RÉPUBLIQUE DE CÔTE D'IVOIRE
Union - Discipline - Travail

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

MÉMOIRE DE FIN D'ÉTUDES

CONCEPTION ET RÉALISATION D'UNE APPLICATION WEB DE GESTION DES RENDEZ-VOUS MÉDICAUX AVEC SYSTÈME DE NOTIFICATIONS

Présenté par : VOTRE NOM

Encadré par : NOM DE L'ENCADRANT

Année académique : 2024 - 2025

TABLE DES MATIÈRES

INTRODUCTION GÉNÉRALE 4

CHAPITRE 1 : CADRE THÉORIQUE ET MÉTHODOLOGIQUE 6

1.1 Contexte et problématique 6

1.2 Objectifs du projet 8

1.3 Méthodologie adoptée 9

CHAPITRE 2 : ANALYSE ET CONCEPTION 11

2.1 Analyse des besoins 11

2.2 Diagrammes UML 14

2.3 Modèle de données 22

2.4 Architecture technique 26

CHAPITRE 3 : RÉALISATION ET IMPLÉMENTATION 30

3.1 Technologies utilisées 30

3.2 Structure du projet 34

3.3 Fonctionnalités implémentées 38

3.4 Sécurité et authentification 52

CHAPITRE 4 : PRÉSENTATION DE L'APPLICATION 56

4.1 Interface d'authentification 56

4.2 Espace Patient 58

4.3 Espace Médecin 68

4.4 Espace Administrateur 78

CHAPITRE 5 : TESTS ET DÉPLOIEMENT 88

5.1 Stratégie de tests 88

5.2 Déploiement 92

5.3 Perspectives d'amélioration 94

CONCLUSION GÉNÉRALE 96

BIBLIOGRAPHIE ET WEBOGRAPHIE 98

ANNEXES 100

LISTE DES FIGURES

Figure 1 : Diagramme de cas d'utilisation global 15

Figure 2 : Diagramme de cas d'utilisation - Patient 16

Figure 3 : Diagramme de cas d'utilisation - Médecin 17

Figure 4 : Diagramme de cas d'utilisation - Admin 18

Figure 5 : Diagramme de séquence - Authentification 19

Figure 6 : Diagramme de séquence - Prise de rendez-vous 20

Figure 7 : Diagramme de séquence - Gestion des notifications 21

Figure 8 : Diagramme de classes 23

Figure 9 : Modèle Conceptuel de Données (MCD) 24

Figure 10 : Modèle Logique de Données (MLD) 25

Figure 11 : Architecture technique du système 27

Figure 12 : Architecture Frontend React 28

Figure 13 : Architecture Backend NestJS 29

Figure 14 : Page de connexion 56

Figure 15 : Page d'inscription 57

Figure 16 : Tableau de bord Patient 59

Figure 17 : Prise de rendez-vous - Étape 1 61

Figure 18 : Prise de rendez-vous - Étape 2 62

Figure 19 : Historique des rendez-vous Patient 64

Figure 20 : Profil Patient 66

Figure 21 : Tableau de bord Médecin 69

Figure 22 : Liste des rendez-vous Médecin 71

Figure 23 : Gestion des patients 73

Figure 24 : Notes médicales 75

Figure 25 : Gestion des créneaux horaires 77

Figure 26 : Tableau de bord Administrateur 79

Figure 27 : Gestion des utilisateurs 81

Figure 28 : Statistiques avancées 83

Figure 29 : Journaux d'audit 85
Figure 30 : Paramètres système 87

LISTE DES TABLEAUX

Tableau 1 : Comparatif des solutions existantes 7
Tableau 2 : Besoins fonctionnels par acteur 12
Tableau 3 : Besoins non fonctionnels 13
Tableau 4 : Description des entités du système 24
Tableau 5 : Technologies Frontend 31
Tableau 6 : Technologies Backend 32
Tableau 7 : Outils de développement 33
Tableau 8 : Structure des modules API 35
Tableau 9 : Endpoints API REST 36
Tableau 10 : Rôles et permissions 53
Tableau 11 : Comptes de test 90
Tableau 12 : Résultats des tests 91

INTRODUCTION GÉNÉRALE

Contexte général

Dans un monde où la digitalisation transforme profondément tous les secteurs d'activité, le domaine de la santé n'échappe pas à cette révolution numérique. La gestion des rendez-vous médicaux, longtemps effectuée de manière manuelle avec des carnets de rendez-vous et des appels téléphoniques, connaît aujourd'hui une transformation majeure grâce aux technologies de l'information et de la communication.

En Côte d'Ivoire, comme dans de nombreux pays en développement, les établissements de santé font face à des défis considérables : files d'attente interminables, rendez-vous manqués, difficultés de communication entre patients et praticiens, et absence de traçabilité des consultations. Ces problèmes engendrent non seulement une perte de temps précieux pour les patients et les médecins, mais aussi une baisse de la qualité des soins dispensés.

Problématique

Face à ces constats, une question fondamentale se pose : *Comment concevoir un système informatique capable de faciliter la prise de rendez-vous médicaux, d'optimiser la gestion du temps des praticiens, et d'améliorer l'expérience globale des patients, tout en intégrant un système de notifications efficace ?*

Objectifs du projet

Ce mémoire présente la conception et la réalisation d'une application web complète de gestion des rendez-vous médicaux avec système de notifications. Les objectifs principaux sont :

• Développer une plateforme web moderne, intuitive et accessible
• Permettre aux patients de prendre des rendez-vous en ligne facilement
• Offrir aux médecins des outils de gestion efficaces de leur emploi du temps
• Implémenter un système de notifications multicanal (email, SMS)
• Assurer la sécurité et la confidentialité des données médicales
• Fournir un tableau de bord administrateur pour la supervision globale

Méthodologie

Pour mener à bien ce projet, nous avons adopté une approche méthodologique structurée combinant les meilleures pratiques du génie logiciel. L'analyse des besoins a été réalisée en utilisant le langage UML (Unified Modeling Language), permettant une modélisation claire et précise du système. Le développement a suivi une architecture moderne séparant le frontend (interface utilisateur) du backend (logique métier et données).

