



ECOLE SUPÉRIEURE PRIVÉE D'INGÉNIERIE ET DE TECHNOLOGIES

www.esprit.tn - E-mail : contact@esprit.tn

Siège Social : 18 rue de l'Usine - Charguia II - 2035 - Tél. : +216 71 941 541 - Fax. : +216 71 941 889

Annexe : Z.I. Chotrana II - B.P. 160 - 2083 - Pôle Technologique - El Ghazala - Tél. : +216 70 685 685 - Fax. : +216 70 685 454



2020 - 2021

PROJET DE FIN D'ÉTUDES

DIPLÔME NATIONAL D'INGÉNIEUR

SPÉCIALITÉ : INFORMATIQUE

**Conception et développement d'une Solution
E-Health pour les malades
D'Alzheimer**

Réalisé par : Atef MADDOURI

Encadrant ESPRIT : Mme Ahlem MARZOUK

Encadrant Entreprise : Mme Soumaya ARGOUBI



J'autorise l'étudiant Atef MADDOURI à faire le dépôt de son rapport de stage en vue d'une soutenance.

Encadrant professionnel, **Madame Soumaya ARGOUBI**

Signature et cachet



J'autorise l'étudiant Atef MADDOURI à faire le dépôt de son rapport de stage en vue d'une soutenance.

Encadrant académique, **Madame Ahlem MARZOUK**

Signature et cachet



Dédicaces

Je dédie ce travail à :

À mes très chers parents **Zhayra et Ali**,

Tant de phrases, aussi expressives soient-elles, ne peuvent exprimer mon respect, mon amour éternel et ma gratitude pour tous les sacrifices que vous avez consentis pour mon éducation et mon bien-être. Ce succès, je vous le dois. Vous le méritez plus que moi.

À mes chers frère et soeur, et à ma grand-mère **Fatma** l'amour de ma vie

À ma tante **Hayate** et mon oncle **Salah**, Je n'oublierai jamais vos encouragements et votre soutien qui m'ont poussé à m'améliorer et à aller de l'avant.

À toute ma famille et tous mes amis

À mes chers **collègues** qui rendent la période de stage agréable et pour leurs encouragements constants.

À toute personne qui a contribué au succès de mon projet de fin d'études,
que ce soit d'une façon direct ou indirect.

Cette aventure était passionnante et agréable.



Atef MADDOURI

Remerciements

Avant toute chose, je tiens à remercier ALLAH le Tout-Puissant de m'avoir donné la foi, la force, le courage et la volonté, ainsi que de m'avoir permis d'y arriver.

*Le chemin que j'ai choisi de suivre n'a pas été facile mais mon encadrante d'entreprise, **Madame Soumaya ARGOUBI**, était toujours présente avec nous, avec ses connaissances et sa sagesse pour assurer un bon déroulement du projet.*

Je tiens à la remercier tout particulièrement du fond du cœur.

*Je suis reconnaissant aussi à **Madame Ahlem MARZOUK** pour l'encadrement, l'aide et la confiance qui nous a été accordée ainsi que pour tous ses conseils et son soutien durant ce stage. Pour le temps qu'elle m'a consacré tout au long de cette période en répondant à toutes mes questions sans oublier sa participation à l'élaboration de ce rapport.*

*Enfin, je tiens à remercier les **membres du jury** de m'avoir fait l'honneur d'accepter d'évaluer ce travail, tout en souhaitant pour eux d'y trouver les qualités de clarté et de motivation qu'ils attendent.*



Table des matières

Introduction générale	1
1 Cadre générale du projet	3
1.1 Présentation de l'organisme d'accueil	4
1.1.1 Structure d'ESPRIT-Tech	5
1.2 Contexte du projet	5
1.3 Problématique	5
1.4 Choix de la méthodologie de développement	5
1.4.1 Etude comparative des méthodologies	6
1.4.2 Méthode adoptée	6
2 Etat de l'art	8
2.1 Étude de l'existant	9
2.1.1 WatchHelp	9
2.1.2 App'Zheimer+	10
2.1.3 Backup Memory app	11
2.2 Critique de l'existant et solution proposée	12
3 Analyse et spécification des besoins	14
3.1 Capture des besoins	15
3.1.1 Identification des acteurs	15
3.1.2 Analyse des besoins fonctionnels	15
3.1.3 Analyse des besoins non fonctionnels	16
3.2 Spécification des besoins	17
3.2.1 Diagramme de cas d'utilisation	17
3.2.2 Diagramme des classes d'analyse	18
4 Analyse technique	19
4.1 Choix de l'architecture de L'application	20
4.1.1 Etude comparative sur les architectures	20

4.1.2	Architecture micro-service	20
4.1.3	Architecture adoptée	21
4.2	Capture des besoins techniques	22
4.3	Cloud patterns	23
4.3.1	Une configuration externe	23
4.3.2	La découverte et l'enregistrement des services	24
4.3.3	Gateway API	24
4.4	La conteneurisation	26
4.5	DevOps	26
4.5.1	L'intégration continue (CI)	27
4.5.2	Déploiement continu (CD)	27
4.6	Autres concepts clés :	27
4.6.1	La Sécurité de système Distribué	27
4.6.2	La documentation des API	28
4.6.3	La qualité de code	28
5	Conception	30
5.1	Architecture globale de la solution	31
5.1.1	Architecture micro-service de l'application	31
5.1.2	Architecture physique	32
5.1.3	Architecture logique	32
5.2	Aspect statique	34
5.2.1	Composants de l'application	34
5.3	Paquetages de l'application	35
5.4	Diagramme de classe de conception	36
5.5	Aspect Dynamique	38
5.5.1	Diagramme de séquence objet	38
6	Réalisation	40
6.1	Environnement de travail	41
6.1.1	Environnement logiciel	41
6.2	Description de l'application	42

Conclusion générale **51**

Bibliographie **52**

Table des figures

1.1	Les étapes du processus 2TUP	7
3.1	Diagramme de cas d'utilisation	17
3.2	Diagramme de classe d'analyse	18
4.1	Architecture Générique	22
4.2	Serveur de configuration	23
4.3	Api gateway	25
5.1	Architecture micro-services de l'application	31
5.2	Architecture Physique	32
5.3	Architecture logique de l'application	33
5.4	Diagramme de composant	34
5.5	Diagramme de composants du cas d'utilisation « gérer ses surveillants »	35
5.6	Diagramme de paquetages de micro service « pillbox service »	36
5.7	Diagramme de classe de conception	37
5.8	diagramme de séquence de conception	38
6.1	La page Home	43
6.2	L'interface d'authentification avec des coordonnées invalides	43
6.3	Les différentes "NavBar" selon le role	44
6.4	Interface de consultation du profil	44
6.5	Interface du mise à jour du profil	45
6.6	Interface du mise à jour du profil	45
6.7	L'interface du gestion des patients	46
6.8	L'interface du gestion de pilulier	46
6.9	Ajout d'un nouvel traitement	47
6.10	La modification et la suppression du traitement	47
6.11	Consulter les détails du médicament	48
6.12	La page de gestion des médicaments	48
6.13	Interface d'ajout d'un nouveau médicament	49

6.14 Consulter ses médecins	49
6.15 Consulter ses proches aidants	50
6.16 Consulter ses traitements données par le médecin	50

Liste des tableaux

1.1	Étude comparative entre la méthodologie lourde et agile	6
3.1	Liste des fonctionnalités par acteur	15
4.1	La différence entre les deux architectures	21
6.1	La liste des logiciels utilisés	41

Liste des abréviations

- **2TUP** = Two Tracks Unified Process
- **API** = Application Programming Interface
- **CD** = Continuous Delivery
- **CI** = Continuous Integration
- **IDE** = Integrated Development Environment
- **JWT** = Json Web Token

Introduction générale

Dans ce projet, nous mettons l'accent sur les patients atteints de la maladie d'Alzheimer, une maladie neurologique qui est devenue très courante au cours de la dernière décennie. Il s'agit d'un type de démence qui touche la plupart des personnes âgées. Dans ce type de maladie, une personne devient inconsciente parce qu'elle n'est pas capable d'effectuer les tâches quotidiennes de manière indépendante et a besoin qu'un membre de sa famille s'occupe toujours de son comportement et surtout de sa santé. Par conséquent, pour les familles ayant des patients atteints de la maladie d'Alzheimer, le coût de l'embauche d'une infirmière ou des soins continus pour ce patient est élevé. Cependant, on espère pouvoir faciliter la tâche pour leurs proches aidants en utilisant les moyens que la technologie peut fournir, afin de réduire les dépenses supplémentaires.

Dans ce contexte, notre travail vise à concevoir et développer une solution destinée aux personnes atteintes de cette maladie. Notre solution consiste à centraliser les informations relatives à leurs traitements. Afin de faciliter l'alternance entre leurs superviseurs (aidants familiaux | médecins). Autrement, tous les proches aidants d'un patient auront une idée du traitement qu'il doit prendre à un moment donné. La date finale de ce traitement. De cette façon, sa famille n'a pas besoin d'engager quelqu'un pour le surveiller 24 heures sur 24, ils doivent juste partager la surveillance entre eux.

En général, les êtres humains sont toujours désireux de trouver de nouvelles idées et d'améliorer et faciliter la vie quotidienne, surtout s'ils sont ingénieurs. Pour cette raison, nous voulons nous assurer que cette première version du projet soit un point de départ solide et fiable pour les prochaines améliorations. Par conséquent, nous avons essayé de choisir et de travailler avec les techniques qui le permettent.

Ce rapport est divisé en 6 chapitres :

Dans son chapitre premier, qui porte le titre « Cadre générale du projet », on commence par la présentation de l'organisme d'accueil, par la suite le Contexte du projet, ainsi que la problématique et le Choix de la méthodologie de développement adoptée selon une étude comparative.

Le deuxième chapitre, appelé « Etat de l'art », est consacré à l'étude de l'existant et à la solution proposée.

Le troisième chapitre, sous le titre « Analyse et spécification des besoins » présente l'étude des besoins fonctionnels et non fonctionnels, Ainsi que la spécification de ces besoins.

Le quatrième chapitre « Analyse technique » définit notre choix de de l'architecture de L'application, et les concepts nécessaires à la réalisation de ce projet, en citant les solutions adoptées et en indiquant leurs valeurs ajoutées pour notre sujet.

Le chapitre consacré à la conception est le quatrième chapitre de notre rapport, intitulé « Conception ». Dans

Introduction générale

ce chapitre, nous présentons nos architectures ainsi que l'aspect statique et dynamique de notre projet.

Enfin, dans le dernier chapitre, dédié à la « Réalisation », nous présentons la palette d'outils utilisés ainsi que le résultat obtenu par un scénario présentant les différentes interfaces de l'application. la « conclusion générale » clôt ce travail.

CADRE GÉNÉRALE DU PROJET

Plan

1	Présentation de l'organisme d'accueil	4
2	Contexte du projet	5
3	Problématique	5
4	Choix de la méthodologie de développement	5

Introduction

Ce chapitre comprend, dans un premier temps, la présentation de l'organisme d'accueil et de sa structure. Ensuite, nous exposons la problématique du projet. Et enfin, nous présentons le Choix de la méthodologie de développement adoptée selon une étude comparative.

1.1 Présentation de l'organisme d'accueil

L'École supérieure privée d'ingénieries et de technologies (ESPRIT) fondée en 2003 à l'initiative de trois universitaires ayant conduit de nombreux projets dans l'enseignement supérieur tunisien, et dans l'enseignement supérieur technique en particulier, entourés de plusieurs dizaines de leurs collègues, ainsi que d'entreprises TIC et de partenaires financiers, a dès le départ mis en place des formations basées sur des valeurs intangibles :

- Par son adhésion en avril 2014 à la CGE qui regroupe environ 110 établissements français et étrangers qui forment leurs diplômés dans un souci constant d'excellence
- Par son adhésion à l'initiative CDIO fondé par le MIT qui regroupe 116 institutions dont 2 seulement sur le continent africain. CDIO a pour objectif est de recentrer la pédagogie des formations d'ingénieurs autour de mises en situation professionnelle. [1]



Le projet a été réalisé au sein de la structure de Recherche-Développement-Innovation (RDI) à ESPRIT.

1.1.1 Structure d'ESPRIT-Tech

ESPRIT-Tech est une structure de recherche dirigée par un directeur qui supervise tous les aspects stratégiques et opérationnels. Pour mener à bien son activité, ESPRIT-Tech a mis en place une structure répondant à sa mission : les équipes se créent afin de réaliser des projets RDI autour de thématiques et d'axes de recherche. Des étudiants peuvent être associés à ces travaux par le biais de leurs projets de fin d'études. ESPRIT-Tech a créé en décembre 2015 un comité de recherche : C-RDI (Comité de RDI), composé de tous les responsables des équipes. Il est chargé de dynamiser les activités de recherche et de veiller à la qualité des productions. Il se réunit en moyenne une fois par mois. Pour un meilleur fonctionnement, le « Comité de RDI » s'est doté d'un secrétariat (SC-RDI) composé de 5 membres qui se réunissent régulièrement. Il a pour rôle de résoudre les problèmes en appliquant les décisions prises par le C-RDI.[2]

1.2 Contexte du projet

Ce projet qui porte le titre « **Conception et développement d'une Solution E-Health pour les malades D'Alzheimer** » est un stage de fin d'études qui nous permet de mettre en pratique ce que nous avons appris à l'École privée d'ingénierie et de technologie afin d'obtenir un diplôme national d'ingénieur en informatique, triplement certifié par les normes internationales EUR-ACE, CGE et CDIO.

1.3 Problématique

Dans ce projet, nous nous focaliserons sur les malades d'Alzheimer, qui est le trouble neurologique le plus courant de la dernière décennie et qui touche la plupart des personnes âgées. Cette maladie rend la personne incapable d'accomplir ses tâches quotidiennes de manière autonome et elle nécessite qu'un membre de la famille s'occupe en permanence de son comportement, de sa santé et de sa sécurité.

1.4 Choix de la méthodologie de développement

Dans le domaine du développement de logiciels, la méthodologie choisie est une étape décisive. De nombreuses méthodologies de développement de logiciels permettent la réalisation d'un logiciel. Celles-ci englobent toutes les étapes de développement d'un logiciel. Les méthodes lourdes et agiles sont les deux grandes familles de méthodologies de développement logiciel.

1.4.1 Etude comparative des méthodologies

Tableau 1.1: Étude comparative entre la méthodologie lourde et agile

Méthodologie	Méthode	Description	Avantages	Inconvénients
Lourde	2TUP	Il s'agit d'une méthode qui permet de séparer les contraintes fonctionnelles des contraintes techniques.	<ul style="list-style-type: none"> — Assure la séparation des exigences fonctionnelles, non fonctionnelles et techniques. — Itérative et incrémental pilotée par les risques. 	Ne propose pas de document standard.
Agile	Scrum	Recourt à de courtes itérations appelées sprints.	S'adapte avec les besoins du client.	Documentation insuffisante.

1.4.2 Méthode adoptée

Le choix de la méthodologie de développement est délicat vu la diversité des méthodologies existantes.

Nous expliquons donc le choix de 2TUP par cette raison

Ce projet a une certaine complexité au niveau technique, et de ce fait, il fallait absolument une phase d'étude de toutes les exigences techniques à prendre en compte avant de démarrer la réalisation des spécifications fonctionnelles.

Le processus 2TUP offre un cycle de développement qui sépare les aspects techniques des aspects fonctionnels ; il est subdivisé en deux grands axes, qui sont un axe fonctionnel et un axe technique, dont les résultats sont combinés afin de réaliser le système. Ce qui conduit à un cycle de développement en forme de Y, illustré dans la figure ci-dessous, que nous suivrons tout au long de notre travail.

