



Université de Bordeaux

Master 1 Génie Logiciel

**Rapport de Projet de Programmation :**

# Conception d'une application web pour la gestion d'un cabinet médical

---

Dépôt Git : <https://github.com/val-1st/Medicolib>

*Réalisé par*

Ephrem Jennifer  
Loustau Valentin  
Duboureau Guillaume  
Goudoussy Diallo Abdoul

*Encadré par*

Mme Farah Sarah Ouada

Année universitaire 2022-2023

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Analyse des besoins</b>	<b>5</b>
2.1	Identification des acteurs . . . . .	5
2.2	Les besoins fonctionnels . . . . .	5
2.2.1	Pour le patient . . . . .	5
2.2.2	Pour le médecin . . . . .	6
2.3	Les besoins non-fonctionnels . . . . .	7
2.4	Diagrammes de cas d'utilisation . . . . .	8
2.4.1	Créer un compte . . . . .	8
2.4.2	S'authentifier . . . . .	8
2.4.3	Gérer les rendez-vous en ligne (Patient) . . . . .	9
2.4.4	Gérer les rendez-vous en ligne (Médecin) . . . . .	9
2.4.5	Consulter les rendez-vous . . . . .	9
2.4.6	Consulter le dossier médical . . . . .	10
2.4.7	Gérer le dossier médical . . . . .	10
2.4.8	Générer le bilan de santé . . . . .	10
<b>3</b>	<b>Conception et Réalisation</b>	<b>11</b>
3.1	Diagrammes de classe du Domain . . . . .	11
3.2	Passage au modèle relationnel . . . . .	12
3.2.1	Règles de passage du diagramme de classes au modèle logique . . . . .	12
3.2.2	Modèle logique de données . . . . .	13
3.3	Outils de développement . . . . .	13
3.4	Côté back-end . . . . .	14
3.5	Côté front-end . . . . .	14
3.6	Base de données . . . . .	15
<b>4</b>	<b>Tests</b>	<b>17</b>
4.0.1	UserControllerTest . . . . .	17
4.0.2	PatientControllerTest . . . . .	18
4.0.3	DoctorControllerTest . . . . .	18
<b>5</b>	<b>Résultats</b>	<b>20</b>
5.1	Navigation en tant que <i>Visiteur</i> (Utilisateur non-connecté) : . . . . .	20
5.1.1	Fonctionnalité de l'entête (Header) : . . . . .	20
5.1.2	Fonctionnalité de la page d'Accueil (Homepage) : . . . . .	20
5.1.3	Fonctionnalité de la page repertoriant les docteurs (Docteurs) : . . . . .	21
5.1.4	Fonctionnalité de la page personnel d'un médecin (Docteur) : . . . . .	21
5.1.5	Fonctionnalité de la page de connexion (Login) : . . . . .	21
5.1.6	Fonctionnalité de la page de création de compte (Register) : . . . . .	21
5.2	Navigation en tant que <i>Patient</i> (connecté) : . . . . .	21
5.2.1	Fonctionnalité de l'entête (Header) : . . . . .	21
5.2.2	Fonctionnalité de la page d'Accueil (Homepage) : . . . . .	22
5.2.3	Fonctionnalité de la page repertoriant les docteurs (Docteurs) : . . . . .	22

5.2.4	Fonctionnalité de la page personnel d'un médecin (Docteur) : . . . . .	22
5.2.5	Fonctionnalité de la page des rendez-vous (Appointments) : . . . . .	22
5.2.6	Fonctionnalité de la page des documents (Documents) : . . . . .	22
5.2.7	Fonctionnalité de la page du compte (Edit) : . . . . .	22
5.3	Navigation en tant que <i>Docteur (connecté)</i> : . . . . .	23
5.3.1	Fonctionnalité de l'entête (Header) : . . . . .	23
5.3.2	Fonctionnalité de la page répertoriant les RDV (Planning) : . . . . .	23
5.3.3	Fonctionnalité de la page répertoriant les patients d'un docteur (Patients) : . . . . .	23
5.3.4	Fonctionnalité de la page de consultation du dossier médical d'un patient : . . . . .	24
5.3.5	Fonctionnalité de la page d'ajout d'une consultation : . . . . .	24
5.3.6	Fonctionnalité de la page du compte (Edit) : . . . . .	24
5.4	Présentation de l'application . . . . .	25
5.4.1	Page d'accueil . . . . .	25
5.4.2	Login . . . . .	26
5.4.3	Register . . . . .	26
5.4.4	Patient Header . . . . .	26
5.4.5	Page des docteurs . . . . .	27
5.4.6	Page de réservation d'un docteur . . . . .	27
5.4.7	Page Mon Compte du patient . . . . .	28
5.4.8	Page RDV du patient (lorsqu'il n'y a pas de RDV) . . . . .	28
5.4.9	Page RDV du patient . . . . .	29
5.4.10	Confirmation de suppression d'un RDV . . . . .	29
5.4.11	Page Documents du patient (sans documents) . . . . .	30
5.4.12	Page Documents du patient . . . . .	30
5.4.13	Doctor Header . . . . .	30
5.4.14	Doctor Planning . . . . .	31
5.4.15	RDV sur le planning du docteur . . . . .	31
5.4.16	Ajout d'une consultation sans prescription médicale . . . . .	31
5.4.17	Ajout d'une consultation avec prescription médicale . . . . .	32
5.4.18	Prescription médicale . . . . .	32
5.4.19	Page Mon Compte du docteur . . . . .	33
5.4.20	Envoie d'un mail . . . . .	33

## 6 Conclusion

34

# Table des figures

2.1	Diagramme de cas d'utilisation : créer un compte . . . . .	8
2.2	Diagramme de cas d'utilisation : s'authentifier . . . . .	8
2.3	Diagramme de cas d'utilisation : Gérer les rendez-vous en ligne (Patient) . . .	9
2.4	Diagramme de cas d'utilisation : Gérer les rendez-vous en ligne (Médecin) . .	9
2.5	Diagramme de cas d'utilisation : Consulter les rendez-vous . . . . .	9
2.6	Diagramme de cas d'utilisation : Consulter le dossier médical . . . . .	10
2.7	Diagramme de cas d'utilisation : Gérer le dossier médical . . . . .	10
2.8	Diagramme de cas d'utilisation : Générer le bilan de santé . . . . .	10
3.1	Diagramme de classes . . . . .	12
3.2	Schéma Relationnel . . . . .	13
5.1	<i>Page d'accueil</i> . . . . .	25
5.2	<i>Login</i> . . . . .	26
5.3	<i>Register</i> . . . . .	26
5.4	<i>Patient Header</i> . . . . .	26
5.5	<i>Page des docteurs</i> . . . . .	27
5.6	<i>Page de réservation d'un docteur</i> . . . . .	27
5.7	<i>Page Mon Compte du patient</i> . . . . .	28
5.8	<i>Page RDV du patient</i> (lorsqu'il n'y a pas de RDV) . . . . .	28
5.9	<i>Page RDV du patient</i> . . . . .	29
5.10	Confirmation de suppression d'un RDV . . . . .	29
5.11	<i>Page Documents du patient</i> (sans documents) . . . . .	30
5.12	<i>Page Documents du patient</i> . . . . .	30
5.13	<i>Doctor Header</i> . . . . .	30
5.14	<i>Doctor Planning</i> . . . . .	31
5.15	RDV sur le planning du docteur . . . . .	31
5.16	Ajout d'une consultation sans prescription médicale . . . . .	31
5.17	Ajout d'une consultation avec prescription médicale . . . . .	32
5.18	Prescription médicale . . . . .	32
5.19	<i>Page Mon Compte du docteur</i> . . . . .	33
5.20	Envoie d'un mail . . . . .	33

# Chapitre 1

## Introduction

La gestion d'un cabinet médical peut vite devenir compliquée pour les professionnels de santé. L'analyse, la collecte et la gestion des données sont des éléments clés pour fournir des soins de qualité aux patients. Pour faciliter ces tâches, nous avons conçu une application de gestion d'un cabinet médical qui permettra aux professionnels de santé de gérer efficacement leurs rendez-vous, les dossiers médicaux ainsi que les ordonnances de leurs patients. Notre interface intuitive et simple d'utilisation permettra aux patients de trouver un rendez-vous avec le spécialiste de leur choix en un rien de temps.

L'objectif et la problématique, ici, est de concevoir une application web qui soit la mieux construite possible. Cela permettra ainsi d'avoir une gestion d'un cabinet médicale la plus simple et fluide possible. Pour cela, nous devons répondre à certains besoins, que nous détaillerons dans les parties suivantes, tels que la centralisation des dossiers médicaux et de faciliter la navigation et l'expérience utilisateur, que se soit pour le médecin ou le patient.

Le présent document expose les différentes étapes franchies afin de mener à bien ce projet et qui se traduisent à travers quatre chapitres. Nous commençons dans le premier chapitre par introduire le cadre général du travail en présentant la problématique traitée dans ce projet. Par la suite, nous établirons une description complète du comportement du système à développer, où nous allons présenter les différents besoins fonctionnels et non-fonctionnels capturés, ainsi qu'une identification des acteurs du système. Ensuite, nous analyserons les besoins à travers l'élaboration des diagrammes de cas d'utilisation.

Le seconde chapitre est consacré à la conception et à la réalisation, dans laquelle nous détaillons notre solution en définissant l'architecture logicielle, la présentation du modèle de conception orientés objets Domain Driven Design (DDD), qu'il est focalisé sur le domaine de l'application, sur la logique associée, et le métier adressé par le logiciel. De plus, la description des choix technologique des outils et langages de développement utilisés pour la réalisation de notre application.

Le troisième chapitre aborde la partie sur les tests implémentés.

Enfin, nous exposons dans le quatrième et dernier chapitre le travail réalisé en présentant les différentes interfaces.

Nous concluons ce rapport par une conclusion générale résumant les principales fonctionnalités réalisées et proposant quelques perspectives en vue d'élargir et d'améliorer ce travail.

# Chapitre 2

## Analyse des besoins

### 2.1 Identification des acteurs

Un acteur représente un rôle joué par une personne externe ou par un processus qui interagit avec le système.

Les acteurs de notre système sont :

- **Patient** : il s'agit d'un acteur qui utilise le site pour gérer (ajouter, consulter, supprimer) ses rendez-vous, consulter son dossier médical. Il peut également mettre à jour ses informations personnelles et communiquer ses informations au médecin.
- **Médecin** : il s'agit d'un acteur qui gère les dossiers des patients, prescrit des ordonnances et les imprime. Il s'occupe de la gestion des rendez-vous de ses patients et les notifie s'il effectue quelque modification.

### 2.2 Les besoins fonctionnels

#### 2.2.1 Pour le patient

##### 1. Création de compte

- Quantifications : Pouvoir créer qu'un seul compte par adresse mail.
- Contraintes ou difficultés techniques : Sécurité des données personnelles des patients.
- Énonciation des risques et parades : Risque de vols des données.  
**Parade** : Protocole de sécurité pour protéger les données.
- Spécification des tests de contrôle : Tests unitaires.