Structure du document

Ce mémoire est organisé en cinq chapitres :

Chapitre 1 : Cadre théorique et méthodologique - Présente le contexte, la problématique détaillée, les objectifs et la méthodologie adoptée.

Chapitre 2 : Analyse et conception - Expose l'analyse des besoins, les diagrammes UML, le modèle de données et l'architecture technique retenue.

Chapitre 3 : Réalisation et implémentation - Détaille les technologies utilisées, la structure du projet, les fonctionnalités développées et les aspects de sécurité.

Chapitre 4 : Présentation de l'application - Présente les différentes interfaces de l'application avec captures d'écran et explications fonctionnelles.

Chapitre 5 : Tests et déploiement - Couvre la stratégie de tests, le déploiement de l'application et les perspectives d'amélioration future.

CHAPITRE 1 : CADRE THÉORIQUE ET MÉTHODOLOGIQUE

1.1 Contexte et problématique

1.1.1 Contexte du projet

Le système de santé ivoirien connaît une croissance significative avec l'augmentation du nombre d'établissements de santé et de professionnels médicaux. Cependant, cette croissance s'accompagne de défis majeurs en termes de gestion et d'organisation. Les patients font souvent face à de longues attentes pour obtenir un rendez-vous, tandis que les médecins peinent à optimiser leur emploi du temps.

La pandémie de COVID-19 a accentué la nécessité de disposer d'outils numériques permettant de :
• Réduire les contacts physiques inutiles
• Faciliter la prise de rendez-vous à distance
• Améliorer la communication entre patients et praticiens
• Assurer un suivi médical continu

1.1.2 Problématique détaillée

L'analyse de l'existant révèle plusieurs problèmes récurrents :

Du côté des patients :
• Difficulté à joindre les cabinets médicaux par téléphone
• Temps d'attente excessif pour obtenir un rendez-vous
• Absence de rappels automatiques entraînant des rendez-vous manqués
• Manque de visibilité sur les disponibilités des médecins
• Impossibilité de consulter l'historique des consultations

Du côté des médecins :
• Gestion manuelle et chronophage des rendez-vous
• Taux élevé de rendez-vous non honorés (no-show)
• Difficulté à gérer les urgences et les reprogrammations
• Absence d'outils de suivi des patients
• Communication limitée avec les patients entre les consultations

Du côté administratif :
• Absence de vue d'ensemble sur l'activité de l'établissement
• Difficulté à générer des statistiques et rapports
• Gestion complexe des accès et des droits utilisateurs

Tableau 1 : Comparatif des solutions existantes

Critères	Doctolib	Medecin Direct	Notre Solution
Prise de RDV en ligne	✓	✓	✓
Notifications SMS	✓	✗	✓
Notifications Email	✓	✓	✓
Espace Patient	✓	✓	✓
Espace Médecin	✓	✓	✓
Espace Admin	✓	Limité	✓
Notes médicales	✗	✗	✓
Statistiques avancées	Payant	✗	✓
Multilingue	✓	✗	✓ (FR/EN)
Open Source	✗	✗	✓
Adapté CI	✗	✗	✓

1.2 Objectifs du projet

1.2.1 Objectif général

Concevoir et développer une application web complète de gestion des rendez-vous médicaux intégrant un système de notifications multicanal, permettant d'améliorer l'efficacité de la prise en charge des patients et l'organisation des praticiens.

1.2.2 Objectifs spécifiques

Pour les patients :
• Offrir une interface intuitive pour la prise de rendez-vous en ligne 24h/24
• Permettre la consultation de l'historique des rendez-vous
• Envoyer des rappels automatiques avant chaque consultation
• Permettre l'annulation ou le report de rendez-vous

Pour les médecins :
• Fournir un tableau de bord complet avec vue sur l'activité
• Permettre la gestion des créneaux de disponibilité
• Offrir un système de notes médicales sécurisé
• Faciliter la communication avec les patients

• Générer des statistiques sur l'activité

Pour les administrateurs :
• Centraliser la gestion des utilisateurs (patients et médecins)
• Fournir des statistiques globales sur l'établissement
• Permettre la configuration du système
• Assurer la traçabilité via des journaux d'audit

1.3 Méthodologie adoptée

1.3.1 Approche de développement

Pour ce projet, nous avons adopté une méthodologie agile inspirée de Scrum, adaptée à un projet individuel. Cette approche nous a permis de :
• Travailler par itérations courtes (sprints de 2 semaines)
• Livrer des fonctionnalités de manière incrémentale
• S'adapter aux retours et aux nouvelles exigences
• Maintenir une documentation à jour

1.3.2 Outils de modélisation

L'analyse et la conception ont été réalisées avec le langage UML (Unified Modeling Language), offrant une représentation visuelle et standardisée du système. Les diagrammes suivants ont été produits :
• Diagrammes de cas d'utilisation
• Diagrammes de séquence
• Diagramme de classes
• Modèle conceptuel de données (MCD)
• Modèle logique de données (MLD)

1.3.3 Architecture logicielle

L'architecture retenue est une architecture moderne de type Client-Serveur avec séparation Frontend/Backend :
• **Frontend** : Application React.js avec Tailwind CSS
• **Backend** : API REST avec NestJS (Node.js/TypeScript)
• **Base de données** : PostgreSQL avec Prisma ORM
• **Authentification** : JWT (JSON Web Tokens)

Cette architecture permet une meilleure maintenabilité, scalabilité et une séparation claire des responsabilités.

CHAPITRE 2 : ANALYSE ET CONCEPTION

2.1 Analyse des besoins

2.1.1 Identification des acteurs

Le système identifie trois acteurs principaux, chacun avec des besoins et des responsabilités spécifiques :

Patient : Utilisateur principal du système qui souhaite prendre des rendez-vous médicaux, consulter son historique et recevoir des notifications.

Médecin : Professionnel de santé qui gère ses rendez-vous, ses patients et rédige des notes médicales.

Administrateur : Gestionnaire du système responsable de la configuration, de la gestion des utilisateurs et de la supervision globale.