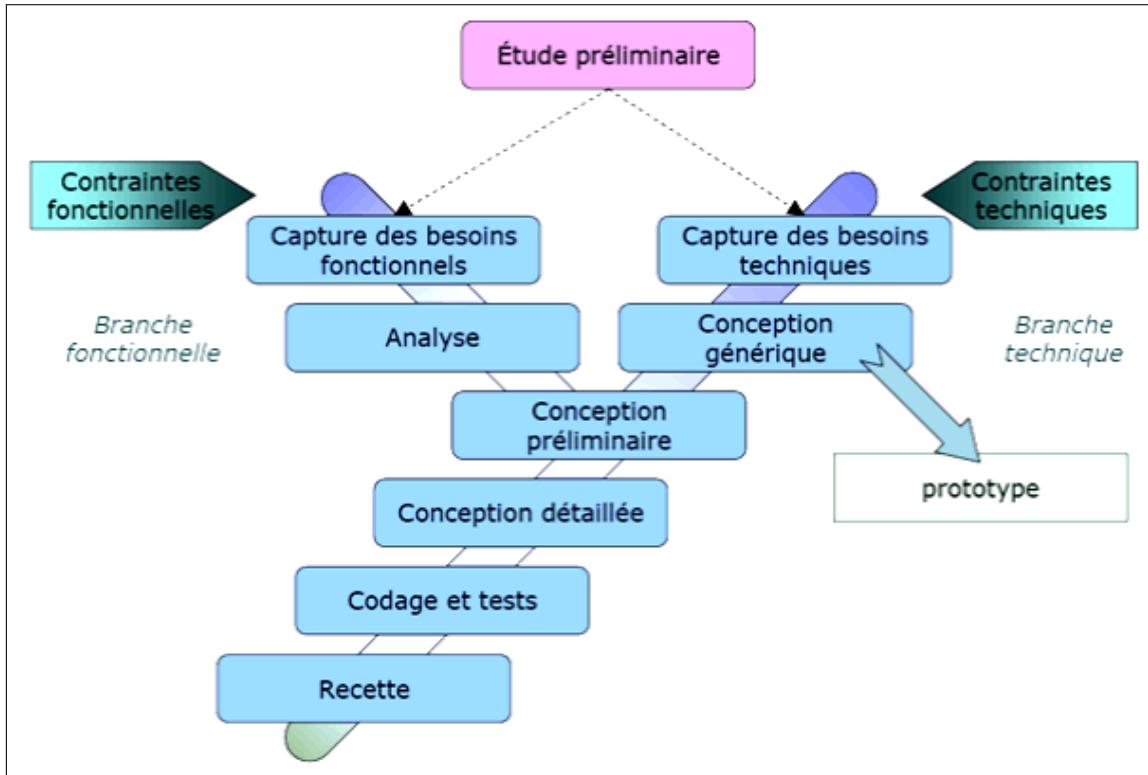


Figure 1.1: Les étapes du processus 2TUP

Conclusion

A travers ce chapitre, nous avons commencé la partie introductive du rapport par une présentation de l'organisation d'accueil, en comprenant le contexte général du projet afin d'exprimer la problématique du travail ainsi que la méthodologie adoptée. Pour se faire une idée plus précise du projet, le chapitre suivant est consacré à l'étude de l'existant et à la solution proposée.

ETAT DE L'ART

Plan

1	Étude de l'existant	9
2	Critique de l'existant et solution proposée	12

Introduction

Après la présentation du cadre général de notre projet. Nous présentons dans ce chapitre une étude des solutions existantes afin de mieux comprendre cet univers de E-Health et les problèmes confrontés par les patients d'Alzheimer et leur famille.

2.1 Étude de l'existant

La réalisation d'une étude détaillée des systèmes existants sur le marché professionnel. permet de mieux définir le périmètre de notre projet, les axes fondamentaux de notre solution et de disposer d'une vision claire du travail. Dans ce contexte, nous avons étudié le fonctionnement de certains applications disponibles sur le marché principalement WatchHelp, APP'ZHEIMER+ et Backup Memory app.

2.1.1 WatchHelp

Cette application permet de programmer les tâches que la personne atteinte de la maladie d'Alzheimer ou autre doit effectuer (accrocher un repas, faire sa toilette, etc.), avec des rappels en temps réel.

Elle comporte trois fonctions :

- **Séquentielle** : les tâches à effectuer, décrites étape par étape,
- **Un planning** : images ou textes présentés sur une journée, une semaine ou un mois,
- **Des mémos** : sortes de fiches pratiques pour rappeler à la personne Alzheimer comment réagir dans certaines situations.



Prenons l'exemple d'une personne atteinte de la maladie d'Alzheimer. L'application et la montre fonctionnent de la manière suivante

L'aideant programme un séquentiel sur le thème par exemple de la toilette. Il définit les différentes tâches que la personne doit effectuer à une heure précise. A 8h00, la smartwatch de la personne Alzheimer vibre et affiche une image avec un rappel qu'il est temps d'aller prendre une douche. Ensuite, les différentes étapes sont indiquées sur la montre : "Mettre du savon", etc. La montre demande à la personne si elle a effectué la tâche, si elle répond non, elle lui envoie immédiatement un rappel. Si le patient Alzheimer répond deux fois non ou ne répond pas du tout, la montre transmet un message pour prévenir l'aideant.



2.1.2 App'Zheimer+

Cette application a pour objectif d'aider au diagnostic de la maladie d'Alzheimer. Conçue par des professionnels de la santé, l'application APP'ZHEIMER+ est conçue comme une aide au diagnostic permettant de détecter et de suivre les troubles de la mémoire et des fonctions intellectuelles, notamment dans la maladie d'Alzheimer. Elle est destinée aux professionnels de santé.

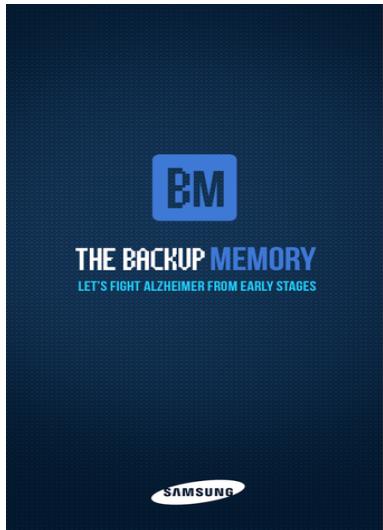


App'Zheimer comporte deux niveaux d'évaluation :

- Un premier niveau qui permet d'identifier les premiers symptômes de la maladie grâce à un test rapide en 4 questions.
 - et un second niveau, un autre test plus long, composé de 30 questions, inspiré de l'échelle du Mini Mental State Examination (MMSE), qui explore la mémoire immédiate, évalue les fonctions cognitives et permet ainsi d'établir un diagnostic plus précis pour assurer un meilleur suivi des différents troubles.
- Il est désormais en vente au prix de 5,99 €.

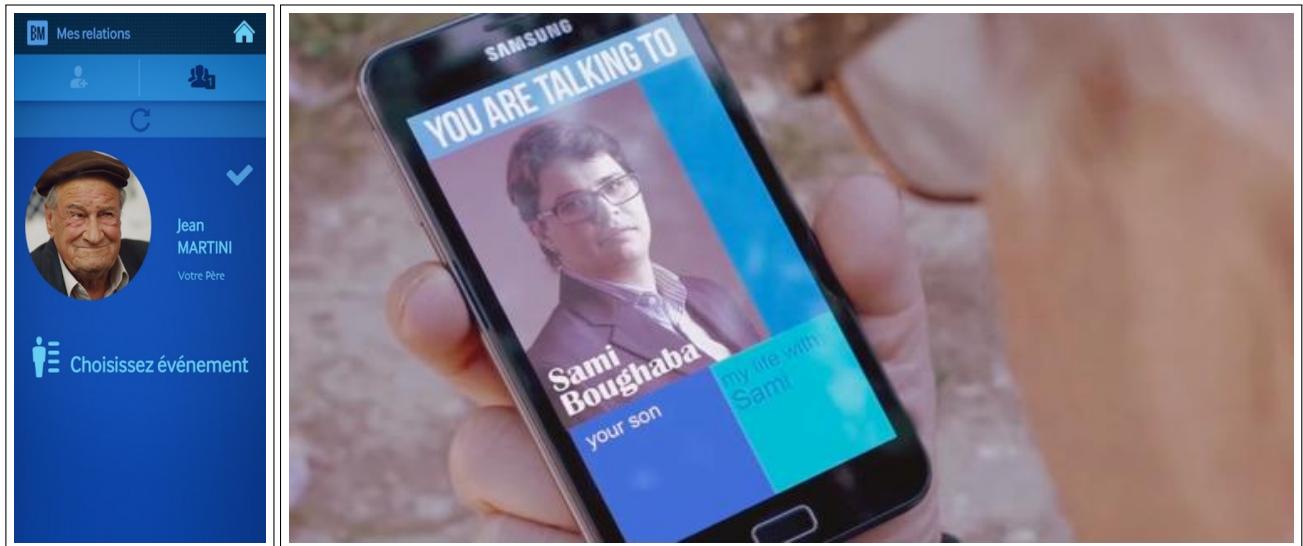
2.1.3 Backup Memory app

Ce projet a été développé par les employés de Samsung Electronics Tunisia (SETN). Azer Jaafoura, responsable marketing des produits mobiles de la SETN, a dirigé le projet. "Nous voulions faciliter la vie des patients atteints de la maladie d'Alzheimer et de leurs soignants, mais nous ne savions pas comment nous y prendre", a déclaré Azer Jaafoura dans un communiqué. "Bien que la maladie d'Alzheimer soit incurable, des études récentes ont montré que la stimulation mentale sous forme de rappels réguliers d'événements passés pouvait potentiellement ralentir la progression de la maladie. C'est là que nous avons vu une opportunité."



Backup Memory, disponible en français et en anglais sur les smartphones Android, est conçue pour ceux qui présentent des signes précoce de la maladie d'Alzheimer ; elle stimule des souvenirs spécifiques. Cette application aide les patients à rester conscients de leur environnement proche en identifiant les membres de la famille à proximité - ceux qui se trouvent dans un rayon de 33 pieds. Les membres de la famille devront avoir une version de l'application téléchargée sur leur téléphone également. L'application rappelle également aux patients la relation qui unit chaque membre de leur famille en leur montrant des photos et des vidéos d'expériences passées qu'ils ont partagées avec eux.

Ces médias sont synchronisés manuellement avec l'application avant d'être utilisés. L'Association tunisienne d'Alzheimer a effectué les premiers tests de l'application. "Backup Memory n'est pas exactement une percée technologique, mais c'est certainement une bonne utilisation de la technologie existante", a déclaré Jaafoura. "Nous sommes satisfaits des progrès réalisés jusqu'à présent, mais nous aimerais en faire plus. Il y a tellement de choses positives que l'on peut faire avec la technologie, et j'aime à penser que nous ne faisons que commencer."



2.2 Critique de l'existant et solution proposée

Etant donné que le coût pour engager une infirmière en permanence est élevé. Et même c'est difficile de trouver quelqu'un qui capable de surveiller le patient 24h/24h et 365jr/an, Et il faut mentionner que le respect de temps de prise est très important. En revanche, le proche aidant peut parfois non seulement de ne pas donner les médicaments à la même heure mais d'en oublier. Les applications existantes n'offrent pas des fonctionnalités qui peuvent remédier à ces problèmes que nous l'avons détectés dans la vie courante. Pour cela, nous pensons à mettre en place un système de surveillance pour les personnes âgées spécifiquement ceux qui sont atteintes de la maladie d'Alzheimer en utilisant les moyens que la technologie peut offrir afin de réduire les dépenses supplémentaires et facilite la tâche pour les proches aidants. Cette plateforme centralisera les informations liées à leurs traitements. Afin de faciliter l'alternance entre leurs superviseurs (proches aidants | médecins). Autrement, tous les proches aidants d'un malade pourront avoir une idée sur le traitement qu'il doit prendre à un moment donné. La date finale de ce traitement. Donc, sa famille a plus besoins d'embaucher quelqu'un pour le surveiller 24h/24h, il suffit seulement de partager entre eux la surveillance.

Conclusion

Pour conclure, nous avons résumé dans ce chapitre les résultats de l'étude préalable que nous avons réalisée durant les premières semaines de notre stage. Cette dernière a permis à la fois de comprendre le système existant et de mieux comprendre la problématique. Cette partie nous a bien préparé à aborder le chapitre suivant "Analyse et spécification des besoins".

ANALYSE ET SPÉCIFICATION DES BESOINS

Plan

1	Capture des besoins	15
2	Spécification des besoins	17

Introduction

Après l'étude de l'existant réalisée dans le chapitre précédent, nous nous intéressons à l'identification des fonctionnalités de l'application que nous envisageons de mettre en œuvre, c'est-à-dire la branche fonctionnelle de la méthodologie 2TUP. Nous commençons ce chapitre par présenter les acteurs, les exigences fonctionnelles et les exigences non fonctionnelles de notre solution. Ensuite, nous aborderons la partie spécification des exigences qui comprend le diagramme des cas d'utilisation, le diagramme des classes d'analyse.

3.1 Capture des besoins

Cette phase est un passage essentiel dans le cycle de développement d'un projet. En effet, elle permet de préciser les fonctionnalités de notre application en identifiant les acteurs et en décrivant leur interaction avec le système.

3.1.1 Identification des acteurs

Cette Application fera intervenir 3 Acteurs principaux

- Patient
- Médecin
- Proche-Aidant

3.1.2 Analyse des besoins fonctionnels

Notre Application mettra à la disposition du chaque acteur les fonctionnalités suivantes

Tableau 3.1: Liste des fonctionnalités par acteur

Acteur	fonctionnalités
Patient	Gérer son compte. Consulter la liste de ses surveillants (un médecin /un proche-Aidant). Recherche multicritère sur la liste des surveillants disponibles. Envoyer une demande de surveillance. Annuler une demande avant qu'elle soit acceptée par le surveillant. Mettre fin à une surveillance. Consulter son pilulier.

Médecin	<ul style="list-style-type: none"> Gere son compte. Consulter la liste de ses patients. Accepter Refuser une demande de surveillance. Recherche multicritère sur la liste des demandes de surveillance. Consulter le pilulier de chaque patient. Ajouter un traitement à un patient. Modifier Supprimer un traitement seulement dans le même jour de son ajout. Consulter la liste de toute les médicaments. Recherche multicritère sur la liste des médicaments. Ajouter un médicament.
Proche-Aidant	<ul style="list-style-type: none"> Gere son compte. Consulter la liste de ses patients. Accepter Refuser une demande de surveillance. Recherche multicritère sur la liste des demandes de surveillance. Consulter le pilulier de chaque patient.

3.1.3 Analyse des besoins non fonctionnels

Notre système devra répondre à un ensemble de critères assurant une meilleure qualité de la solution développée. Nous citons parmi ces critères

- **Utilisabilité (La facilité d'utilisation)** : L'application doit garantir une meilleure expérience utilisateur. Son interface doit être conviviale, facile à utiliser et à comprendre afin de rendre la navigation plus fluide et de repérer les services souhaités.
- **Sécurité** : Il est nécessaire de procéder à une authentification pour que l'utilisateur puisse utiliser l'application. De plus, les droits d'accès et les permissions doivent être pris en compte pour protéger les ressources.
- **Maintenabilité** : Réduire le temps nécessaire pour détecter une anomalie dans l'application et la corriger. Il est donc nécessaire de structurer correctement le code source.

3.2 Spécification des besoins

Dans cette partie, nous modélisons les exigences fonctionnelles du système à travers un diagramme de cas d'utilisation et une classe d'analyse, afin d'avoir un aperçu global du fonctionnement de notre application.

3.2.1 Diagramme de cas d'utilisation

La figure ci-dessus représente le diagramme des cas d'utilisation qui résume les différentes fonctionnalités de l'application.