##### 2. Gestion et consultation des rendez-vous en ligne

- Quantifications : Pouvoir prendre plusieurs rendez-vous selon les créneaux libres sur le planning. De plus, possibilité de visualiser ses rendez-vous en temps réel.
- Éléments de faisabilité : Mettre en place un système de gestion de rendez-vous en ligne tel que doctolib et vérifier sa faisabilité.
- Contraintes ou difficultés techniques : Assurer la synchronisation en temps réel avec le calendrier du médecin, garantir la disponibilité d'un rendez-vous.
- Énonciation des risques et parades : Chevauchement de prise de rendez-vous pour des patients différents, ou sélectionner un rendez-vous qui n'existe plus.  
**Parade** : Bloquer la prise d'un rendez-vous pour le premier arrivé.
- Spécification des tests de contrôle : Tests unitaires.

### 2.2.2 Pour le médecin

#### 1. Gérer (ajouter/modifier/supprimer documents) les dossiers médicaux des patients

- Quantifications : Vue complète des dossiers des patients en temps réel.
- Éléments de faisabilité : Évaluation de différents systèmes de dossiers médicaux pour comprendre les fonctionnalités.
- Contraintes ou difficultés techniques : Sécurité des données.
- Énonciation des risques et parades : Risques de pertes de données à cause d'un souci technique, risque de non synchronisation/mise à jour.  
**Parade** : Historique de sauvegarde.
- Spécification des tests de contrôle : Tests de sécurité.

#### 2. Gestion de ses rendez-vous en ligne

- Quantification : Possibilité de visualiser son calendrier de rendez-vous en temps réel.
- Éléments de faisabilité : Mettre en place un système de gestion de rendez-vous en ligne tel que doctolib et vérifier sa faisabilité.
- Contraintes ou difficultés techniques : Assurer la synchronisation en temps réel avec le calendrier, garantir la disponibilité d'un rendez-vous (ne pas placer le rendez-vous sur le créneau d'un autre patient, dû à une synchronisation non effectué).
- Énonciation des risques et parades : Rendez-vous en simultanée pour des patients différents, sélectionner un rendez-vous qui n'existe plus.  
**Parade** : Bloquer la prise d'un rendez-vous pour le premier arrivé.
- Spécification des tests de contrôle : Tests unitaires.

#### 3. Communication avec les patients

- Éléments de faisabilité : Mise en place d'un système de notifications par e-mail.
- Contraintes ou difficultés techniques : Aucune contrainte.

#### 4. Consulter son planning des rendez-vous

- Quantification : Le médecin doit pouvoir consulter son planning pour chaque jour de la semaine, ainsi que pour une période donnée : 1 semaine. Le planning doit être mis à jour en temps réel lorsque des rendez-vous sont ajoutés ou supprimés.
- Éléments de faisabilité : La consultation du planning peut être réalisée en utilisant une interface graphique simple, qui affiche les rendez-vous (stockés dans une base de données) en fonction de l'heure et de la date tel que le logiciel Google Calendar.
- Contraintes ou difficultés techniques : Garantir la confidentialité des informations stockées dans la base de données.
- Énonciation des risques et parades : Risque que les rendez-vous soient compromis si la base de données est piratée. Risque que le planning du médecin ne soit pas à jour après modification.  
**Parade** : Le système peut être conçu pour utiliser des méthodes de cryptage pour protéger les données sensibles. Actualisation du planning après chaque modification.
- Spécification des tests de contrôle : Des tests de validation peuvent être effectués pour vérifier que le système affiche correctement les rendez-vous dans l'interface utilisateur. Des tests de contrôle peuvent être effectués pour vérifier que les données sont correctement stockées dans la base de données et affichées.

dans l'interface utilisateur.

5. Bilan de santé (Le système doit assurer l'impression des fiches malades et les bilans) :

- Contraintes ou difficultés techniques : Les fichiers doivent être au format PDF exclusivement pour ne pas perdre d'informations et pour une meilleure impression.
- Énonciation des risques et parades : Risque de mauvais transfert/perte de données lors de l'envoi du bilan par le médecin au patient.  
**Parade** : Nous pourrions vérifier si le document reçu par le patient est vide (null) ou non.

## 2.3 Les besoins non-fonctionnels

Ce sont des besoins en relation avec la performance du système, la facilité d'utilisation, l'ergonomie des interfaces, la sécurité etc. Et parmi ces besoins nous citons :

1. Sécurité des données médicales des patients

- Éléments de faisabilité : Mise en place d'un système d'identification et d'authentification de chaque utilisateur (patients/médecins) pour garantir que seuls les utilisateurs autorisés aient accès au système et aux données sensibles.
- Contraintes ou difficultés techniques : Complexité de la gestion des autorisations d'accès pour les différents utilisateurs.
- Énonciation des risques et parades : Risque de faille et accès à un intrus aux données des patients.  
**Parade** : Il faudrait pour cela protéger ces données.
- Spécification des tests de contrôle : Connexion au logiciel avec les différents acteurs et vérifier leur autorisation (un patient n'a pas accès au rendez-vous de tous les patients par exemple).

2. Simplicité et ergonomie de l'interface graphique

- Éléments de faisabilité : Mise en place d'une interface intuitive et facile à utiliser pour les utilisateurs, en utilisant des icônes, des boutons et d'autres éléments de conception familiers (système de navigation logique), avec des choix judicieux de couleurs, de polices et d'images pour renforcer la clarté et la lisibilité de l'interface.
- Contraintes ou difficultés techniques : Temps nécessaire pour trouver le juste équilibre entre simplicité et fonctionnalité mais aussi pour la conception et la mise en œuvre d'une telle interface.
- Énonciation des risques et parades : Risque de confusion pour les utilisateurs et bug de navigation.  
**Parade** : Il faudrait faire tester le logiciel par plusieurs utilisateurs et recueillir un feedback.
- Spécification des tests de contrôle : Tests unitaires.

3. Performance du système en temps de réponse, stockage mémoire

- Quantification : Temps de réponses en fonction des différentes actions (pour une requête utilisateur, pour le chargement d'une page).
- Éléments de faisabilité : Un essai peut être implémenté pour mesurer la performance actuelle du système et trouver des améliorations à faire.



- Contraintes ou difficultés techniques : Posséder un stockage suffisant pour contenir toutes les données enregistrées, réussir à élaborer des algorithmes pour exécuter les requêtes en temps réel.
- Énonciation des risques et parades : Les risques peuvent être un stockage insuffisant en mémoire.  
**Parade** : Utilisation de technologies de stockage plus performantes.
- Spécification des tests de contrôle : Tests de performance pour mesurer les temps de réponse, tests de stockage pour vérifier que la mémoire est suffisante.

## 2.4 Diagrammes de cas d'utilisation

### 2.4.1 Créer un compte

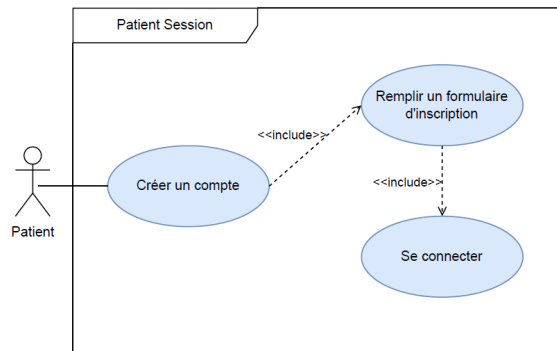


Figure 2.1 – Diagramme de cas d'utilisation : créer un compte

### 2.4.2 S'authentifier

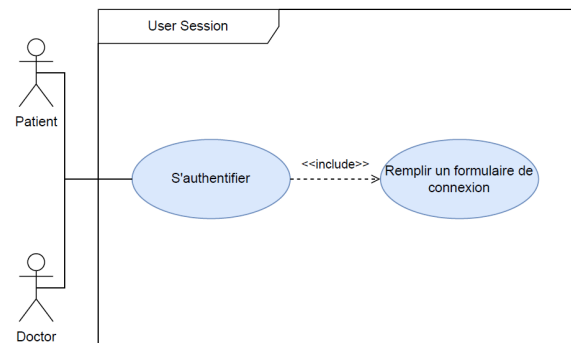


Figure 2.2 – Diagramme de cas d'utilisation : s'authentifier

### 2.4.3 Gérer les rendez-vous en ligne (Patient)

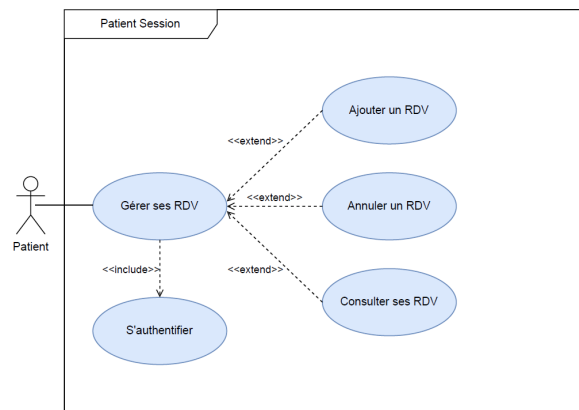


Figure 2.3 – Diagramme de cas d'utilisation : Gérer les rendez-vous en ligne (Patient)

### 2.4.4 Gérer les rendez-vous en ligne (Médecin)

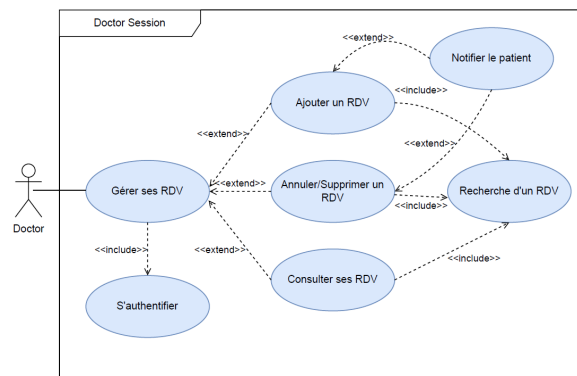


Figure 2.4 – Diagramme de cas d'utilisation : Gérer les rendez-vous en ligne (Médecin)