Tableau 2 : Besoins fonctionnels par acteur

Acteur	Besoins fonctionnels
Patient	• S'inscrire et se connecter • Prendre un rendez-vous • Consulter l'historique des RDV • Annuler/Reporter un RDV • Recevoir des notifications • Modifier son profil
Médecin	• Se connecter • Gérer son planning • Confirmer/Annuler des RDV • Consulter la liste des patients • Rédiger des notes médicales • Gérer ses créneaux disponibles
Admin	• Gérer les utilisateurs • Valider les comptes médecins • Consulter les statistiques • Configurer le système • Consulter les journaux d'audit • Gérer les notifications globales

Tableau 3 : Besoins non fonctionnels

Catégorie	Exigence	Description

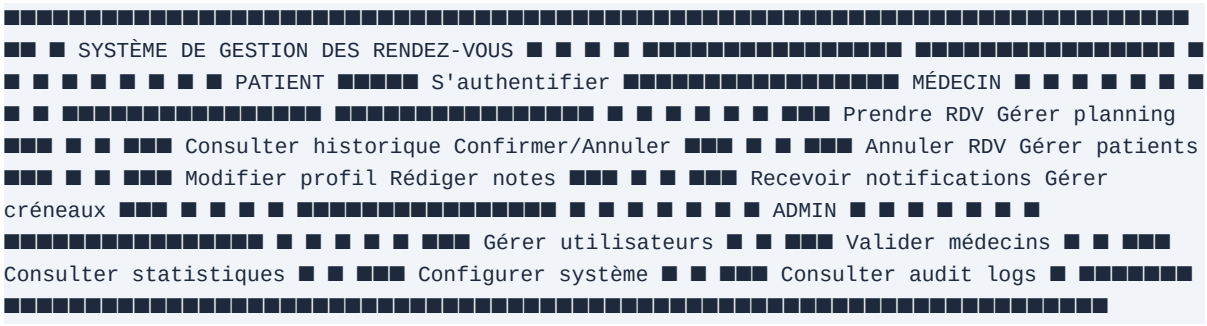
Performance	Temps de réponse	< 2 secondes pour chaque requête
Performance	Disponibilité	99% de disponibilité
Sécurité	Authentification	JWT avec tokens de rafraîchissement
Sécurité	Mots de passe	Hashage bcrypt avec salt
Sécurité	Données sensibles	Chiffrement des données médicales
Ergonomie	Responsive	Adapté mobile, tablette et desktop
Ergonomie	Accessibilité	Conformité WCAG 2.1
Ergonomie	Multilingue	Français et Anglais
Fiabilité	Sauvegarde	Backup automatique quotidien
Fiabilité	Logs	Journalisation des actions critiques

2.2 Diagrammes UML

2.2.1 Diagramme de cas d'utilisation global

Le diagramme de cas d'utilisation ci-dessous présente une vue d'ensemble des fonctionnalités du système et des interactions entre les différents acteurs.

Figure 1 : Diagramme de cas d'utilisation global



2.2.2 Diagrammes de séquence

Les diagrammes de séquence illustrent les interactions entre les acteurs et le système pour les processus clés.

Processus d'authentification :
1. L'utilisateur saisit ses identifiants (email/mot de passe)
2. Le frontend envoie une requête POST à /api/auth/login
3. Le backend vérifie les identifiants dans la base de données

4. Si valides, génération d'un accessToken (15 min) et refreshToken (7 jours)
5. Retour des tokens et des informations utilisateur au frontend
6. Stockage des tokens dans le localStorage
7. Redirection vers le tableau de bord approprié selon le rôle

Processus de prise de rendez-vous :
1. Le patient sélectionne une spécialité médicale
2. Le système affiche les médecins disponibles
3. Le patient choisit un médecin
4. Le système affiche les créneaux disponibles
5. Le patient sélectionne une date et un créneau horaire
6. Le patient saisit le motif de la consultation
7. Confirmation et création du rendez-vous (statut: EN_ATTENTE)
8. Envoi de notifications (email/SMS) au patient et au médecin

2.3 Modèle de données

2.3.1 Description des entités

Le système repose sur plusieurs entités interconnectées, chacune représentant un concept métier fondamental.

Tableau 4 : Description des entités du système

Entité	Description	Attributs principaux
User	Utilisateur du système (Patient, Médecin ou Admin)	id, nom, prenom, email, motDePasse, role, telephone
RendezVous	Rendez-vous médical entre patient et médecin	id, patientId, medecinId, date, statut, motif
TimeSlot	Créneau de disponibilité d'un médecin	id, medecinId, jour, heureDebut, heureFin, isAvailable
NoteMedicale	Note/observation médicale sur un patient	id, medecinId, patientId, contenu, statut, piecesJointes
Notification	Notification envoyée à un utilisateur	id, userId, type, titre, description, lue
AuditLog	Journal d'audit des actions	id, userId, action, entity, status, ip

2.3.2 Schéma relationnel de la base de données

User (id, nom, prenom, email, motDePasse, role, telephone, dateNaissance, adresse, isActive, specialite*, numeroOrdre*, statutValidation*, preferencesNotifEmail, preferencesNotifSms, theme, langue)

RendezVous (id, #patientId, #medecinId, date, statut, motif, createdAt, updatedAt)
patientId référence User(id)
medecinId référence User(id)

TimeSlot (id, #medecinId, jour, heureDebut, heureFin, isAvailable)
medecinId référence User(id)
Contrainte UNIQUE(medecinId, jour, heureDebut)

NoteMedicale (id, #medecinId, #patientId, contenu, statut, piecesJointes[], createdAt)
medecinId référence User(id)
patientId référence User(id)

Notification (id, #userId, type, titre, description, lue, createdAt)
userId référence User(id)

AuditLog (id, #userId, userName, userRole, action, entity, entityId, ip, userAgent, status, metadata, createdAt)
userId référence User(id) - nullable

2.4 Architecture technique

2.4.1 Architecture globale

L'application suit une architecture moderne de type Client-Serveur avec une API RESTful. Cette architecture offre plusieurs avantages :

• **Séparation des préoccupations** : Le frontend et le backend sont développés et déployés indépendamment
• **Scalabilité** : Chaque composant peut être mis à l'échelle séparément
• **Réutilisabilité** : L'API peut être consommée par d'autres clients (mobile, etc.)
• **Maintenabilité** : Les modifications sur un composant n'impactent pas les autres

2.4.2 Architecture Frontend

Le frontend est construit avec React.js et suit une architecture par composants :

