

Remerciements

Je tiens, au terme de ce travail, à présenter mes vifs remerciements à toutes les personnes qui ont contribué, de près ou loin, à son bon déroulement.

Mesdames **Selma BELGACEM** et **Leila HELALI** qui m'ont guidé tout au long de la période de stage, pour leur encadrement, leur suivi et pour l'aide qu'elles m'ont prodigué durant ce rapport.

Monsieur **Haythem SOUISSI**, pour son encadrement au sein de l'entreprise **LORDROID**, ainsi que pour son suivi et son encouragement tout au long de ce travail.

L'équipe de **LORDROID**, pour l'expérience enrichissante au sein de l'entreprise.

Tout qui a collaboré de près ou de loin à l'élaboration de ce modeste travail. Je remercie également mes enseignants et les membres de jury qui ont accepté d'évaluer mon travail.

Dédicaces

Je dédie ce travail à :

Mes très chers parents qui sont toujours là pour moi tout au long de mes études et qui m'ont donné un magnifique modèle de labeur et de persévérence.

*Mes sœurs **Ousseyma**, **Ameni** et mon frère **Ahmed** pour leurs soutien moral.*

*Mon cher ami **Jihed** pour ses encourages.*

Que Dieu leur accorde la santé et la prospérité.

Résumé

Ce travail a été élaboré dans le cadre du projet de fin d'étude pour l'obtention du diplôme d'ingénieur en informatique qui à été réalisé au sein de la société LORDROID.

Ce projet consiste à concevoir et développer une application mobile intelligente pour permettre aux patients de bénéficier des consultations médicales en ligne, de prendre des rendez-vous en ligne, de localiser un médecin en se basant sur les coordonnées géographiques. Notre application donne aussi la possibilité d'estimer automatiquement la maladie à partir d'un ensemble de symptômes entrés par le patient à travers l'intégration d'un chatbot. Ce chatbot propose aussi le traitement médical adéquat.

Mots clés

Application mobile, Android, Consultation médicale en ligne, Chatbot, Clean Architecture, Dagger2, Kotlin

Table des matières

1 Cadre général du projet	8
1.1 Présentation de l'organisme d'accueil	9
1.2 Présentation du projet	10
1.2.1 Problématique	10
1.2.2 Solution proposée	11
1.2.3 Objectifs	11
1.3 Étude de l'existant	12
1.4 Processus de développement	13
1.4.1 Développement incrémental	13
1.4.2 Planning prévisionnel des tâches	13
2 Spécification des besoins	15
2.1 Identification des acteurs	16
2.2 Besoins Fonctionnels	17
2.3 Besoins non Fonctionnels	17
2.4 Diagrammes de cas d'utilisation	18
2.4.1 Diagramme de cas d'utilisation globale	19
2.4.2 Raffinement des cas d'utilisation	21
2.4.2.1 Raffinement du cas d'utilisation «S'inscrire» .	21
2.4.2.2 Raffinement du cas d'utilisation «Traiter un rendez-vous»	23
2.4.2.3 Raffinement du cas d'utilisation «Saisir symptômes»	25
2.4.2.4 Raffinement du cas d'utilisation «Estimer une maladie»	27
3 Conception	29
3.1 Architecture globale	30
3.1.1 Définition du patron Clean Architecture	30
3.1.2 Application du patron "Clean Architecture"	32
3.2 Vue dynamique de l'application	38

3.2.1	Diagramme de séquences détaillé du cas d'utilisation "s'inscrire"	38
3.2.2	Diagramme de séquence détaillé du cas d'utilisation "Traiter des rendez-vous"	39
3.2.3	Diagramme de séquences détaillé du cas d'utilisation "Estimer une maladie"	40
4	Réalisation	42
4.1	Spécification technique	43
4.1.1	Environnement matériel	43
4.1.2	Environnement logiciel	43
4.1.2.1	Outils logiciels	44
4.1.2.2	API et bibliothèques utilisée	48
4.1.2.3	Langage de développement	51
4.1.3	Diagramme de déploiement de l'application	51
4.2	Interfaces de l'application	53
4.2.1	Lancement de l'application	53
4.2.2	Authentification	54
4.2.3	Inscription	55
4.2.4	Interface d'accueil	56
4.2.5	Menu principal	57
4.2.5.1	Profile utilisateur	58
4.2.6	Recherche d'un médecin	59
4.2.7	Prendre rendez-vous	60
4.2.8	Consultation en ligne	61
4.2.9	Chatbot	62
4.2.10	Offres	63

Table des figures

1.1	Établissement d'accueil-Lordroid	10
1.2	Diagramme de Gantt	14
2.1	Diagramme de cas d'utilisation global	20
2.2	Diagramme détaillé de cas d'utilisation «S'inscrire»	21
2.3	Diagramme détaillé de cas d'utilisation «Traiter un rendez-vous»	23
2.4	Diagramme détaillé du cas d'utilisation «Saisir symptômes» . .	25
2.5	Diagramme détaillé de cas d'utilisation «Estimer une maladie»	27
3.1	Clean Architecture [12]	31
3.2	Diagramme de classes représentant la couche "Entités"	33
3.3	Diagramme de dépendance générale	36
3.4	Diagramme de dépendance détaillé dans le cas de la fonctionnalité «afficher la liste des utilisateurs»	37
3.5	Diagramme de séquence détaillé du cas d'utilisation "s'inscrire"	38
3.6	Diagramme de séquences détaillé du cas d'utilisation "Traiter des rendez-vous"	39
3.7	Diagramme de séquences détaillé du cas d'utilisation "Estimer une maladie"	40
4.1	Android Studio	44
4.2	PostMan	45
4.3	Draw.io	46
4.4	Firebase	47
4.5	Adobe illustrator cc 2017	48
4.6	Diagramme de déploiement de notre application	52
4.7	Interfaces de lancement de l'application	54
4.8	Interfaces «Authentification »	55
4.9	Interfaces « Incription de patient »	55
4.10	Interfaces « Incription de médecin »	56
4.11	« Interface d'accueil »	57
4.12	Interface «Menu principal »	58

4.13 Interfaces « Profile utilisateur »	59
4.14 Interfaces « Recherche médecin »	59
4.15 Interfaces « Prendre rendez-vous »	61
4.16 Interface « Consultation en ligne »	62
4.17 Interfaces « Sélection des symptômes »	62
4.18 Interfaces « Estimation de maladie »	63
4.19 Interface «Offres »	64

Liste des tableaux

1.1	Étude de l'existant	12
2.1	Cas d'utilisation«S'inscrire»	22
2.2	Cas d'utilisation«Traiter un rendez-vous»	24
2.3	Cas d'utilisation«Saisir symptômes»	26
2.4	Cas d'utilisation«Estimer une maladie»	28
4.1	Bibliothèques	50

Introduction générale

Le domaine du développement mobile est très bien évolué ces dernières années vu l'utilisation montante des Smartphones. Ce domaine témoigne de multiples investissements importants grâce aux développeurs qui s'intéressant de plus en plus à ce domaine et proposent une infinité d'applications aussi utiles que divertissantes.

Aujourd'hui, s'informer de la météo, commander des produits en ligne, suivre les résultats des compétitions sportives, etc... sont tous des services offerts à travers des fenêtres de discussions réalisées avec des chatbots.

Un chatbot est un agent conversationnel ayant le même comportement qu'un être humain grâce à des outils d'intelligence artificielle qui lui permettent de comprendre et générer des phrases écrites en langage naturel. Ce chatbot est disponible 7j/7 et 24h/24, et peut répondre aux requêtes de plusieurs utilisateurs à la fois.

Ce chatbot permet de construire des applications intelligentes tel que dans le domaine de la santé. En faisant une recherche dans ce domaine, nous avons constaté que les gens trouvent des difficultés pour choisir et trouver le médecin adapté à leurs maladies (mal estimation de la maladie, problème de localisation du plus proche médecin adéquat...). Les difficultés peuvent être aussi au niveau de la prise de rendez-vous (communication téléphonique payante, perte de temps principalement dans les hôpitaux publics...). Cette étude révèle aussi d'autres difficultés chez les médecins débutants pour avoir des patients.

Cette étude a mené à la proposition du sujet de notre stage qui est le développement d'une application intelligente mobile de consultation en ligne et d'estimation de maladie. En effet, cette application doit permettre de prendre des rendez-vous à distance, de localiser un médecin en se basant sur les coordonnées géographiques. Notre application doit aussi donner la possibilité d'estimer automatiquement la maladie à partir d'un ensemble de symptômes entrés par le patient à travers l'intégration d'un chatbot. Ce chatbot pourra aussi proposer le traitement médical adéquat.

Notre projet de fin d'études intitulé "Conception et développement d'une application mobile intelligente de e_consultation et d'estimation des maladies" vient conclure notre formation d'ingénieur en informatique à l'Institut

Supérieur des Sciences Appliquées et de Technologie de Sousse.

Le projet a été réalisé dans une durée de quatre mois, au sein de l'entreprise Lordroid. Le présent rapport synthétise les étapes de réalisation de ce projet. Il a pour but de situer le contexte du projet, de décrire l'application réalisée, les méthodes et outils utilisés ainsi que les résultats obtenus.

Le présent rapport suit l'organisation suivante :

Le premier chapitre s'intitule " Cadre général du projet ", qui est un chapitre introductif présentant l'entreprise d'accueil , la problématique, la solution proposée et les objectifs du projet, une étude de l'existant et le processus de développement de notre application.

Le deuxième chapitre, "Spécification des besoins", sert à identifier les acteurs de notre application et ensuite à spécifier les besoins fonctionnels et non fonctionnels auxquels notre application doit répondre permettant de dégager ses principales fonctionnalités.

Le troisième chapitre, "Conception", qui sert à décrire les schémas conceptuels et l'architecture appliquée à notre solution proposée.

Le quatrième et le dernier chapitre, "Réalisation", illustre la réalisation de notre projet par la présentation de l'environnement et les outils de développement ainsi que la visualisation des résultats de notre travail à travers les principales interfaces de l'application.

Enfin, nous terminons le rapport par une conclusion générale dans laquelle nous récapitulons le travail réalisé et nous présentons les perspectives futures.

CHAPITRE 1

CADRE GÉNÉRAL DU PROJET

1.1	Présentation de l'organisme d'accueil	9
1.2	Présentation du projet	10
1.2.1	Problématique	10
1.2.2	Solution proposée	11
1.2.3	Objectifs	11
1.3	Étude de l'existant	12
1.4	Processus de développement	13
1.4.1	Développement incrémental	13
1.4.2	Planning prévisionnel des tâches	13

Introduction

Nous abordons ce chapitre par une présentation générale de l'organisme d'accueil. Nous détaillerons ensuite la problématique de notre stage ainsi que la solution proposée qui sera une tentative de résolution des inconvénients des solutions qui existent préalablement. Nous finirons ce chapitre par décrire le calendrier de notre stage à travers le diagramme de Gantt ainsi que le processus de développement.

1.1 Présentation de l'organisme d'accueil

Lordroid est un centre de développement informatique basé à Jemmal en Tunisie. Créé en 2015 par une équipe de jeunes promoteurs, il est spécialisé dans la conception et la réalisation d'Applications Mobiles. Il s'agit actuellement d'une équipe de 5 développeurs (2 iOS, 2 android et un développeur web)

Ses principaux partenaires sont de grands comptes en Europe. La maturité et la compétence de ses équipes ainsi que la synergie avec ses partenaires étrangers il permettent de leur apporter une entière satisfaction.

Lordroid conseille, conçoit et développe des solutions mobiles et web sur-mesure, adaptées aux besoins de tout type d'entreprise.

Selon les besoins de ses clients, il eux proposera des solutions Web/Mobile ou des applications natives sur les plateformes les plus populaires du marché :iOS (iPhone / iPad / iPod), Android.

Les client peuvent également le confier :

- Le développement des sites web personnalisés sur des framework PHP tel que Symfony, Laravel, Yii ou Zend.
- Le développement de sites portails, blogs et sites événementiels utilisant des CMS leader du marché (WordPress ou Drupal).
- La mise en place de boutiques e-commerce avec les solutions Prestashop ou Magento.

Lordroid offre plusieurs modes de fonctionnement :

- Mode Projet : Réalisation de vos projets en intégralité ou en partie sur un périmètre défini.
- Équipe Dediée : Mise à disposition de ressources dédiées en régie.

- Offres de service : Création et intégration de vos propres services et adaptation à votre stratégie commerciale.



FIGURE 1.1 – Établissement d'accueil-Lordroid

1.2 Présentation du projet

Dans cette partie,nous mettons notre travail dans son contexte général.En premier lieu, nous exposons la problématique et les raisons qui ont mené Lordroid à proposer ce sujet. En second lieu, nous présentons la solution proposée afin de résoudre la problématique. Enfin, nous décrirons les objectifs de ce projet.

1.2.1 Problématique

Nous avons constaté que les gens trouvent des difficultés pour choisir et trouver le médecin adapté à leurs maladies (mal estimation de la maladie, problème de localisation du plus proche médecin adéquat...). Les difficultés peuvent être aussi au niveau de la prise de rendez-vous (communication téléphonique payante, perte de temps principalement dans les hôpitaux publics...).

1.2.2 Solution proposée

Nous proposons une application mobile de consultation médicale en ligne et d'estimation des maladies. Cette application doit permettre également de prendre des rendez-vous à distance, de localiser un médecin en se basant sur les coordonnées géographiques. Notre application doit aussi donner la possibilité d'estimer automatiquement la maladie à partir d'un ensemble de symptômes entrés par le patient à travers l'intégration d'un chatbot. Ce chatbot pourra aussi proposer le traitement médical adéquat.

1.2.3 Objectifs

Garantir la satisfaction des utilisateurs en assurant les :

Objectifs fonctionnels :

1. Consultation à distance.
2. Prendre les rendez-vous en ligne.
3. Estimation des maladie.
4. Localisation des médecins.