### 2.4.5 Consulter les rendez-vous

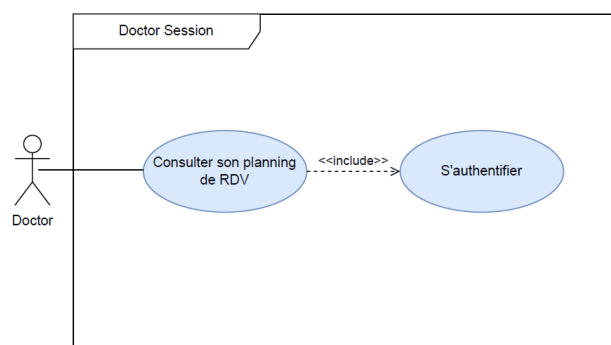


Figure 2.5 – Diagramme de cas d'utilisation : Consulter les rendez-vous

## 2.4.6 Consulter le dossier médical

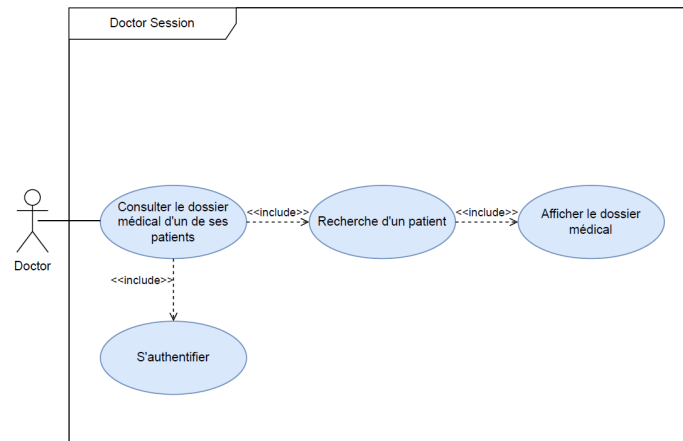


Figure 2.6 – Diagramme de cas d'utilisation : Consulter le dossier médical

## 2.4.7 Gérer le dossier médical

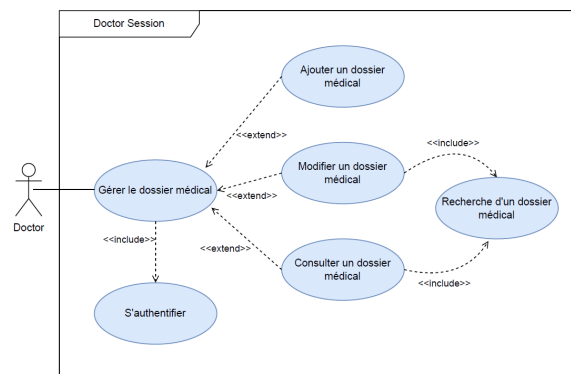


Figure 2.7 – Diagramme de cas d'utilisation : Gérer le dossier médical

## 2.4.8 Générer le bilan de santé

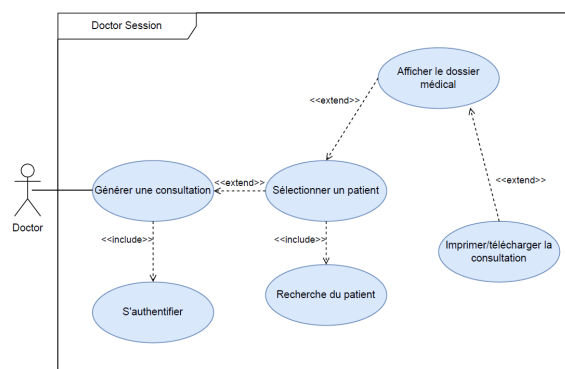


Figure 2.8 – Diagramme de cas d'utilisation : Générer le bilan de santé

# Chapitre 3

## Conception et Réalisation

L'architecture logicielle de notre application web pour la gestion d'un cabinet médical se compose d'un front-end, d'un back-end et d'une base de données. Pour le back-end, nous avons décidé d'utiliser une architecture DDD (Domain-Driven Design) qui se compose de quatre couches distinctes : une couche domaine, une couche infrastructure, une couche application et une couche interface utilisateur.

Dans la couche domaine, on trouve les objets métiers qui représentent les éléments clés de notre application tels que les patients, les médecins, les rendez-vous, les créneaux horaires, les consultations, les adresses des patients et les prescriptions médicales. Nous avons distingué les "values object" des Entity et de notre Aggrégat, le dossier médical. Ainsi, nous pouvons recréer nos objets en les récupérant de la base de données ou alors sauvegarder ces objets. C'est ce qui nous permet de faire le pont entre la base de données et le frontend qui reçoit et met en forme les données.

Nous avons également mis en place une couche infrastructure en utilisant des bases de données relationnelles pour stocker les informations de l'application. Cette couche infrastructure est responsable de créer pour chaque classe du domaine une table et de gérer les requêtes et les mises à jour dans celles-ci.

Dans la couche interface utilisateur, on a des controllers qui servent d'intermédiaires entre la partie client et la partie serveur. Les controllers reçoivent les entrées utilisateurs, les envoient à la couche application et envoient une réponse au client. Les controllers renvoient en général une réponse HTTP afin de distinguer les différents code HTTP que l'on reçoit (ok, forbiden...)

Avec cette architecture DDD, nous avons créé une application structurée pour répondre aux besoins spécifiques au cahier des charges. Le code est ainsi plus facile à lire car les données sont structurées par couches.

Concernant le frontend, nous nous occupons ici de présenter les données, et c'est donc une partie de conception graphique et de présentation aux utilisateurs et non pas une partie de conception d'un modèle pour la modélisation du cabinet médical. Les données sont donc traitées dans le backend pour pouvoir être stockées dans la base de données.

### 3.1 Diagrammes de classe du Domain

Concernant le diagramme de classe, nous avons représenté seulement les classes du Domain car elle représente le coeur de l'architecture DDD.

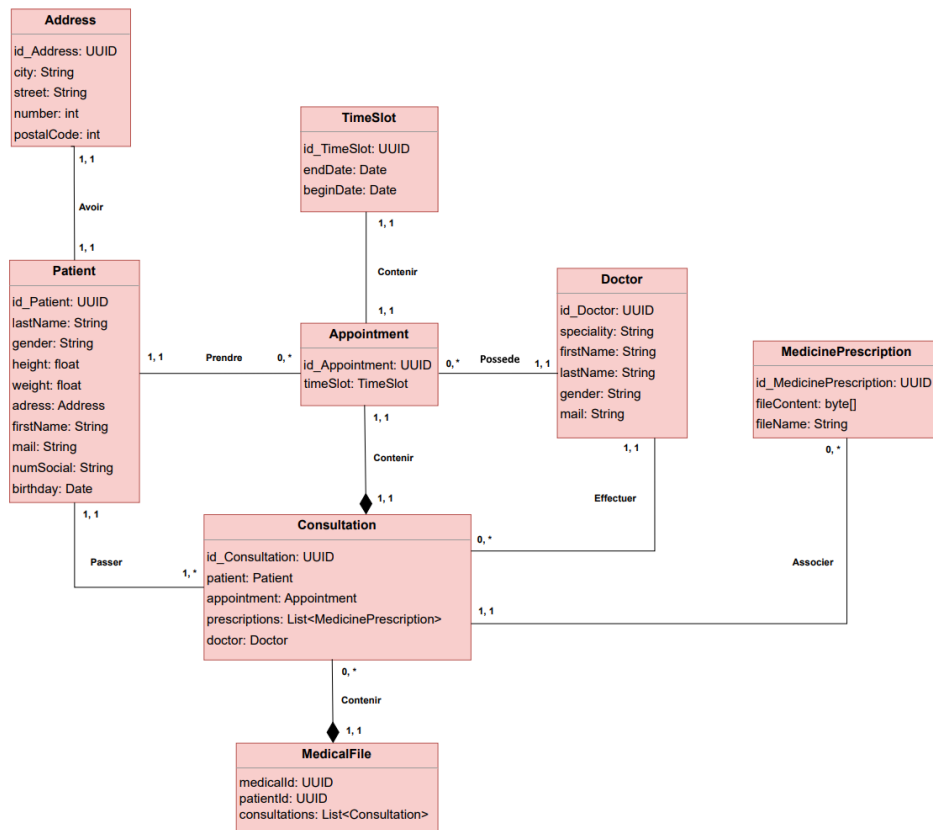


Figure 3.1 – Diagramme de classes

## 3.2 Passage au modèle relationnel

Le modèle relationnel est le modèle logique de données qui correspond à l'organisation des données dans les bases de données relationnelles. Un modèle relationnel est composé de relations, appelée table. Ces tables sont décrites par des attributs ou champs. Pour décrire une relation, on indique tout simplement son nom, suivi du nom de ses attributs entre parenthèses. L'identifiant d'une relation est composé d'un ou plusieurs attributs qui forment la clé primaire. Une relation peut faire référence à une autre en utilisant une clé étrangère, qui correspond à la clé primaire de la relation référencée.

### 3.2.1 Règles de passage du diagramme de classes au modèle logique

A partir du modèle de classes UML de notre application, nous avons obtenu notre modèle relationnel en appliquant les règles de passage suivantes :

R1 : chaque entité (classe) devient une relation ayant un identifiant comme clé primaire.

R2 : une association (X, n)..(X, 1) (association père fils) provoque la migration d'une clé primaire, l'identifiant coté (X, 1) migre vers l'entité coté (X, n) comme étant une clé étrangère.

R3 : une association 1..1 dans ce cas on choisit une classe et sa clé migre dans la seconde comme étant clé étrangère.

R4 : dans le cas de la composition la clé primaire du composé migre vers le composant et devient à la fois une clé primaire et étrangère.

### 3.2.2 Modèle logique de données

En appliquant les règles de passages pour notre diagramme de classes, nous aboutissons au schéma relationnel suivant :