• **Pages** : Composants de haut niveau correspondant aux routes
• **Components** : Composants réutilisables (Layout, Modals, Forms)
• **Context** : Gestion de l'état global (Auth, Theme, Notifications)
• **Services** : Couche d'abstraction pour les appels API
• **Utils** : Fonctions utilitaires (validation, formatage)
• **Hooks** : Hooks personnalisés pour la logique réutilisable

2.4.3 Architecture Backend

Le backend est construit avec NestJS et suit une architecture modulaire :

• **Modules** : Regroupement logique des fonctionnalités (auth, patients, medecins, admin)
• **Controllers** : Gestion des requêtes HTTP et des routes
• **Services** : Logique métier et accès aux données
• **DTOs** : Validation et typage des données entrantes
• **Guards** : Protection des routes (authentification, autorisation)
• **Decorators** : Annotations personnalisées (@CurrentUser, @Roles)

CHAPITRE 3 : RÉALISATION ET IMPLÉMENTATION

3.1 Technologies utilisées

Le choix des technologies a été guidé par plusieurs critères : modernité, performance, maintenabilité, et adéquation avec les besoins du projet.

Tableau 5 : Technologies Frontend

Technologie	Version	Rôle
React	19.x	Bibliothèque UI pour construire l'interface
React Router	7.x	Gestion du routage côté client
Tailwind CSS	3.x	Framework CSS utilitaire pour le styling
Headless UI	2.x	Composants UI accessibles (modals, menus)
Heroicons	2.x	Bibliothèque d'icônes SVG
i18next	23.x	Internationalisation (FR/EN)
Axios	1.x	Client HTTP pour les appels API
Recharts	2.x	Graphiques et visualisation de données
date-fns	3.x	Manipulation des dates

Tableau 6 : Technologies Backend

Technologie	Version	Rôle
Node.js	20.x LTS	Environnement d'exécution JavaScript
NestJS	10.x	Framework backend modulaire et scalable
TypeScript	5.x	Superset JavaScript avec typage statique
Prisma	5.22	ORM moderne pour PostgreSQL
PostgreSQL	15.x	Base de données relationnelle
JWT	-	Authentification par tokens
bcrypt	5.x	Hashage sécurisé des mots de passe
Nodemailer	6.x	Envoi d'emails (notifications)
Twilio	5.x	Envoi de SMS (notifications)

class-validator	0.14	Validation des DTOs

Tableau 7 : Outils de développement

Outil	Usage
Visual Studio Code	Éditeur de code principal
Git / GitHub	Gestion de versions et collaboration
Postman	Test des endpoints API
Prisma Studio	Interface graphique pour la base de données
Docker	Conteneurisation pour le déploiement
Render	Plateforme de déploiement cloud
Figma	Conception des maquettes UI/UX

3.2 Structure du projet

3.2.1 Organisation du code source

Le projet est organisé en monorepo avec deux applications principales :

```
projet_memoire/ ■■■ medical-appointment-api/ # Backend NestJS ■ ■■■ prisma/ ■ ■ ■■■
schema.prisma # Schéma de la base de données ■ ■ ■■■ seed.ts # Données de test ■ ■■■ src/ ■
■ ■■■ auth/ # Module authentification ■ ■ ■■■ patients/ # Module patient ■ ■ ■■■ medecins/
# Module médecin ■ ■ ■■■ admin/ # Module administration ■ ■ ■■■ notifications/ # Module
notifications ■ ■ ■■■ timeslots/ # Module créneaux horaires ■ ■ ■■■ common/ # Guards,
decorators, filters ■ ■ ■■■ prisma/ # Service Prisma ■ ■ ■■■ main.ts # Point d'entrée ■
■■■ package.json ■ ■■■ medical-appointment-frontend/ # Frontend React ■ ■■■ src/ ■ ■ ■■■
components/ # Composants réutilisables ■ ■ ■ ■■■ layout/ # Layouts par rôle ■ ■ ■ ■■■
modals/ # Modales ■ ■ ■ ■■■ common/ # Composants communs ■ ■ ■■■ pages/ # Pages par rôle ■
■ ■ ■■■ auth/ # Login, Register ■ ■ ■ ■■■ patient/ # Pages patient ■ ■ ■ ■■■ medecin/ #
Pages médecin ■ ■ ■ ■■■ admin/ # Pages admin ■ ■ ■■■ context/ # Contexts React ■ ■ ■■■
services/ # Services API ■ ■ ■■■ utils/ # Utilitaires ■ ■ ■■■ locales/ # Traductions i18n ■
■ ■■■ App.jsx # Composant racine ■ ■■■ package.json ■ ■■■ CLAUDE.md # Documentation
technique
```

Tableau 8 : Structure des modules API

Module	Controller	Service	Description
auth	auth.controller.ts	auth.service.ts	Inscription, connexion, refresh token
patients	patients.controller.ts	patients.service.ts	Gestion profil, RDV, notifications patient

medecins	medecins.controller.ts	medecins.service.ts	Gestion RDV, patients, notes, créneaux
admin	admin.controller.ts	admin.service.ts	Gestion utilisateurs, stats, audit
notifications	notifications.controller.ts	notifications.service.ts	Email, SMS, notifications in-app
timeslots	timeslots.controller.ts	timeslots.service.ts	Créneaux publics des médecins

Tableau 9 : Endpoints API REST

Méthode	Endpoint	Description	Rôle requis
POST	/api/auth/register	Inscription utilisateur	Public
POST	/api/auth/login	Connexion	Public
POST	/api/auth/refresh	Rafraîchir le token	Authentifié
GET	/api/patients/profile	Profil patient	PATIENT
POST	/api/patients/rendezvous	Créer un rendez-vous	PATIENT
GET	/api/patients/rendezvous	Liste des RDV patient	PATIENT
GET	/api/medecins/rendezvous	Liste des RDV médecin	MEDECIN
PATCH	/api/medecins/rendezvous/:id	Modifier statut RDV	MEDECIN
GET	/api/medecins/patients	Liste des patients	MEDECIN
POST	/api/medecins/notes	Créer une note médicale	MEDECIN
GET	/api/admin/users	Liste des utilisateurs	ADMIN
PATCH	/api/admin/medecins/:id/validate	Valider un médecin	ADMIN
GET	/api/admin/statistics	Statistiques globales	ADMIN
GET	/api/timeslots/medecin/:id	Créneaux d'un médecin	Public

