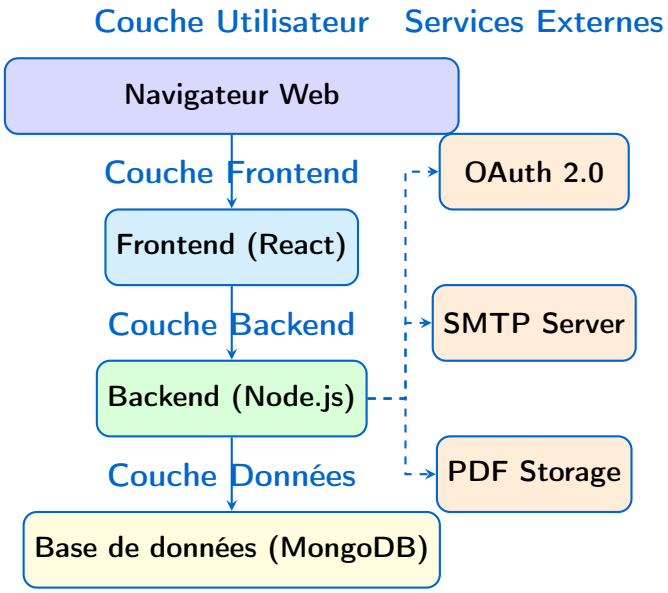

FIGURE 4.2 – Architecture physique de déploiement du système

4.2.1 Description de l'Architecture Physique

Couche Utilisateur (Navigateur Web) :

— Application React SPA chargée initialement

— Communication via HTTPS avec le frontend

— Stockage local des tokens JWT

Couche Frontend (Port 5173) :

— Serveur de développement Vite en local

— Build de production déployé sur serveur web

— React Router pour la navigation côté client

Couche Backend (Port 3000) :

— Serveur Node.js avec Express.js

— Middleware JWT pour l'authentification

— Routes API REST structurées par domaine

— Modèles Mongoose pour la validation

Couche Base de Données (Port 27017) :

— MongoDB NoSQL avec 9 collections

— Indexation optimisée pour les requêtes fréquentes

— Replica Set pour la haute disponibilité

Services Externes :

— Google OAuth 2.0 pour l'authentification sociale

— Serveur SMTP (Gmail) pour l'envoi d'emails

— Stockage local pour les PDF générés

Composants de déploiement : Le déploiement repose sur trois composants principaux interconnectés. Le **Frontend** consiste en un build Vite optimisé déployé sur serveur Nginx avec compression gzip, mise en cache des ressources statiques et configuration reverse proxy. Le **Backend** s'exécute sur Node.js géré par PM2 assurant haute disponibilité avec redémarrage automatique, load balancing multi-cœurs CPU et gestion centralisée des logs. La **Database** utilise MongoDB offrant scalabilité automatique et réplication pour haute disponibilité.

Chapitre 5

Réalisation et Implémentation

5.1 Stack Technologique

Technologie	Utilisation
React 19	Composants modulaires, hooks (useState, useEffect, useContext)
Node.js + Express	API REST, middleware, routing
MongoDB + Mongoose	Base NoSQL, schémas validés, indexation
JWT + BCrypt	Authentification stateless, hachage mots de passe
PDFKit	Génération ordonnances et certificats PDF
Tailwind CSS 4	Design responsive, classes utilitaires
Google OAuth 2.0	Social login