Objectifs non fonctionnels :

1. Disponibilité
2. Ergonomie
3. Fiabilité
4. Extensibilité

Objectifs techniques :

1. Organisation de l'application selon le *clean architecture*[2].
2. L'injection de dépendance avec le framework Dagger[5].
3. Intégration de chatbot.
4. Utilisation du langage de programmation orienté objet "Kotlin"[8].

1.3 Étude de l'existant

Dans cette partie, nous analysons et critiquons les applications existantes actuellement à travers le tableau 1.1 afin de proposer une solution qui résout leurs inconvénients.

Applications	Avantages	Inconvénients
Med.tn	Med.tn est une plateforme qui permet de localiser les médecins, prendre des rendez-vous et poser des questions.	Med.tn n'a pas un chatbot permettant à un patient d'estimer sa maladie.
TunMedic	TunMedic est une application mobile qui permet de localiser les professionnels de la santé tunisienne.	TunMedic ne permet au patient ni de communiquer avec les médecins ni de prendre des rendez vous en ligne. Cette application n'intègre pas un chatbot également.
k Health	k Health est une application mobile qui donne une estimation de la maladie à partir des symptômes donnés par l'utilisateur au cours d'une conversation à distance gratuite..	k Health ne donne pas les informations sur les médecins ainsi que leurs localisations. Aussi cette application ne permet pas de prendre un rendez-vous en ligne.

TABLE 1.1 – Étude de l'existant

1.4 Processus de développement

Un processus de développement logiciel est un ensemble d'activités connexes suivis par une équipe mené à la production du logiciel au sein de l'organisation. Il consiste en un plan détaillé décrivant comment développer, concevoir, tester, déployer et maintenir le produit.

1.4.1 Développement incrémental

Dans ce contexte, nous adoptons le processus de développement incrémental comme démarche pour la réalisation de notre projet. Selon ce processus, les besoins du client sont spécifiés, le logiciel est globalement conçu, puis la réalisation se fait par incrément de fonctionnalités. Chaque incrément est considéré comme une partie exécutable du système final. Ces incréments sont intégrés successivement au produit final et à chaque étape le logiciel est testé, exploité et maintenu dans son ensemble. Implémenter le logiciel par incrément permet de prendre en compte l'analyse des risques pour faciliter la détection des erreurs au plus tôt selon les retours du client et de réduire les délais et le coût de production, ce qui aide à la réalisation d'un logiciel qualifié.

1.4.2 Planning prévisionnel des tâches

Le diagramme de gantt est un outil graphique qui représente la gestion du projet dans le temps ce qui facilite sa réalisation.

En effet, la figure 1.2 représente l'avancement des activités et des tâches exécutées tout le long de notre projet.

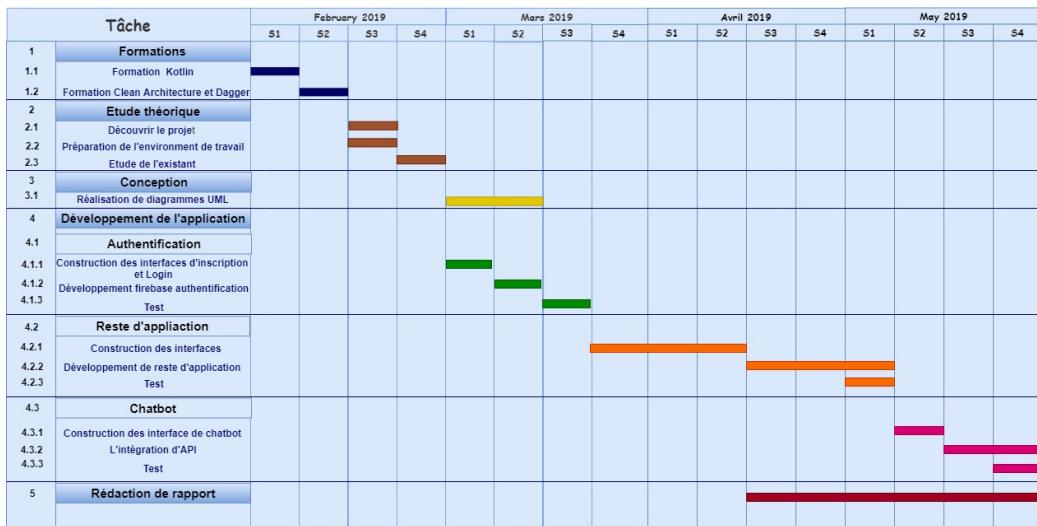


FIGURE 1.2 – Diagramme de Gantt

Conclusion

Dans ce chapitre, nous avons introduit le contexte de notre projet en représentant l'organisme d'accueil en premier lieu. En seconde lieu, nous avons dégagé le problématique. Ensuite, nous avons décrit la solution proposée et les objectifs à atteindre. En troisième lieu, nous avons analysé les applications existantes. En quatrième lieu, nous avons représenté l'avancement des activités tout au long du projet selon le processus de développement adopté. Dans le prochain chapitre, nous allons spécifier les besoins fonctionnels et les besoins non fonctionnels.

SPÉCIFICATION DES BESOINS

2.1	Identification des acteurs	16
2.2	Besoins Fonctionnels	17
2.3	Besoins non Fonctionnels	17
2.4	Diagrammes de cas d'utilisation	18
2.4.1	Diagramme de cas d'utilisation globale	19
2.4.2	Raffinement des cas d'utilisation	21
2.4.2.1	Raffinement du cas d'utilisation «S'inscrire»	21
2.4.2.2	Raffinement du cas d'utilisation «Traiter un rendez-vous»	23
2.4.2.3	Raffinement du cas d'utilisation «Saisir symptômes» .	25
2.4.2.4	Raffinement du cas d'utilisation «Estimer une maladie»	27

Introduction

Dans ce chapitre, nous allons identifier les acteurs, ensuite nous allons spécifier les besoins fonctionnels et non fonctionnels auxquels la solution proposée doit répondre. Finalement, nous présentons les diagrammes de cas d'utilisation qui expliquent les principales fonctionnalités de notre application.

2.1 Identification des acteurs

Un acteur représente une entité externe qui interagit directement avec le système. Il peut être soit une personne humaine soit un système. On distingue deux types d'acteur, un acteur principale et un acteur secondaire. En effet, un acteur principal obtient un résultat observable du système tandis qu'un acteur secondaire est sollicité pour des informations complémentaires.

Dans notre application nous avons 4 acteurs :

Les acteurs principales :

Administrateur :

Cette personne est le responsable de :

- La gestion des utilisateurs
- La validation de compte médecin
- L'envoie des notifications

Patient :

C'est un utilisateur qui doit être authentifié pour qu'il puisse :

- Consulter en ligne un médecin
- Prendre un rendez-vous en ligne
- Rechercher un médecin
- Localiser le cabinet d'un médecin
- Discuter avec le chatbot pour estimer sa maladie
- Gérer son profile
- Exprimer son avis

Médecin :

Cet utilisateur doit être aussi authentifié pour qu'il puisse :

- Communiquer avec les patient
- Traiter les rendez-vous
- Annoncer des offres
- Gérer son profile

L'acteur secondaire :**Chatbot :**

Cet acteur est un programme qui représente un agent conversationnel qui :

- Analyse les requêtes envoyées par le patient
- donne une estimation de la maladie
- donne le spécialiste le plus proche du patient
- propose un traitement médical selon la maladie du patient

2.2 Besoins Fonctionnels

Les besoins fonctionnels sont exprimés par l'utilisateur de l'application qui permettent de dégager les fonctionnalités de cette application. Dans notre cas, les besoins fonctionnels sont :

- Inscription
- Authentification
- Validation des comptes
- Consultation en ligne
- Prendre des rendez-vous à distance
- Estimation de la maladie
- Annonce des offres
- Envoie des réclamations
- Envoie des notification
- Localisation du cabinet du médecin
- Proposition de médecin

2.3 Besoins non Fonctionnels

Les besoins non fonctionnels représentent les caractéristiques du système. Ils concernent les contraintes à prendre en considération pour mettre en place une solution adéquate.

Pour notre application, les besoins non fonctionnels sont :

Ergonomie :

Les interfaces de notre application donne à l'utilisateur ce qu'il cherche rapidement et facilement.

Extensibilité :

L'application peut être étendue par l'ajout d'autres scénarios et d'autres fonctionnalités.

Performance :

Le temps de réponse aux requêtes des utilisateurs doit être raisonnable.

Disponibilité :

L'agent conversationnel doit être disponible en service 7/7j et 24/24h pour répondre aux requêtes des utilisateurs à tout moment.

Adaptabilité :

Notre application doit être adaptable aux différentes tailles d'écran (Smartphone / Tablette).

Sécurité :

Tous les utilisateurs doivent s'authentifier avant d'accéder à l'application.

2.4 Diagrammes de cas d'utilisation

Dans cette section, nous allons dégager les fonctionnalités du système à partir des besoins fonctionnels cités précédemment en se basant sur les un diagrammes UML¹ qui regroupe l'ensemble des cas d'utilisation du système.

1. Unified Modeling Language

2.4.1 Diagramme de cas d'utilisation globale

La figure 2.1 illustre le diagramme de cas d'utilisation globale de notre application.

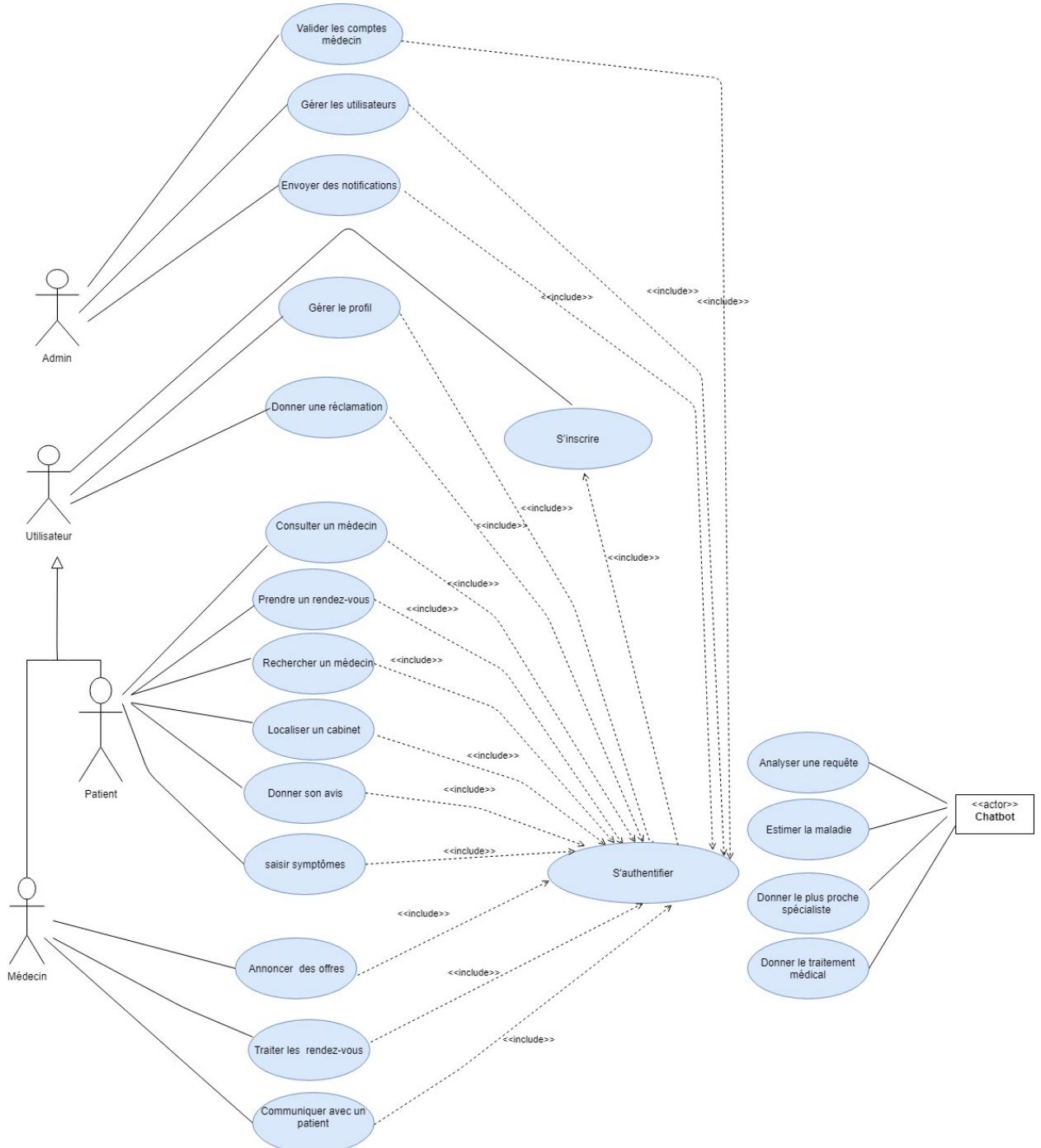


FIGURE 2.1 – Diagramme de cas d'utilisation global

2.4.2 Raffinement des cas d'utilisation

Dans cette section, nous allons détailler les principaux cas d'utilisation.

2.4.2.1 Raffinement du cas d'utilisation «S'inscrire»

Dans notre application, un utilisateur doit s'inscrire qu'il soit un patient ou un médecin avant de bénéficier des services de notre application.