3.3 Fonctionnalités implémentées

3.3.1 Système d'authentification

L'authentification utilise JWT (JSON Web Tokens) avec une stratégie de double token :
• **Access Token** : Durée de vie courte (15 minutes), utilisé pour les requêtes API
• **Refresh Token** : Durée de vie longue (7 jours), permet de renouveler l'access token

Processus d'authentification :
1. L'utilisateur s'inscrit ou se connecte
2. Le serveur génère et retourne les deux tokens
3. Le frontend stocke les tokens dans localStorage
4. Chaque requête API inclut l'access token dans le header Authorization
5. Quand l'access token expire, le frontend utilise le refresh token pour en obtenir un nouveau

3.3.2 Gestion des rendez-vous

Le système de rendez-vous est le cœur de l'application :

Cycle de vie d'un rendez-vous :
1. **Création** : Le patient crée un RDV → Statut: EN_ATTENTE
2. **Confirmation** : Le médecin confirme → Statut: CONFIRME
3. **Annulation** : Patient ou médecin annule → Statut: ANNULE

Règles métier importantes :
• Un patient ne peut PAS confirmer son propre rendez-vous (seul le médecin peut)
• Un patient peut uniquement annuler ses rendez-vous
• Les créneaux déjà réservés ne sont plus disponibles
• Les notifications sont envoyées automatiquement à chaque changement de statut

3.3.3 Système de notifications

Le système de notifications est multicanal :

Types de notifications :
• RAPPEL : Rappel de rendez-vous à venir
• CONFIRMATION : Confirmation de rendez-vous
• ANNULATION : Annulation de rendez-vous
• CHANGEMENT_HORAIRE : Modification d'horaire
• RECOMMANDATION : Recommandation médicale

Canaux de diffusion :
• Email via Nodemailer (SMTP)
• SMS via Twilio
• Notifications in-app (base de données)

3.3.4 Gestion des créneaux horaires

Les médecins peuvent définir leurs disponibilités :
• Configuration par jour de la semaine
• Définition d'heures de début et de fin
• Possibilité de désactiver temporairement un créneau
• Les créneaux sont publics pour permettre aux patients de voir les disponibilités

3.3.5 Notes médicales

Les médecins peuvent documenter les consultations :

• Création de notes associées à un patient
• Statut: ACTIF ou ARCHIVE
• Possibilité d'attacher des pièces jointes (PDF, images)
• Recherche et filtrage des notes

3.4 Sécurité et authentification

3.4.1 Mesures de sécurité implémentées

La sécurité est un aspect critique d'une application médicale. Plusieurs mesures ont été mises en place :

Authentification et autorisation :
• Mots de passe hashés avec bcrypt (salt rounds: 10)
• Tokens JWT signés avec une clé secrète
• Guards NestJS pour protéger les routes
• Décorateur @Roles() pour le contrôle d'accès basé sur les rôles

Validation des données :
• Validation des DTOs avec class-validator
• Sanitization des entrées utilisateur
• Validation côté frontend avant soumission

Protection des endpoints :
• CORS configuré pour n'accepter que les origines autorisées
• Rate limiting pour prévenir les attaques par force brute
• Journalisation des actions sensibles (audit logs)

Tableau 10 : Rôles et permissions

Permission	PATIENT	MEDECIN	ADMIN
Consulter son profil	✓	✓	✓
Modifier son profil	✓	✓	✓
Prendre un rendez-vous	✓	✗	✗
Annuler un rendez-vous	✓	✓	✓
Confirmer un rendez-vous	✗	✓	✓
Gérer les créneaux	✗	✓	✗
Créer des notes médicales	✗	✓	✗

Voir liste des patients	✗	✓	✓
Gérer les utilisateurs	✗	✗	✓
Valider les médecins	✗	✗	✓
Voir les statistiques globales	✗	✗	✓
Consulter les audit logs	✗	✗	✓

CHAPITRE 4 : PRÉSENTATION DE L'APPLICATION

Ce chapitre présente les différentes interfaces de l'application, organisées par espace utilisateur. Chaque interface est décrite avec ses fonctionnalités et son ergonomie.

Note : Les captures d'écran mentionnées dans ce chapitre doivent être ajoutées lors de la finalisation du document.

4.1 Interface d'authentification

4.1.1 Page de connexion

La page de connexion est la porte d'entrée de l'application. Elle présente un design moderne et épuré avec :

• Un formulaire centré avec champs email et mot de passe
• Validation en temps réel des champs
• Messages d'erreur explicites
• Lien vers la page d'inscription
• Fond avec effet de gradient animé
• Mode sombre/clair supporté

[Figure 14 : Page de connexion - Insérer capture d'écran ici]

4.1.2 Page d'inscription

L'inscription se fait en deux étapes :

Étape 1 : Choix du type de compte
• Patient : accès aux fonctionnalités de prise de rendez-vous
• Médecin : accès aux outils de gestion médicale

Étape 2 : Formulaire d'inscription
• Informations communes : nom, prénom, email, téléphone, mot de passe
• Champs spécifiques médecin : spécialité, numéro d'ordre
• Validation complète avec messages d'erreur

[Figure 15 : Page d'inscription - Insérer capture d'écran ici]

Particularité pour les médecins :
Après inscription, le compte médecin est en statut "PENDING" et doit être validé par un

administrateur avant de pouvoir être utilisé. Cette mesure garantit que seuls les professionnels de santé légitimes peuvent accéder à l'espace médecin.