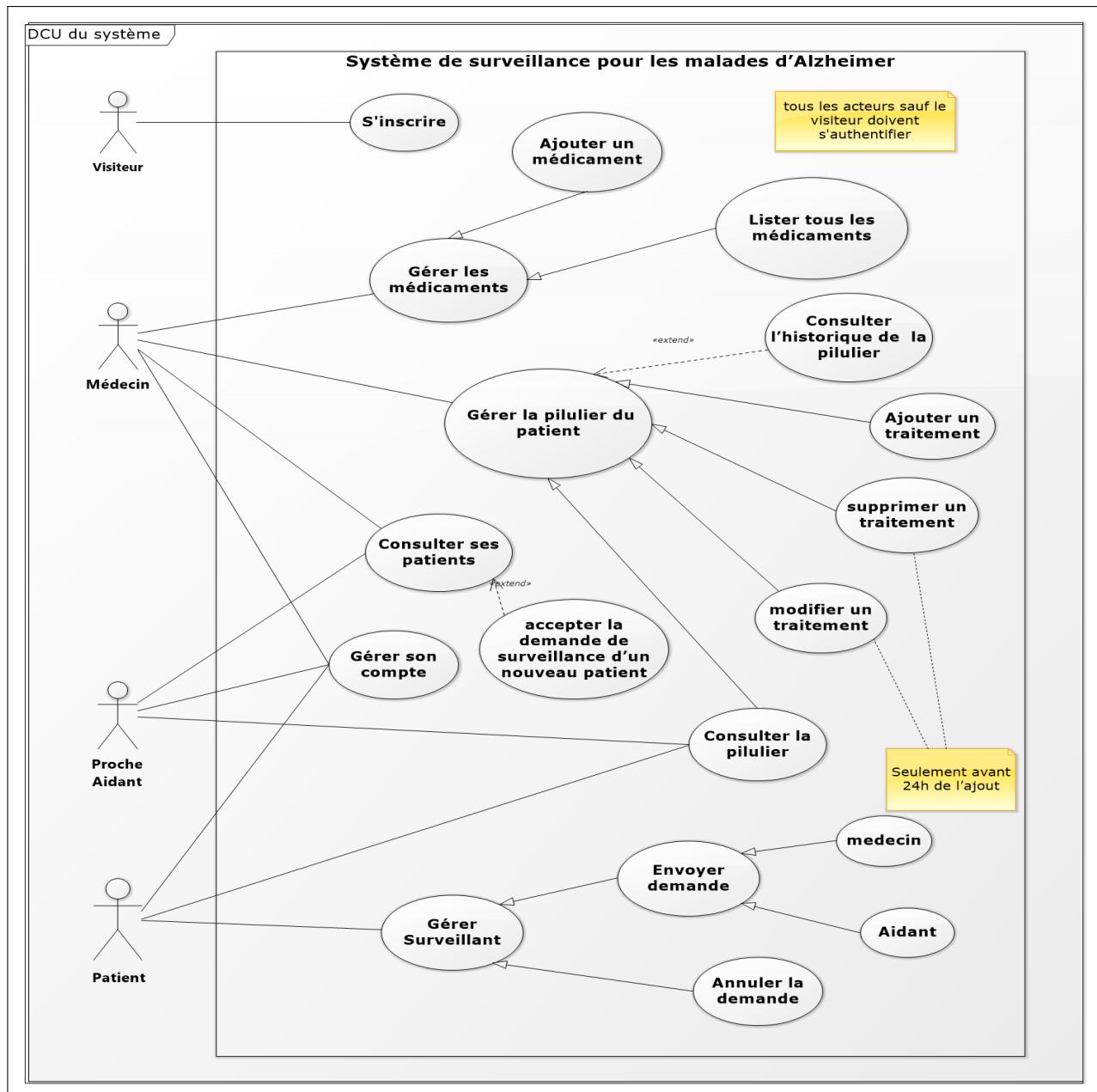


Figure 3.1: Diagramme de cas d'utilisation

3.2.2 Diagramme de classes d'analyse

La figure ci-dessous représente le diagramme de classes d'analyse qui résume les différentes entités de l'application.

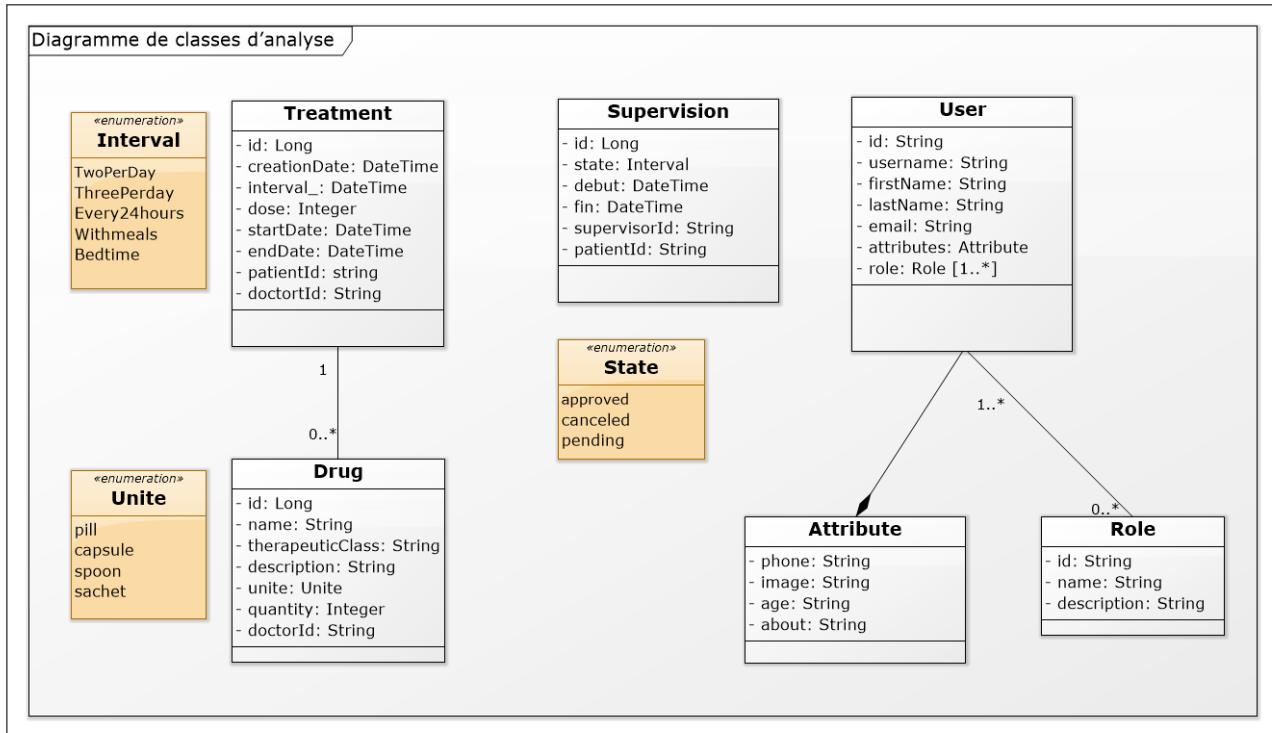


Figure 3.2: Diagramme de classe d'analyse

Conclusion

Dans ce chapitre, nous avons mis l'accent sur l'analyse et la spécification des besoins de notre solution. Nous avons commencé par la capture des besoins, en identifiant les parties prenantes, l'analyse des besoins fonctionnels et l'analyse des besoins non fonctionnels. Ensuite, nous avons modélisé le projet par un diagramme de cas d'utilisation détaillé avec un diagramme de classe d'analyse.

ANALYSE TECHNIQUE

Plan

1	Choix de l'architecture de L'application	20
2	Capture des besoins techniques	22
3	Cloud patterns	23
4	La conteneurisation	26
5	DevOps	26
6	Autres concepts clés :	27

Introduction

Après avoir présenté le chapitre sur l'analyse et la spécification des besoins. Dans ce chapitre, nous allons explorer l'architecture, les principaux concepts et techniques dont nous nous servons pour réaliser notre projet. Pour ce faire, nous démarrons la partie capture des besoins techniques par une étude comparative des architectures existantes et nous présentons l'architecture générique du système. Par la suite, nous aborderons chaque besoin en citant la solution adoptée. Puis, nous présenterons l'architecture physique et logique finale du projet. Pour finir, nous décrirons l'environnement technique.

4.1 Choix de l'architecture de L'application

Nous allons commencer cette section par une étude comparative en définissant les deux architectures monolithique et micro-service et en précisant leurs avantages et inconvénients puis nous citons notre choix en expliquant la raison.

4.1.1 Etude comparative sur les architectures

4.1.1.1 Architecture monolithique

Une application monolithique est une application développée dans un seul module, dans laquelle toutes les exigences fonctionnelles sont centralisées dans une seule application réalisée par une même technologie.

4.1.2 Architecture micro-service

Les microservices décomposent une application monolithique en plusieurs petits services autonomes qui sont faiblement couplés entre eux, appelés Micro-services. Chacun des microservices est une mini-application qui possède sa propre logique métier. La plupart des microservices mettent à disposition une API destinée à d'autres microservices ou aux clients légers.

4.1.2.1 La différence entre les deux architectures

Tableau 4.1: La différence entre les deux architectures

	Monolithique	Micro-service
ATOUTS	Simplicité Facile à déployer	-Couplage faible -Extensibilité (monté en charge) -Livraison en continu (approche CI/CD) -Facile à tester et à déployer - Une équipe hétérogène.
DÉFAUT	-Une seule technologie utilisée -Un point de défaillance unique (single point of failure) -Difficile à faire évoluer au niveau fonctionnel Chaque modification nécessite de : - Tester les régressions - Redéployer toute l'application - Livraison en bloc	Complexité

4.1.3 Architecture adoptée

Nous voulons que ce projet soit une base solide pour les prochaines mises à jour. De ce fait, nous allons opterons pour l'architecture micro-service. L'idée est de décomposer ce système en petites unités implémentées dans des micro-services. Chaque service est responsable d'une fonction et il est développé, testé et déployé séparément des autres micro-services. La technologie utilisée par chaque micro-service n'est pas obligatoirement la même que les autres.

Ceci est une Architecture Générique :

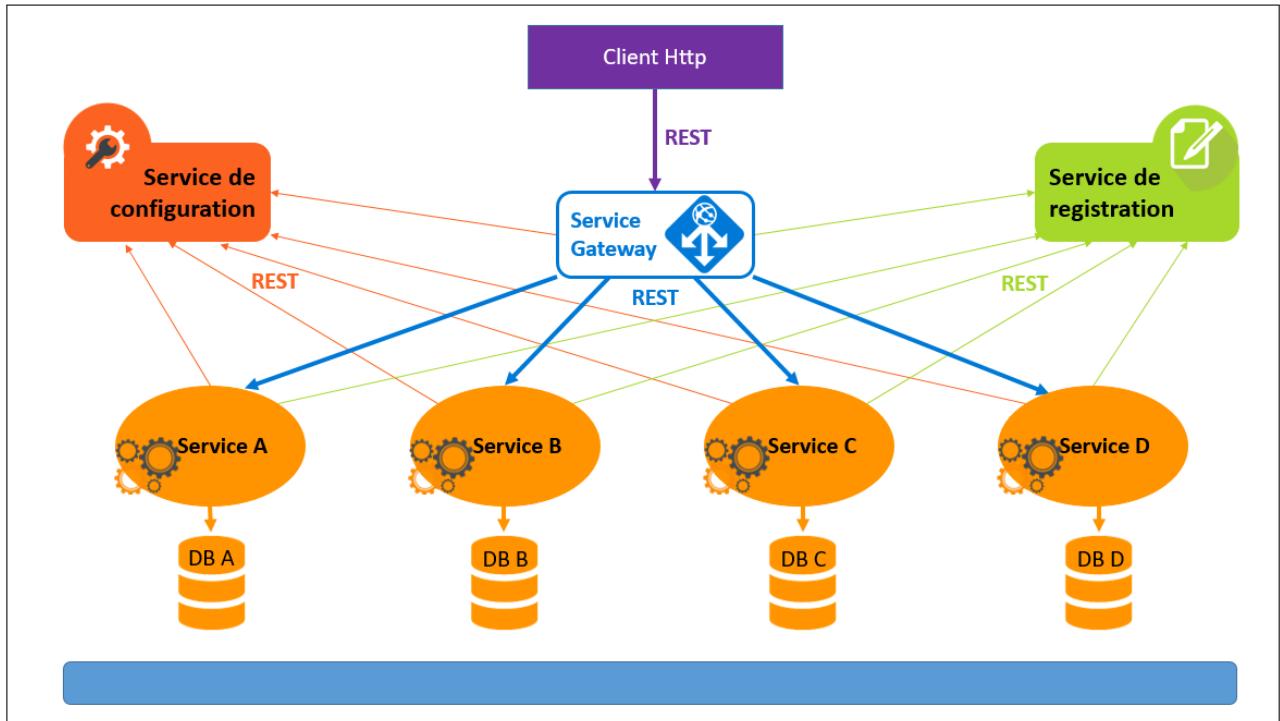


Figure 4.1: Architecture Générique

4.2 Capture des besoins techniques

Lorsque vous travaillez pour la première fois avec une architecture de microservices, vous devez vous interroger sur plusieurs points, dont :

- Comment on va assurer la communication entre les micro-services ?
- Comment centraliser la configuration de ces micro-services ?
- Comment on va gérer l'autorisation et l'authentification entre ces différentes micro-services ?

Et donc on pense automatiquement aux patrons. Ces patrons sont utiles pour créer des applications fiables, évolutives et sécurisées. Un patron est une solution réutilisable à un problème qui se pose dans un contexte particulier. Nous allons traiter ces patrons :

- Une configuration externe
- La découverte et l'enregistrement des services
- Gateway API

En ce qui concerne les besoins non fonctionnels, la sécurité et la maintenabilité ont été mentionnées. Il est

donc nécessaire de définir les méthodes permettant de résoudre ces problèmes avant de commencer la partie développement. Nous allons traiter comme solution :

- Keycloak
- La documentation des API
- SonarCloud

4.3 Cloud patterns

4.3.1 Une configuration externe

Les fichiers de configuration individuels limitent la configuration à une seule application alors que, dans certains cas, il est utile de partager les paramètres de configuration entre plusieurs applications. Prenons l'exemple des chaînes de connexion à la base de données utilisées par un ensemble d'applications liées. Supposons que nous devions mettre à jour l'url de la base de données. Cette opération doit être effectuée pour chaque micro-service. En effet, si nous oublions de mettre à jour cette donnée quelque part, cela peut entraîner une différence dans les paramètres de configuration des instances pendant le déploiement de la mise à jour.

Spring Cloud Config intervient à ce niveau en offrant une solution permettant de séparer complètement les fichiers de configuration de l'application. Ce service est pour centraliser la configuration de l'ensemble des micro-services dans un seul fichier de configuration qui est généralement stocké dans un repo GIT. Au démarrage le micro service demande sa configuration de ce service via les RESTs.