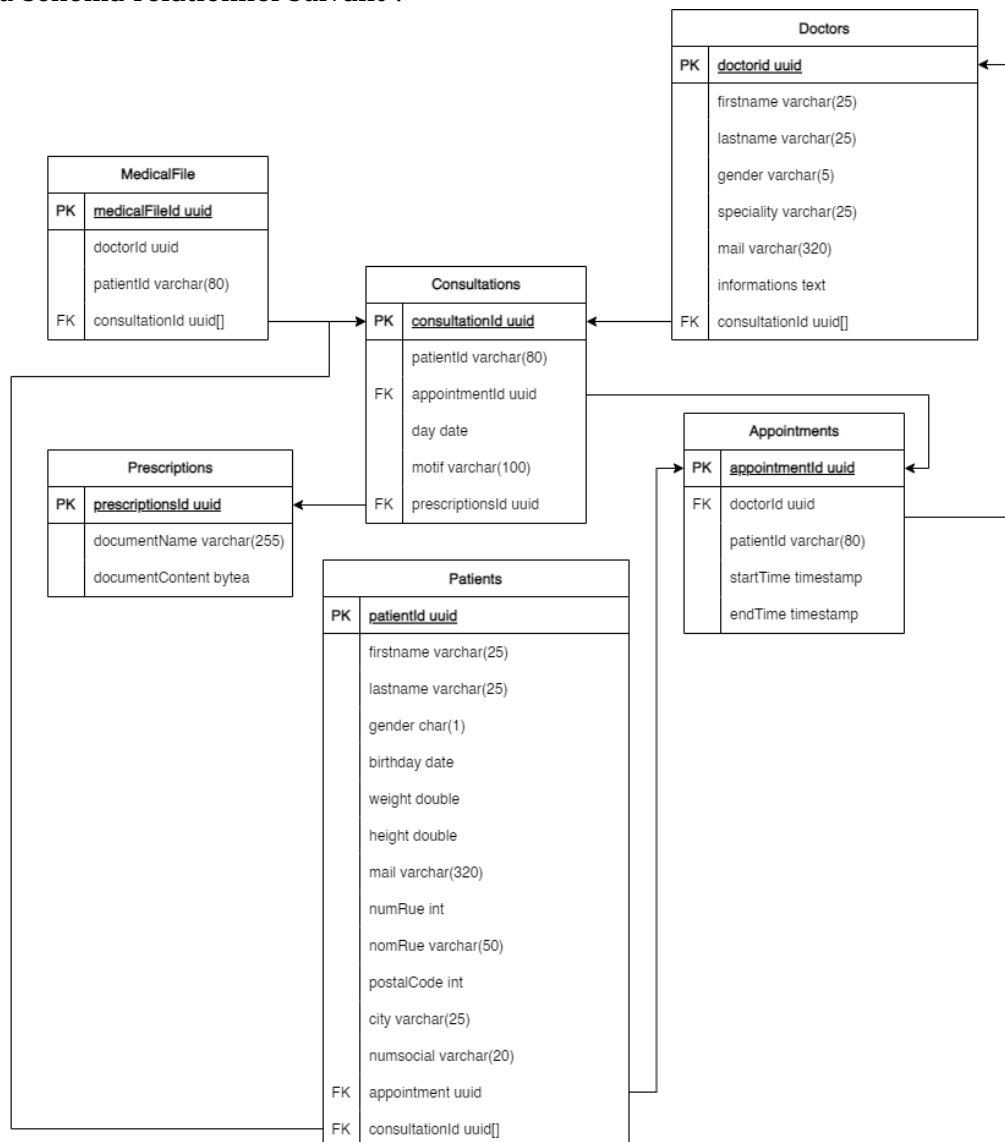


Figure 3.2 – Schéma Relationnel

## 3.3 Outils de développement

- **Java** : Nous avons utilisé le langage de programmation orienté objet Java pour notre projet [Jav].
- **ReactJS** : Pour le côté Front-end, nous avons utilisé la bibliothèque JavaScript ReactJS qui est très populaire dans le monde du développement web et très facile d'utilisation. Elle fonctionne avec des composants où chaque composant représente une partie de l'interface utilisateur. Ces composants sont réutilisables ce qui nous permet de gagner du temps. Elle possède également une documentation bien organisée et facile à comprendre [Rea].

- Maven : Pour la construction du projet nous avons choisi Maven, un outil de gestion de projet facile à configurer et à utiliser. Il peut être étendu grâce à des plugins pour répondre à des besoins spécifiques. [Apa]
- Springboot : Pour créer notre application web, nous avons décidé d'utiliser Spring Boot qui est un framework Java qui fournit un ensemble de fonctionnalités prêtes à l'emploi pour simplifier la configuration, le déploiement et la gestion des applications web [Piv].
- Git : Pour pouvoir au mieux travailler sur ce projet chacun de notre côté, nous avons décidé d'utiliser Git qui est un logiciel de gestion de versions décentralisé. Il permet de travailler sur des parties différentes du code en même temps et il est très utile pour le travail en équipe [Git].
- Postgresql : Pour la gestion de notre base de données nous avons utilisé Postgresql qui est un système de gestion de bases de données relationnelles. Il est fiable et robuste, de plus nous avons déjà utilisé ce système donc c'était facile pour nous de l'utiliser pour notre projet [pos].
- JUnit : Pour les tests, nous avons utilisé JUnit qui est un framework de test unitaire pour Java et qui permet de tester des fragments de code de manière isolée et automatisée [JUn].

### 3.4 Côté back-end

Pour la partie back-end, nous avons entamé le processus par l'implémentation de la couche domaine en y incluant les protagonistes majeurs du cabinet médical : les docteurs et les patients. Nous avons poursuivi notre travail en créant des value objects et des entités tels que les adresses postales, les rendez-vous, les consultations, les ordonnances et les utilisateurs. Nous avons conclu en créant l'agrégat, qui est un dossier médical.

Ensuite, nous avons développé la couche infrastructure en créant quatre dépôts distincts : un pour les médecins, un pour les patients, un pour les utilisateurs et un pour le cabinet médical dans son ensemble. Ces dépôts contiennent les requêtes et les insertions dans les tables de la base de données.

Par la suite, nous avons travaillé sur la couche application, qui récupère les informations de la couche infrastructure afin de les transmettre à la couche interface utilisateur.

Enfin, nous avons implémenté la couche interface utilisateur, dans laquelle se trouvent des contrôleurs pour les patients, les médecins et les utilisateurs qui font le lien entre la partie serveur et la partie client.

### 3.5 Côté front-end

Le développement de l'interface utilisateur de notre application web a été effectué en utilisant React JS. Ce framework a permis de créer une interface utilisateur attrayante et réactive pour les utilisateurs.

Nous avons commencé par concevoir la page principale, qui comprend une liste déroulante permettant aux utilisateurs de sélectionner la spécialité du médecin recherché (Dentiste, Généraliste, etc.).

Ensuite, nous avons créé les pages d'inscription et de connexion pour permettre aux patients de se connecter à leur compte ou d'en créer un nouveau. Une partie d'administration a également été développée, avec un router dédié permettant d'accéder à toutes les pages disponibles pour les patients connectés, telles que la page "Mes rendez-vous", "Mes documents" et "Mon compte", où les patients peuvent modifier leurs informations personnelles. Un header a également été ajouté pour faciliter la navigation entre ces différentes pages.

Dans la partie publique de l'application, accessible à tous les utilisateurs, y compris ceux qui ne sont pas connectés, nous avons créé des pages dédiées à chaque médecin pour afficher les créneaux horaires disponibles. Nous avons ensuite créé une partie réservée uniquement aux médecins, où ils peuvent accéder à leur planning, visualiser leurs patients et modifier leurs informations personnelles.

Pour appliquer ceci, nous avons défini un router grâce au module React.Router permettant de gérer les routes de notre application web en fonction des URL qui sont appelées par l'utilisateur. En d'autres termes, le router permet de faire correspondre une URL donnée à un composant spécifique dans l'application. Cette méthode nous permet également de gérer les redirections et les transitions d'une page à l'autre, pour gérer l'historique de navigation et les paramètres d'URL, et pour autoriser ou refuser l'accès à certaines pages en fonction des autorisations de l'utilisateur.

## 3.6 Base de données

Nous avons créé une base de données à l'aide du système Postgresql dans laquelle nous avons ajouté des tables pour chaque classe de notre domaine DDD.

- **Table doctors** : on stocke les informations relatives aux docteurs comme leur nom, prénom, spécialité et les informations qu'on retrouve sur leur page Medicolib.
- **Table patients** : on stocke les informations personnelles des patients comme leur nom, prénom, genre, poids, taille, adresse mail et date de naissance.
- **Table address** : on stocke les différentes parties d'une adresse postale d'un patient tel que le numéro et nom de la rue, le code postal et le nom de la ville.
- **Table users** : on stocke les adresses mails et mots de passe cryptés des utilisateurs ainsi que leur rôle (patient ou doctor).
- **Table availableTimeSlots** : on stocke les créneaux de rendez-vous disponibles pour chaque médecin.
- **Table appointments** : on stocke tous les rendez-vous pris pour chaque médecin et chaque patient.
- **Table documents** : on stocke les documents que les patients peuvent ajouter en amont de leur rendez-vous.
- **Table consultations** : on stocke les informations d'une consultation entre un médecin et un patient et un identifiant d'ordonnance si il y en a une à la suite de la consultation.
- **Table prescriptions** : on stocke les ordonnances que le médecin prescrit au patient.



- **Table medicalFile** : on stocke les dossiers médicaux des patients pour chaque médecin, qui correspondent à une liste de leurs consultations.
- **Table price** : on stocke le prix des services proposés pour tous les médecins c'est-à-dire l'identifiant du docteur avec l'intitulé de la prestation et le prix.

# Chapitre 4

## Tests

Nous allons maintenant aborder la partie sur les tests. Nous avons utiliser JUnit pour effectuer des tests unitaires sur les contrôleurs. JUnit est un framework permettant de développer des tests unitaires automatisables. Ainsi, nous pouvons nous assurer que le code répond toujours aux besoins après avoir implémenté ou effectué des modifications. Ces tests correspondent à des assertions qui permettent de tester les résultats attendus. L'objectif ici, est de tester les méthodes des différents contrôleurs que nous avons implémentés pour vérifier que le code que nous avons produit est bien correct. Les tests sont séparés du code de la classe permettant ainsi de la tester. Le principe d'implémentation de ces tests est le suivant :

- Création d'une instance du contrôleur et de tout autre objet nécessaire aux tests
- Appel de la méthode à tester
- Comparaison du résultat attendu avec le résultat que l'on vient d'obtenir

L'avantage est de pouvoir écrire un certain nombre de tests permettant de tester chacun différents cas de figure et de trouver le plus de bugs possibles dans notre application. Nous utilisons également, des objets Mock (du framework Mockito). MockMvc est en fait un objet de Spring [Spr21] permettant de simuler un objet qu'on aura besoin d'utiliser pour effectuer nos tests. Cela permet notamment, de simuler des accès à la base de données. Tout cela nous permet ainsi de rédiger nos tests sur les contrôleurs. Nous pouvons ainsi vérifier que les requêtes effectuées et les objets retournés et/ou intégrés dans la base de données sont conformes au résultat attendu. Nous avons trois contrôleurs donc trois classes de tests correspondantes.