5.1.1 Logos des Technologies

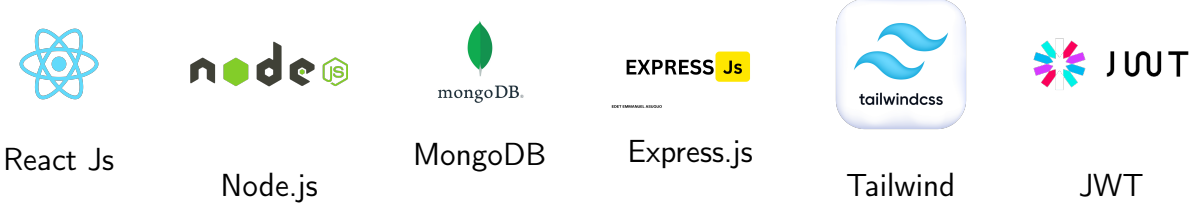

FIGURE 5.1 – Stack technologique complète du projet

5.2 Interfaces Utilisateur

5.2.1 Dashboard Patient

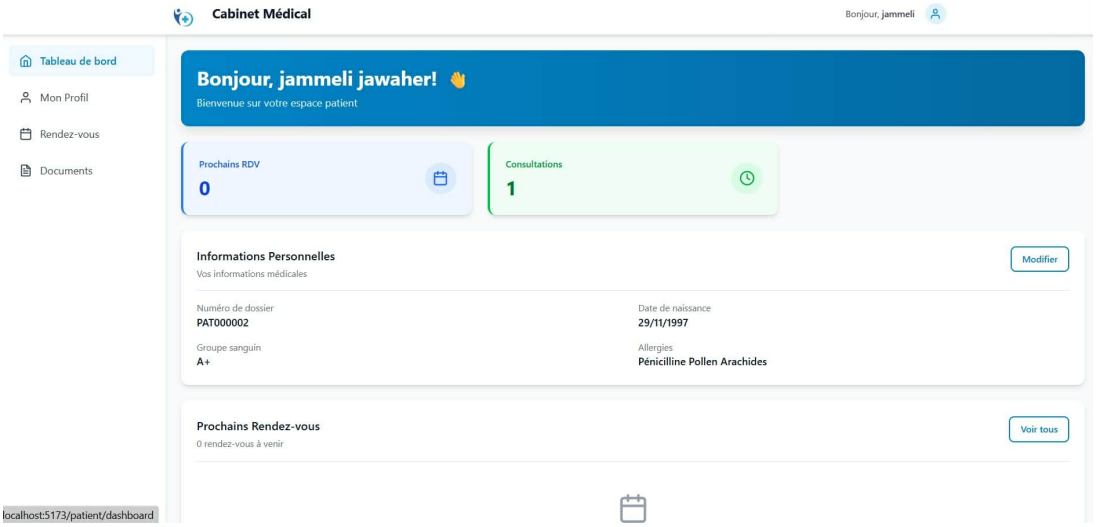

FIGURE 5.2 – Interface Dashboard Patient - Vue d'ensemble

Fonctionnalités clés : Le dashboard patient offre une vue personnalisée avec statistiques (RDV à venir, documents récents) et rappels automatiques. L'interface de prise de RDV intègre un calendrier interactif avec disponibilités temps réel pour sélection rapide du créneau. La section dossier médical centralise allergies, antécédents et traitements en cours avec suivi chronologique. Le module documents permet téléchargement sécurisé des ordonnances et certificats PDF avec recherche par date ou type.