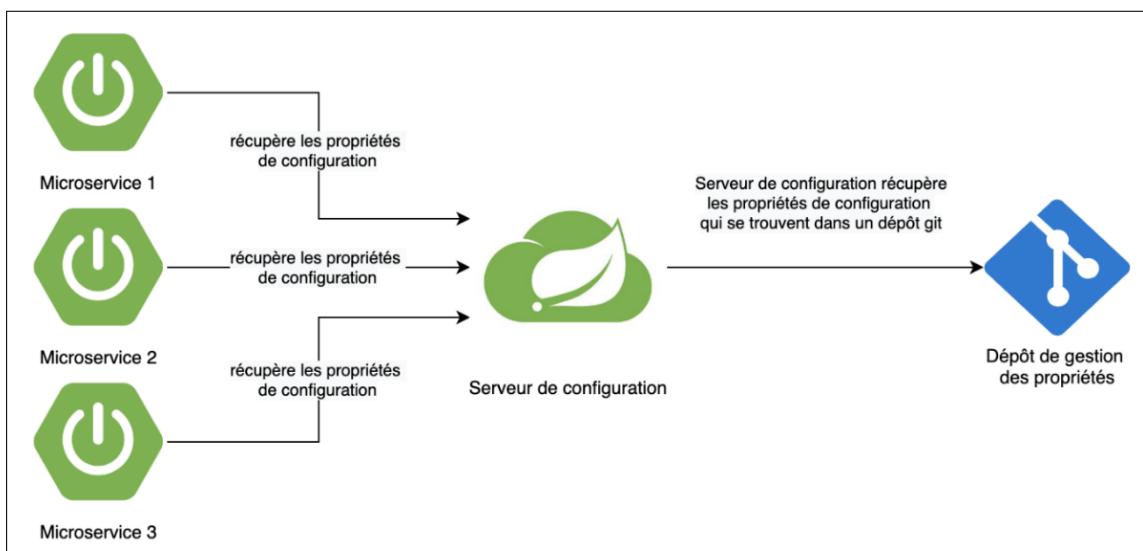


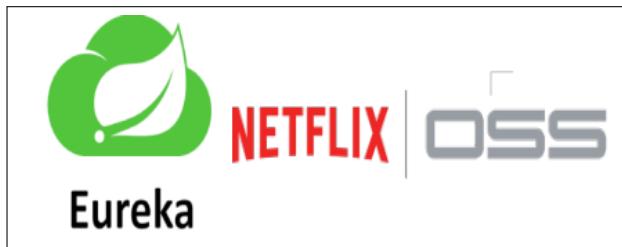
Figure 4.2: Serveur de configuration

4.3.2 La découverte et l'enregistrement des services

Dans le monde des microservices, il est important de disposer de services d'enregistrement et de découverte. En effet, nous exécutons souvent plusieurs instances de services. Et afin que ces services puissent communiquer entre eux, chacun d'entre eux doit être capable de déterminer l'emplacement du réseau (adresse IP et port) d'un service donné. Cependant, toutes ces adresses sont aléatoirement distribuées. En outre, un service peut avoir plusieurs instances qui peuvent ne pas être les mêmes en raison de l'auto-scaling, de pannes, etc.

Il nous faut donc un mécanisme d'enregistrement et de découverte automatique des services.

Le service de registration joue le rôle d'un annuaire qui contient Le nom, l'adresse et le port de chaque micro-service. Ce dernier(micro-service) s'enregistre automatiquement lors de son démarrage. Le service API Gateway sollicite ce service à chaque fois qu'il reçoit une requête pour l'acheminer au bon endroit. C'est là où le Eureka server intervient.



4.3.3 Gateway API

L'Api Gateway est **un proxy inverse** amélioré avec des fonctionnalités plus avancées, notamment l'orchestration, la sécurité et le monitoring.

Un proxy inverse est un serveur qui reçoit les requêtes d'un client, les transfère à un serveur qui peut y répondre et retourne au client les réponses du serveur.

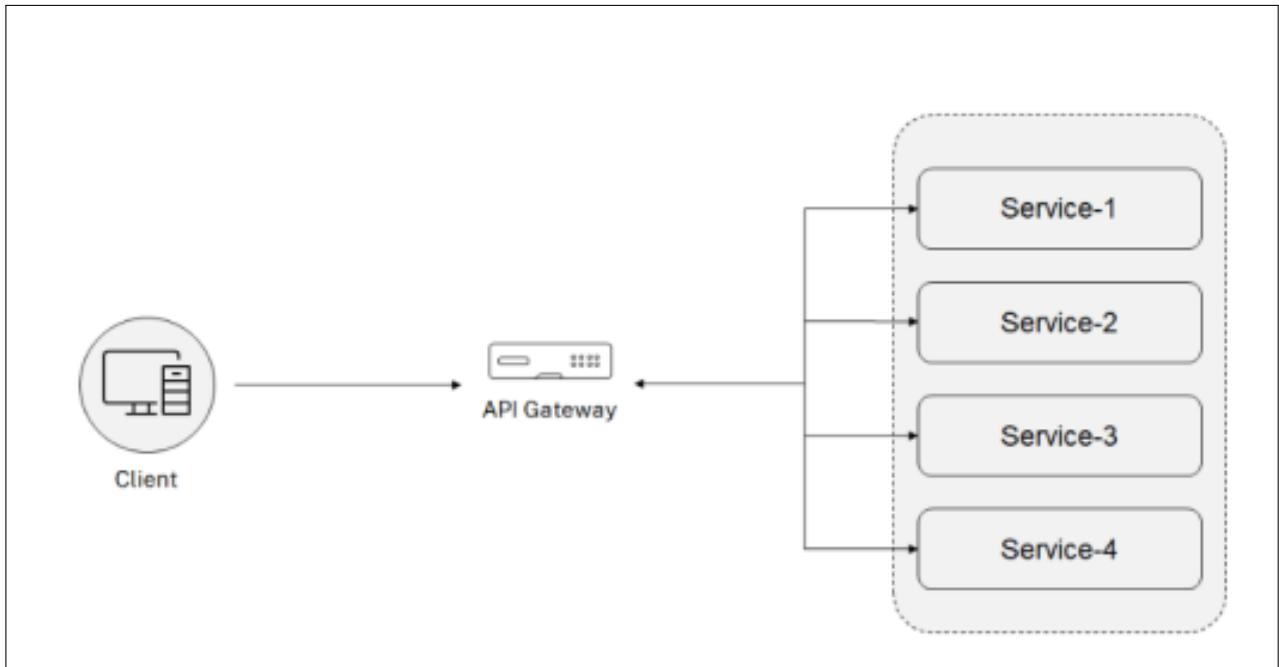


Figure 4.3: Api gateway

Parmi les implémentations les plus utilisées on cite Spring Cloud Gateway et Netflix Zuul Proxy.

4.3.3.1 Netflix Zuul Proxy

La première version de Zuul est basée sur le framework Servlet. Il assure un traitement synchrone bloquant des entrées/sorties. Cela signifie que les requêtes sont traitées en suivant un thread par connexion, qui restera bloqué jusqu'à la fin du traitement.

La version 2 de Zuul, basée sur Responsive Systems, a été publiée par Netflix, cependant Spring n'a pas intégré cette nouvelle version dans son écosystème. Il dispose actuellement de son propre serveur proxy appelé Spring Cloud Gateway.



4.3.3.2 Spring Cloud Gateway :

Spring Cloud Gateway est basé sur l'écosystème **Reactive Spring** et s'exécute sur le serveur Netty pour assurer la gestion asynchrone non bloquante des requêtes. En plus de cela, il fournit un moyen simple et efficace de transmission des requêtes reçues au bon destinataire grâce au Gateway Handler Mapping. Ce qui nous encourage à opter pour cette solution.



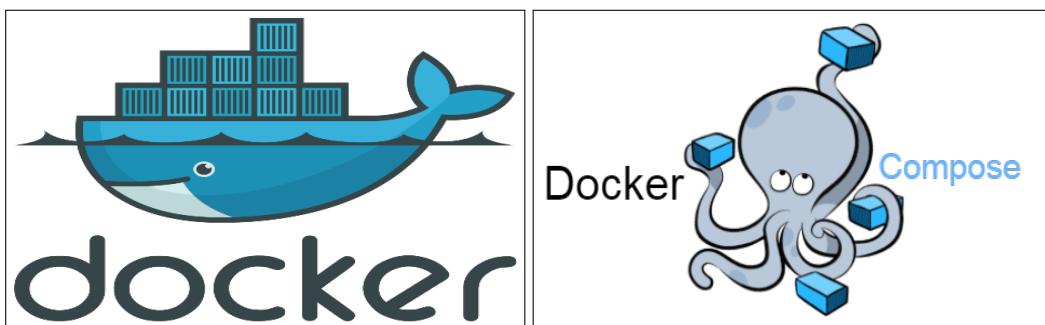
4.4 La conteneurisation

Afin de protéger notre application des risques qui sont liés aux mises à jour de l'environnement d'exécution. Et comme les services sont déployables séparément, nous devons simplifier le processus de déploiement de notre système. D'où la nécessité de conteneuriser nos micro-services.

Cette solution nous garantit **un déploiement plus rapide** car les conteneurs sont plus rapides à lancer et à terminer. En outre, elle permet **une isolation** complète du système d'exploitation hôte. Cette isolation protège notre application de nombreux risques liés à la mise à jour de l'OS hôte. Ces conteneurs peuvent être exécutés pratiquement n'importe où (**Portabilité**), ce qui nous facilite grandement le développement et le déploiement : sur Linux, Windows, Mac et, bien sûr, dans le cloud.

Toutes les applications de notre système distribué seront déployées indépendamment dans des conteneurs à l'aide de **Docker**.

Pour éviter d'avoir à se préoccuper de nombreuses lignes de commande pour lancer nos images Docker. Nous utilisons le **docker Compose** qui va lancer tout le système en tapant une seule commande dans l'invite de commande.



4.5 DevOps

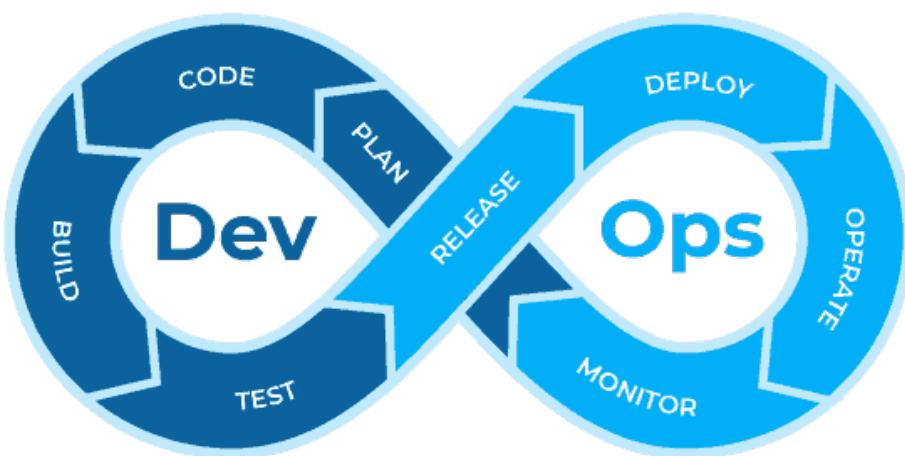
Une fois nos différents microservices et l'application frontale sont déployés, la mise en production continue d'un nouveau produit ou service devient une tâche épuisante. En effet, nos aspirations pour ce projet vont dans le sens d'innover toujours davantage pour répondre à la pression du marché, avec plus de fonctionnalités qui doivent être déployées de manière plus rapide. Dans cette optique, nous avons pensé à la philosophie Dev Ops. Grâce à elle, les évolutions de notre système se font de manière la plus fluide et transparente possible pour l'utilisateur, en d'autres termes, les mises à jour dont nous avons parlé deviennent naturelles dans le cycle de vie du produit. Les éléments clés de DevOps sont les meilleures pratiques appliquées pendant le cycle de développement du projet.

4.5.1 L'intégration continue (CI)

Cela nous permettra d'apporter des modifications permanentes à notre code. Tout en garantissant que chaque livraison de code est testée et compilée, de sorte que la base de code reste propre à tout moment.

4.5.2 Déploiement continu (CD)

Cette opération est la dernière étape d'un pipeline CI/CD. Elle a pour but d'automatiser le démarrage d'une application dans un environnement de production.



4.6 Autres concepts clés :

4.6.1 La Sécurité de système Distribué

Contrôler les failles de sécurité dans nos microservices est essentiel, car les menaces sont de plus en plus puissantes et répandues. D'autant plus qu'ils sont connectés au cloud. Cela augmente le degré de vulnérabilité aux menaces. Il est donc essentiel de s'assurer que les utilisateurs n'ont accès qu'aux données auxquelles ils sont autorisés à accéder dans leurs applications.

Pour y parvenir, nous utiliserons Keycloak comme serveur de gestion d'identité. Ainsi, nous allons intégrer un adaptateur keycloak au système de sécurité de chaque microservice qui sera Spring Security. Ce qui nous permettra de déléguer la partie gestion de session et authentification à keycloak.

Au niveau du front-end, nous associons à toutes les requêtes envoyées au back-end un Access-Token crypté contenant l'identité du demandeur de ressources. Cet Access-Token permet au microservice sollicité de vérifier si cet utilisateur est autorisé à accéder aux ressources demandées.

Pour assurer la communication entre microservices sans se soucier de l'Access-Token et du RestTemplate. Keycloak nous propose une classe KeycloakRestTemplate qui hérite les mêmes fonctionnalités que RestTemplate et qui permet d'ajouter automatiquement le Access-Token dans les headers des appels entre microservices.



4.6.2 La documentation des API

Les API ont pour vocation de servir à de nombreux développeurs. Par conséquent, le développement d'une API nécessite une documentation accessible et facilement utilisable. Cette dernière doit toujours être à jour au fur et à mesure que le code et les fonctionnalités de l'API évoluent. Et dans le but d'offrir un aperçu sur les services exposés par nos microservices aux prochains développeurs sans avoir besoin de se plonger dans le code. Nous avons implémenté swagger3 dans nos microservices pour simplifier la création et la maintenance de cet aperçu. Une fois que nous avons fini de développer nos microservices, nous pouvons passer à l'application frontale et consommer ces API. Avec swagger, nous n'avons plus besoin d'exécuter les microservices dans l'éditeur pour y retourner et vérifier les détails des fonctions, des classes, des types de retour et des arguments. Il suffit de lancer rapidement nos images docker et d'accéder via le navigateur à la documentation préparée par swagger pour leurs endpoints.



4.6.3 La qualité de code

En informatique, une application de qualité est définie comme une appréciation générale du produit. En effet, cette qualité est directement liée à la qualité du code. De plus, le coût de correction d'une erreur augmente considérablement avec le temps, ce qui implique que la détection précoce des erreurs est d'autant plus importante. Ainsi, l'une des étapes de la chaîne devops consiste à tester cette fiabilité avant de passer au déploiement. Dans ce contexte, nous intégrerons sonarcloud dans notre système qui nous donnera donc des indications claires sur la manière de résoudre les problèmes de qualité et de sécurité du code détectés. Il

nous aidera donc à détecter rapidement les failles de sécurité, les bugs.



Conclusion

Dans ce chapitre, nous nous sommes concentrés sur l'analyse technique de notre solution. Nous avons commencé par le choix de notre architecture. Ensuite, nous avons cité les besoins techniques. Et nous avons terminé en présentant la solution adoptée pour chaque besoin.

CONCEPTION

Plan

1	Architecture globale de la solution	31
2	Aspect statique	34
3	Paquetages de l'application	35
4	Diagramme de classe de conception	36
5	Aspect Dynamique	38

Introduction

La partie conception est une étape essentielle pour le développement de notre solution, il s'agit de la phase fondamentale de la méthodologie 2TUP, dont le but est la conception des exigences définies dans la spécification et qui représente une conception détaillée de notre système. Dans ce chapitre, nous commençons par une architecture de la solution pour avoir une idée sur les styles architecturaux utilisés. Ensuite, nous présentons la partie conception approfondie permettant de mieux comprendre la conception liée à l'aspect statique et à l'aspect dynamique. basée sur des diagrammes de séquence de composants, de packages, de classes et d'objets.

5.1 Architecture globale de la solution

Dans cette rubrique nous allons présenter les modèles architecturaux de notre solution par l'architecture micro-service, physique et logique.

5.1.1 Architecture micro-service de l'application

La figure ci-dessus illustre l'interaction entre les composants de notre application.