La figure 2.2 représente le diagramme de cas d'utilisation «S'inscrire»

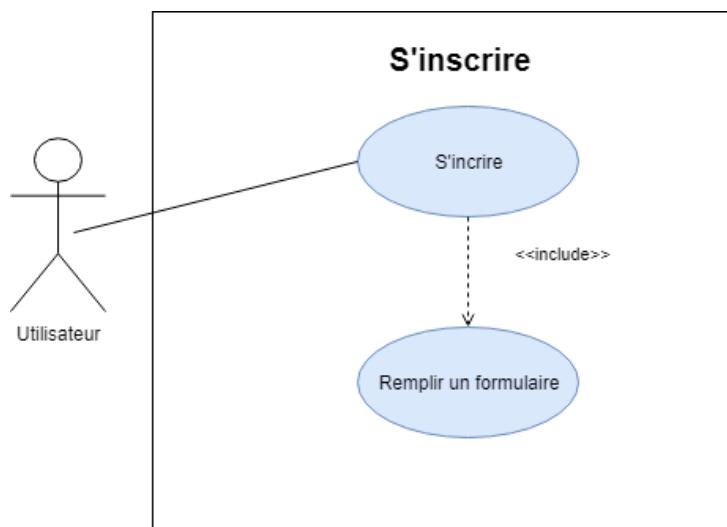


FIGURE 2.2 – Diagramme détaillé de cas d'utilisation «S'inscrire»

Le tableau 2.1 détaille les tâches à exécuter par l'utilisateur pour s'inscrire à note application.

SOMMAIRE	
Titre	S'inscrire
Objectif	Permettre à l'utilisateur de s'inscrire pour pouvoir s'authentifier à chaque consultation de l'application
Acteurs	Médecin et patient
Description des enchainements	
Pré-condition	L'utilisateur doit démarrer l'application et aller sur l'interface d'inscription.
Post-condition	L'utilisateur est inscrit et ses coordonnées sont enregistrés dans la base de données.
Scénario normal	<ol style="list-style-type: none"> 1. L'utilisateur accède à l'interface d'inscription. 2. L'utilisateur remplit le formulaire et confirme. 3. L'inscription est effectué avec succès pour le patient. Mais, le médecin attend la validation du compte par l'administrateur.
Scénario alternatif	<ol style="list-style-type: none"> 1. Des champs vides : Le système envoie un message d'erreur «Vous devez remplir tous les champs» 2. L'email saisi ou le mot de passe saisi ne sont pas valides : Le système affiche un message d'erreur décrivant les conditions de validation de ces champs. 3. L'email est déjà utilisé : Une alerte est envoyée par le système.
Contraintes non fonctionnels	<ol style="list-style-type: none"> 1. L'interface doit être ergonomique. 2. Les messages d'erreur doivent être compréhensibles et clairs.

TABLE 2.1 – Cas d'utilisation «S'inscrire»

2.4.2.2 Raffinement du cas d'utilisation «Traiter un rendez-vous»

La figure 2.3 illustre le diagramme de cas d'utilisation «Traiter un rendez-vous»

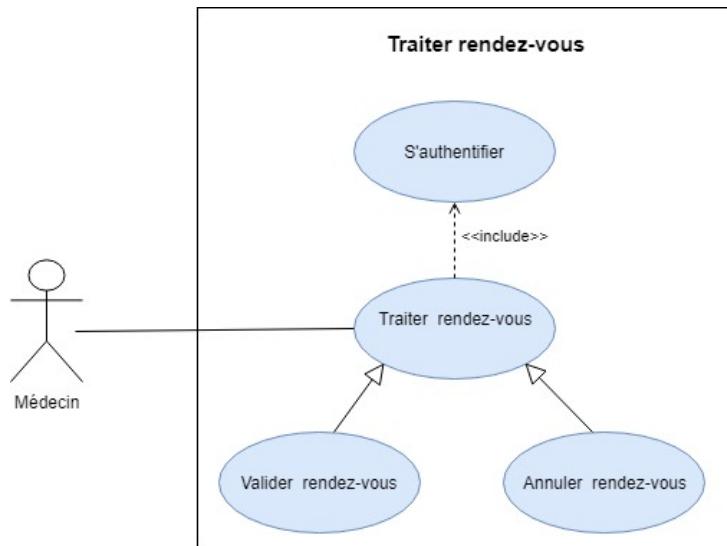


FIGURE 2.3 – Diagramme détaillé de cas d'utilisation «Traiter un rendez-vous»

Le tableau 2.2 décrit les tâches à réaliser par un médecin pour gérer les rendez-vous avec ses patients.

SOMMAIRE	
Titre	Traiter un rendez-vous
Objectif	Permettre au médecin de gérer les rendez-vous proposés par les patients.
Acteurs	Médecin
Description des enchainements	
Pré-condition	Le médecin est authentifié
Post-condition	Les rendez-vous sont traités et enregistrés dans la base de données.
Scénario normal	<ol style="list-style-type: none"> 1. Le médecin doit s'authentifier 2. Le médecin affiche la liste des rendez-vous demandés 3. Le médecin confirme ou refuse les demandes
Scénario alternatif	Le système affiche un message d'erreur en cas de confirmation d'un rendez-vous dont la date et l'heure sont déjà réservées
Contraintes non fonctionnels	<ol style="list-style-type: none"> 1. L'interface doit être ergonomique 2. Les messages d'erreur doivent être compréhensibles et clairs

TABLE 2.2 – Cas d'utilisation «Traiter un rendez-vous»

2.4.2.3 Raffinement du cas d'utilisation «Saisir symptômes»

La figure 2.4 illustre le diagramme de cas d'utilisation «Saisir symptômes»

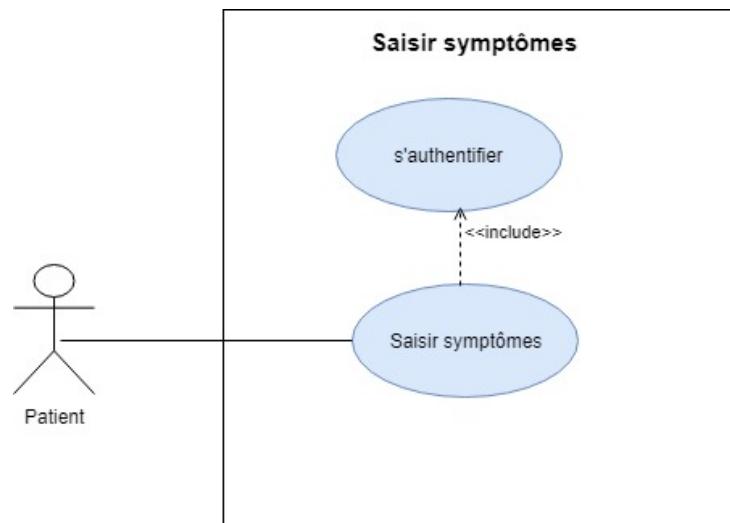


FIGURE 2.4 – Diagramme détaillé du cas d'utilisation «Saisir symptômes»

Le tableau 2.3 décrit les tâches à réaliser par un patient pour saisir les symptômes de sa maladie qui vont être traités par le Chatbot afin d'estimer sa maladie et lui proposer un traitement adéquat.

SOMMAIRE	
Titre	Saisir symptômes
Objectif	Permettre au patient de saisir les symptômes de sa maladie.
Acteurs	Patient
Description des enchainements	
Pré-condition	Le patient est authentifié
Post-condition	Les symptômes sont saisies.
Scénario normal	<ol style="list-style-type: none"> 1. Le patient doit s'authentifier 2. Le patient affiche l'interface du chatbot 3. Le patient sélectionne les symptômes de sa maladie parmi une liste de symptôme proposées par le chatbot
Contraintes non fonctionnels	<ol style="list-style-type: none"> 1. L'interface doit être ergonomique 2. L'agent conversationnel(chatbot) doit être toujours disponible.

TABLE 2.3 – Cas d'utilisation «Saisir symptômes»

2.4.2.4 Raffinement du cas d'utilisation «Estimer une maladie»

La figure 2.5 illustre le diagramme de cas d'utilisation «Estimer une maladie»

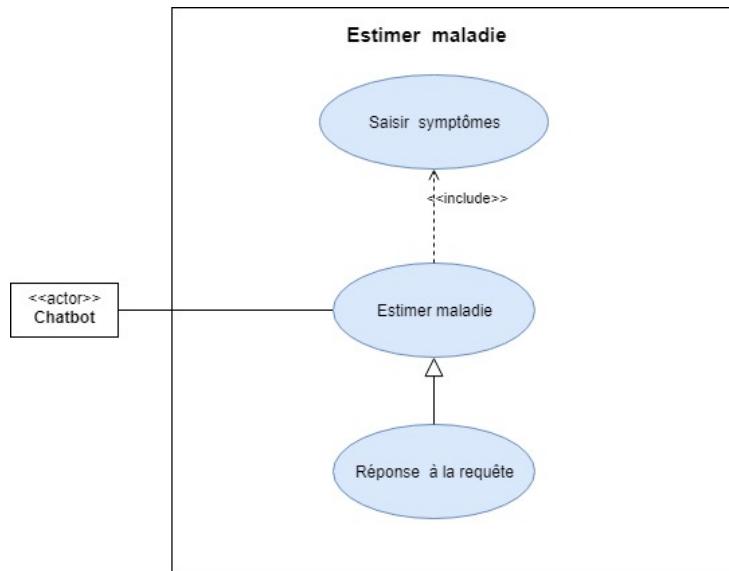


FIGURE 2.5 – Diagramme détaillé de cas d'utilisation «Estimer une maladie»

SOMMAIRE	
Titre	Estimer une maladie
Objectif	Déclencher un chatbot qui répondra à la requête du patient en estimant sa maladie
Acteurs	Chatbot
Description des enchainements	
Pré-condition	L'utilisateur s'est authentifié, il sélectionne les symptômes de sa maladie
Post-condition	La maladie est estimée et la réponse est affichée
Scénario normal	<ol style="list-style-type: none"> 1. le chatbot est lancé à la réception d'une requête 2. le chatbot analyse la requête 3. le chatbot prépare la réponse 4. le chatbot envoie la réponse
Scénario alternatif	Un message d'erreur s'affiche au cas où la maladie est non reconnue par le système.
Contraintes non fonctionnels	<ol style="list-style-type: none"> 1. L'interface doit être ergonomique 2. Le chatbot doit être toujours disponible

TABLE 2.4 – Cas d'utilisation «Estimer une maladie»

Conclusion

Au cours de ce chapitre, nous avons décrit les phases de spécification des besoins de notre application élaborées afin d'identifier les différents acteurs ainsi que les fonctionnalités et les services que notre application doit fournir. Nous avons détaillé ces fonctionnalités par des diagrammes de cas d'utilisation. Le prochain chapitre sera consacré à la phase de conception.

CHAPITRE 3

CONCEPTION

3.1	Architecture globale	30
3.1.1	Définition du patron Clean Architecture	30
3.1.2	Application du patron "Clean Architecture"	32
3.2	Vue dynamique de l'application	38
3.2.1	Diagramme de séquences détaillé du cas d'utilisation "s'inscrire"	38
3.2.2	Diagramme de séquence détaillé du cas d'utilisation "Traiter des rendez-vous"	39
3.2.3	Diagramme de séquences détaillé du cas d'utilisation "Estimer une maladie"	40

Introduction

Après avoir identifié les besoins fonctionnels et non fonctionnels et les principales fonctionnalités de notre application. Nous entamons dans cette partie l'étude conceptuelle en décrivant l'architecture générale de notre système et sa modélisation interne détaillée à travers les diagrammes de classes, de séquences.

3.1 Architecture globale

Dans cette section, nous allons donner un aperçu sur la définition du patron d'architecture choisi pour modéliser notre application et nous allons détailler la projection de ce patron sur notre application.

3.1.1 Définition du patron Clean Architecture

La *clean architecture* est une philosophie de conception logicielle proposée par Robert C. Martin, alias Oncle Bob. Cet architecture est basée sur[2]

- les principes SOLID dont dérivent le principe d'inversion de contrôle et le principe d'injection de dépendance (un framework externe à l'application crée les composantes de l'application et injecte les dépendances entre ces composantes) [12].
- le principe de séparation des règles métiers "*business rules*" (indépendantes des applications) et des règles de l'application "*application rules*".
- le principe d'indépendance du noyau de l'application des technologies nécessaires pour son fonctionnement.

Grâce à ces principes, les composants de l'application deviennent facile à gérer, à tester, à modifier et à réutiliser.

Le patron *clean architecture* est divisé en 4 couches (figure 3.1) dont le sens de dépendance est entrant :

— Entités

Les entités qui représentent les objets du domaine de l'application avec leurs méthodes (*Business Rules*).

— Cas d'utilisation

Cette couche représente la logique de l'application, c'est à dire les règles de gestion spécifique à une application (*Application Rules*).

— Interfaces / Adaptateurs

cette couche représente l'interface entre les systèmes externes à l'application.

tion (framework, databases, UI...) et le noyau de l'application (entités + cas d'utilisation). Cette couche permet de réaliser le formatage, la structuration et l'adaptation des données entrantes et sortantes.

— Technologies externes

Cette couche représente la couche externe à l'application regroupant les technologies nécessaires à son fonctionnement (*APIs, frameworks, Databases, UI, network, drivers, devices...*).