### 4.0.1 UserControllerTest

Cette section correspond aux tests sur toute la partie "connexion" de notre site web. Plus précisément, nous testons ici que les connexions et réinitialisation de mot de passe s'effectuent bien. Nous avons effectué des tests positifs et négatifs, c'est-à-dire avec des bons identifiants de connexion et des mauvais et nous nous assurons que les réponses HTTP renvoyées sont bien identiques. Par exemple, si un utilisateur se connecte avec un mauvais mot de passe, il faut que la réponse attendue soit "Forbidden". Ainsi, si un des tests ne passent pas, il est simple de savoir où l'erreur se produit. Voici un exemple d'un de nos tests pour vérifier qu'un utilisateur ne peut pas réinitialiser son mot de passe s'il n'est pas enregistré en tant qu'utilisateur (qu'il n'a pas de compte).

```
@Test
public void testResetPasswordWithNonUser() throws Exception {
    Map<String, String> map = new HashMap<>();
    map.put("mail", "test@gmail.com");

    when(medicalPractice.checkUserExist("test@gmail.com")).thenReturn(false);
```

```

mockMvc.perform(post("/new-password")
    .contentType(MediaType.APPLICATION_JSON)
    .content(asJsonString(map)))
    .andExpect(status().isNotFound());
}

```

#### 4.0.2 PatientControllerTest

Cette section va correspondre aux tests du controleur du patient et va comme précédemment, tester les différentes fonctionnalités de l'API Rest (avec les codes HTTP). Nous testons ainsi toutes les demandes liées aux patients : modifications d'informations, prises et annulations de RDV, ajout de documents, etc. Comme précédemment, nous avons, pour certaines méthodes, tester plusieurs cas, lorsque la méthode en question peut retourner une réponse "ok" ou "forbidden". Voici un exemple d'un de nos tests pour vérifier qu'un utilisateur peut supprimer un document :

```

@Test
void testDeleteDocument() throws SQLException, IOException {
    HashMap<String, String> mockMap = new HashMap<>();
    mockMap.put("apptid", "94b9b73a-f561-4ca9-a6c3-6ae7e0361773");
    mockMap.put("mail", "johndoe@gmail.com");
    MockMultipartFile file = new MockMultipartFile("test", "test.txt", "text/plain", "file content".getBytes());
    patientController.uploadFile(file, mockMap.get("mail"), mockMap.get("apptid"));
    HashMap<String, String> map = new HashMap<>();
    map.put("id", "94b9b73a-f561-4ca9-a6c3-6ae7e0361773");
    map.put("name", file.getName());
    ResponseEntity<String> response = patientController.deleteDocument(map);
    assertEquals(HttpStatus.OK, response.getStatusCode());
    assertEquals("File deleted successfully", response.getBody());
}

```

#### 4.0.3 DoctorControllerTest

Cette section va correspondre aux tests du controleur concernant les docteurs. Nous allons, comme précédemment effectuer des tests sur les différentes méthodes du contrôleur et plus précisément sur les points suivants : la consultation du planning par le docteur, l'ajout de consultations, l'annulation de RDV etc. Voici un exemple de test pour récupérer la liste des patients d'un docteur :

```

@Test
void testGetAllPatientByDoctor() throws ParseException, SQLException {
    SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
    Date date = format.parse("2023-04-03");
    List<Patient> expectedPatients = new ArrayList<>();
    Patient patient = new Patient(
        UUID.fromString("dccf9cfd-f2cc-4e44-8357-dd4140e17b73"),
        "John", "Doe", "M", date, "54165", "johndoe@gmail.com", null, 0, 0);
    expectedPatients.add(patient);
    Map<String, String> requestBody = new HashMap<>();
    requestBody.put("mail", "doctor@gmail.com");
    when(medicalPractice.getPatientsByDoctor(medicalPractice.
        getInformationsDoctorByMail("doctor@gmail.com"))).thenReturn(
        expectedPatients);
    ResponseEntity<List<Patient>> response = doctorController.
        getAllPatientByDoctor(requestBody);
}

```

```
    assertEquals(HttpStatus.OK, response.getStatusCode());  
    assertEquals(expectedPatients, response.getBody());  
}
```

Pour conclure cette partie sur les tests, nous pouvons dire que les tests représentent une partie très importante du projet car elles nous permettent de suivre l'évolution de notre application et de détecter des bugs rapidement et lors de l'ajout de nouvelles fonctionnalités. L'avantage de JUnit est que c'est très simple à utiliser, et très rapide à mettre en place et cela nous permet d'écrire beaucoup de tests permettant de couvrir une grande partie de notre programme.

# Chapitre 5

## Résultats

La partie résultats de ce rapport présente les conclusions de notre projet. Dans cette section, nous présentons les fonctionnalités de l'application (les besoins fonctionnels et non fonctionnels), son interface utilisateur, sa sécurité et sa facilité d'utilisation. Nous avons fait au mieux pour établir une mise en relation simple et efficace entre patient et professionnel de santé. Notamment grâce à une prise de rendez-vous simple et intuitive selon les besoins des patients mais aussi permettre aux professionnels de santé de pouvoir suivre les dossiers médicaux de chacun de leurs patients et avoir une vue générale sur l'ensemble de leurs rendez-vous sur un planning. Les informations présentées dans cette section ont été obtenues à partir d'une étude approfondie des besoins, d'une analyse comparative d'autres applications similaires et d'une phase de conception et de développement rigoureuses. Nous sommes convaincus que cette application représente une solution efficace et facile d'utilisation pour la gestion des cabinets médicaux et qu'elle répondra aux besoins spécifiques des professionnels de santé et de leurs patients. En effet, nous avons conceptualisé au mieux cette application en proposant un *responsive web design*, permettant à l'application de s'adapter à toutes les tailles de fenêtres.

### 5.1 Navigation en tant que *Visiteur* (Utilisateur non-connecté) :

#### 5.1.1 Fonctionnalité de l'entête (Header) :

- Cliquer sur se connecter pour être redirigé vers la page *Login*.

#### 5.1.2 Fonctionnalité de la page d'Accueil (Homepage) :

- Se renseigner sur la nature du site, les services qu'il propose et la qualité de ces services.
- Utiliser la barre de navigation :
  - Rechercher sans spécifier une spécialité médicale : redirection vers la page *Docteurs* en répertoriant l'ensemble des praticiens du cabinet, toute spécialité confondue.
  - Rechercher avec une spécialité choisie dans la liste déroulante (qui est à jour en direct, en affichant chaque spécialité disponible selon les médecins enregistrés dans la base de données), et ce par une redirection vers la page *Docteurs* répertoriant les praticiens correspondant à la spécialité recherchée.

### 5.1.3 Fonctionnalité de la page repertoriant les docteurs (Docteurs) :

- Cliquer sur le bouton prendre RDV pour accéder à la page *Docteur* (page personnelle du docteur).
- Naviguer sur le planning, en changeant de semaine pour trouver un jour et un créneau.
- Afficher toutes les horaires des jours de la semaine choisie.
- Prendre directement un rendez-vous sur le planning selon disponibilité (les créneaux déjà réservés ne sont pas affichés), en cliquant sur un créneau, l'utilisateur sera alors redirigé vers la page *Login* car il est impossible de réserver un créneau si l'utilisateur n'est pas connecté.

### 5.1.4 Fonctionnalité de la page personnel d'un médecin (Docteur) :

- Consulter les informations pratiques du docteur en question (tarifs, informations, contact).
- Naviguer sur le planning pour trouver un jour et un créneau.
- Prendre un rendez-vous sur le planning selon disponibilité (les créneaux déjà réservés seront affichés mais barrés et non cliquable), en cliquant sur un créneau, l'utilisateur sera alors redirigé vers la page *Login*.

### 5.1.5 Fonctionnalité de la page de connexion (Login) :

- Se connecter via son email et son mot de passe (que l'on peut masquer ou non).  
Remarque : le mot de passe est stocké de manière crypté dans la base de données grâce à un encodage de celui-ci, défini par sa valeur de hashage.
- Indiquer "mot de passe oublié", ce qui nous invite à entrer notre email pour recevoir un mail contenant un nouveau mot de passe aléatoire. Remarque : si le mail n'existe pas dans la base de données, alors aucun mail ne sera envoyé.
- Cliquer sur "Enregistrez-vous", ce qui redirigera l'utilisateur sur la page *Register*.

### 5.1.6 Fonctionnalité de la page de création de compte (Register) :

- Renseigner l'intégralité des informations demandées pour créer son compte en tant que patient (les professionnels sont ajoutés par le biais des administrateurs), puis se connecter.

## 5.2 Navigation en tant que *Patient* (connecté) :

### 5.2.1 Fonctionnalité de l'entête (Header) :

- Cliquer sur "Mes rendez-vous" pour être redirigé vers la page *Appointments*.
- Cliquer sur "Mes documents" pour être redirigé vers la page *Documents*.
- Cliquer sur le prénom pour faire apparaître une liste déroulante :
  - Cliquer sur "Mon compte" pour être redirigé vers la page *Edit*.
  - Cliquer sur "Déconnexion" pour être déconnecté et fermer la session.
- Si la taille de la page est réduite, tous les boutons du header se retrouvent dans la liste déroulante en cliquant sur son prénom.

### 5.2.2 Fonctionnalité de la page d'Accueil (Homepage) :

- Décrite dans la partie 6.1.2.

### 5.2.3 Fonctionnalité de la page repertoriant les docteurs (Docteurs) :

- Décrite dans la partie 6.1.3.
- Cependant il est désormais possible de réserver un créneau car le patient est connecté, il sera alors redirigé vers la page *Appointments*. Remarque : le patient recevra un mail lui indiquant que son rendez-vous à la date X et l'horaire Y a été réservé.

### 5.2.4 Fonctionnalité de la page personnel d'un médecin (Docteur) :

- Décrite dans la partie 6.1.4.
- Cependant il est désormais possible de réserver un créneau car le patient est connecté, il sera alors redirigé vers la page *Appointments*. Remarque : le patient recevra un mail lui indiquant que le rendez-vous à la date X et l'horaire Y a été réservé.

### 5.2.5 Fonctionnalité de la page des rendez-vous (Appointments) :

- Affiche les rendez vous à venir pris par le patient, si aucun rendez vous n'est planifié alors un message sera affiché pour indiquer au patient qu'aucun rendez vous n'est planifié pour le moment.
- Possibilité de cliquer sur "voir mes rendez-vous passés" pour afficher les rendez-vous passés, si aucun rendez-vous n'est passé, de même un message sera présent pour prévenir le patient. Remarque : au bout d'un an les rendez vous passés ainsi que les documents qu'ils comportent seront effacé.
- Ajouter des documents pour un rendez-vous, ils seront alors répertoriés sur la page *Documents*. Remarque : les documents ajoutés seront visibles uniquement par le médecin indiqué sur le rendez-vous.
- Annuler un rendez vous, par conséquent le créneau disparaîtra de la page, le rendez-vous à nouveau disponible pour d'autres patients. Remarque : un mail sera alors envoyé indiquant que le rendez vous à la date X et l'horaire Y a été annulé.

### 5.2.6 Fonctionnalité de la page des documents (Documents) :

- Affiche tous les documents déposés par le patient avec la date de la consultation concernée, avec possibilité de télécharger le document ou de le retirer (ce qui va le retirer pour le praticien également).
- Affiche les ordonnances prescrite par le praticien avec la date de la consultation concernée, il est possible de la télécharger sous le format pdf.
- Si aucun document n'est disponible alors un message sera affiché pour indiquer au patient qu'aucun document n'est disponible pour le moment.

### 5.2.7 Fonctionnalité de la page du compte (Edit) :

- Modifier les informations personnelles : prénom, nom, numéro de sécurité sociale et adresse postale. Remarque : le prénom et nom sera parsé sous la forme Xxxxx et Xxxxx permettant une bonne lisibilité des informations. De plus, si le patient souhaite modifier son adresse et que celle-ci n'est pas complète, elle ne sera pas prise en compte, en laissant l'ancienne adresse.
- Modifier les informations médicale : sexe, date de naissance, poids et taille. Remarque : il n'est pas possible de rentrer une date de naissance supérieure à la date actuelle.
- Modifier les informations de connexion : mail et mot de passe.