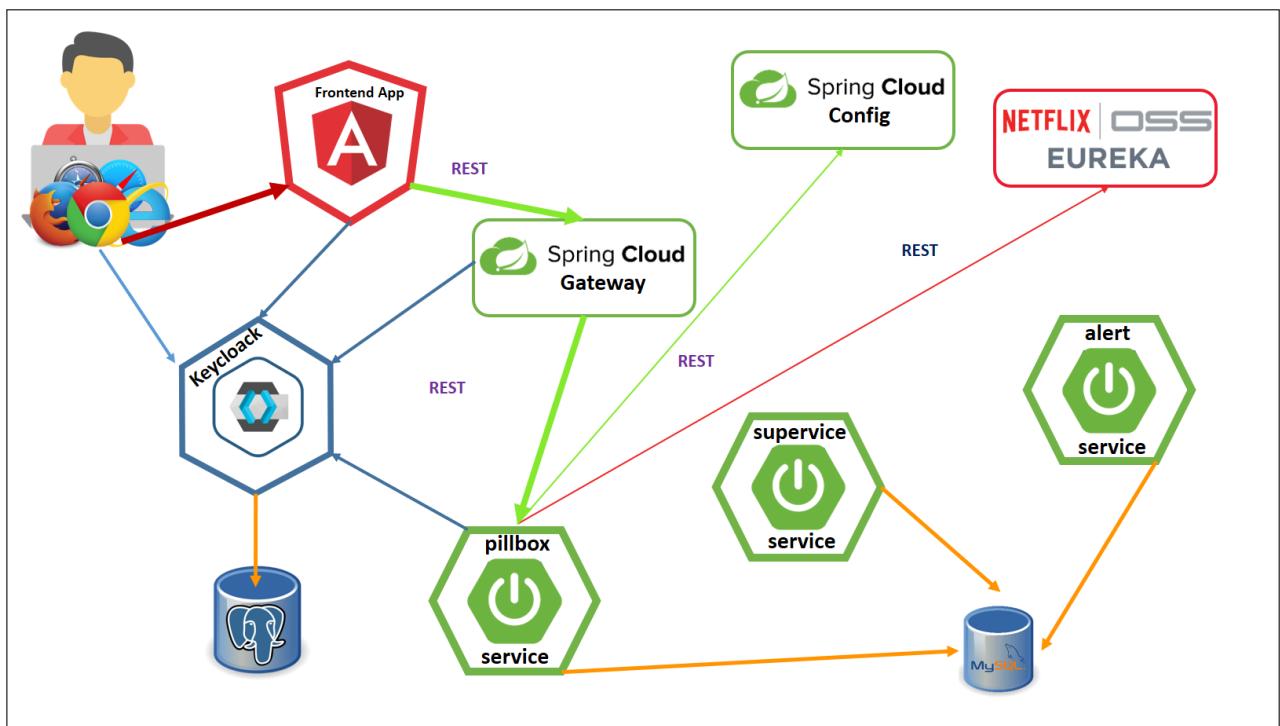


Figure 5.1: Architecture micro-services de l'application

5.1.2 Architecture physique

Pour pouvoir définir l'architecture physique du système, ainsi que pour évaluer les implications de la distribution et de l'allocation des ressources, le schéma représenté ci-dessous a été utilisé.

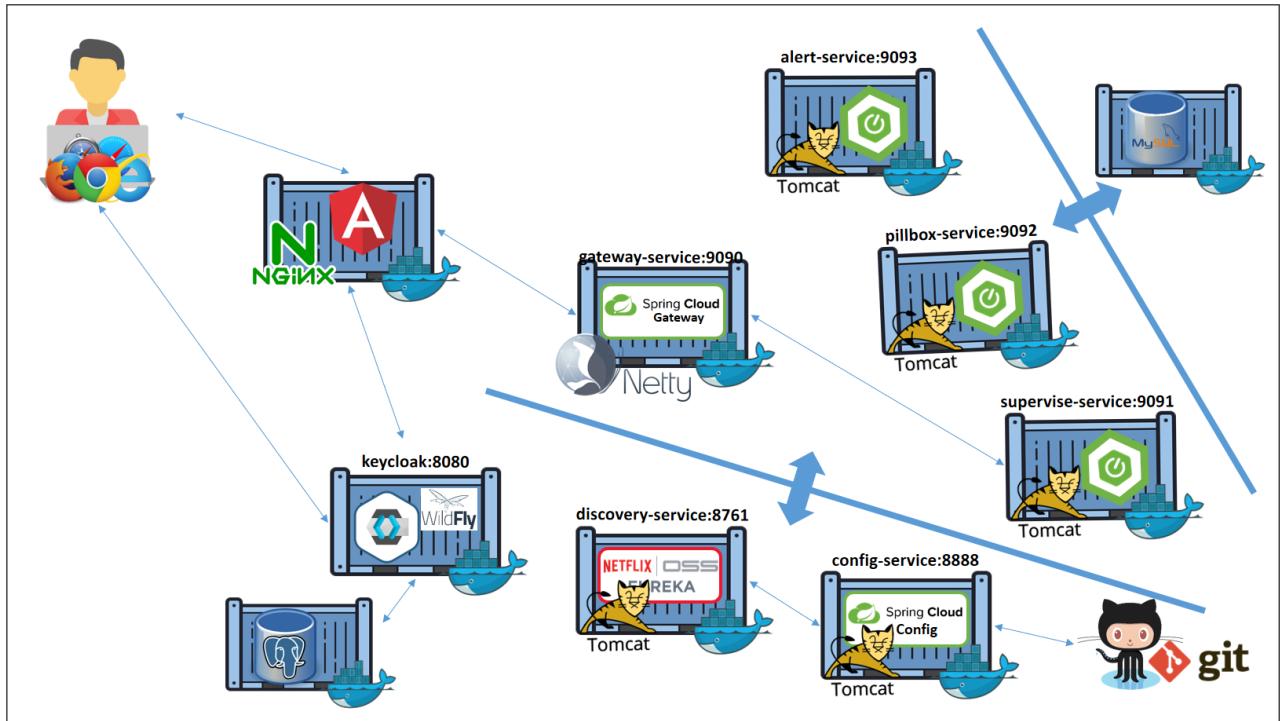


Figure 5.2: Architecture Physique

L'architecture physique de notre projet est composée de plusieurs types de serveurs dont les échanges se font via différents protocoles. Les requêtes http venant du front-end passent par notre proxy Spring cloud gateway et celui-ci recherche l'emplacement et le numéro de port du microservice demandé auprès du serveur d'enregistrement Eureka. Ensuite, le microservice traite la demande et la renvoie au proxy qui, à son tour, la renvoie au front-end et ce, après une vérification des droits d'accès et de la validité du jeton d'accès réalisé par keycloak.

Lors du démarrage, tous les microservices demandent leur configuration au serveur de configuration "spring cloud config" et ils s'enregistrent dans Eureka en fournissant 3 paramètres principaux : le nom, les adresses IP et le numéro de port.

5.1.3 Architecture logique

Contrairement à l'architecture physique, où l'objectif était de distinguer les différents niveaux physiques de l'application, l'architecture logique vise plutôt la division logique de l'application et la manière de

regrouper les composants suivant le type de fonction et de traitement qu'ils effectuent. La figure ci-dessous illustre l'architecture logique de l'application.

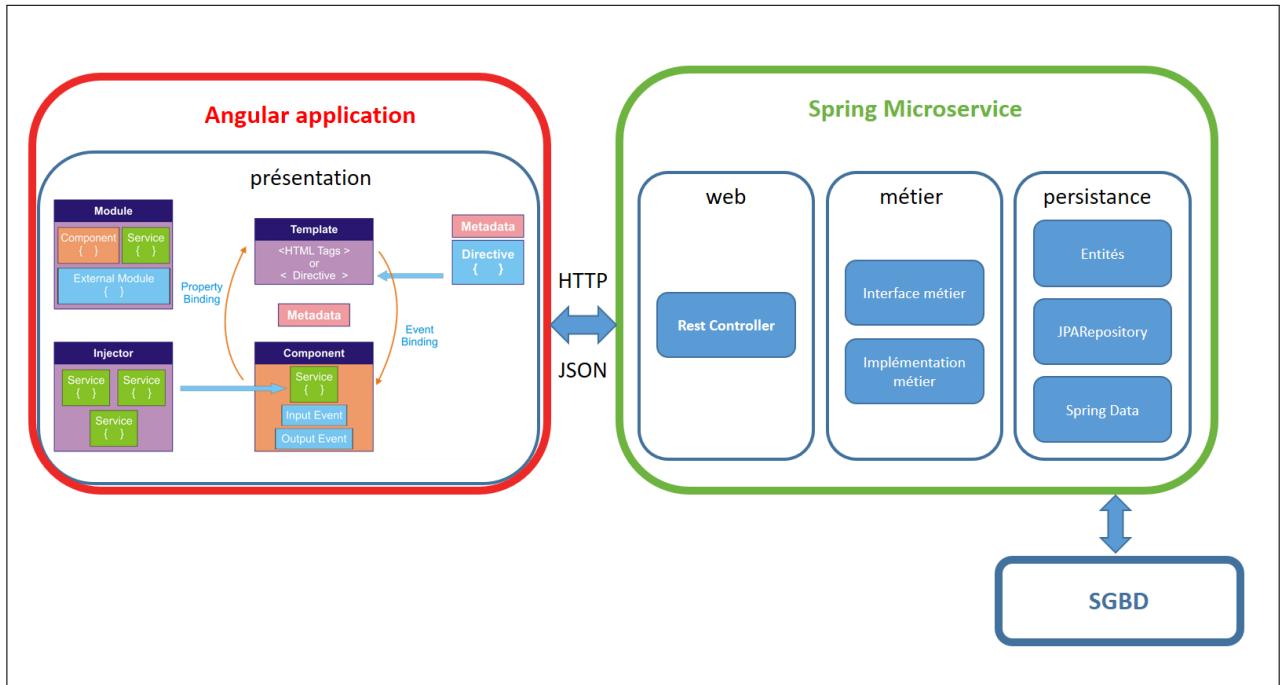


Figure 5.3: Architecture logique de l'application

- **La couche présentation :** cette couche gère les interactions avec l'utilisateur final. Pour chaque ensemble de fonctionnalités, nous avons des composants, qui sont formés d'une page web implémentée avec HTML, CSS et Bootstrap. Ainsi, un contrôleur ayant comme langage le Type Script.
- **La couche web :** Cette couche sert à exposer les services de la couche métier à la couche présentation via REST.
- **La couche métier :** Cette couche est le cœur du projet. Celle-ci comprend les aspects fonctionnels et opérationnels de la spécification des exigences.
- **La couche persistance :** Cette couche permet d'accéder aux données à l'aide de Spring data JPA et des entités qui correspondent aux données persistantes (tables) de la base de données.

5.2 Aspect statique

Dans cette partie, nous aborderons l'aspect statique de notre solution à travers les diagrammes de composants, de classes et de paquets. Dans cette rubrique, nous détaillons notamment le microservice "pillbox-service" à titre d'exemple. Les autres microservices sont pour la plupart similaires.

5.2.1 Composants de l'application

5.2.1.1 Diagramme de composants du cas d'utilisation « gérer le pilulier du patient »

La figure ci-dessous illustre le diagramme de composants du cas d'utilisation « gérer le pilulier du patient ».

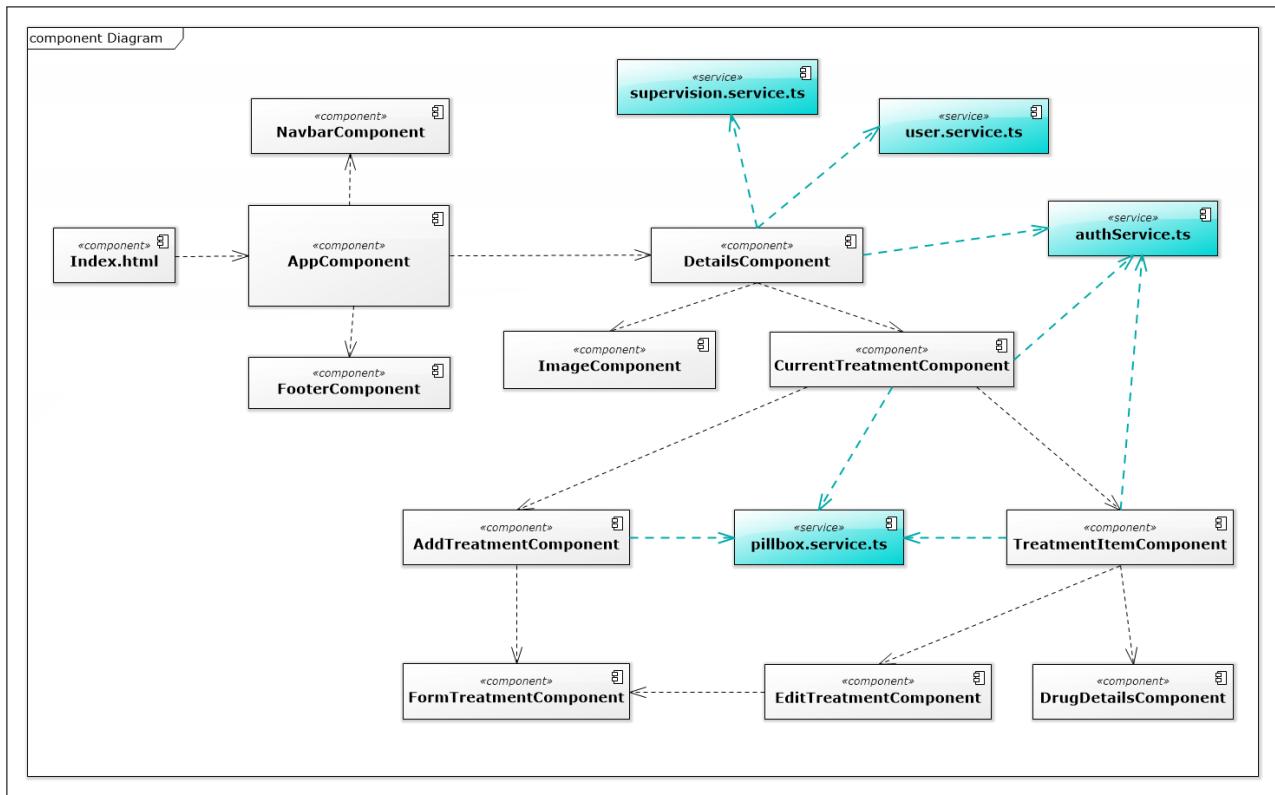


Figure 5.4: Diagramme de composant

- **index.html** : C'est notre seule page web qui utilise le AppComponent, celle-ci est simplement un arbre de composants.
- **AppComponent** : C'est le composant racine de notre module, en fait c'est le seul composant qui sera appelé au démarrage de l'application.

- **NavbarComponent, FooterComponent, DetailsComponent, CurrentTreatmentComponent, AddTreatmentComponent, EditTreatmentComponent, FormTreatmentComponent, DrugDetailsComponent** : Tous les composants sont eux-mêmes composés d'un fichier HTML (.html) et d'un fichier TypeScript (.ts). Tous ces éléments sont définis par un mot clé appelé (selector), en l'appelant, l'élément sera injecté dans le fichier HTML de votre choix.
- **authService.ts, user.service.ts, supervision.service.ts, pillbox.service.ts** : Il est possible d'injecter un service dans différents composants, et un composant peut ainsi injecter un ou plusieurs services.

5.2.1.2 Diagramme de composants du cas d'utilisation « gérer ses surveillants »

La figure ci-dessous illustre le diagramme de composants du cas d'utilisation « gérer ses surveillants ».