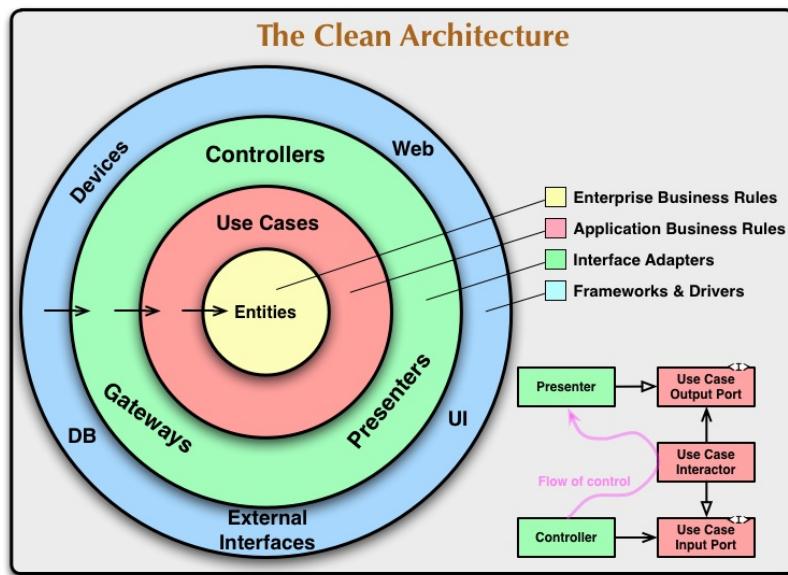


FIGURE 3.1 – Clean Architecture [12]

3.1.2 Application du patron "Clean Architecture"

Nous expliquons dans ce paragraphe les détails de la projection du patron Clean Architecture sur notre application.

La figure 3.2 représente le diagramme de classes qui illustre la liaison entre les entités(classes) adoptées du domaine médicale. En effet, ce diagramme traduit le contenu de la couche **Entités**.

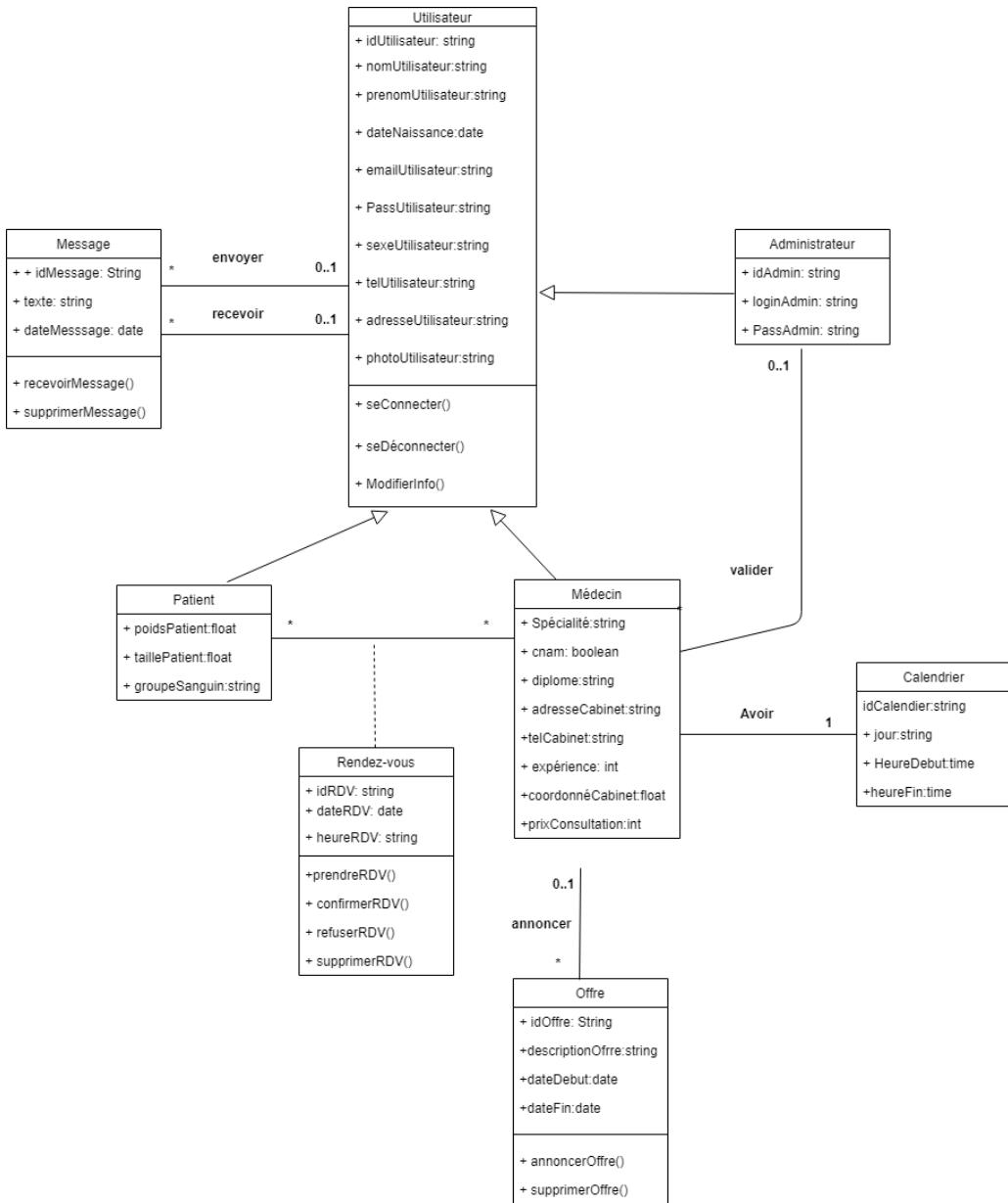


FIGURE 3.2 – Diagramme de classes représentant la couche "Entités"

Description des classes qui représentent nos entités :

— Utilisateur :

Cette classe représente les utilisateurs de notre application. Dans notre application, un utilisateur peut se connecter, déconnecter et aussi de mo-

difier ses coordonnées. Les attributs de cette classe sont :

- idUtilisateur : l'identifiant de l'utilisateur.
- nomUtilisateur : nom de l'utilisateur.
- prénomUtilisateur : le prénom de l'utilisateur.
- dateNaissance : la date de naissance de l'utilisateur.
- emailUtilisateur : l' email de l'utilisateur.
- PassUtilisateur : le mot de passe de l'utilisateur.
- sexeUtilisateur : le sexe de l'utilisateur.
- telUtilisateur : le numéro de téléphone de l'utilisateur.
- adresseUtilisateur : l'adresse de l'utilisateur.
- adresseUtilisateur : l'adresse de l'utilisateur.

— **Administrateur :**

L'administrateur est un utilisateur. Il gère les utilisateurs : patients et médecins et valide le compte médecin. Ses attributs sont :

- idAdmin : l'identifiant de l'administrateur.
- LoginAdmin : l'email de l'administrateur .
- PassAdmin : le mot de passe de l'administrateur .

— **Patient :**

Le patient est un utilisateur. Dans le cas où, il veut consulter un médecin en ligne, il suffit qu'il lui envoie un message.

Ses attributs sont :

- PoidsPatient : le poids du patient.
- taillePatient : la taille du patient.
- groupeSanguin : le groupe sanguin du patient.

— **Médecin :**

Le médecin est un utilisateur. Chaque médecin doit avoir un calendrier. Comme il peut discuter avec son patient en ligne.

Ses attributs sont :

- spécialité : la spécialité de médecin.
- cnam¹ : cet attribut retourne vrai si le médecin est conventionné avec le cnam sinon il retourne faux.
- diplome : l'image de diplôme du médecin devant être validée.
- adresseCabinet : l'adresse du cabinet.
- telCabinet : le numéro de téléphone du cabinet.
- coordonnéCabinet : la position géographique localisé par GPS, elle est définie par une longitude et une latitude.
- expérience : le nombre d'année d'expérience du médecin.
- prixConsultation : le prix de consultation.

— **Calendrier :**

1. Caisse nationale d'assurance maladie

Un calendrier regroupe les dates de travail d'un médecin.

Ses attributs sont :

- idCalendrier : l'identifiant de calendrier.
- jour : les jours de travail de médecin dans une semaine.
- heureDebut : l'heure de début de travail pendant une journée.
- heureFin : l'heure de fin de travail pendant une journée.

— **Message :**

cette classe représente le message écrit par l'utilisateur(patient ou médecin) pour discuter.

Ses attributs sont :

- idMessage : l'identifiant de message.
- texte : le texte écrit par l'utilisateur .
- dateMessage : la date d'envoi ou de réception du message.

— **Rendez-vous :**

cette classe représente le rendez-vous que le patient le réserve en ligne. Il peut être accepté ou refusé par le médecin. Comme il peut être supprimé par l'utilisateur.

Ses attributs sont :

- idRDV : l'identifiant du rendez-vous.
- dateRDV : la date du rendez-vous.
- heureRDV : l'heure du rendez-vous.

— **Offre :**

cette classe représente l'offre annoncée par le médecin. il peut être supprimée par le médecin ou par le système si la période associée est terminée.

Ses attributs sont :

- idOffre : l'identifiant de l'offre.
- descriptionOffre : la description de l'offre.
- dateDebut : la date de début de l'offre.
- dateFin : la date de fin de l'offre.

Les classes que nous avons définies dans la couche "Entités" sont liées au domaine médical, employables dans différents types d'application médicale et indépendantes de notre application "MyDoctor". Ainsi, nous avons réalisé le principe d'indépendance entre les règles métier (*business rules*) et les règles d'application (*application rules*).

Pour le reste des couches du patron Clean architecture, nous allons décrire les dépendances entre ces couches en donnant un exemple de liaison entre un ensemble de classes appartenant à ces couches.

La figure 3.3 représente la dépendance générale dans notre cas entre les couches du patron *clean architecture*.

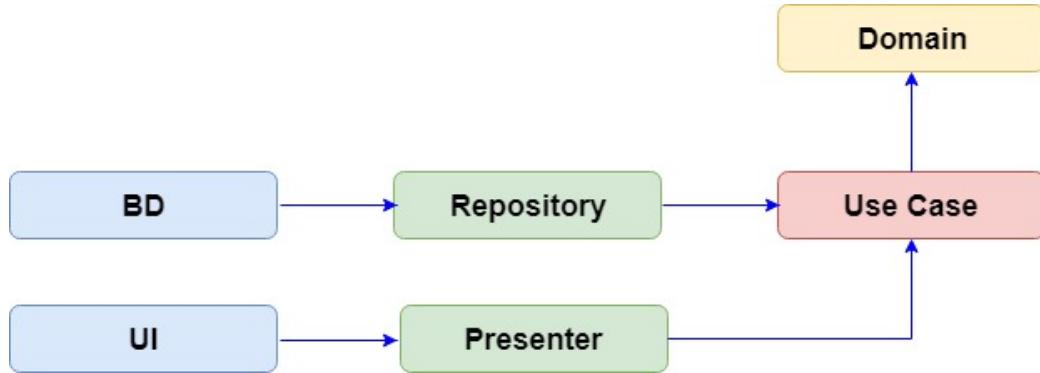


FIGURE 3.3 – Diagramme de dépendance générale

- La couche de domaine(Entités) est indépendante de toutes les couches.
- La couche d'application(Usecase) dépend de la couche de domaine et indépendante des autres couches. Cette couche prépare les réponses aux requêtes de l'utilisateur en utilisant les entités.
- La couche d'*interface adapters* dépend de la couche application(Usecase). Cette couche transmet la requête de l'utilisateur et les données nécessaires à son traitement à la couche *Use Cases* et récupère la réponse et la structure pour le composant *User Interface*.
- Le composant d'interface utilisateur(UI) de la couche "Technologies externes" dépend de la couche *interface adapters* à travers le *Presenter* afin de lui passer la requête de l'utilisateur et récupérer la réponse.
- Le composant de bases de données de la couche "Technologies externes" permet à la couche *Use Case* la récupération des données à travers un *Gateway*(Repository) appartenant à la couche *Interface Adapter*.

La figure 3.4 représente un exemple de dépendance détaillé entre les couches du patron *clean architecture* dans le cas de notre application. Ces dépendances lient un ensemble de classes appartenant aux différentes couches. Ces classes réalisent l'exemple de la fonctionnalité d'affichage de la liste des utilisateurs.

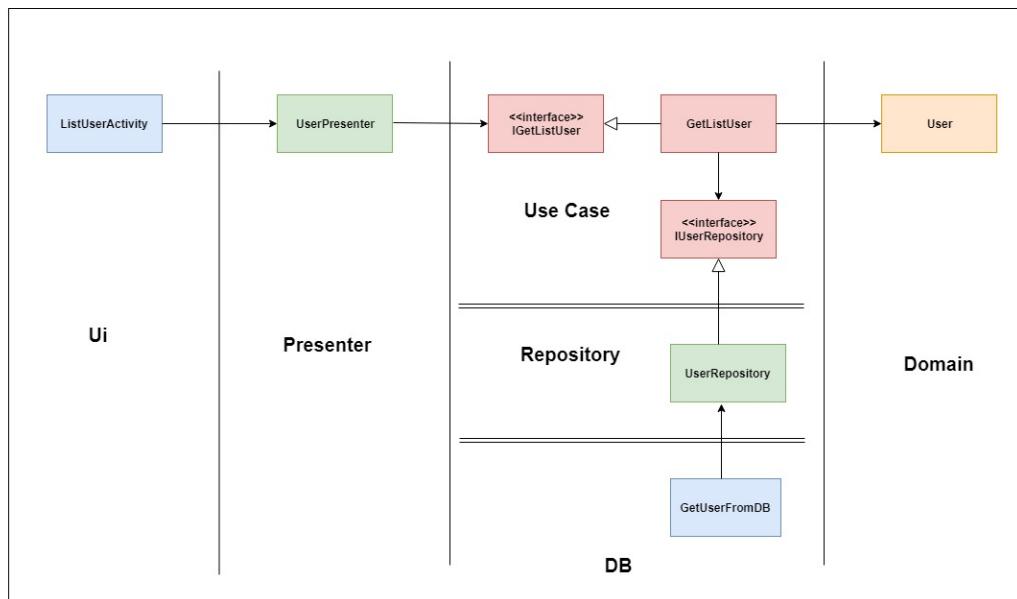


FIGURE 3.4 – Diagramme de dépendance détaillé dans le cas de la fonctionnalité «afficher la liste des utilisateurs»

Selon la figure 3.4, la couche *Interface Adapter*(`UserPresenter` et `UserRepository`) communique (passer et récupérer les données) avec la couche *Use cases* en implémentant ou instanciant ses interfaces `IGetListUser` et `IUserRepository`. Ces interfaces représentent les "ports" de la couche *Use Cases*. Cet exemple montre comment nous avons réalisé le principe d'indépendance du noyau de l'application des technologies utilisées. Aucune instance de la couche interface adapters ou de la couche externe n'existe dans le noyau de l'application.