- Enregistrer les modifications, une notification apparaîtra alors sur l'écran à la suite de la requête axios pour indiquer la bonne prise en compte des modifications.
- Supprimer le compte, un pop-up apparaîtra demandant une confirmation, à la suite de cette confirmation le patient sera déconnecté et le compte supprimé.

## 5.3 Navigation en tant que *Docteur* (connecté) :

### 5.3.1 Fonctionnalité de l'entête (Header) :

- Cliquer sur "Planning" pour être redirigé vers la page *Planning*.
- Cliquer sur "Mes patients" pour être redirigé vers la page *Patients*.
- Cliquer sur le prénom pour faire apparaître une liste déroulante :
  - Cliquer sur "Mon compte" pour être redirigé vers la page *Edit*.
  - Cliquer sur "Déconnexion" pour être déconnecté et fermer la session.
- Si la taille de la page est réduite, tous les boutons du header se retrouvent dans la liste déroulante en cliquant sur son prénom.

### 5.3.2 Fonctionnalité de la page répertoriant les RDV (Planning) :

- Se déplacer de semaine en semaine et visualiser tous les jours de la semaine sélectionnée.
- Pouvoir revenir au jour actuel en cliquant sur un bouton "Aujourd'hui" pour faciliter la navigation.
- Cliquer sur un RDV apparaissant sur le planning (dont on peut prévisualiser le nom, prénom du patient et l'heure du RDV) et ouverture d'un popup avec différentes fonctionnalités :
  - Affichage des informations (Nom et prénom du patient, heure et date).
  - Voir les documents ajoutés par le patient (s'il en a ajouté sinon un message indiquant qu'il n'y a pas de documents apparaît) et pouvoir les télécharger.
  - Cliquer sur un bouton amenant directement au dossier médical du patient.
  - Cliquer sur un bouton permettant d'ajouter une consultation au patient au moment du RDV.
  - Pour les RDV qui ne sont pas encore passés, un bouton "Annuler" est disponible permettant ainsi au docteur de pouvoir annuler ce RDV. Une notification est envoyée au patient par mail.
  - Pouvoir cliquer sur le bouton pour fermer ce popup.

Le docteur peut voir ses RDV qui sont passés et ceux à venir dans le planning en se déplaçant de semaine en semaine. Ainsi, il peut retrouver des documents transmis par le patient même lorsqu'un RDV est passé.

### 5.3.3 Fonctionnalité de la page répertoriant les patients d'un docteur (Patients) :

Pour qu'un docteur puisse avoir des patients, il faut que ceux-ci prennent un RDV pour la première fois avec ce docteur. Un dossier médical entre le patient et le docteur est alors créé même si le patient ne reprend plus de RDV après.

- Si le docteur n'a jamais eu de patients, la page sera vide. Si des patients ont déjà pris RDV avec lui, un tableau de patients est disponible :



- Affichage du nom, prénom, date de naissance, sexe et mail de chaque patient.
- Possibilité d'accéder au *dossier médical* d'un patient.
- Les patients sont affichés 10 par 10, donc on peut accéder aux pages suivantes si besoin.

#### 5.3.4 Fonctionnalité de la page de consultation du dossier médical d'un patient :

- En haut à gauche : affichage du numéro du dossier médical.
- Sur la partie gauche de la page, affichage des informations personnelles du patient : Nom, prénom, sexe, date de naissance, taille, poids, numéro de sécurité sociale, son adresse et son mail. Pour les informations qui n'ont pas été transmises (le poids par exemple), "non renseigné" apparaît.
- Sur la partie droite de la page, affichage des consultations de ce patient :
  - Si le premier RDV du patient n'est pas encore passé, alors cette partie sera vide car aucune consultation n'aura encore été créée par le médecin.
  - Si le patient a déjà eu au moins un RDV, alors le docteur a normalement créé une consultation lors de ce RDV. On retrouve ainsi la date de chaque consultation, le motif de la venue du patient et les différentes ordonnances si le médecin en a généré (voir en détail *Consultations* ). Si des ordonnances sont donc présentes, le docteur peut les télécharger. Notons que le docteur ne peut consulter que les consultations qui lui sont liées et non pas les consultations des autres docteurs avec le même patient.

#### 5.3.5 Fonctionnalité de la page d'ajout d'une consultation :

- Lors du RDV, le docteur ajoute une consultation au patient :
  - Il peut ajouter la date de la consultation. Par défaut, c'est la date actuelle qui est mise.
  - Il peut ajouter le motif de la consultation.
  - Il peut ensuite ajouter une ordonnance ou non. S'il n'ajoute pas d'ordonnance, dans le cas où le patient n'a aucun traitement à prendre et qu'il enregistre la consultation, celle-ci s'ajoute dans la section du dossier médical du patient. S'il ajoute une ordonnance, des cases de texte apparaissent pour ajouter les différentes prescriptions (de médicaments par exemple). On peut ajouter autant de prescriptions que l'on veut. Lorsqu'on enregistre, la consultation est créée avec l'ordonnance qui est générée sous forme d'un fichier pdf. Ce fichier contient les informations du docteur (Nom, l'adresse du cabinet, etc), ainsi que les différentes prescriptions. Ce pdf est également transmis au patient.

#### 5.3.6 Fonctionnalité de la page du compte (Edit) :

Le docteur peut modifier ses différentes informations :

- Dans la partie en haut à gauche, il peut modifier ses informations personnelles : nom, prénom et sexe. Pour pouvoir modifier sa spécialité, il doit contacter les administrateurs du site (au survol de la case, un message apparaît pour le préciser). En cliquant sur le bouton "enregistrer", les informations sont modifiées et un message de confirmation apparaît.
- Dans la partie en bas à gauche, il peut modifier ses informations de connexion : adresse email et mot de passe. Comme précédemment, lorsqu'il clique sur le bouton "enregistrer", ses informations de connexion sont modifiées et un message de confirmation apparaît.
- Dans la partie droite, il peut modifier ses informations professionnelles. Il peut ainsi, modifier ses différents tarifs de consultation (ainsi que leur nom), en ajouter et/ou

en supprimer. Il peut également, modifier son texte disponible pour les patients au moment de la réservation du RDV. Comme précédemment, un bouton "enregister" est disponible.

## 5.4 Présentation de l'application

Nous présentons ici les différentes pages de notre site web.

### 5.4.1 Page d'accueil

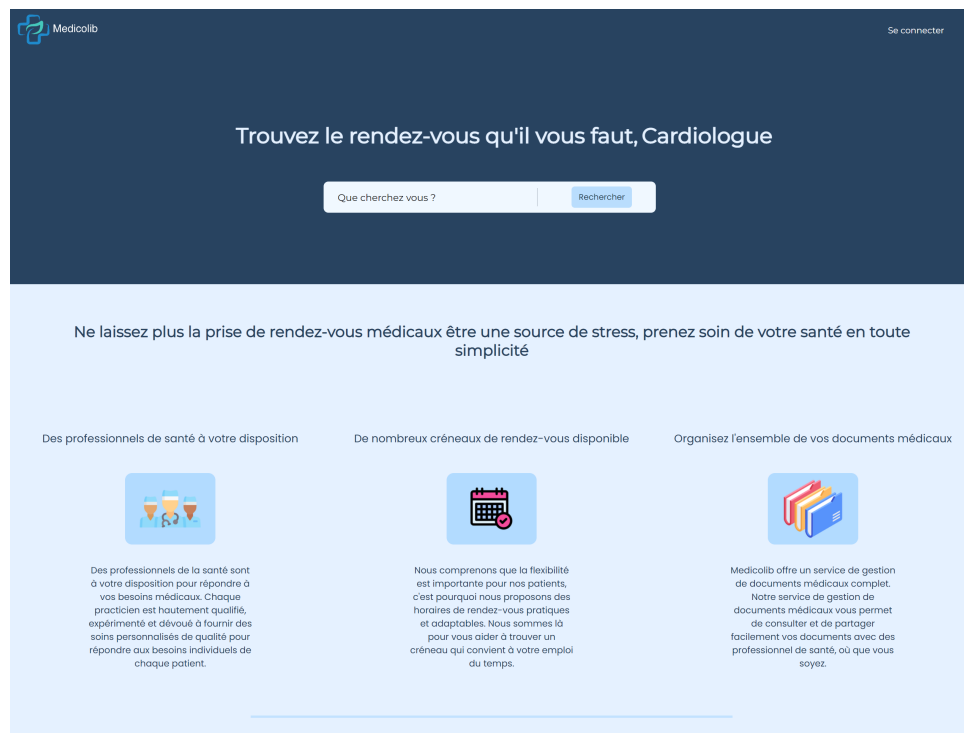
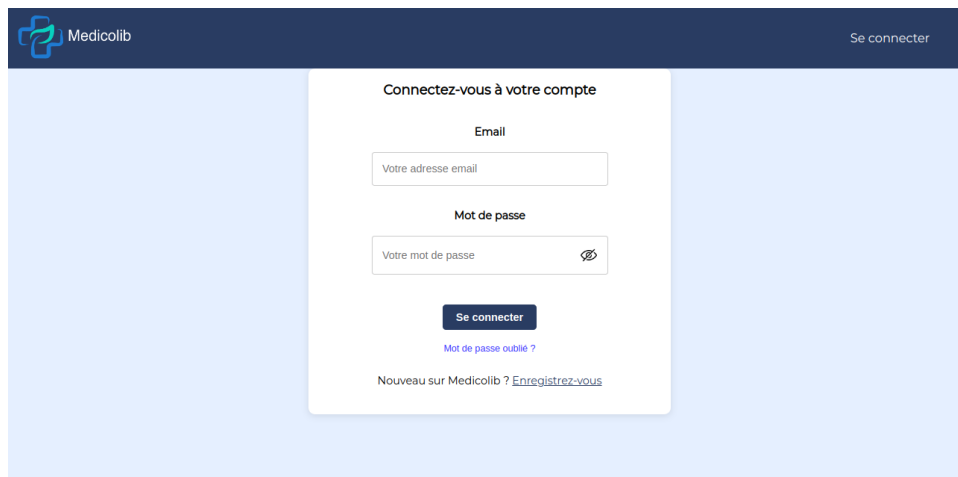


Figure 5.1 – Page d'accueil

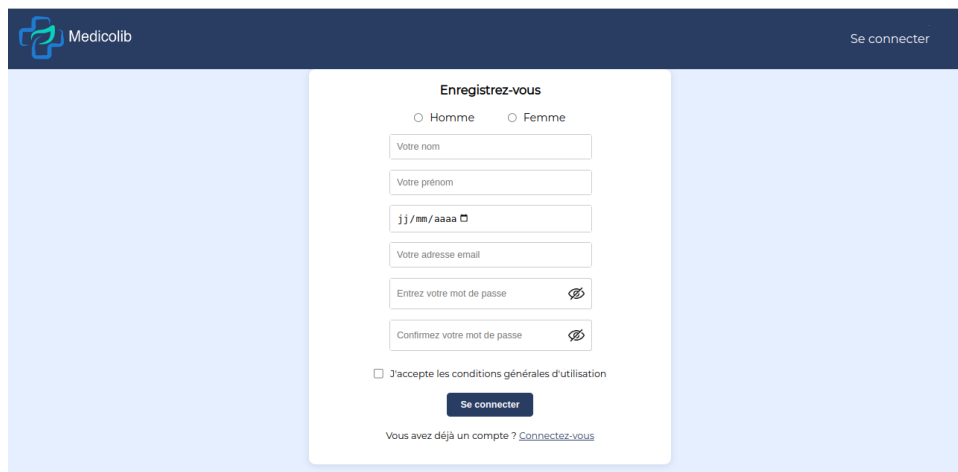
## 5.4.2 Login



The login form is titled "Connectez-vous à votre compte". It features a dark blue header with the Medicolib logo on the left and a "Se connecter" link on the right. The form itself is a white card with a light blue background. It contains two input fields: "Email" with the placeholder "Votre adresse email" and "Mot de passe" with the placeholder "Votre mot de passe" and an eye icon for toggling visibility. Below the password field is a "Se connecter" button. Underneath the button is a link "Mot de passe oublié ?". At the bottom of the card, it says "Nouveau sur Medicolib ? [Enregistrez-vous](#)".