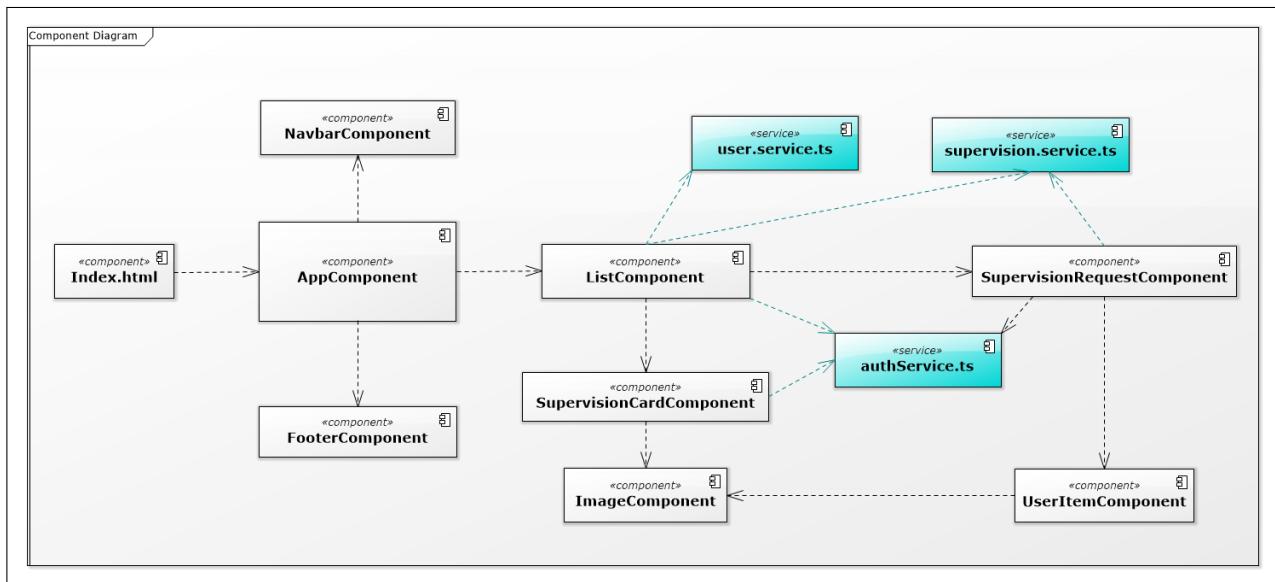


Figure 5.5: Diagramme de composants du cas d'utilisation « gérer ses surveillants »

5.3 Paquetages de l'application

Comme l'architecture de notre application est en couches, il est important de regrouper les classes à l'aide de packages pour permettre de mieux comprendre le rôle de chaque partie et favoriser la maintenabilité du code. La figure ci-dessous illustre les paquetages de micro service « pillbox service »

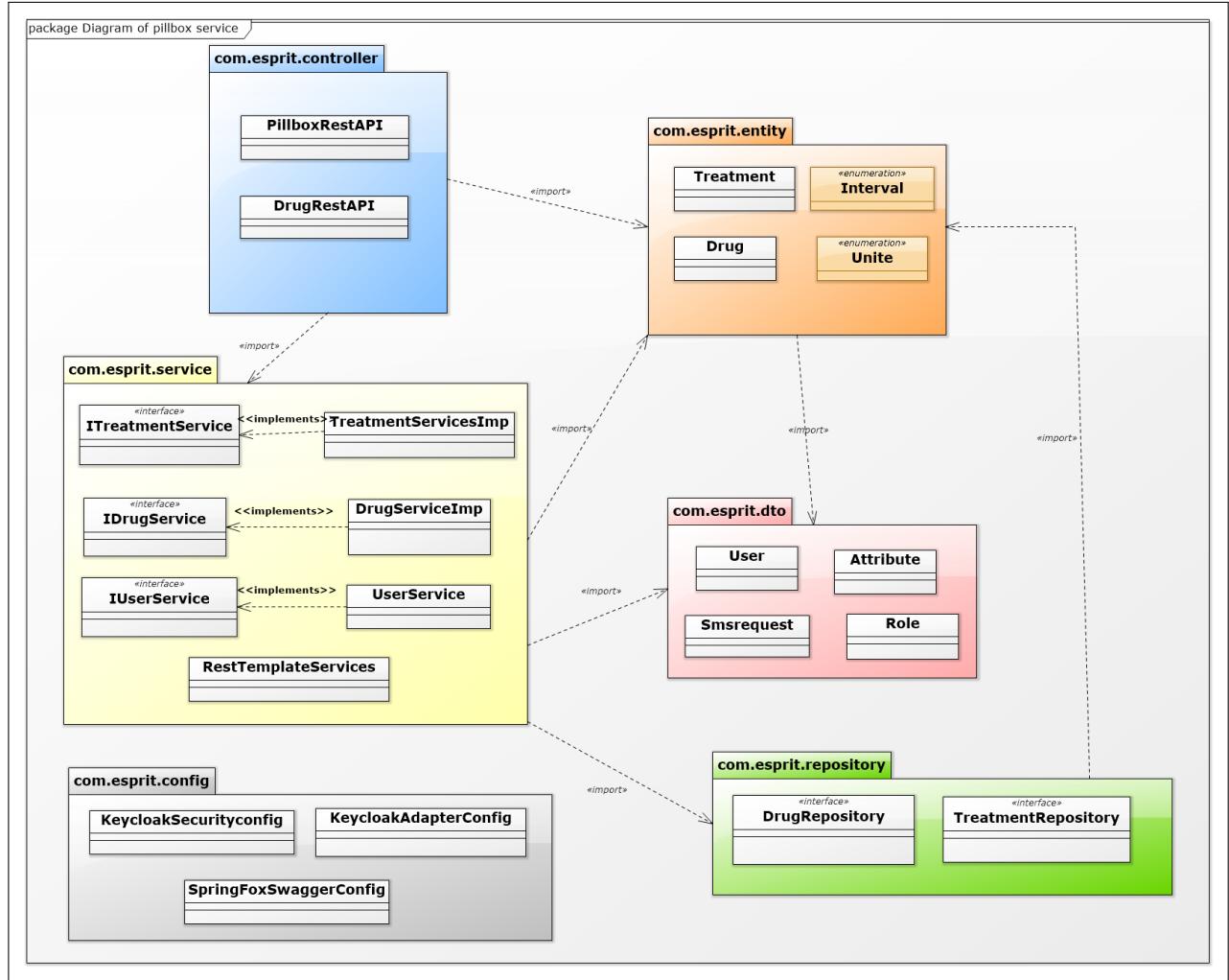


Figure 5.6: Diagramme de paquetages de micro service « pillbox service »

5.4 Diagramme de classe de conception

La figure ci-dessous décrit les classes liées au micro service « pillbox-service » :

Chapitre 5. Conception

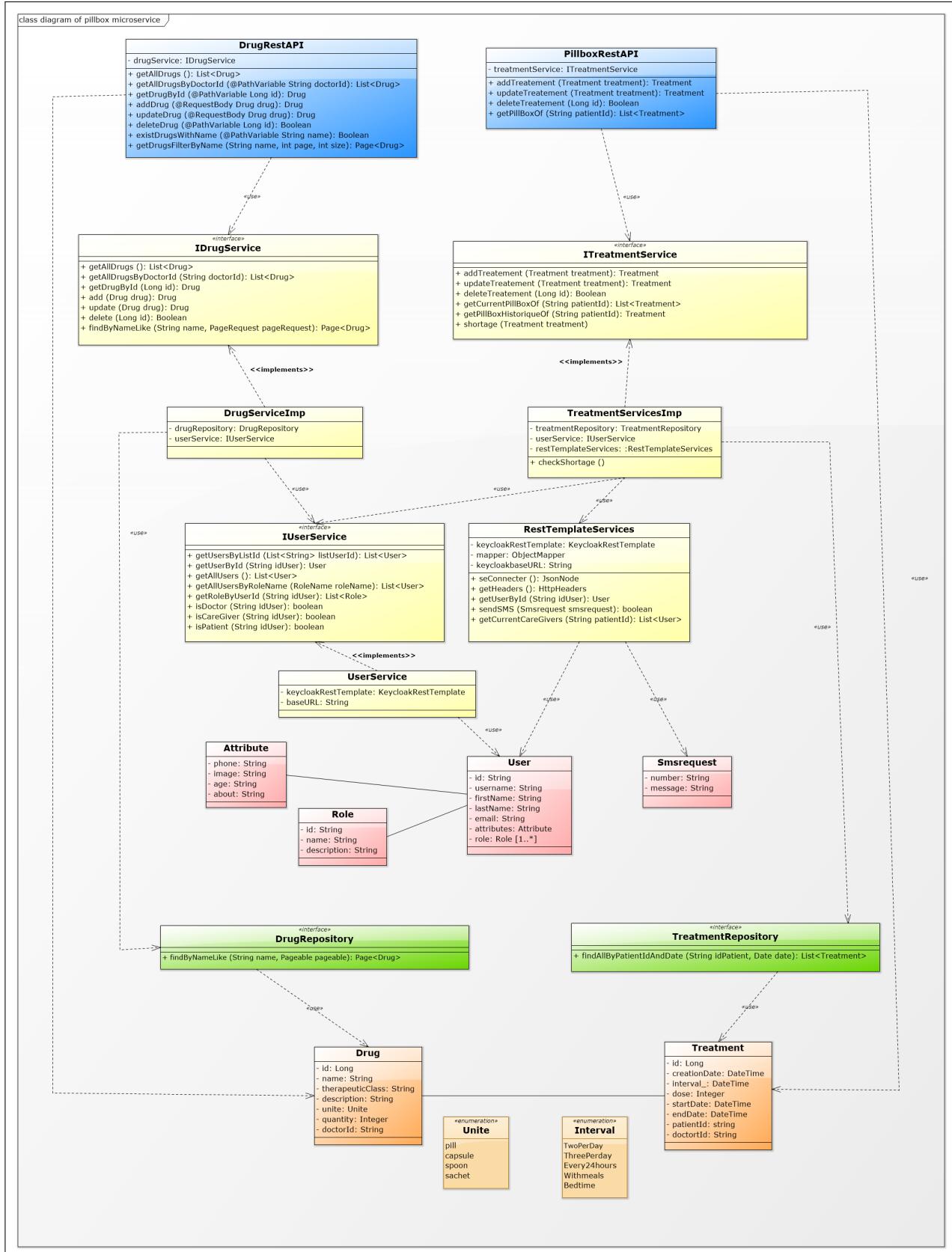


Figure 5.7: Diagramme de classe de conception

5.5 Aspect Dynamique

La vue dynamique est utilisée pour la modélisation du comportement dynamique de notre système en montrant comment ces objets agissent entre eux au moment de l'exécution. Pour cela, des diagrammes de séquence d'objets sont utilisés.

5.5.1 Diagramme de séquence objet

La figure ci-dessous illustre l'interaction entre les objets intervenant dans le cas d'utilisation « consulter la liste de ses proches aidants »

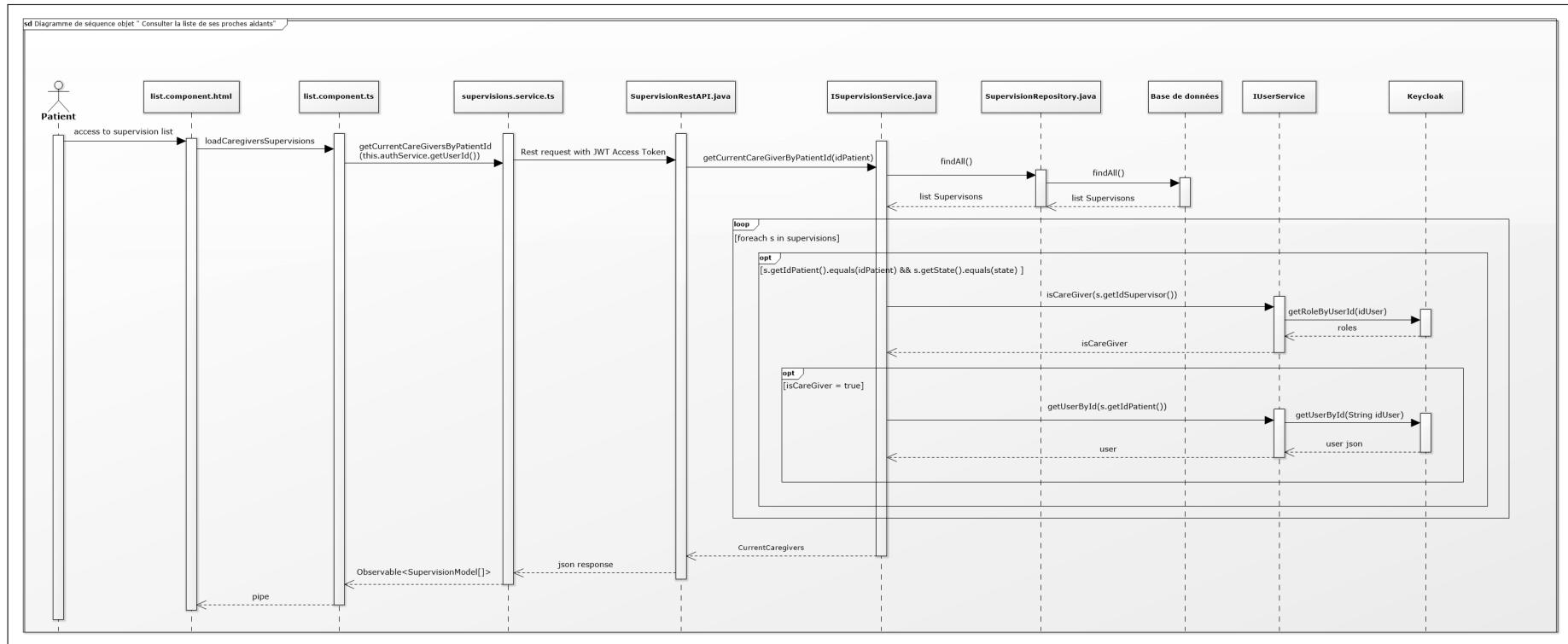


Figure 5.8: diagramme de séquence de conception

Conclusion

Dans ce chapitre, Nous avons examiné la possibilité de réaliser notre application. Tout d'abord, nous avons présenté les modèles architecturaux à travers une architecture micro-service, physique et logique. Puis, nous avons abordé la conception de l'application à travers la présentation des diagrammes de composants, du diagramme de paquetage et du diagramme de classe qui exposent l'aspect statique du système. Finalement, nous avons exposé l'aspect dynamique par les diagrammes de séquence d'objets, qui nous semblaient nécessaires pour illustrer le comportement de l'application.

RÉALISATION

Plan

1	Environnement de travail	41
2	Description de l'application	42

Introduction

Ce chapitre décrit la réalisation de notre application, que nous avons détaillée, tout en tenant compte des exigences spécifiées dans les chapitres précédents. Ce chapitre est composé de trois parties : nous commençons par décrire l'environnement de développement de l'application. Ensuite, nous détaillons les différentes technologies adoptées dans notre projet. Enfin, nous présentons le travail effectué durant la période de stage.

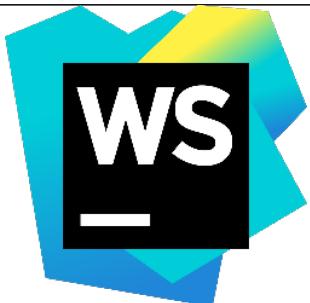
6.1 Environnement de travail

Cette section présente notre environnement logiciel avec lequel nous avons développé notre solution tout au long de ce projet de fin d'études.