3.2 Vue dynamique de l'application

Dans cette section, nous allons décrire la dynamique interne de notre application en utilisant les diagrammes de séquences et les diagrammes d'activités.

3.2.1 Diagramme de séquences détaillé du cas d'utilisation "s'inscrire"

La figure 3.5 illustre le diagramme de séquences détaillé du cas d'utilisation "s'inscrire".

Ce cas d'utilisation commence par l'ouverture d'une interface d'inscription pour l'utilisateur. Ensuite, l'utilisateur saisit ses coordonnées et les confirme. Le système affiche un message d'erreur si l'email existe déjà sinon il rédige l'utilisateur vers la page d'accueil d'un patient. Dans le cas où l'utilisateur est un médecin, le système lui envoie un message de confirmation puisque son compte doit être validé par l'administrateur.

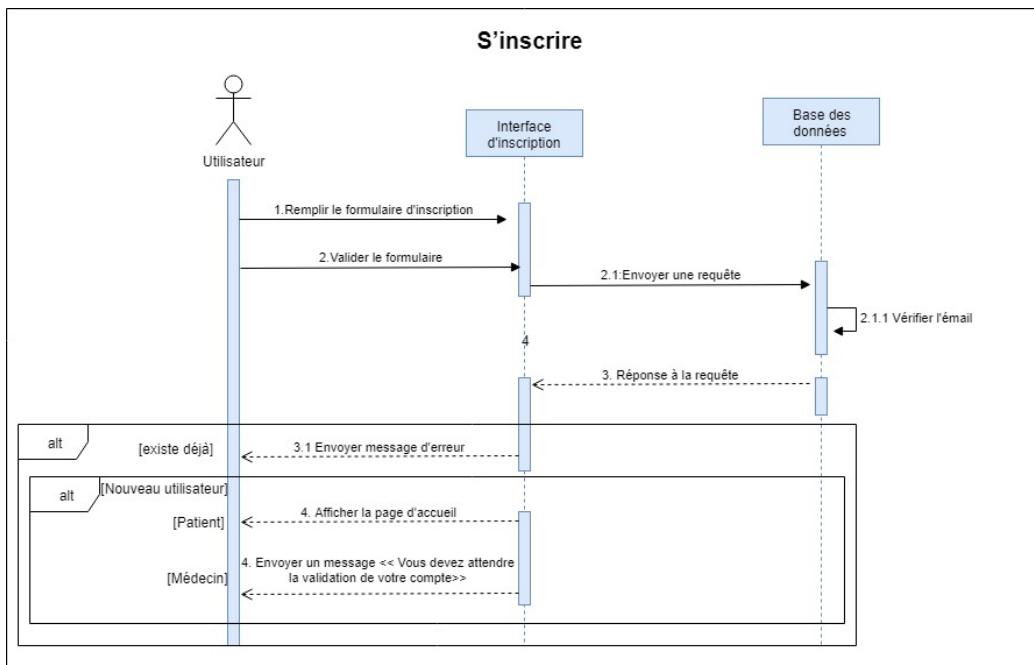


FIGURE 3.5 – Diagramme de séquence détaillé du cas d'utilisation "s'inscrire"

3.2.2 Diagramme de séquence détaillé du cas d'utilisation "Traiter des rendez-vous"

Le diagramme 3.6 décrit un scénario de cas d'utilisation "Traiter rendez-vous".

Le médecin ouvre l'interface d'accueil de son compte et demande l'interface de traitement des rendez-vous. Puis, il confirme un rendez-vous. Le système, par la suite, vérifie la disponibilité du médecin et affiche un message d'erreur si la date est déjà réservée sinon la demande est acceptée.

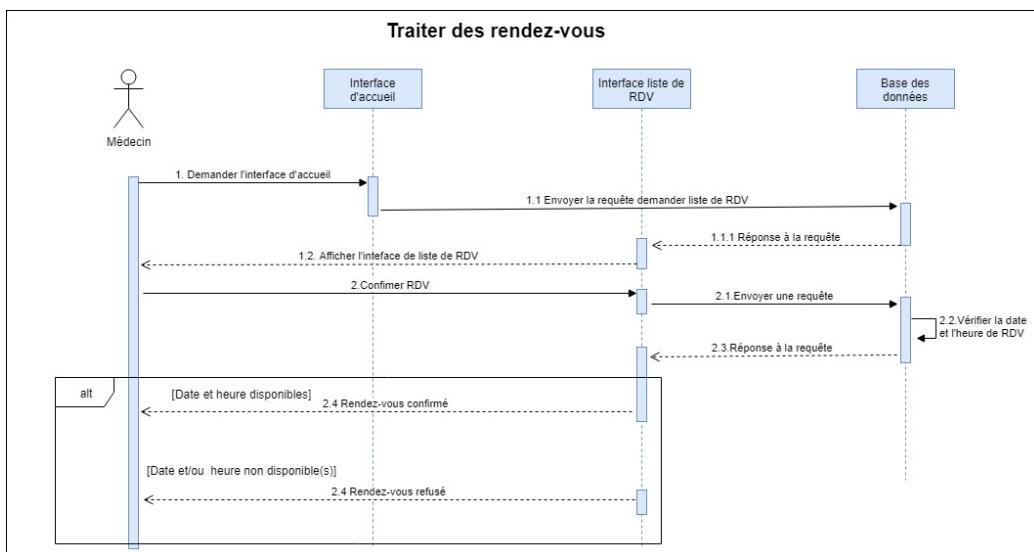


FIGURE 3.6 – Diagramme de séquences détaillé du cas d'utilisation "Traiter des rendez-vous"

3.2.3 Diagramme de séquences détaillé du cas d'utilisation "Estimer une maladie"

Le diagramme 3.7 décrit un scénario de cas d'utilisation "Estimer une maladie".

Suite à la réception de la requête de l'utilisateur, le chatbot estime la maladie si elle est la reconnue sinon il envoie un message d'erreur.

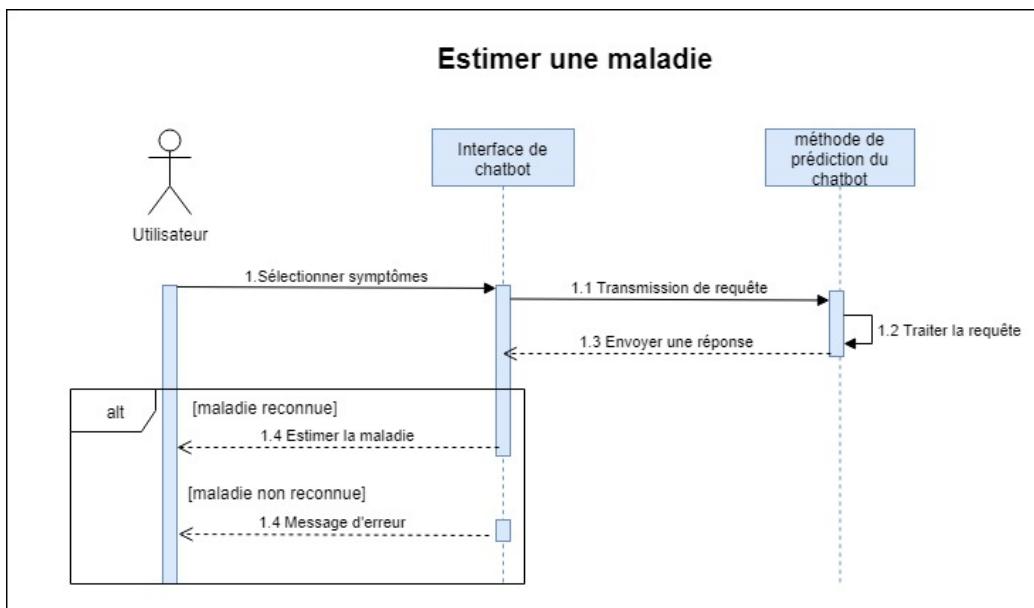


FIGURE 3.7 – Diagramme de séquences détaillé du cas d'utilisation "Estimer une maladie"

Conclusion

Ce chapitre présente l'une des plus importantes phases du processus de développement d'un projet : la conception subdivisée en conception globale et en conception détaillée.

Nous avons présenté le patron d'architecture *Clean Architecture* et nous avons expliqué son application dans notre cas. Nous avons décrit la vue dynamique du système à travers un ensemble de diagrammes de séquences et de diagrammes d'activités.

Le chapitre suivant traitera la réalisation du projet illustrée par des imprimés écran des différentes interfaces. Nous décrirons l'environnement de travail et les outils utilisés.

RÉALISATION

4.1	Spécification technique	43
4.1.1	Environnement matériel	43
4.1.2	Environnement logiciel	43
4.1.2.1	Outils logiciels	44
4.1.2.2	API et bibliothèques utilisée	48
4.1.2.3	Langage de développement	51
4.1.3	Diagramme de déploiement de l'application	51
4.2	Interfaces de l'application	53
4.2.1	Lancement de l'application	53
4.2.2	Authentification	54
4.2.3	Inscription	55
4.2.4	Interface d'accueil	56
4.2.5	Menu principal	57
4.2.5.1	Profile utilisateur	58
4.2.6	Recherche d'un médecin	59
4.2.7	Prendre rendez-vous	60
4.2.8	Consultation en ligne	61
4.2.9	Chatbot	62
4.2.10	Offres	63

Introduction

Dans ce chapitre, nous décrivons l'environnement de travail utilisés pendant la réalisation de notre application. Nous allons aussi décrire sa disposition physique à l'aide d'un diagramme de déploiement. Ensuite, nous détaillerons le travail réalisé et les résultats obtenus à l'aide d'un ensemble d'imprimés écran représentant les interfaces des différentes fonctionnalités de notre application.

4.1 Spécification technique

Dans cette section, nous allons présenter les choix techniques relatifs à l'environnement matériel et logiciel qui ont contribué à la réalisation de notre application.

4.1.1 Environnement matériel

Au cours des différents étapes de notre projet, à savoir la documentation, l'implémentation du code et le test, nous avons disposé :

— D'un ordinateur personnel dont la configuration est la suivante :

Marque : Asus ;
Processeur : Intel CORE i7 ;
RAM : 8 Go ;
Disque dur : 1To ;
Type de système : Système d'exploitation 64 bit.

— D'un Smartphone ayant les caractéristiques suivantes :

Marque : HTC ;
RAM : 3 Go ;
Processeur : Quad Core Qualcomm Snapdragon 810 à 2,3 GHz ;
Version du système : Android 7.0 Nougat.

4.1.2 Environnement logiciel

Tout au long de la phase de développement, nous avons utilisé les outils logiciels et les langages de développement suivantes :

4.1.2.1 Outils logiciels

Durant l'implémentation du notre projet nous avons utilisé les logiciels suivant :

Android Studio

Android Studio est un environnement de développement intégré de la plate-forme Android de Google . Les développeurs utilisent cet environnement afin de développer des applications Android . Android Studio permet principalement d'éditer les fichiers Java, les fichiers XML de description des interfaces, et les fichiers de configuration d'une application Android[10].

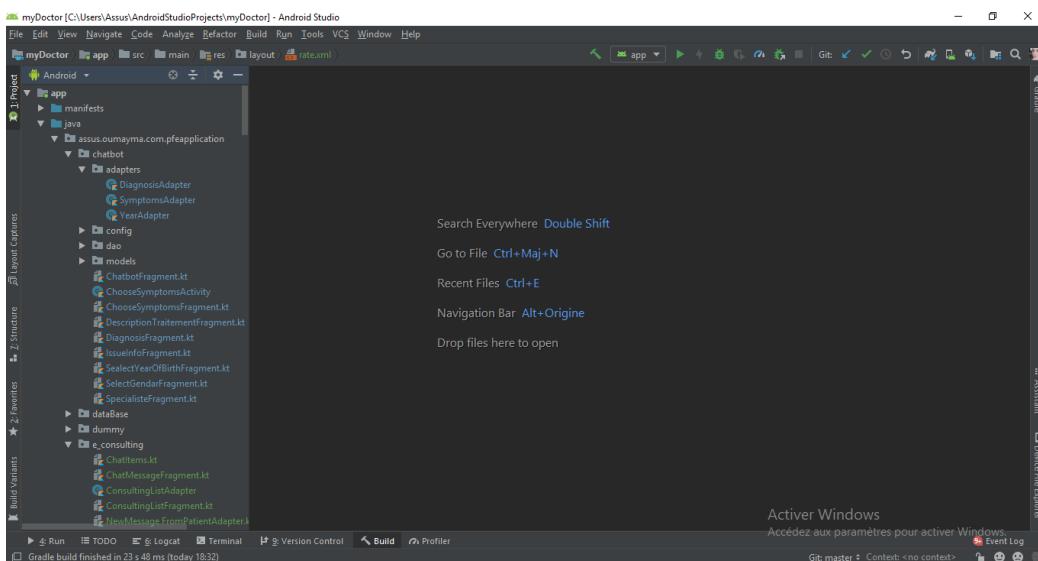


FIGURE 4.1 – Android Studio

PostMan

PostMan est une extension pour le navigateur Chrome qui rend le développement d'API plus rapide et plus simple[9].

Dans notre application, on l'utilise pour tester les APIs.

Environnement de conception : Draw.io

La conception est une phase importante dans le développement des logiciels.