Figure 5.2 – Login

## 5.4.3 Register



The registration form is titled "Enregistrez-vous". It features a dark blue header with the Medicolib logo on the left and a "Se connecter" link on the right. The form is a white card with a light blue background. It starts with gender selection: "Homme" and "Femme" with radio buttons. Below are input fields for "Votre nom", "Votre prénom", and a date field "jj/mm/aaaa" with a calendar icon. Then "Votre adresse email", "Entrez votre mot de passe" (with an eye icon), and "Confirmez votre mot de passe" (with an eye icon). Below these is a checkbox "J'accepte les conditions générales d'utilisation". A "Se connecter" button is at the bottom of the card. At the very bottom, it says "Vous avez déjà un compte ? [Connectez-vous](#)".

Figure 5.3 – Register

## 5.4.4 Patient Header



The patient header is a dark blue bar. On the left is the Medicolib logo. In the center, there are two links: "Mes rendez-vous" and "Mes documents". On the right, the user's name "Jennifer Ephrem" is displayed with a dropdown arrow. Below the name is a white button with a user icon and the text "Mon compte". Below that is another white button with a logout icon and the text "Déconnexion".

Figure 5.4 – Patient Header

## 5.4.5 Page des docteurs

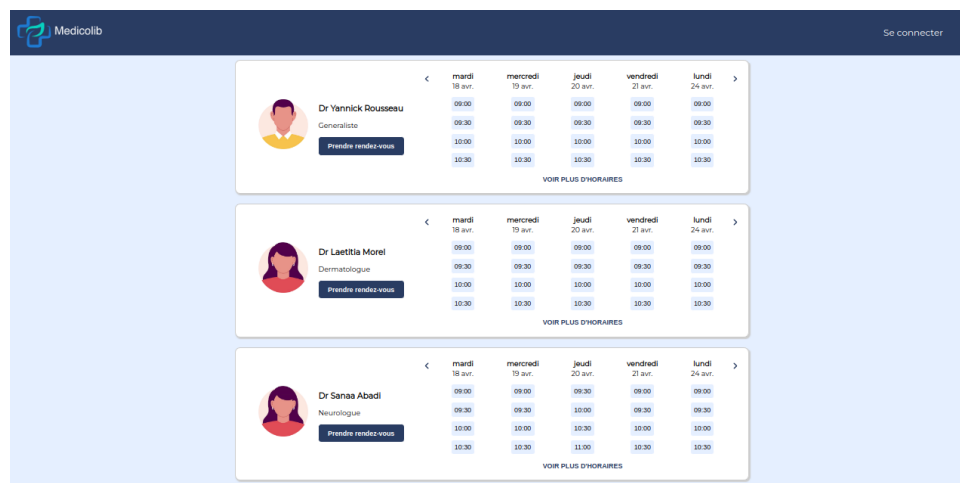


Figure 5.5 – Page des docteurs

## 5.4.6 Page de réservation d'un docteur

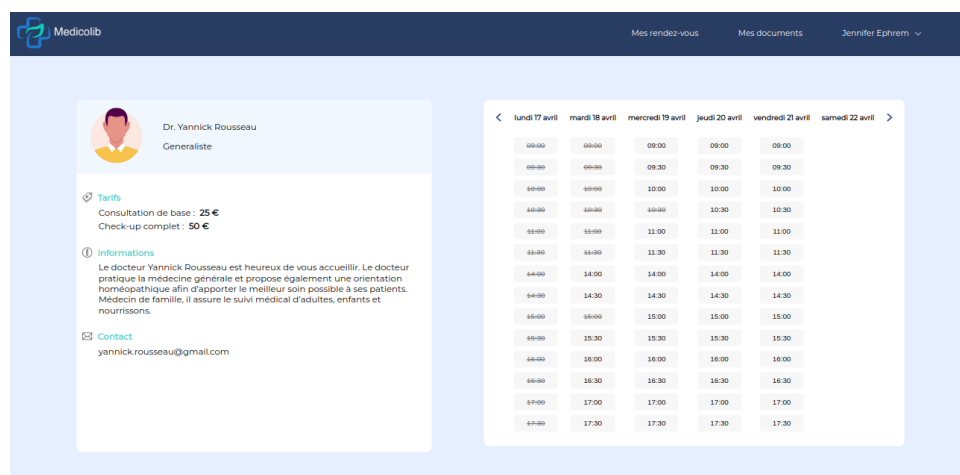


Figure 5.6 – Page de réservation d'un docteur

### 5.4.7 Page Mon Compte du patient

The screenshot shows the 'Mon Compte' page in the Medicolib application. The header is dark blue with the Medicolib logo on the left and navigation links 'Mes rendez-vous', 'Mes documents', and 'Jennifer Ephrem' on the right. The main content area is light blue and contains three white form boxes. The first box, 'Mes informations personnelles', has fields for 'Prénom' (Jennifer), 'Nom' (Ephrem), 'Numéro de sécurité sociale' (Votre numéro de sécurité soc), and a radio button for 'Adresse actuelle' (selected) and 'Nouvelle adresse' (Adresse non renseigné). The second box, 'Mes informations médicale', has radio buttons for 'Homme' and 'Femme' (selected), a 'Date de naissance' field (24/08/2001), and fields for 'Poids (en kg)' and 'Taille (en cm)' (both 0). The third box, 'Mes informations de connexion', has a 'Mail' field (ephrem.jennifer@gmail.com) and a 'Mot de passe' field (Votre nouveau mot de passe). A dark blue button 'Enregistrer les modifications' is below the forms, and a red button 'Supprimer le compte' is at the bottom right.

Figure 5.7 – Page Mon Compte du patient

### 5.4.8 Page RDV du patient (lorsqu'il n'y a pas de RDV)

The screenshot shows the 'RDV' page in the Medicolib application. The header is dark blue with the Medicolib logo on the left and navigation links 'Mes rendez-vous', 'Mes documents', and 'Jennifer Ephrem' on the right. The main content area is light blue. At the top, it says 'Rendez-vous à venir'. Below this, a message states: 'Vous n'avez aucun rendez-vous à venir pour le moment. Veuillez revenir ultérieurement ou prendre rendez-vous avec un professionnel.' Further down, there is a link 'VOIR MES RENDEZ-VOUS PASSÉS'.

Figure 5.8 – Page RDV du patient (lorsqu'il n'y a pas de RDV)