6.1.1 Environnement logiciel

Tableau 6.1: La liste des logiciels utilisés

Environnement logiciel	Description
 <i>Software Ideas Modeler</i>	Software Ideas Modeler est un outil UML (Unified Modeling Language) qui prend en charge la norme UML
 <i>Github</i>	GitHub est une plateforme d'hébergement de code pour le contrôle de version et la collaboration. Il vous permet, à vous et à d'autres, de travailler ensemble sur des projets de n'importe où.[3]

	<p>STS est l'outil IDE proposé par SpringSource pour les développeurs d'applications Spring. Il comprend un certain nombre de plug-ins, qui fournissent un support pour le développement d'applications basées sur Spring, et s'intègre à d'autres plug-ins Eclipse (tels que m2e, AspectJ Development Tool, etc.)[4]</p>
	<p>Postman est Un logiciel qui permet de tester les web services.</p>
	<p>WebStorm est un IDE pour les langages Web (HTML, CSS et JavaScript), développé par l'entreprise JetBrains et basé sur la plateforme IntelliJ IDEA.[5]</p>
	<p>Docker Desktop est une application native conçue pour Windows et MAC OS pour exécuter, créer et expédier des applications ou des services dockerisés /conteneurisés.[6]</p>
	<p>Azure est la plateforme Cloud de Microsoft. Elle regroupe divers services de Cloud Computing.[7]</p>
	<p>Jenkins est un serveur d'automatisation open source par excellence. il permet de construire, de déployer et d'automatiser n'importe quel projet.</p>

6.2 Description de l'application

Nous présenterons dans cette rubrique, au moyen d'une série de captures d'écran, une description générale des différentes fonctionnalités de l'application. En tapant l'URL de l'application, la page "Home" (la figure ci-dessous) s'affiche.



Figure 6.1: La page Home

En appuyant sur le bouton "Sign In", l'utilisateur est redirigé vers une page d'authentification gérée par keycloak. Il peut soit saisir ses informations de connexion soit s'enregistrer. En cas d'erreur, un message s'affiche comme suit :

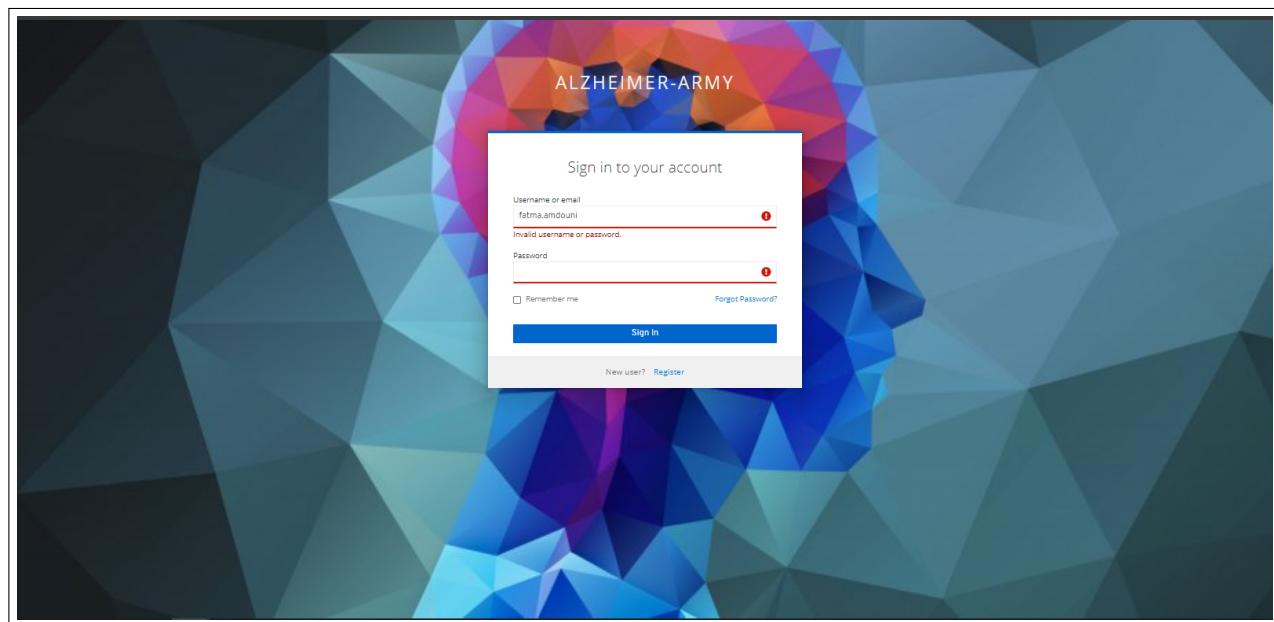


Figure 6.2: L'interface d'authentification avec des coordonnées invalides

Après une authentification réussie, la page d'accueil s'affiche à nouveau et la "Navbar" est adaptée en fonction du profil de l'utilisateur, comme le montre la figure ci-dessous :

Chapitre 6. Réalisation

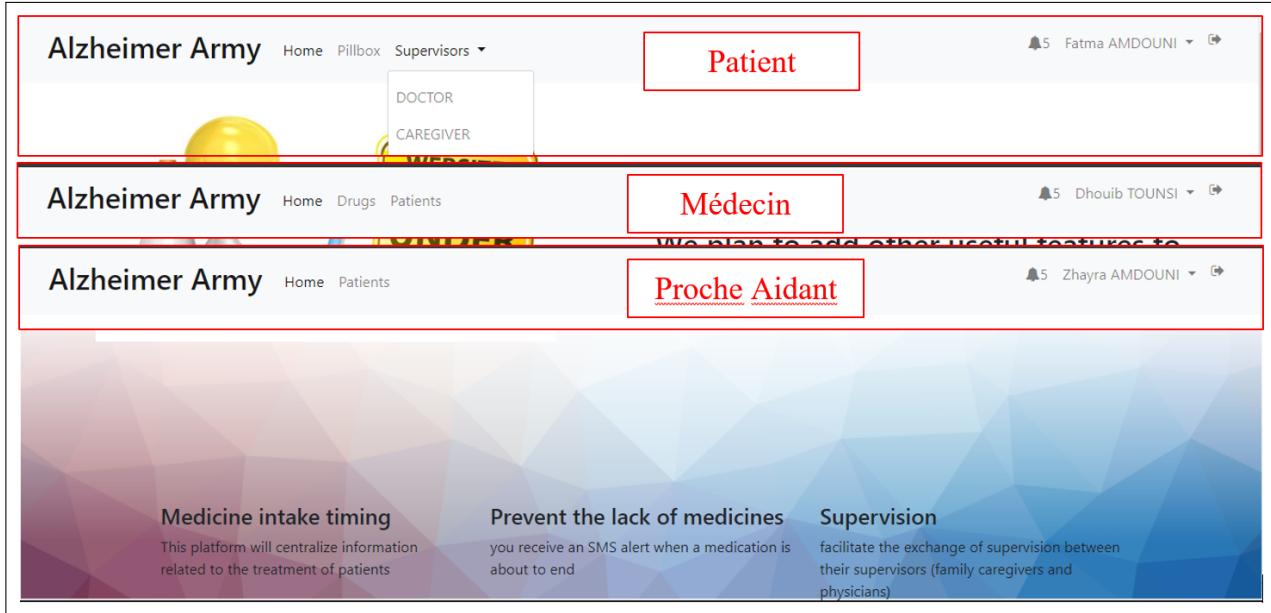


Figure 6.3: Les différentes "NavBar" selon le rôle

Chaque utilisateur dispose d'un espace personnel (son profil), il peut consulter ses informations personnelles comme l'illustre la figure suivante

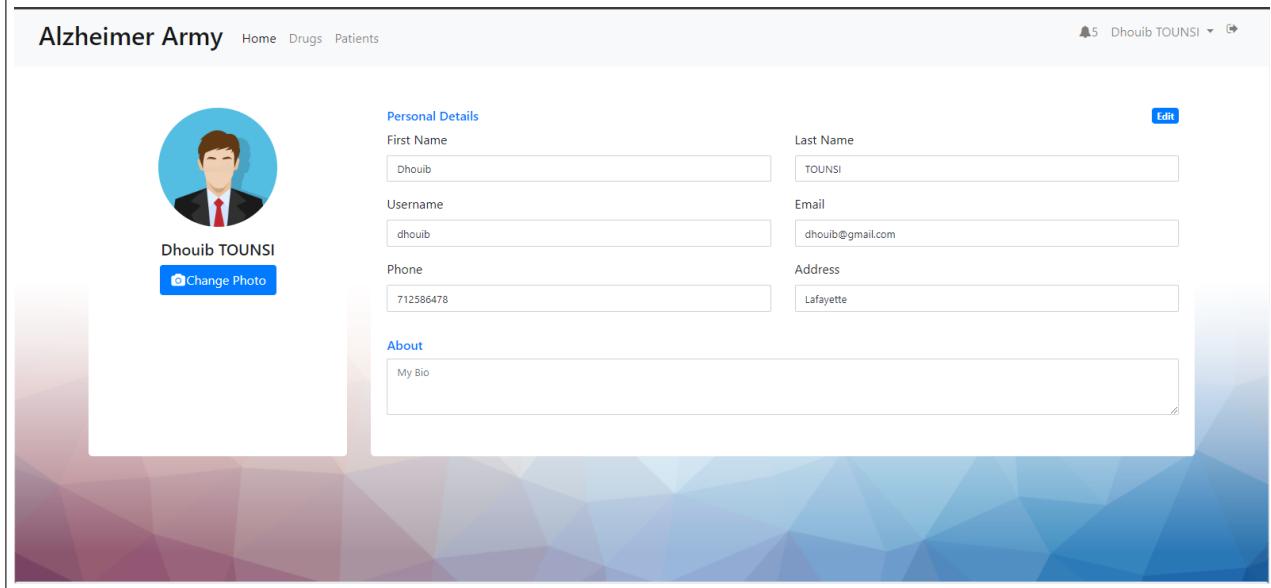


Figure 6.4: Interface de consultation du profil

Il peut ainsi modifier sa photo de profil et mettre à jour ses coordonnées sur cette même page.

Chapitre 6. Réalisation

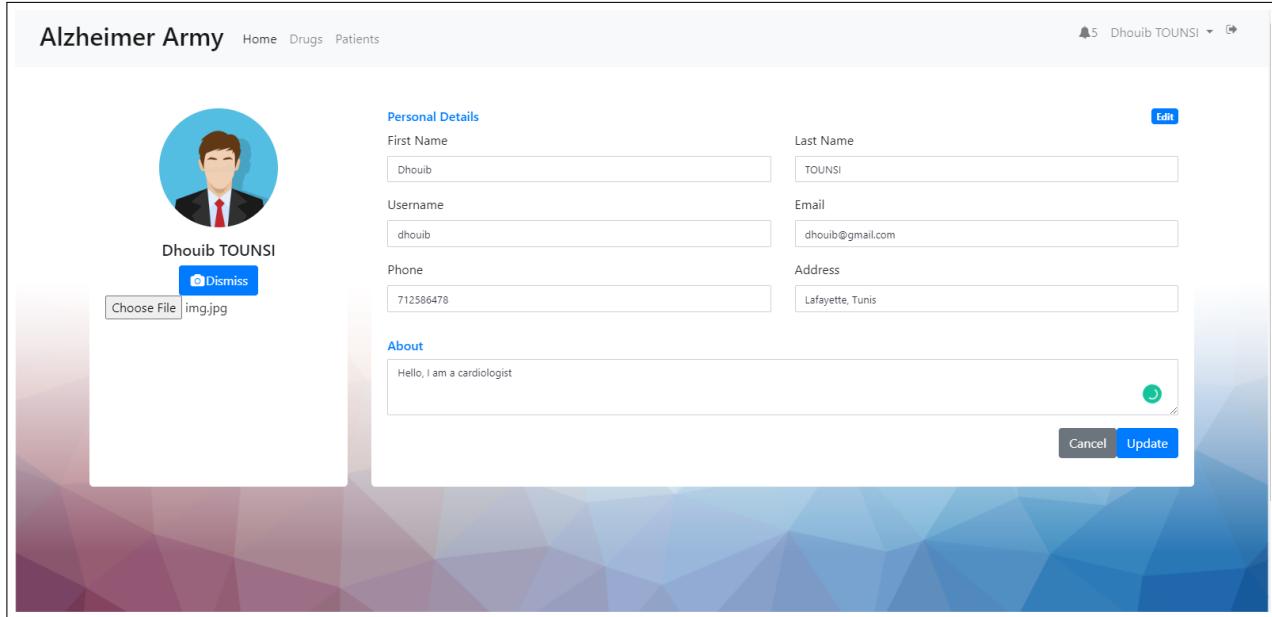


Figure 6.5: Interface du mise à jour du profil

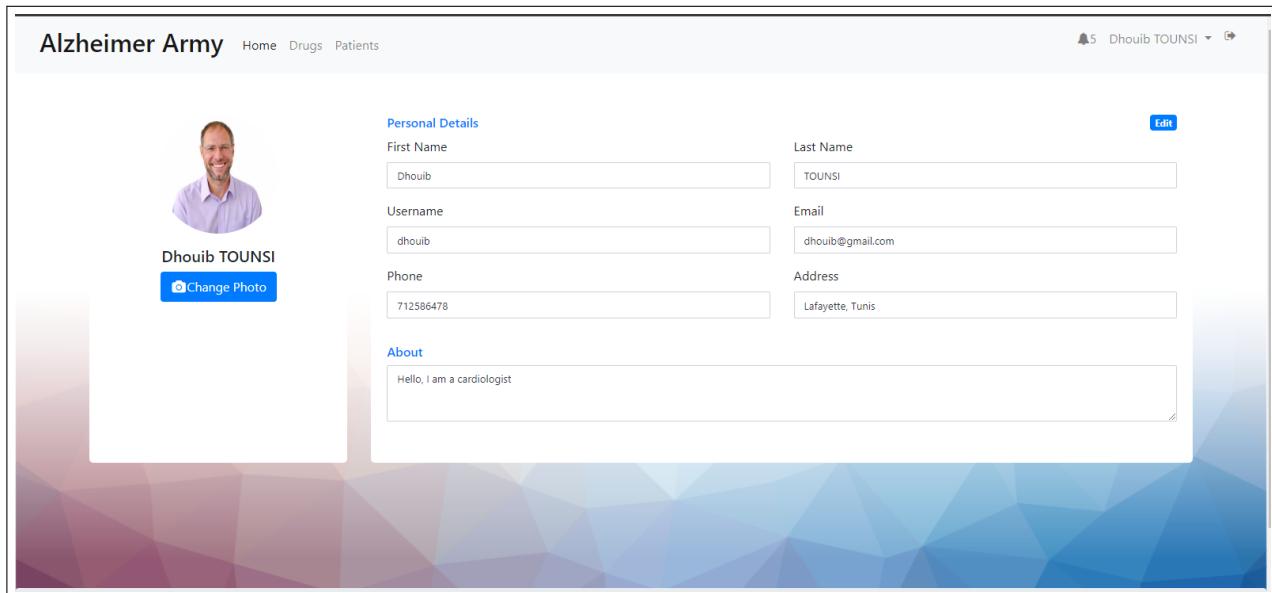


Figure 6.6: Interface du mise à jour du profil

Pour que le médecin ou le proche aidant puisse gérer ses supervisions, il suffit de cliquer sur l'élément "Patients", comme le montre la figure ci-dessous :

Chapitre 6. Réalisation

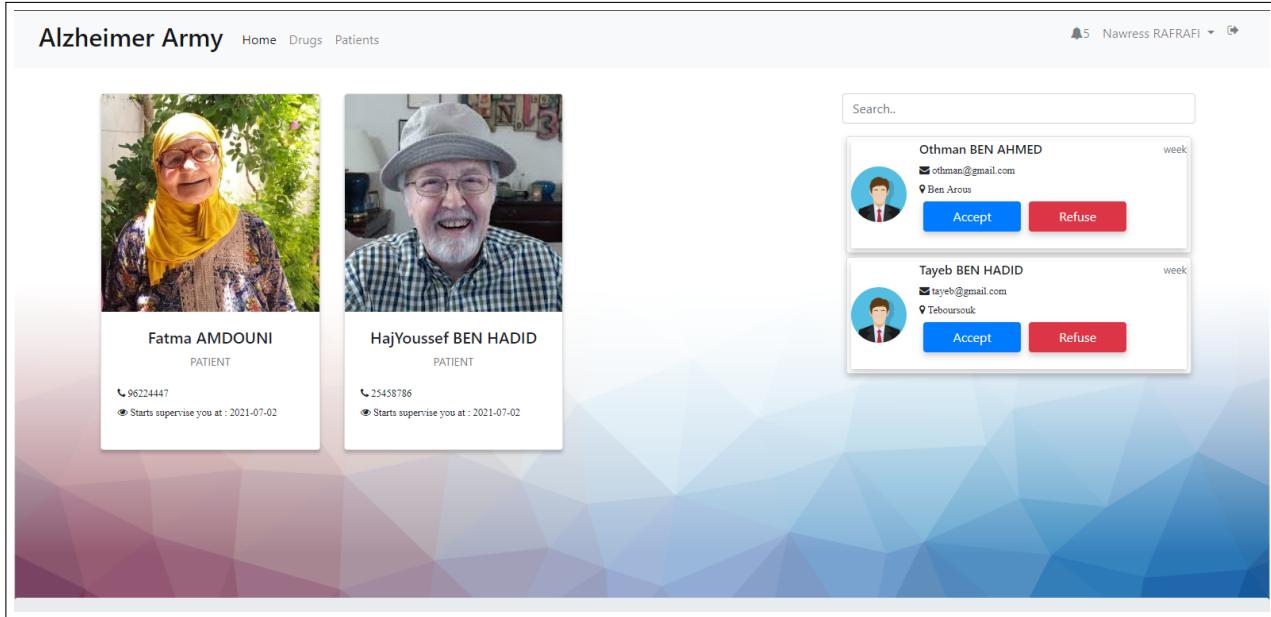


Figure 6.7: L'interface du gestion des patients

En cliquant sur la carte d'un patient, les détails de ce dernier s'affichent. Si l'utilisateur connecté est un médecin, d'autres éléments seront affichés pour l'aider à gérer le pilulier du patient.