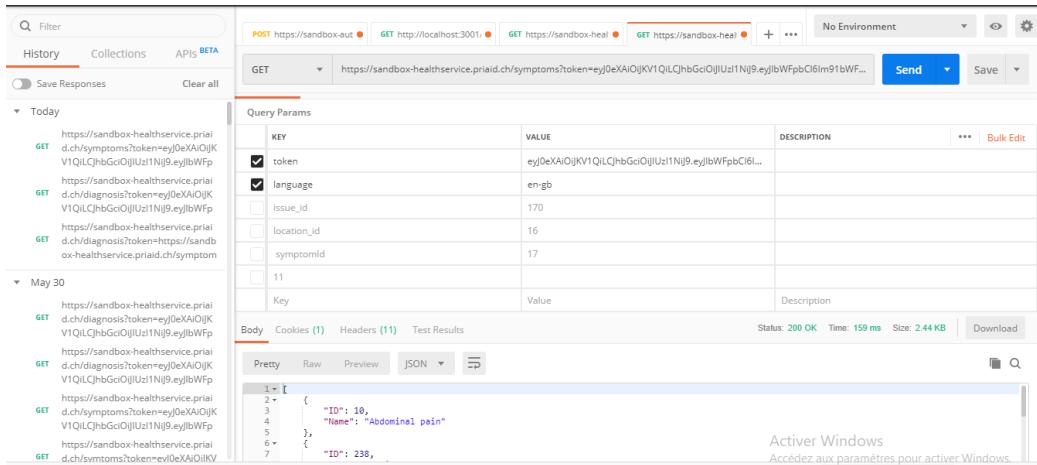


FIGURE 4.2 – PostMan

Pour cette raison, le choix de l'environnement de conception est important. Draw.io est une application Web développée par Google qui permet de dessiner des diagrammes UML, des diagrammes ERD¹, des schémas réseaux, Business Process Models, Charts, des circuits électroniques et des maquettes d'interfaces, des organigrammes en ligne gratuitement[11].

cet outil propose plusieurs fonctionnalités :

- Enregistrement de travail directement sur Dropbox ou Google Drive.
- Large bibliothèque d'icônes.
- Interface intuitive en glisser-déposer.
- Recherche d'image et ajout dans le diagramme.
- Export en format png, jpg, gif, svg, html, intégration en pages web.
- Import d'un diagramme au format XML.
- Support d'interfaces tactiles.
- Edition collaborative en temps réel.
- Intégration de diagrammes dans les blogs et les wikis.

Cette application a été très utile pour la conception des diagrammes UML et d'autres diagrammes dans ce travail.

1. Entity-Relationship Diagram

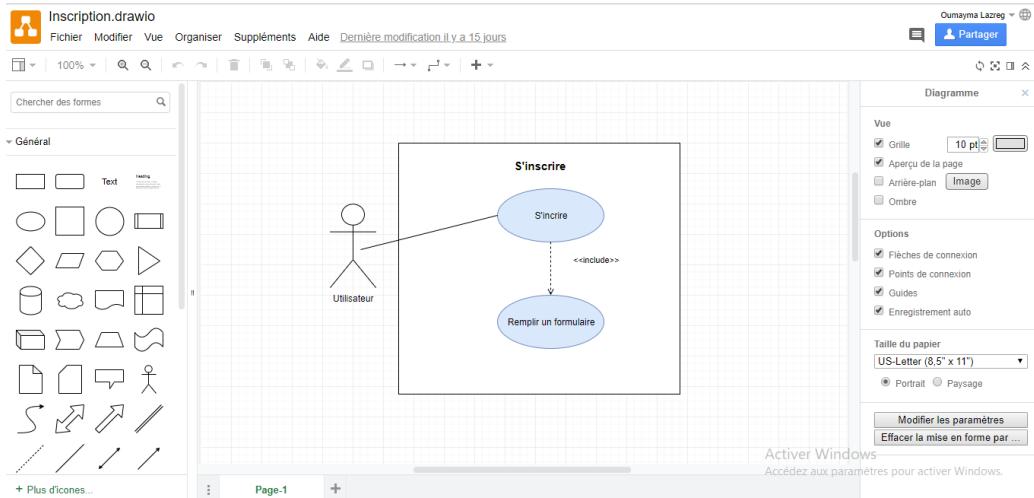


FIGURE 4.3 – Draw.io

Firebase

Firebase est une plateforme de développement des application mobile(android et iOS) et des application web soutenu par Google. Cette plateforme fournit plusieurs fonctionnalités tel que authentication, crashlytics, realtime database, analytics, Cloud Storage.[7]

Parmi ses fonctionnalités nous avons utilisé :

- **Authentification Firebase :**

Firebase gère les utilisateurs de façon sécurisé.

- **Realtime Database :**

Realtime Database est une base de données NoSQL qui peut être hébergé sur une plateforme ou dans un cloud. Les données sont stockées sous le format de JSON de manière synchronisées en temps réel.

- **FCM(Firebase Cloud Messaging) :**

FCM est solution de messagerie qui permet d'envoyer et de recevoir des message et des notification sur plusieurs plateforme sans frais.

- **Cloud Storage :**

Cloud Storage permet de stocker différents types de données :image, son, vidéo dans un cloud ou sur plusieurs plateformes.

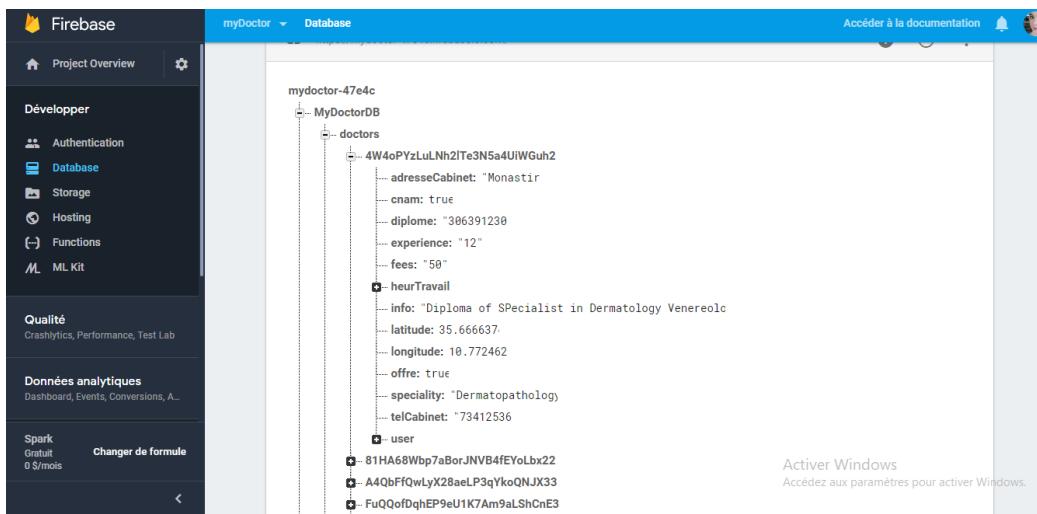


FIGURE 4.4 – Firebase

Adobe illustrator cc 2017

Pour la création du logo de l'application nous avons utilisé Adobe illustrator cc 2017. Adobe illustrator cc 2017 est un logiciel de dessin graphique vectoriel dont la qualité d'image reste la même en modifiant sa taille. Ce logiciel sert à créer des logos, des icônes et des typographies.



FIGURE 4.5 – Adobe illustrator cc 2017

4.1.2.2 API et bibliothèques utilisée

API Symptom Checker

L’API Symptom Checker est un API de vérification de symptômes médicales proposé par priaid.com. Cet API permet d’estimer les maladies possibles en se basant sur des symptômes saisies par un patient. Le fonctionnement de cette API est basé sur les techniques d’intelligence artificielle. Nous pouvons l’intégrer à des applications mobile ou web indépendamment du langage de programmation utilisé [6].

Cette API nous a permis de créer le **chatbot** qui aide le patient à reconnaître et traiter sa maladie dans le cas des maladies non complexes. Cette API est intégrée dans notre application en utilisant l’API **Retrofit** (voir tableau4.1).

Bibliothèques

Bibliothèque	Utilisation
Retrofit	Cette bibliothèque est un client REST pour Android et java qui consomme les web services d'une manière sécurisée. Retrofit consomme les données sous format JSON ou XML ² qui sont par la suite analysées par des objets POJO ³ . Elle exécute les méthodes HTTP GET, POST, PUT, DELETE et PATCH[3]. Dans notre application nous utilisons Retrofit pour consommer les données sous format JSON de l'API Symptom Checker et la méthode GET pour l'exécution.
Facebook API	la bibliothèque Facebook API assure l'authentification à travers le réseau social Facebook.
Google Maps API	Nous avons utilisé l'API Google Maps pour la localisation des cabinets des médecins.
Picasso	Nous avons utilisé la bibliothèque Picasso dans notre application afin d'afficher les images stockées dans le firebase.

Firebase API	la bibliothèque Firebase API utilisée pour profiter des services Firebase
Dagger 2	<p>Dagger 2 est une bibliothèque d'injection de dépendances basée sur les annotations qui génèrent automatiquement une série de classe pour fournir des dépendances à différentes parties d'une application[5]. Dagger 2 utilise les annotation suivante :</p> <ul style="list-style-type: none"> — <code>@Module</code> :employée avec la classe injecteur d'objets qui représentent les dépendances. — <code>@Provides</code> :employée avec les méthodes de la classe Module qui retournent les objets à injecter.. — <code>@Inject</code> :employée avec un constructeur, un champ ou une méthode et indique qu'une dépendance a été demandée. — <code>@Component</code> : la classe Module ne fournit pas la dépendance directement à la classe qui la demande. Pour cela, une interface de composant utilisée sert de pont entre <code>@Module</code> et <code>@Inject</code>.
RXJava	<p>RXJava est une implémentation de la bibliothèque ReactiveX⁴ qui compose les programmes asynchrones et se base sur les événements en utilisant des séquences observables[4]. RxJava utilisent les termes suivante :</p> <ul style="list-style-type: none"> — Asynchrone : les composants de l'application s'exécute simultanément. — Événement : le code est exécuté en fonction des événements générés. — Observable : défini en tant qu'un appel asynchrone.

TABLE 4.1 – Bibliothèques

4.1.2.3 Langage de développement

Kotlin

Kotlin est défini comme un langage de programmation orientée objet de type statique, interopérable par la machine virtuelle Java, les bibliothèques Java et Android. Ce langage est parfait pour développement des applications Android étant donné que le code kotlin est plus court et moins redondant[8].

XML⁵

Le langage XML permet au développeur de créer des formats d'information et de la partager via les réseaux. Ce standard utilisé pour décrire les données.

Json⁶

Json est un format d'échange de données léger, facile à lire et à écrire et fréquemment utilisé par les développeurs[1].

4.1.3 Diagramme de déploiement de l'application

Le diagramme de déploiement modélise l'architecture physique d'un système. Ce diagramme montre la topologie des éléments logiciels qui sont déployés sur les éléments matériels.

La figure 4.6 représente le diagramme de déploiement de notre application.

5. Extensible Markup Language

6. JavaScript Object Notation

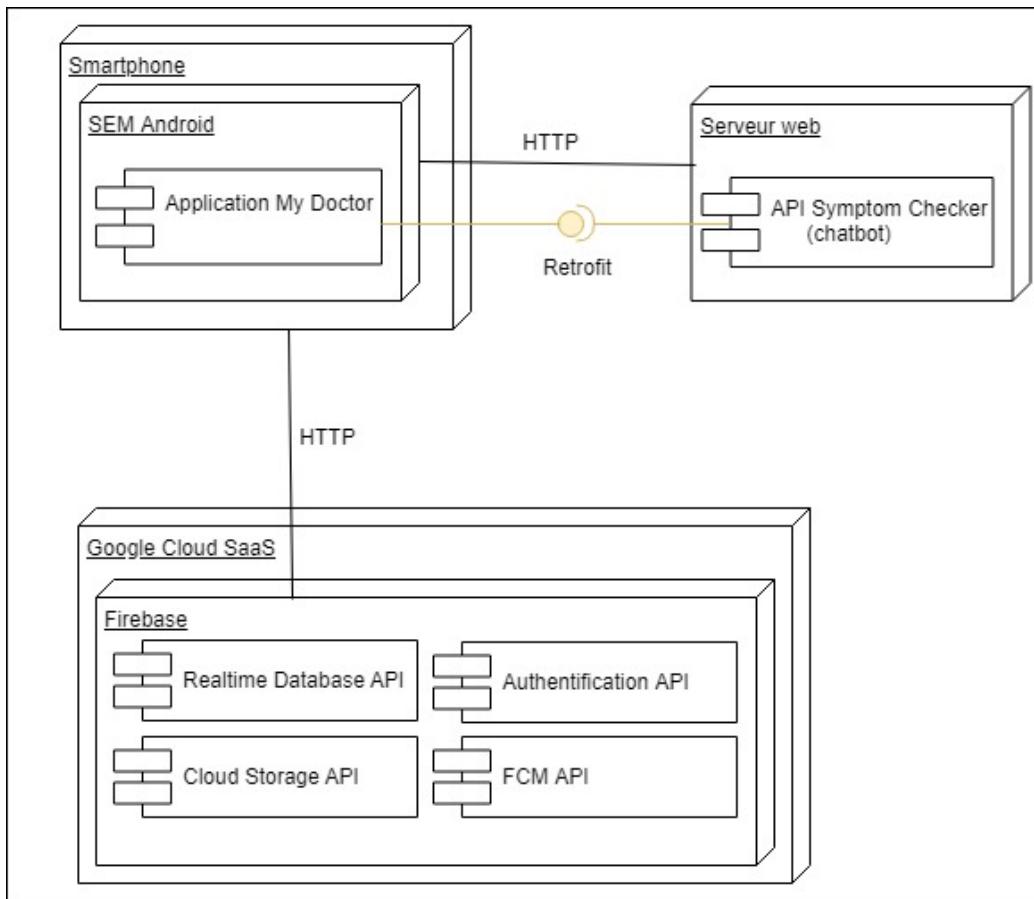


FIGURE 4.6 – Diagramme de déploiement de notre application

Le diagramme de déploiement de notre application se compose de 3 nœuds :

- Smartphone : l'appareil qui embarque le système d'exploitation mobile Android sur lequel le fichier .apk de notre application doit être installé.
- Serveur web : c'est le serveur web qui embarque l'API Symptom checker consommée par l'application et qui représente le back-end de notre Chatbot.
- Cloud SAAS⁷ : c'est le modèle qui distribue le firebase à travers le cloud pour l'utiliser comme un backend de notre application. Nous avons utilisé : Authentification Firebase API, Cloud Storage API, FCM(Firebase Cloud Messaging) et RealTime Database API.