5.2.2 Dashboard Médecin

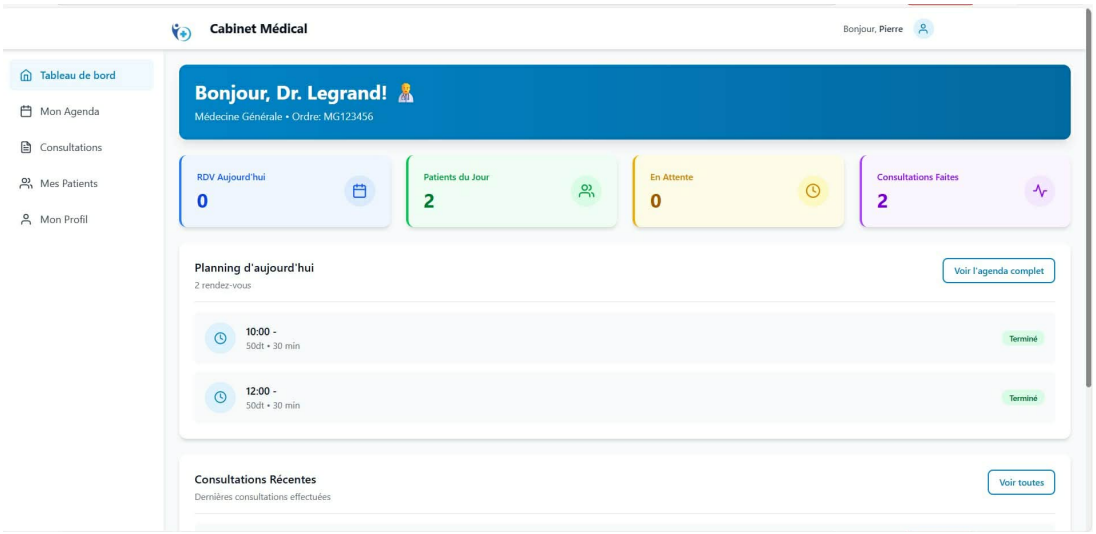

FIGURE 5.3 – Interface Dashboard Médecin - Agenda

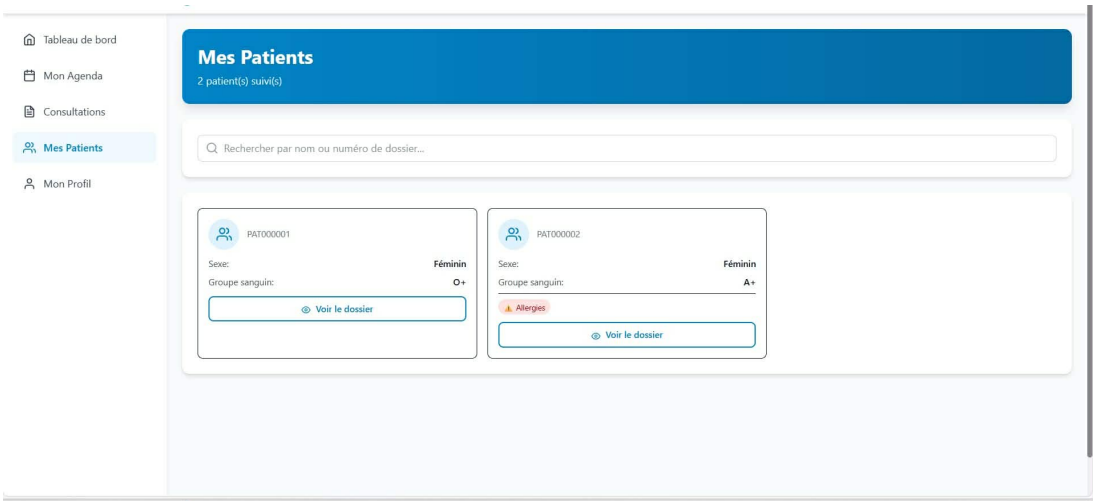

FIGURE 5.4 – Interface Dashboard Médecin - Consultation

Fonctionnalités clés : L'agenda hebdomadaire affiche tous les RDV avec code couleur par type et rappels automatiques 15 minutes avant. Le formulaire consultation permet saisie du diagnostic, constantes vitales (pression, fréquence cardiaque, saturation), observations cliniques et recommandations thérapeutiques. La génération d'ordonnances utilise templates professionnels avec en-tête praticien, prescriptions détaillées, mentions légales et signature électronique. Les certificats couvrent tous types : arrêts travail, aptitude sportive, scolarité et aptitude professionnelle.