4.2 Espace Patient

L'espace patient est conçu pour offrir une expérience utilisateur fluide et intuitive. Il comprend les sections suivantes :

4.2.1 Tableau de bord Patient

Le tableau de bord offre une vue synthétique de l'activité du patient :

En-tête personnalisée :
• Message de bienvenue avec le nom du patient
• Date et heure actuelles
• Accès rapide aux notifications et au profil

Cartes statistiques :
• Prochain rendez-vous avec compte à rebours
• Nombre de rendez-vous à venir
• Rendez-vous passés ce mois
• Rendez-vous annulés

Mini-calendrier :
• Vue mensuelle avec code couleur par statut
• Bleu = Confirmé, Orange = En attente, Rouge = Annulé

Graphiques statistiques :
• Évolution des consultations par mois
• Répartition par spécialité consultée

[Figure 16 : Tableau de bord Patient - Insérer capture d'écran ici]

4.2.2 Prise de rendez-vous

Le processus de prise de rendez-vous est guidé en 4 étapes :

Étape 1 : Sélection de la spécialité
• Liste des spécialités disponibles avec icônes
• Suggestions basées sur l'historique du patient

[Figure 17 : Prise de rendez-vous - Étape 1 - Insérer capture d'écran ici]

Étape 2 : Choix du médecin

• Cartes de présentation des médecins disponibles
• Informations : nom, spécialité, photo
• Filtre par disponibilité

Étape 3 : Sélection de la date et du créneau
• Calendrier interactif avec dates disponibles
• Grille des créneaux horaires
• Code couleur : Vert = disponible, Gris = indisponible

[Figure 18 : Prise de rendez-vous - Étape 2 et 3 - Insérer capture d'écran ici]

Étape 4 : Motif et confirmation
• Champ pour le motif de consultation
• Récapitulatif complet du rendez-vous
• Bouton de confirmation
• Notifications automatiques après confirmation

4.2.3 Historique des rendez-vous

Cette page permet de consulter tous les rendez-vous passés et à venir :

Fonctionnalités :
• Liste chronologique des rendez-vous
• Filtres par statut, médecin, période
• Barre de recherche
• Actions : annuler (si à venir)
• Détails complets en clic sur un rendez-vous

Informations affichées :
• Date et heure
• Médecin et spécialité
• Statut avec badge coloré
• Motif de la consultation

[Figure 19 : Historique des rendez-vous Patient - Insérer capture d'écran ici]

4.2.4 Notifications

Centre de notifications du patient :
• Liste des notifications reçues
• Icônes par type (rappel, confirmation, annulation)
• Marquage lu/non lu
• Filtres par type

• Actions : marquer comme lu, supprimer

4.2.5 Profil et Paramètres

[Figure 20 : Profil Patient - Insérer capture d'écran ici]

Page Profil :
• Visualisation des informations personnelles
• Photo de profil (initiales par défaut)
• Formulaire de modification
• Changement de mot de passe

Page Paramètres :
• Préférences de notifications (email, SMS)
• Thème (clair/sombre)
• Langue (Français/Anglais)

4.3 Espace Médecin

L'espace médecin offre des outils professionnels pour la gestion de l'activité médicale.

4.3.1 Tableau de bord Médecin

[Figure 21 : Tableau de bord Médecin - Insérer capture d'écran ici]

Éléments du tableau de bord :
• Message de bienvenue personnalisé "Bonjour Dr [Nom]"
• Rendez-vous du jour avec liste des patients
• Statistiques : RDV confirmés, en attente, annulés
• Mini-calendrier du mois
• Graphiques d'activité
• Alertes et recommandations

4.3.2 Gestion des rendez-vous

[Figure 22 : Liste des rendez-vous Médecin - Insérer capture d'écran ici]

Fonctionnalités :
• Liste complète des rendez-vous
• Filtres multiples (statut, date, patient)
• Actions : confirmer, annuler, voir détails
• Vue calendrier optionnelle
• Export des données

Workflow de confirmation :
1. RDV créé par patient → Statut: EN_ATTENTE
2. Médecin clique "Confirmer" → Statut: CONFIRME
3. Notification automatique envoyée au patient

4.3.3 Gestion des patients

[Figure 23 : Gestion des patients - Insérer capture d'écran ici]

Liste des patients :
• Tous les patients ayant eu un RDV avec ce médecin
• Recherche par nom
• Informations : nom, téléphone, dernier RDV
• Accès au dossier patient

Fiche patient :
• Informations de contact
• Historique des rendez-vous
• Notes médicales associées

4.3.4 Notes médicales

[Figure 24 : Notes médicales - Insérer capture d'écran ici]

Fonctionnalités :
• Création de notes pour chaque patient
• Éditeur de texte riche
• Statut : ACTIF ou ARCHIVE
• Pièces jointes (PDF, images)
• Recherche et filtrage
• Historique complet

4.3.5 Gestion des créneaux

[Figure 25 : Gestion des créneaux horaires - Insérer capture d'écran ici]

Configuration des disponibilités :
• Par jour de la semaine
• Heures de début et de fin
• Activation/désactivation des créneaux
• Création de créneaux multiples
• Suppression des créneaux

Règles de validation :
• Heure de fin > Heure de début
• Durée minimum : 30 minutes
• Horaires entre 6h et 22h
• Pas de chevauchement

4.4 Espace Administrateur

L'espace administrateur offre une vue globale et des outils de gestion du système.

4.4.1 Tableau de bord Administrateur

[Figure 26 : Tableau de bord Administrateur - Insérer capture d'écran ici]

Vue d'ensemble :
• Nombre total de patients
• Nombre total de médecins
• Rendez-vous du jour
• Rendez-vous en attente de confirmation
• Médecins en attente de validation

Graphiques :
• Évolution des inscriptions
• Répartition des RDV par spécialité
• Taux de RDV honorés vs annulés

4.4.2 Gestion des utilisateurs

[Figure 27 : Gestion des utilisateurs - Insérer capture d'écran ici]

Gestion des patients :
• Liste complète avec pagination
• Recherche et filtres
• Activation/désactivation de comptes
• Consultation des détails

Gestion des médecins :
• Liste avec statut de validation
• Workflow de validation : PENDING → APPROVED/REJECTED
• Modification des informations
• Gestion des spécialités

4.4.3 Statistiques avancées

[Figure 28 : Statistiques avancées - Insérer capture d'écran ici]

Indicateurs clés (KPI) :
• Nombre total de RDV
• Taux de RDV confirmés
• Taux d'annulation
• Temps moyen de confirmation
• Médecin le plus consulté
• Spécialité la plus demandée

Graphiques interactifs :
• Courbe d'évolution mensuelle
• Histogramme par spécialité
• Camembert de répartition

4.4.4 Journaux d'audit

[Figure 29 : Journaux d'audit - Insérer capture d'écran ici]