### 5.4.9 Page RDV du patient

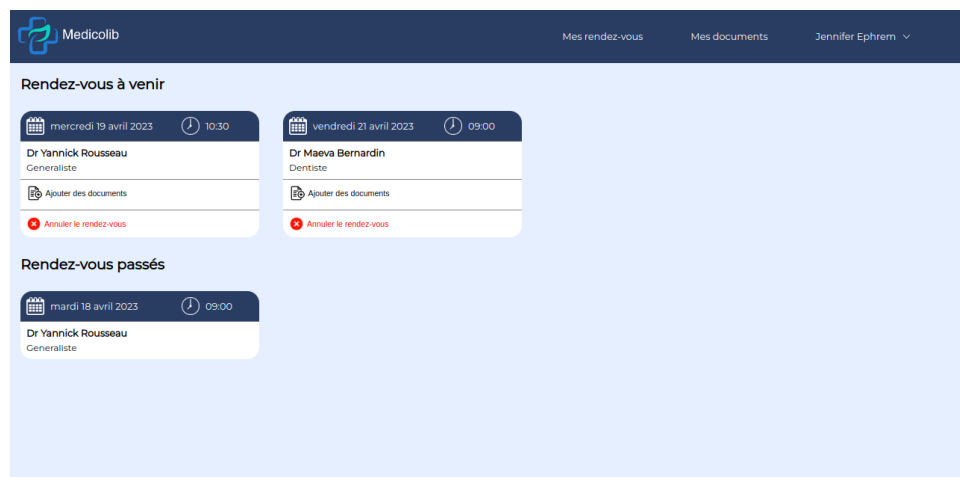


Figure 5.9 – Page RDV du patient

### 5.4.10 Confirmation de suppression d'un RDV

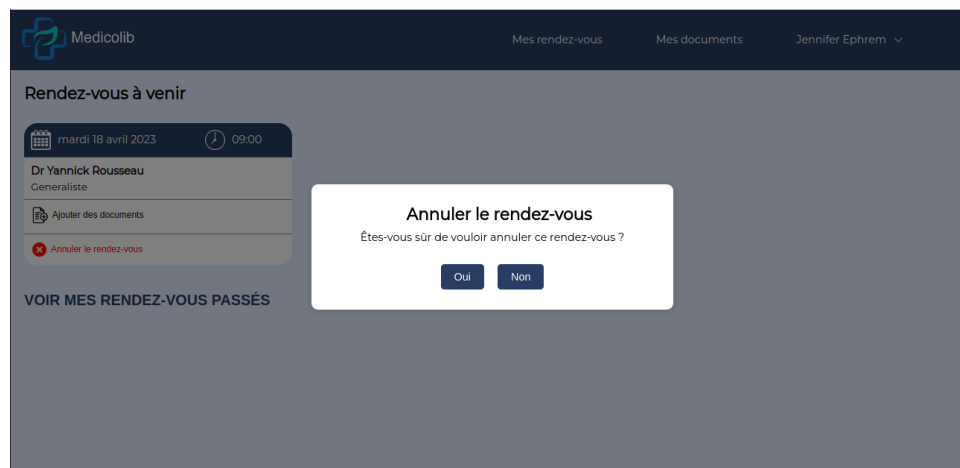


Figure 5.10 – Confirmation de suppression d'un RDV

#### 5.4.11 Page Documents du patient (sans documents)



Figure 5.11 – *Page Documents du patient (sans documents)*

#### 5.4.12 Page Documents du patient

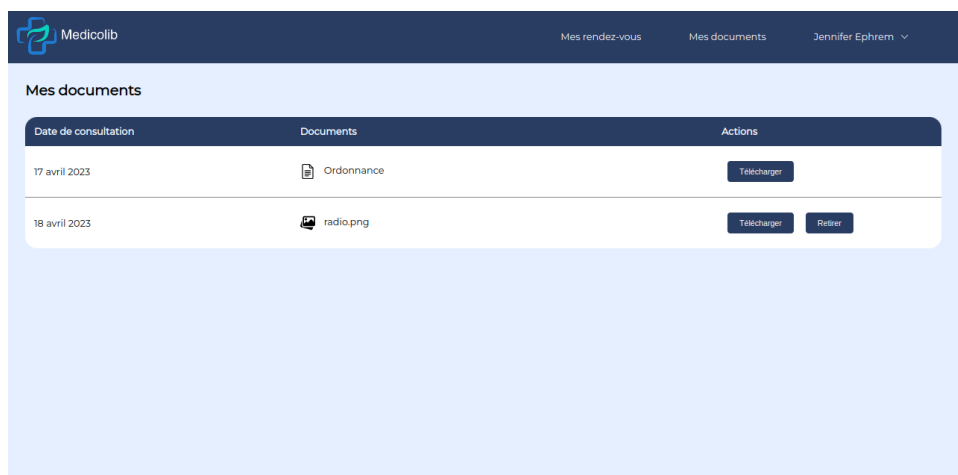


Figure 5.12 – *Page Documents du patient*

#### 5.4.13 Doctor Header



Figure 5.13 – *Doctor Header*

#### 5.4.14 Doctor Planning

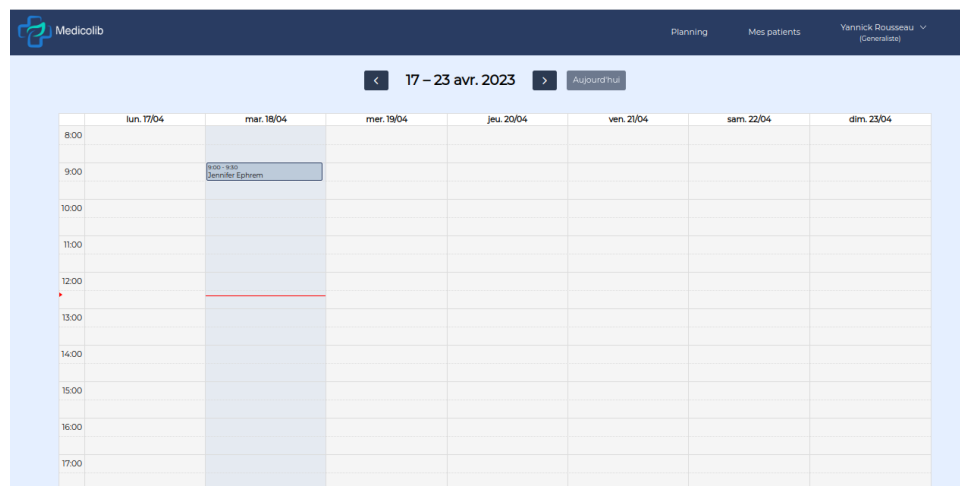


Figure 5.14 – Doctor Planning

#### 5.4.15 RDV sur le planning du docteur

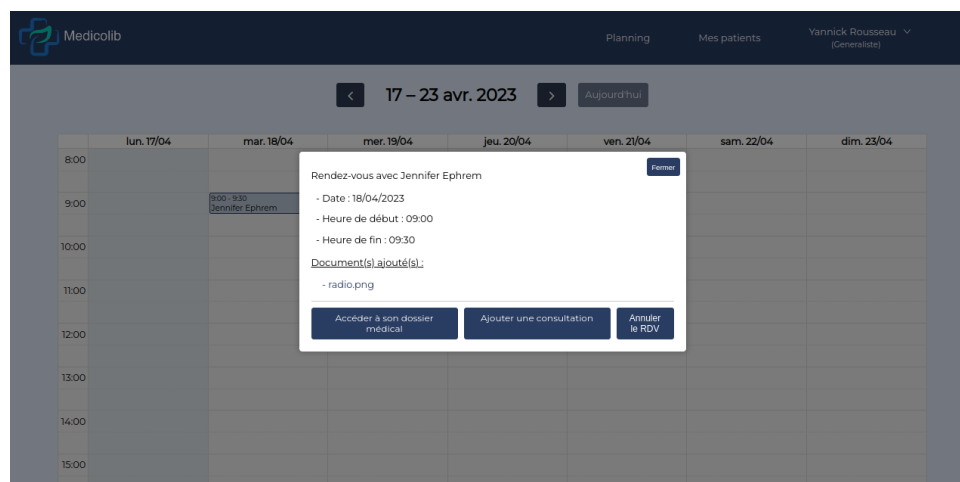


Figure 5.15 – RDV sur le planning du docteur

#### 5.4.16 Ajout d'une consultation sans prescription médicale

Figure 5.16 – Ajout d'une consultation sans prescription médicale



#### 5.4.17 Ajout d'une consultation avec prescription médicale

The screenshot shows the 'Ajouter une consultation' form in the Medicolib application. The form is centered on a light blue background. At the top, the header bar includes the Medicolib logo, 'Planning', 'Mes patients', and the user 'Yannick Rousseau (Generaliste)'. The form fields are as follows: 'Date' with a calendar icon showing '17/04/2023'; 'Motif de la consultation' with a dropdown menu showing 'maladie'; 'Ajouter une ordonnance' with radio buttons for 'Oui' (selected) and 'Non'; 'Prescription 1:' with a text area containing 'Doliprane : 3 par jour pendant 5 jours' and '+' and '-' buttons; and an 'Enregistrer' button at the bottom. A link 'Télécharger le PDF' is located below the 'Enregistrer' button.

Figure 5.17 – Ajout d'une consultation avec prescription médicale

#### 5.4.18 Prescription médicale

The screenshot shows a medical prescription document. At the top left, it reads 'Docteur Yannick Rousseau' followed by 'Generaliste'. Below this, the address 'Medicolib 351 Cours de la Libération - 33400 Talence' and email 'yannick.rousseau@gmail.com' are listed. On the right side, the patient's name 'Jennifer Ephrem' and the date 'Talence, le 17/04/2023' are printed. At the bottom left, the prescription is written as '- DOLIPRANE : 3 PAR JOUR PENDANT 5 JOURS'.

Figure 5.18 – Prescription médicale

### 5.4.19 Page Mon Compte du docteur

**Mes informations personnelles**

Prénom: Yannick, Nom: Rousseau, Spécialité: Generaliste, Homme/Femme: Homme

**Mes informations de connexion**

Mail: yannick.rousseau@gmail.com, Mot de passe: Nouveau mot de passe

**Mes informations professionnelles**

Vos tarifs appliqués:

Nom de la consultation	Tarifs
Consultation de base	25 €
Check-up complet	50 €

Ajouter un nouveau tarif

Informations affichées aux patients lors de la réservation:

Le docteur Yannick Rousseau est heureux de vous accueillir. Le docteur pratique la médecine générale et propose également une orientation homéopathique afin d'apporter le meilleur soin possible à ses patients.

Figure 5.19 – Page Mon Compte du docteur

### 5.4.20 Envoie d'un mail

Confirmation de prise de rendez-vous du 19 avril 2023

noreply.medicalpractice@gmail.com

Nous vous confirmons le rendez-vous du 19 avril 2023 à 10:30. Nous vous remercions de bien vouloir annuler votre rendez-vous si vous ne pouvez pas l'assurer. Si vous avez des documents à transmettre à votre docteur, n'hésitez pas à les lui faire suivre depuis votre compte, dans la section mes rendez-vous. A très bientôt, votre cabinet médical Medicolib.

Répondre Transférer

Figure 5.20 – Envoie d'un mail

# Chapitre 6

## Conclusion

En conclusion, nous avons développé une application web pour un cabinet médical complète qui répond aux besoins spécifiques de ce secteur d'activité. Cette application permet aux médecins et aux patients de gagner du temps et de simplifier les processus de gestion des rendez-vous et des consultations médicales.

Au fil de notre projet, nous avons instauré une solution de gestion de rendez-vous à la fois facile et rapide pour nos patients. Ils peuvent désormais aisément planifier, annuler ou encore joindre des documents à leur rendez-vous. Les médecins, quant à eux, peuvent accéder aux dossiers médicaux et documents de leurs patients afin de leur prodiguer les soins les plus adaptés à leur situation. Ils bénéficient également d'une interface intuitive et conviviale pour visualiser leur emploi du temps, ainsi que la possibilité d'annuler un rendez-vous en cas d'indisponibilité. En un simple clic, l'ajout d'une consultation se fait aisément, avec la possibilité de télécharger une ordonnance en cas de prescription. Enfin, pour une gestion personnalisée, tous nos utilisateurs ont la possibilité de modifier leurs informations personnelles depuis l'onglet "Mon compte".

Cependant, bien que notre application soit une réussite, nous sommes conscients qu'il y a toujours place à l'amélioration. Parmi les perspectives d'amélioration, nous souhaitons renforcer la sécurité de nos utilisateurs en cryptant l'ensemble des données sensibles, notamment les dossiers médicaux, et non pas simplement les mots de passe. Nous voulons ainsi assurer la protection maximale des informations médicales de nos patients. Nous souhaitons également mettre en place un système de téléconsultation pour que les patients puissent bénéficier des soins nécessaires directement depuis chez eux. Enfin, dans le contexte de notre application web, notre objectif serait de réduire au maximum les risques d'injections SQL dans notre base de données afin d'assurer la fiabilité et la confidentialité des données.

Dans l'ensemble, nous sommes fiers de ce que nous avons accompli avec cette application web pour un cabinet médical. Nous sommes convaincus que notre solution répondra aux besoins des médecins et des patients, et nous sommes impatients de recevoir les commentaires et les suggestions de nos utilisateurs pour continuer à améliorer notre application.

# Bibliographie

- [Apa] Apache maven. <https://maven.apache.org/>.
- [Git] Git - documentation. <https://git-scm.com/doc>.
- [Jav] Java se 17 jdk 17 documentation. <https://docs.oracle.com/en/java/javase/17/docs/api/index.html>.
- [JUn] Junit 5 user guide. <https://junit.org/junit5/docs/current/user-guide/>.
- [Piv] Pivotal. Spring Boot. <https://spring.io/projects/spring-boot>.
- [pos] PostgreSQL documentation. <https://www.postgresql.org/docs/>.
- [Rea] Reactjs. <https://reactjs.org/>.
- [Spr21] Spring. Testing the web layer - spring boot, 2021.