5.2.3 Dashboard Secrétaire

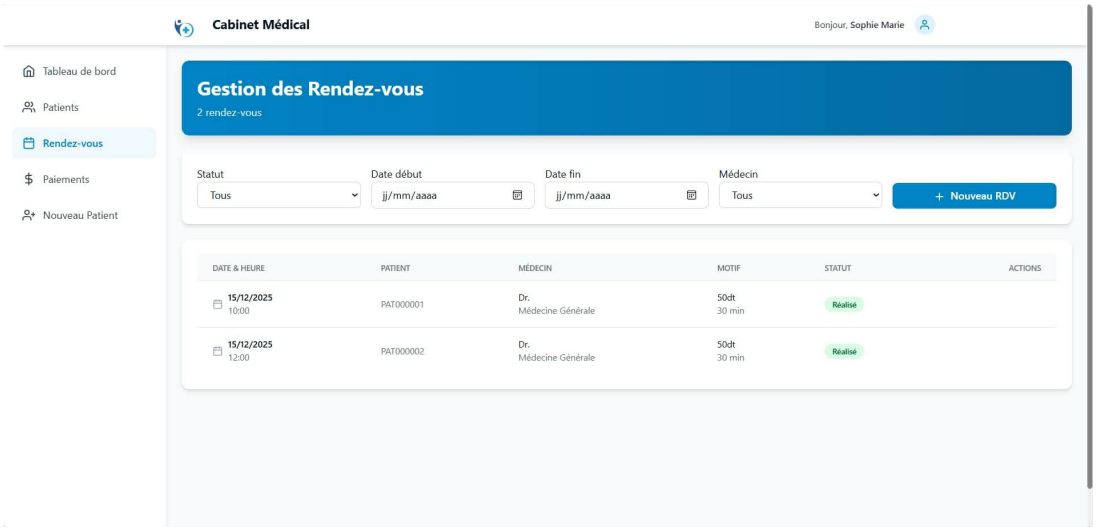

FIGURE 5.5 – Interface Dashboard Secrétaire - Gestion des patients

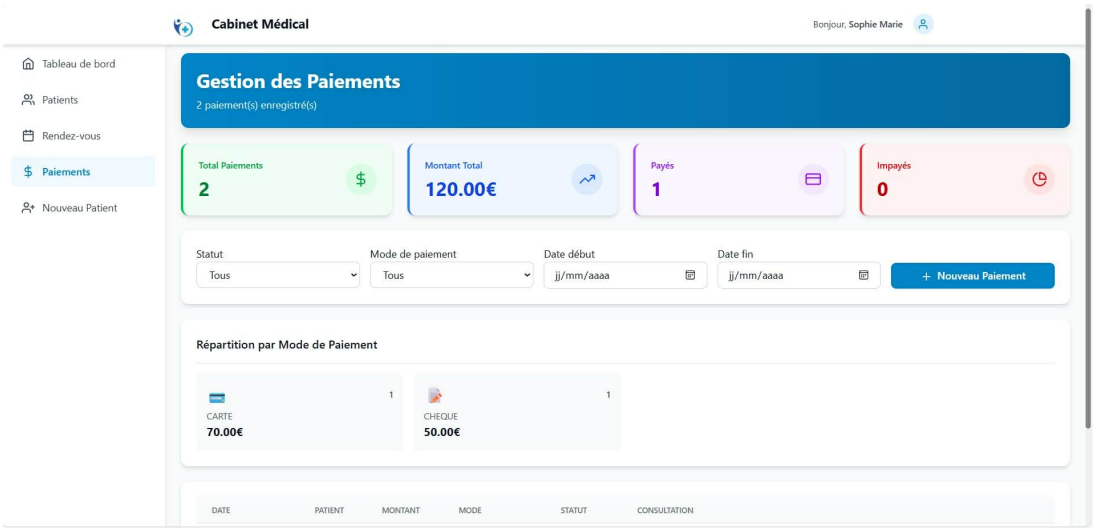

FIGURE 5.6 – Interface Dashboard Secrétaire - Planification des rendez-vous

Fonctionnalités clés : La gestion patients offre création complète des dossiers, modification avec historique des changements, et recherche multicritère (nom, dossier, date). La planification RDV propose vue synthétique avec filtrage par médecin, confirmation avec emails de rappel automatiques, annulation avec raison obligatoire et notification, et reprogrammation avec suggestion de nouveaux créneaux. Le système paiements supporte tous modes de règlement (espèces, carte, chèque, virement) et génère reçus PDF officiels avec numérotation séquentielle.