Informations tracées :
• Actions effectuées (login, création, modification, suppression)
• Utilisateur concerné
• Date et heure
• Adresse IP
• Statut (succès, échec)
• Entité impactée

Fonctionnalités :
• Recherche par utilisateur, action, date
• Filtres multiples
• Pagination
• Export (prévu)

4.4.5 Paramètres système

[Figure 30 : Paramètres système - Insérer capture d'écran ici]

Paramètres configurables :
• Gestion des spécialités médicales
• Paramètres de notifications globales
• Configuration du thème par défaut

- Paramètres de sauvegarde

CHAPITRE 5 : TESTS ET DÉPLOIEMENT

5.1 Stratégie de tests

5.1.1 Types de tests réalisés

Tests unitaires :
Les services backend ont été testés individuellement pour vérifier leur bon fonctionnement. Les tests vérifient notamment :
• La logique métier des services
• Les validations de données
• Les cas d'erreur

Tests d'intégration :
Les tests d'intégration vérifient le bon fonctionnement des modules ensemble :
• Authentification complète (inscription, connexion, refresh)
• Cycle de vie des rendez-vous
• Système de notifications

Tests manuels :
Des tests manuels exhaustifs ont été réalisés sur l'ensemble des fonctionnalités :
• Parcours utilisateur complets
• Tests de responsive design
• Tests cross-browser

5.1.2 Comptes de test

Tableau 11 : Comptes de test

Rôle	Email	Mot de passe
Admin	admin@medical.com	password123
Médecin (Cardiologie)	jean.kouadio@medical.com	password123
Médecin (Pédiatrie)	sophie.kone@medical.com	password123
Médecin (Dermatologie)	michel.traore@medical.com	password123
Patient	marie.yao@example.com	password123
Patient	kouassi.bamba@example.com	password123
Patient	fatou.diallo@example.com	password123

Tableau 12 : Résultats des tests

Fonctionnalité	Statut	Observations
Inscription Patient	✓ OK	Validation complète fonctionnelle
Inscription Médecin	✓ OK	Statut PENDING par défaut
Connexion	✓ OK	Tokens JWT générés correctement
Refresh Token	✓ OK	Renouvellement automatique
Prise de RDV	✓ OK	Workflow complet testé
Confirmation RDV	✓ OK	Seul médecin peut confirmer
Annulation RDV	✓ OK	Patient et médecin peuvent annuler
Notifications Email	✓ OK	Envoi via Nodemailer
Notifications SMS	✓ OK	Envoi via Twilio
Notes médicales	✓ OK	CRUD complet fonctionnel
Créneaux horaires	✓ OK	Validation des chevauchements
Validation médecin	✓ OK	Admin peut approuver/rejeter
Statistiques	✓ OK	Calculs corrects
Audit logs	✓ OK	Traçabilité complète
Responsive	✓ OK	Mobile, tablette, desktop
Mode sombre	✓ OK	Basculement sans rechargement
Multilingue	✓ OK	FR/EN avec persistance

5.2 Déploiement

5.2.1 Infrastructure de déploiement

L'application est déployée sur la plateforme cloud Render, offrant :
• Hébergement gratuit pour les petits projets
• Déploiement automatique depuis GitHub
• Support Docker
• Base de données PostgreSQL managée
• Certificat SSL automatique

5.2.2 Configuration Docker

L'application utilise Docker pour le déploiement. Le Dockerfile combine le backend NestJS et le frontend React dans une seule image :

Processus de build :

1. Construction du frontend React (npm run build)
2. Construction du backend NestJS
3. Copie des fichiers statiques du frontend dans le répertoire public du backend
4. Le backend NestJS sert les fichiers statiques du frontend

5.2.3 Variables d'environnement

Les variables suivantes doivent être configurées :
• DATABASE_URL : URL de connexion PostgreSQL
• JWT_SECRET : Clé secrète pour les tokens
• JWT_REFRESH_SECRET : Clé pour les refresh tokens
• EMAIL_HOST, EMAIL_USER, EMAIL_PASS : Configuration SMTP
• TWILIO_SID, TWILIO_TOKEN, TWILIO_PHONE : Configuration Twilio

5.2.4 Procédure de déploiement

1. Push du code sur la branche main (GitHub)
2. Render détecte le changement et lance le build
3. Exécution des migrations Prisma
4. Démarrage de l'application
5. Vérification du health check

5.3 Perspectives d'amélioration

5.3.1 Améliorations fonctionnelles

• **Téléconsultation** : Intégration de la visioconférence pour les consultations à distance
• **Paiement en ligne** : Intégration d'un système de paiement (Orange Money, MTN Money, Visa)
• **Rappels intelligents** : Rappels automatiques basés sur l'historique médical
• **Ordonnances numériques** : Génération d'ordonnances signées électroniquement
• **Dossier médical partagé** : Historique médical complet accessible par tous les praticiens

5.3.2 Améliorations techniques

• **Application mobile** : Développement d'applications iOS et Android natives
• **Notifications push** : Notifications en temps réel sur mobile
• **Cache Redis** : Amélioration des performances avec mise en cache
• **Tests automatisés** : Couverture de tests à 80%+
• **CI/CD** : Pipeline d'intégration et déploiement continus

5.3.3 Sécurité renforcée

• **2FA** : Authentification à deux facteurs obligatoire pour les médecins

- **Chiffrement des données** : Chiffrement au repos des données sensibles
- **Audit de sécurité** : Pentest régulier par des experts
- **Conformité RGPD** : Mise en conformité complète avec le règlement européen

CONCLUSION GÉNÉRALE

Bilan du projet

Ce projet de fin d'études nous a permis de concevoir et de réaliser une application web complète de gestion des rendez-vous médicaux avec système de notifications. L'application répond aux besoins identifiés lors de l'analyse, offrant une solution moderne et efficace pour la prise de rendez-vous médicaux en Côte d'Ivoire.

Objectifs atteints

Les objectifs initiaux ont été largement atteints :

✓ **Interface intuitive** : L'application offre une expérience utilisateur fluide et moderne, accessible à tous les profils d'utilisateurs.

✓ **Prise de rendez-vous en ligne** : Les patients peuvent facilement prendre, modifier ou annuler leurs rendez-vous 24h/24.