Les interactions entre composants logiciels sont :

7. Software as a Service

- entre notre application "MyDoctor" et les APIs Firebase via les requêtes HTTP. L'échange des données est sous format JSON à travers les APIs REST⁸.
- entre notre application "MyDoctor" et l'API Symptom Checker via les requêtes HTTP. L'échange des données se fait à travers le retrofit sous format JSON.

4.2 Interfaces de l'application

Dans cette partie, nous présentons les différentes fonctionnalités de notre application en présentant les interfaces graphiques réalisées :

4.2.1 Lancement de l'application

La figure 4.7(a) représente le splash screen qui est la première fenêtre affichée de l'application, elle indique l'ouverture de l'application, et elle reste affichée pendant quelques secondes le temps que l'application se charge. Cette interface inclut le logo de l'application.

La figure 4.7(b) représente l'interface du choix du rôle. Cette interface contient deux types de texte : Patient et Doctor et deux images : une image pour le patient et l'autre pour le médecin. Dès que l'utilisateur clique sur l'image qui définit son rôle une interface d'authentification s'affiche.

8. representational state transfer)

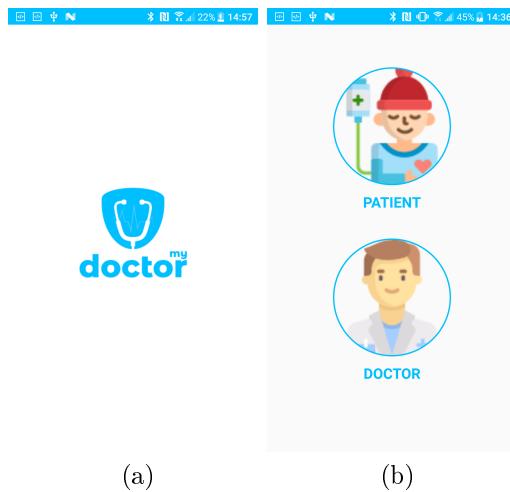


FIGURE 4.7 – Interfaces de lancement de l’application

4.2.2 Authentification

La figure 4.8 explique le processus d’authentification. Elle contient deux champs d’édition de texte : Login et Password ,un bouton « Login » et une case à cocher avec le texte « Keep me signed in» qui permet à un utilisateur d’accéder à toutes ses données, même après l’expiration de sa session. Au cas où, l’utilisateur oublie son mot de passe, il doit cliquer sur le champs de texte « Forget Password ?», ainsi l’interface 4.8(c) s’affiche. Le système envoie un émail à l’utilisateur contenant un lien qui lui permet de réinitialiser son mot de passe.

Les boutons « Contenu with Facebook »,« Contenu with Twitter» et « Contenu with Google+» permet de se connecter à partir du réseau sociaux.

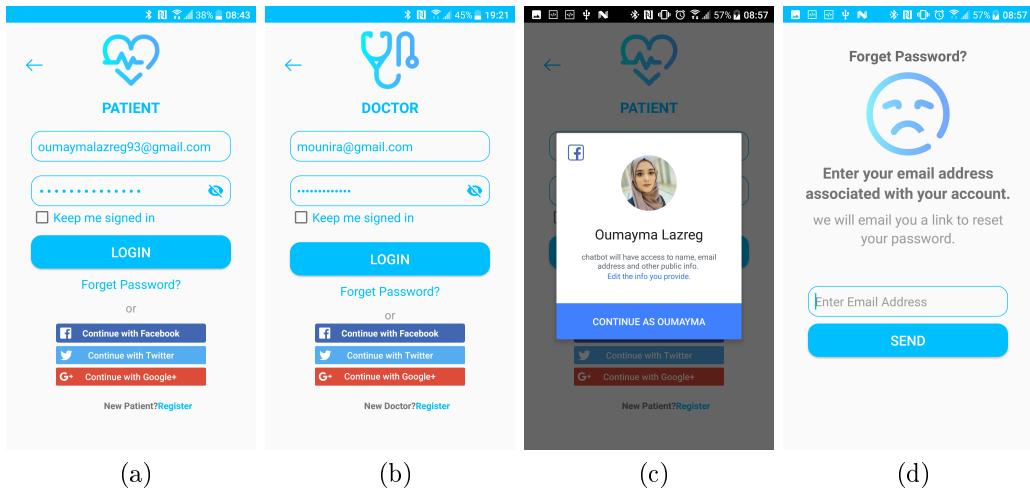


FIGURE 4.8 – Interfaces «Authentification »

4.2.3 Inscription

L’interface inscription contient les champs d’éditions : FirstName, LastName, Date Of Birth, Email, Password, Phone Number, une image, un spinner pour la sélection de gouvernement, un bouton switch pour définir le sexe de l’utilisateur et un bouton submit.

Si l’utilisateur est un patient (figure 4.9) alors son interface contient de plus deux champs : Weight et Height et un spinner Blood group pour sélectionner son groupe sanguin.

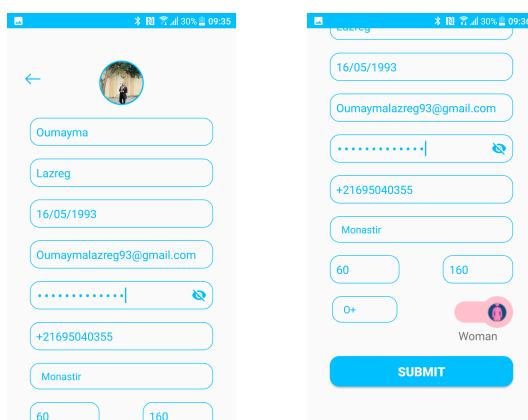


FIGURE 4.9 – Interfaces « Incription de patient »

Si l'utilisateur est un médecin (figure 4.10) alors alors son interface contient de plus les champs : phone office, fees, more information ,deux spinner : select your government, select your number of year experience, une zone d'image pour ajouter l'image de son diplôme. Quand, le médecin clique sur le champs Add your location un google maps s'affiche pour récupérer ses coordonnées géographique. Cette interface contient aussi deux boutons radios Cnam et Offer et bouton submit. Quand, le médecin clique sur le champs «Select your opening time », l'interface 4.10(c) s'affiche.

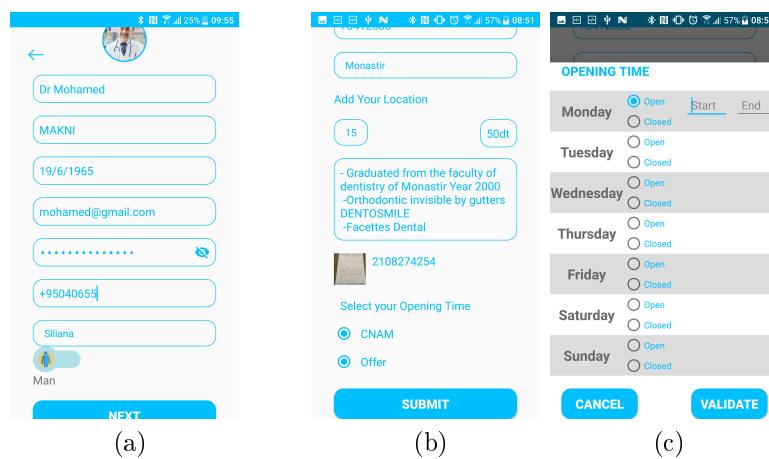


FIGURE 4.10 – Interfaces « Incription de médecin »

4.2.4 Interface d'accueil

Une fois l'utilisateur s'est authentifié, il est redirigé vers l'interface d'accueil de l'application (figure 4.11).

Cette interface contient les listes des médecins les plus proches, les médecins les plus évalués par les patients, les médecins qui sont conventionnés avec le cnam ainsi qu'une liste d'articles contenant des études médicales.

Quand l'utilisateur clique sur la liste « Medical Studies » un article détaillé s'affiche (figure 4.11(c)).

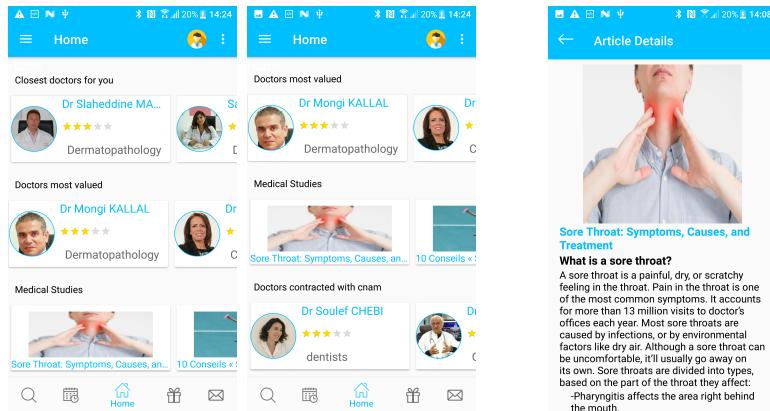


FIGURE 4.11 – « Interface d'accueil »

4.2.5 Menu principal

La figure 4.19 représente l'interface du menu principal du compte de l'utilisateur. Dans cette interface la photo et les informations personnelles de l'utilisateur seront affichées. De plus l'utilisateur aura l'accès au menu principal de l'application qui permet de basculer entre les fonctionnalités suivantes :

- Afficher le profil de l'utilisateur
- Afficher l'interface d'accueil
- Afficher l'interface de Recherche d'un médecin
- Afficher l'interface de consultation en ligne
- Appeler l'administrateur
- Afficher l'interface des paramètres de l'application.
- Se déconnecter

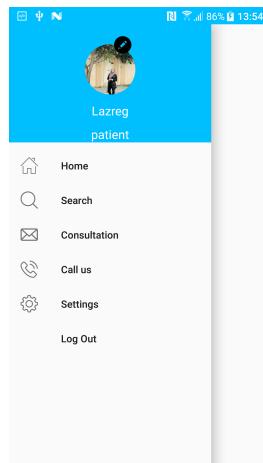


FIGURE 4.12 – Interface «Menu principal »

4.2.5.1 Profile utilisateur

L'interface 4.13(a) nous présente toutes les informations liées au profil de chaque patient :la photo de patient, le nom et le prénom, le sexe, la taille, le poids, le groupe sanguin, la date de naissance, l'email, le mot de passe, le numéro de téléphone et l'adresse.

les interfaces 4.13(b) et 4.13(c) présentent toutes les informations liées au profil de chaque médecin :la photo de médecin, le nom et le prénom, le sexe, la date de naissance, l'email, le mot de passe, le numéro de téléphone, la spécialité, le numéro de téléphone de cabinet, l'adresse de cabinet qui est affiché dans le google maps, le prix de consultation, le nombre d'années d'expérience et la calendrier de travail.

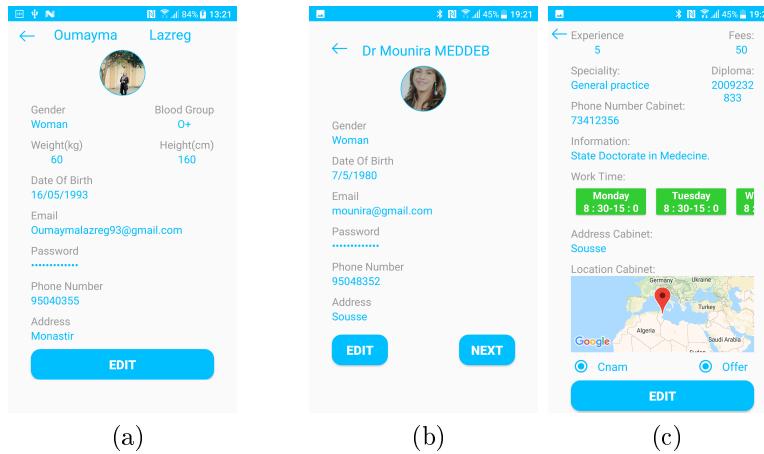


FIGURE 4.13 – Interfaces « Profile utilisateur »

4.2.6 Recherche d'un médecin

La figure 4.14 représente les interfaces de la fonctionnalité de recherche d'un médecin. L'interface principale permet de rechercher un médecin selon son nom, sa spécialité ou selon son gouvernement. Après la recherche d'un médecin une liste de médecins s'affiche avec leurs coordonnées : nom, prénom , le nombre d'étoiles données par les patients, spécialité, numéro de téléphone du cabinet et le gouvernement du cabinet. Pour que le patient réserve un rendez vous, il suffit qu'il clique sur le bouton « réserve ».