5.2.4 Sécurité Implémentée

Le système implémente une sécurité multicouche conforme aux standards de l'industrie. L'**authentification** utilise des tokens JWT signés HS256 valides 24h, transmis dans les en-têtes HTTP Authorization. L'**autorisation** RBAC vérifie à chaque requête les permissions selon le rôle (Patient, Médecin, Secrétaire, Admin) avec blocage HTTP 403 si non autorisé. Les **mots de passe** sont hachés avec BCrypt (10 rounds) rendant impossible leur récupération. La **validation** Express-validator vérifie format, longueur et type de toutes les entrées pour prévenir injections SQL et XSS. Le protocole **HTTPS** avec TLS 1.3 est obligatoire et la configuration **CORS** utilise une whitelist stricte des origines autorisées.

Chapitre 6

Conclusion

6.1 Bilan du Projet

Le projet a abouti à une application web complète et fonctionnelle répondant aux 23 besoins identifiés. Tous les objectifs stratégiques sont atteints : automatisation des rendez-vous avec prévention des conflits, centralisation sécurisée des dossiers patients MongoDB, génération automatique de documents PDF conformes aux standards, interfaces ergonomiques adaptées par rôle, et sécurité robuste (JWT, BCrypt, HTTPS) conforme aux exigences du secteur médical.

6.2 Difficultés Rencontrées

Le développement a présenté plusieurs défis techniques résolus avec succès. La **gestion des conflits horaires** a été traitée par un index unique composite MongoDB sur (date, heure, idMédecin) empêchant les doublons au niveau base de données. La **génération PDF** avec PDFKit nécessitait optimisations car consommait beaucoup de CPU ; implémentation de cache des templates et génération asynchrone ont amélioré les performances. L'**authentification Google OAuth** demandait attention sur les URLs de callback entre environnements développement et production. Le **responsive design** a été facilité par Tailwind CSS et son système de classes utilitaires responsives pour adaptation mobile.

6.3 Compétences Acquises

— Modélisation UML professionnelle (cas d'utilisation, classes, séquence)

— Architecture full-stack moderne (React + Node.js + MongoDB)

— Développement API REST sécurisée

— Gestion état avec React Hooks et Context

— Authentification JWT et OAuth 2.0

— Génération documents PDF côté serveur

— Méthodologie Agile et travail en équipe

6.4 Perspectives Futures

— Téléconsultation en temps réel (WebRTC)

— Notifications push en temps réel (WebSocket)

— Application mobile native (React Native)

— Intégration pharmacies et laboratoires

— Analytics avancées et dashboard BI

— Certificat HIPAA et conformité RGPD complète

Annexe A

Installation et Configuration

A.1 Prérequis

— Node.js >= 16.x

— npm >= 8.x

— MongoDB >= 5.x (local ou Atlas)

A.2 Installation Backend

```
cd backend
npm install
cp .env.example .env
npm run dev
```

A.3 Installation Frontend

```
cd frontend
npm install
npm run dev
Acceder a http://localhost:5173
```

Annexe B

Variables d'Environnement

```
PORT=3000
MONGO_URI=mongodb://localhost:27017/medical-cabinet
JWT_SECRET=your-super-secret-key
FRONTEND_URL=http://localhost:5173
GOOGLE_CLIENT_ID=your_google_client_id
GOOGLE_CLIENT_SECRET=your_google_secret
SMTP_HOST=smtp.gmail.com
SMTP_USER=your-email@gmail.com
SMTP_PASS=your-app-password
```

Annexe C

Références

— React Documentation : https://react.dev

— Node.js Documentation : https://nodejs.org

— MongoDB Documentation : https://www.mongodb.com/docs

— Express.js Guide : https://expressjs.com

— JWT.io : https://jwt.io

— PlantUML : https://plantuml.com