✓ **Gestion médicale efficace** : Les médecins disposent d'outils complets pour gérer leur planning, leurs patients et leurs notes médicales.

✓ **Système de notifications** : Les notifications par email et SMS garantissent une communication efficace et réduisent les rendez-vous manqués.

✓ **Administration centralisée** : Les administrateurs peuvent superviser l'ensemble du système et gérer les utilisateurs.

✓ **Sécurité** : L'authentification JWT, le hashage des mots de passe et le contrôle d'accès par rôles garantissent la sécurité des données.

Compétences acquises

Ce projet nous a permis de développer et renforcer de nombreuses compétences :

• **Développement Frontend** : Maîtrise de React.js, Tailwind CSS, gestion d'état, routage
• **Développement Backend** : Architecture NestJS, API REST, TypeScript, ORM Prisma
• **Base de données** : Conception, modélisation et manipulation avec PostgreSQL
• **Sécurité** : Authentification JWT, hashage, protection des données
• **DevOps** : Docker, déploiement cloud, variables d'environnement
• **Méthodologie** : Analyse UML, conception orientée objet, approche agile

Difficultés rencontrées

Plusieurs défis ont été relevés durant ce projet :

• La gestion de l'authentification avec refresh tokens
• La synchronisation entre frontend et backend
• La gestion des créneaux horaires sans chevauchement
• L'optimisation des performances pour les listes volumineuses
• L'internationalisation complète de l'application

Perspectives

L'application peut être étendue avec de nouvelles fonctionnalités telles que la téléconsultation, le paiement en ligne, une application mobile native, ou encore un dossier médical partagé. Ces évolutions permettraient de proposer une solution encore plus complète répondant aux besoins croissants de la e-santé en Côte d'Ivoire.

Conclusion

En définitive, ce projet a été une expérience enrichissante qui nous a permis de mettre en pratique les connaissances acquises durant notre formation tout en découvrant de nouvelles technologies. L'application développée représente une contribution concrète à l'amélioration de l'accès aux soins de santé grâce aux technologies de l'information.

BIBLIOGRAPHIE ET WEBOGRAPHIE

Ouvrages et articles

[1] GAMMA E., HELM R., JOHNSON R., VLISSIDES J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[2] FOWLER M., *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2002.

[3] ROQUES P., *UML 2 par la pratique*, 7e édition, Eyrolles, 2011.

[4] BANKS A., PORCELLO E., *Learning React: Modern Patterns for Developing React Apps*, O'Reilly Media, 2020.

[5] KAMINSKI M., *NestJS: A Progressive Node.js Framework*, Packt Publishing, 2021.

Documentation officielle

[6] React Documentation, *https://react.dev/*, consulté en 2024.

[7] NestJS Documentation, *https://docs.nestjs.com/*, consulté en 2024.

[8] Prisma Documentation, *https://www.prisma.io/docs*, consulté en 2024.

[9] Tailwind CSS Documentation, *https://tailwindcss.com/docs*, consulté en 2024.

[10] PostgreSQL Documentation, *https://www.postgresql.org/docs/*, consulté en 2024.

[11] JWT.io, *https://jwt.io/introduction*, consulté en 2024.

Ressources en ligne

[12] MDN Web Docs, *https://developer.mozilla.org/*, consulté en 2024.

[13] Stack Overflow, *https://stackoverflow.com/*, consulté en 2024.

[14] GitHub, *https://github.com/*, consulté en 2024.

[15] Render Documentation, *https://render.com/docs*, consulté en 2024.

Tutoriels et cours

[16] Traversy Media, *React Crash Course*, YouTube, 2023.

[17] Academind, *NestJS Complete Course*, Udemy, 2023.

[18] The Net Ninja, *Tailwind CSS Tutorial*, YouTube, 2023.

ANNEXES

Annexe A : Schéma complet de la base de données (Prisma)

Le schéma Prisma complet définit les modèles suivants :
• User (avec tous les champs et relations)
• RendezVous
• TimeSlot
• NoteMedicale
• Notification
• MedecinIndisponibilite
• AuditLog

Le fichier schema.prisma complet est disponible dans le code source du projet.

Annexe B : Configuration du fichier .env

Variables d'environnement requises pour le backend :

```
# Base de données
DATABASE_URL="postgresql://user:password@localhost:5432/medical_db" # JWT
JWT_SECRET="votre_secret_jwt_très_long_et_sécurisé"
JWT_REFRESH_SECRET="votre_secret_refresh_très_long" # Email (Nodemailer)
EMAIL_HOST="smtp.gmail.com" EMAIL_PORT=587 EMAIL_USER="votre_email@gmail.com"
EMAIL_PASS="votre_mot_de_passe_app" # Twilio (SMS)
TWILIO_ACCOUNT_SID="ACxxxxxxxxxx" TWILIO_AUTH_TOKEN="votre_token"
TWILIO_PHONE_NUMBER="+1234567890" # Application PORT=3002 NODE_ENV=production
```

Annexe C : Commandes utiles

Backend (medical-appointment-api) :

```
# Installation des dépendances npm install # Générer le client Prisma npx prisma
generate # Exécuter les migrations npx prisma migrate dev # Peupler la base avec les
données de test npx prisma db seed # Lancer en mode développement npm run start:dev
# Lancer en mode production npm run start:prod # Lancer les tests npm run test
```

Frontend (medical-appointment-frontend) :

```
# Installation des dépendances npm install # Lancer en mode développement npm start
# Build pour la production npm run build # Lancer les tests npm test
```

Annexe D : Structure des réponses API

Toutes les réponses de l'API suivent un format standardisé :

Succès :
{ "statusCode": 200, "data": { ... }, "message": "Opération réussie" }

Erreur :
{ "statusCode": 400/401/403/404/500, "message": "Description de l'erreur", "error": "Type d'erreur" }

Annexe E : Guide d'installation rapide

1. Cloner le dépôt GitHub
2. Installer PostgreSQL et créer la base de données
3. Configurer les fichiers .env (backend et frontend)
4. Installer les dépendances (npm install) dans les deux dossiers
5. Exécuter les migrations Prisma
6. Lancer le seed pour les données de test
7. Démarrer le backend (port 3002)
8. Démarrer le frontend (port 3000)
9. Accéder à l'application : http://localhost:3000