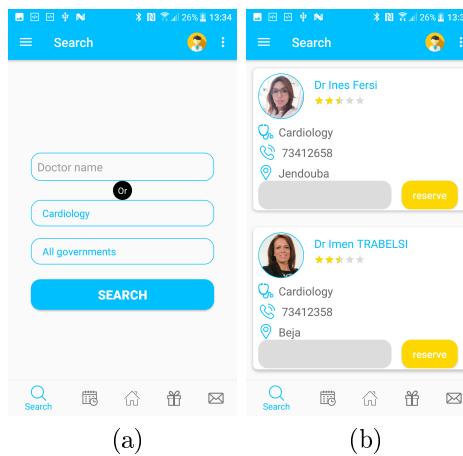


FIGURE 4.14 – Interfaces « Recherche médecin »

4.2.7 Prendre rendez-vous

Quand le patient clique sur le bouton « réserve » de l'interface « Liste des médecins » (figure 4.14(b)), il sera redirigé vers l'interface « Prendre rendez-vous » (figure 4.15(a)). Cette interface contient des informations concernant le médecin choisi : son calendrier de travail qui s'affiche dans une liste pour que le patient choisisse la date et l'heure du rendez-vous, sa localisation qui s'affiche dans le google maps.

Quand le patient prend un rendez-vous, une notification est envoyée au médecin concerné qui peut accepter ou refuser le rendez-vous.

Ensuite, l'interface 4.15(b) s'affiche. Cette interface contient une liste de tous les rendez-vous du patient connecté. Cette interface contient aussi la date et l'heure du rendez-vous, des informations sur le médecin concerné, une icône de poubelle permettant de supprimer le rendez-vous et un bouton « Wait confirmation ». Le patient peut donner son avis sur la consultation du médecin (figure 4.15(c)).

Dès que le médecin reçoit la notification, l'interface de figure 4.15(d) s'affiche. Cette interface contient une liste de tous les rendez-vous du médecin connecté. Cette interface contient aussi la date et l'heure de rendez-vous, des informations sur le patient. L'interface de figure 4.15(d) contient aussi deux boutons : si le médecin veut accepter le rendez-vous il clique sur le bouton « Accepte » et dans ce cas la couleur du bouton « Wait Confirmation » de l'interface de figure 4.15(b) change en vert et le texte devient « Confirmed » sinon il clique sur le bouton « Refuse » et la couleur de bouton « Wait Confirmation » de l'interface de figure 4.15(b) change en rouge et le texte devient « Refused ».

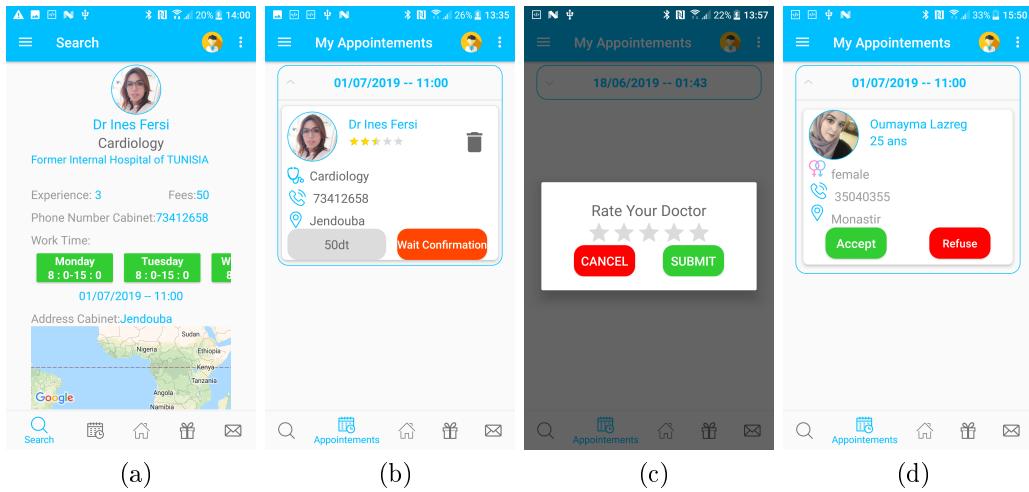


FIGURE 4.15 – Interfaces « Prendre rendez-vous »

4.2.8 Consultation en ligne

La figure 4.16 représente les interfaces de la fonctionnalité consultation en ligne. L'interface de figure 4.16(a) affiche la liste des médecins avec leurs noms et leurs spécialités. Le patient peut sélectionner le médecin désiré pour passer une consultation en ligne.

Ensuite, il sera redirigé vers l'interface (figure 4.16(b)) L'interface de figure 4.16 (b) contient un champs d'édition « New message » permettant à l'utilisateur de saisir un message et de le transmettre, comme il peut envoyer et recevoir des images qui peuvent contenir des informations médicales (ordonnance, bilan, radiologie...).

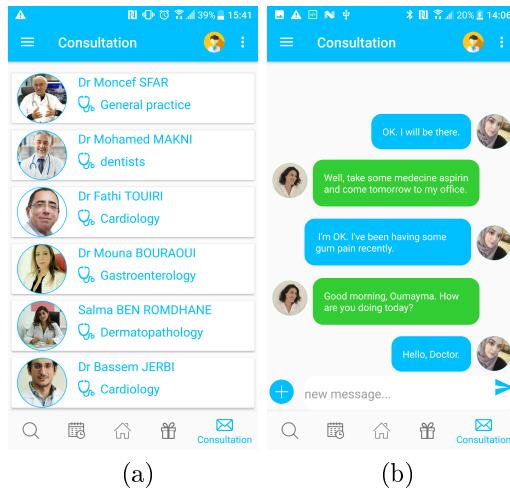


FIGURE 4.16 – Interface « Consultation en ligne »

4.2.9 Chatbot

Les figures 4.17 et 4.18 représentent les interfaces du Chatbot.

Lorsque l'utilisateur clique sur l'icône du chatbot, l'interface de figure 4.17(a) affiche une liste de symptômes. Le système change la couleurs des symptômes sélectionnés en doré. Lorsque l'utilisateur clique sur le bouton « Next » de l'interface 4.17(a), l'interface 4.17(b) s'affiche. Après la sélection du sexe ,l'utilisateur sera redirigé automatiquement vers l'interface 4.17(c).

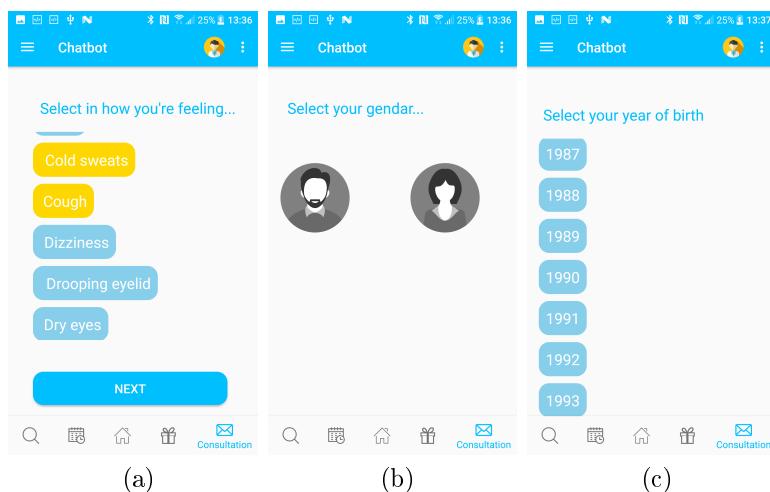


FIGURE 4.17 – Interfaces « Sélection des symptômes »

Quand l'utilisateur sélectionne l'année de sa naissance, l'interface 4.18(a) s'affiche. Cette interface représente la réponse du Chatbot qui est la liste des maladies estimées selon les informations saisies par l'utilisateur, pondérées par des pourcentages probabilistes. Cette interface contient aussi de courtes descriptions de ces maladies.

Lorsque l'utilisateur clique sur « What to do next », le système affiche l'interface 4.18(b) qui donne une description des traitements que le patient peut suivre selon sa maladie. Le patient peut également voir la liste des médecins les plus proches, il suffit qu'il clique sur « Click here when you're ready » et l'interface 4.18(d) s'affiche.

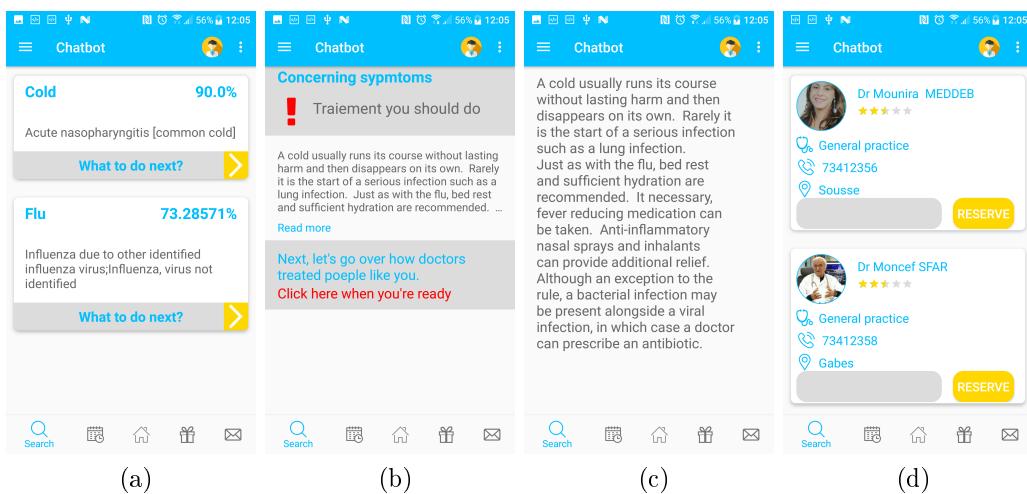


FIGURE 4.18 – Interfaces « Estimation de maladie »

4.2.10 Offres

L'interface 4.19 contient des offres annoncées par les médecins. A chaque offre, un ensemble d'informations est associé : coordonnées du médecin, description, délai...

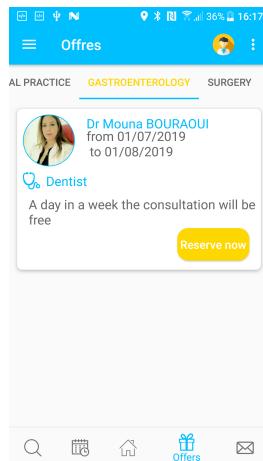


FIGURE 4.19 – Interface «Offres »

Conclusion

Dans ce chapitre, nous avons présenté la spécification technique de notre application par l'introduction des environnements matériel et logiciel. Finalement, nous avons décrit les fonctionnalités réalisées de notre application, illustrées par des imprimés écrans de ses interfaces utilisateurs. Nous clôturons notre rapport par la conclusion générale et les différentes perspectives futures.

Conclusion et perspectives

Certes, la diversité des applications mobiles est un enrichissement au monde de la mobilité, et à la vie sociale, ou tout passe par ces applications pour faciliter les tâches quotidiennes et apporter plus de confort, de gain de temps et d'argent.

L'objectif visé à travers ce rapport est de présenter l'application réalisée au cours de mon stage de projet fin d'étude au sein de la société « LorDroid ». Notre application permet aux patients de faire des consultations médicales et de prendre des rendez-vous en ligne et de chercher le médecin le plus proche, elle permet également aux médecins de publier des offres. L'application réalisée intègre aussi un Chatbot qui donne la possibilité d'estimer automatiquement la maladies à partir d'un ensemble de symptômes entrés. Ce chatbot propos aussi le traitement médical adéquat.

Ce stage m'a permis d'améliorer mes connaissances techniques étant donné que nous avons utilisés une variété de technologies. D'autre part, durant ce stage, je me suis habituée à la vie professionnelle et au travail en équipe.

Pour conclure, notre application réalisée peut être étendue en développant la fonctionnalité d'administrateur prévue dans la conception, en ajoutant la possibilité de consultation en ligne audiovisuelle (communication directe entre le patient et le médecin à travers un appel vocal ou un appel vidéo)... Il est possible aussi de rendre notre application multilingues, et de point de vue technique, il est possible de la rendre multi-plateformes. Une application mobile multi-plateformes est une application qui peut être installée sur différents systèmes d'exploitation mobiles, principalement Android et iOS.

Références

- [1] *JSON*. <https://www.json.org/>. [Accés le 1 Juin 2019].
- [2] *Clean Coder*. <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>, 2012. [Accés le 10 février 2019].
- [3] *Meduim*. <https://medium.com/cr8resume/make-your-hand-dirty-with-retrofit-2-a-type-safe-http-client-for-android-and-java-c546f88b3a51>, 2017. [Accés le 16 Mai 2019].
- [4] *MindORks*. <https://blog.mindorks.com/rxjava-anatomy-what-is-rxjava-how-rxjava-is-designed-and-how-rxjava-works-d357b3aca586>, 2017. [Accés le 17 Mars 2019].
- [5] *Developer Life*. <https://developerlife.com/2018/10/21/getting-started-with-dagger2/>, 2018. [Accés le 17 Mars 2019].
- [6] *ApiMedic*. <https://apimedic.com/>, 2019. [Accés le 15 Mai 2019].
- [7] *Firebase*. <https://firebase.google.com/products>, 2019. [Accés le 3 Février 2019].
- [8] *LeMagIT*. <https://www.lemagit.fr/definition/Kotlin>, 2019. [Accés le 10 Février 2019].
- [9] *POSTMAN*. <https://www.getpostman.com/>, 2019. [Accés le 2 Mai 2019].
- [10] *SearchMobileComputing*. <https://searchmobilecomputing.techtarget.com/definition/Android-Studio>, 2019. [Accés le 1 Février 2019].

- [11] *TICE Education.* <https://www.tice-education.fr/index.php/tous-les-articles-er-ressources/articles-internet/819-draw-io-un-outil-pour-dessiner-des-diagrammes-en-ligne>, 2019. [Accés le 15 Février 2019].
- [12] Belgacem, Dr. Selma: *Les patrons d'architecture (2) Clean Architecture. (ISSAT de Sousse)*. (35), 2019. [Accés le 18 juin 2019].