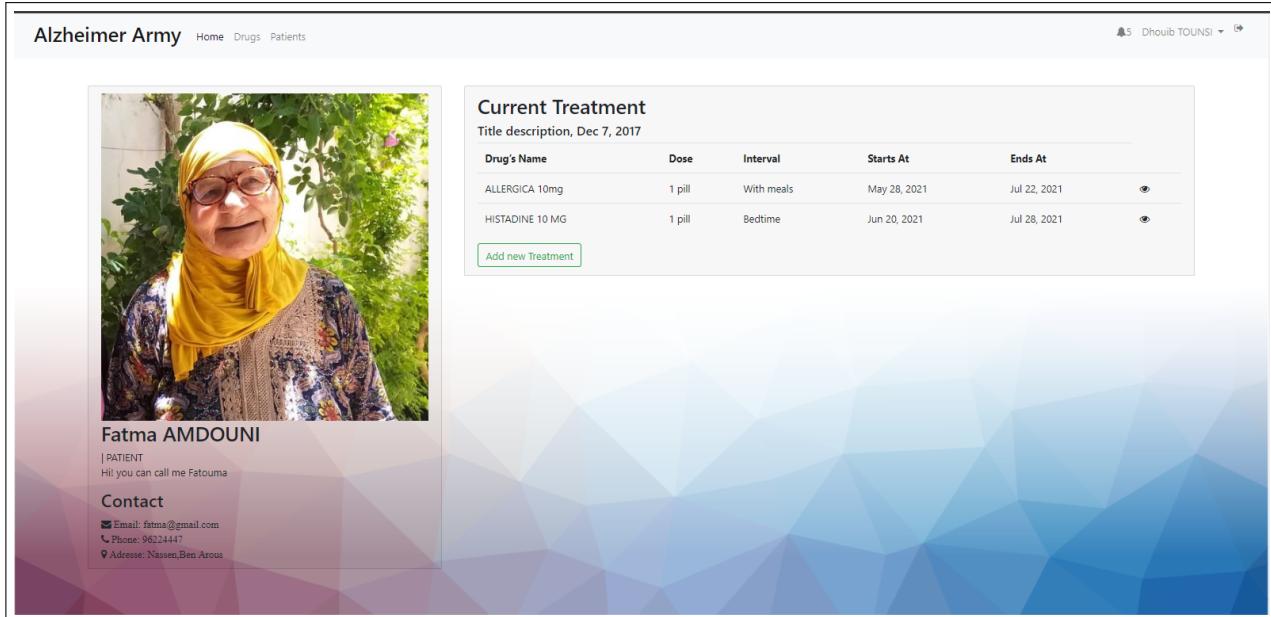


Figure 6.8: L'interface du gestion de pilulier

Pour ajouter un nouveau traitement, il suffit de cliquer sur le bouton "Add new Treatment" et de remplir le formulaire affiché dans la fenêtre pop-up modale. Ainsi, le système facilite la recherche du médicament par le médecin en lui proposant les noms des médicaments similaires.

Chapitre 6. Réalisation

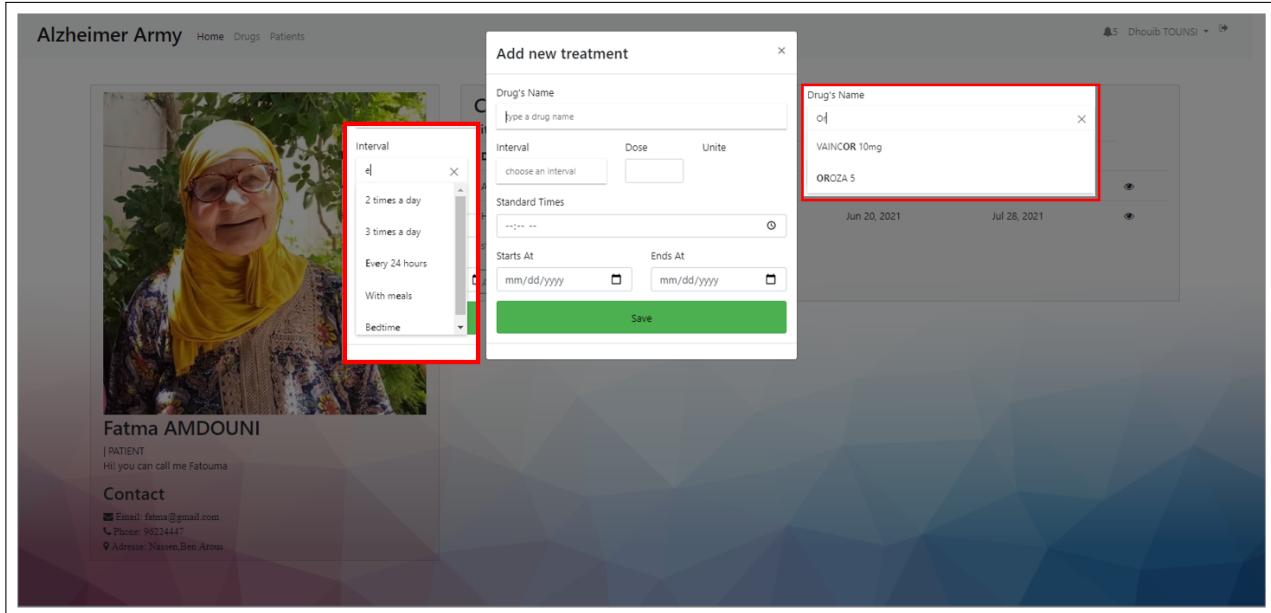


Figure 6.9: Ajout d'un nouvel traitement

En appuyant sur le bouton "Save". Si l'ajout du traitement est réussi, alors un message de réussite s'affiche. Ce nouveau traitement ajouté par ce médecin sera affiché en bleu. Ainsi, deux icônes qui lui permettront de modifier ou de supprimer ce traitement avant le lendemain afin que nous puissions toujours garder la trace de tous les traitements pris par un patient.

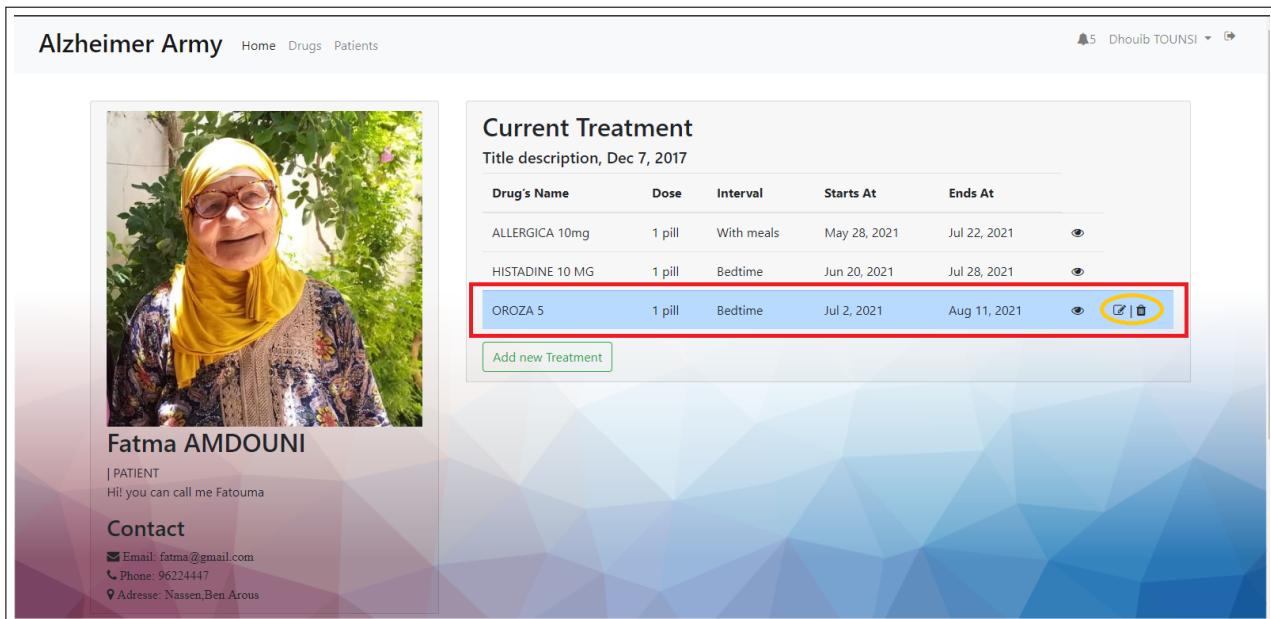


Figure 6.10: La modification et la suppression du traitement

Par un clic sur le petit œil, une fenêtre pop-up apparaîtra contenant tous les détails du médicament

Chapitre 6. Réalisation

utilisé dans ce traitement, Comme le montre la figure suivante :

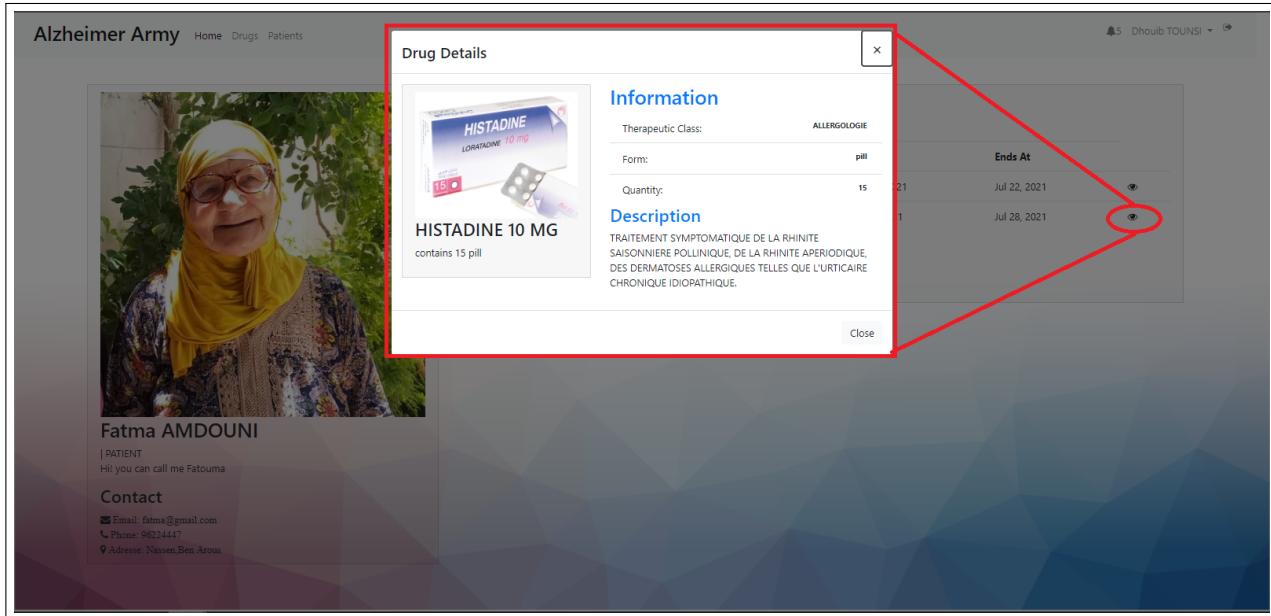


Figure 6.11: Consulter les détails du médicament

Au cas où le médecin ne trouve pas la drogue à inclure dans un traitement, il lui faut cliquer sur l'onglet « Drugs » et puis le bouton « Add new Drug » pour l'ajouter. Comme suit :

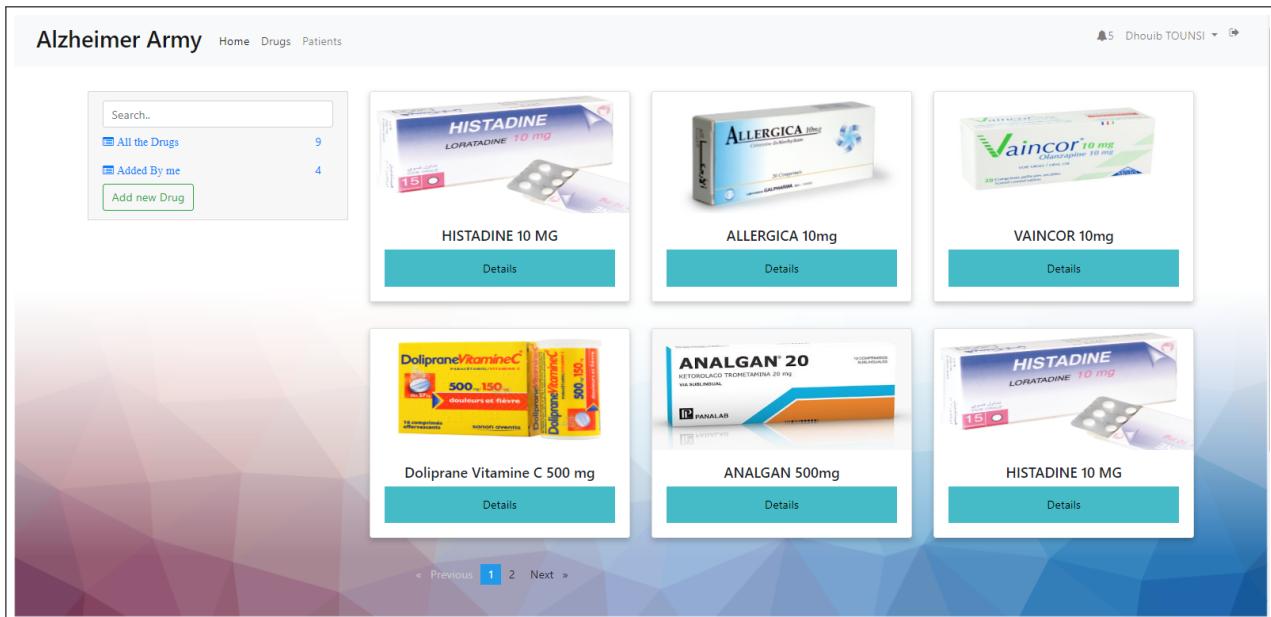


Figure 6.12: La page de gestion des médicaments

Chapitre 6. Réalisation

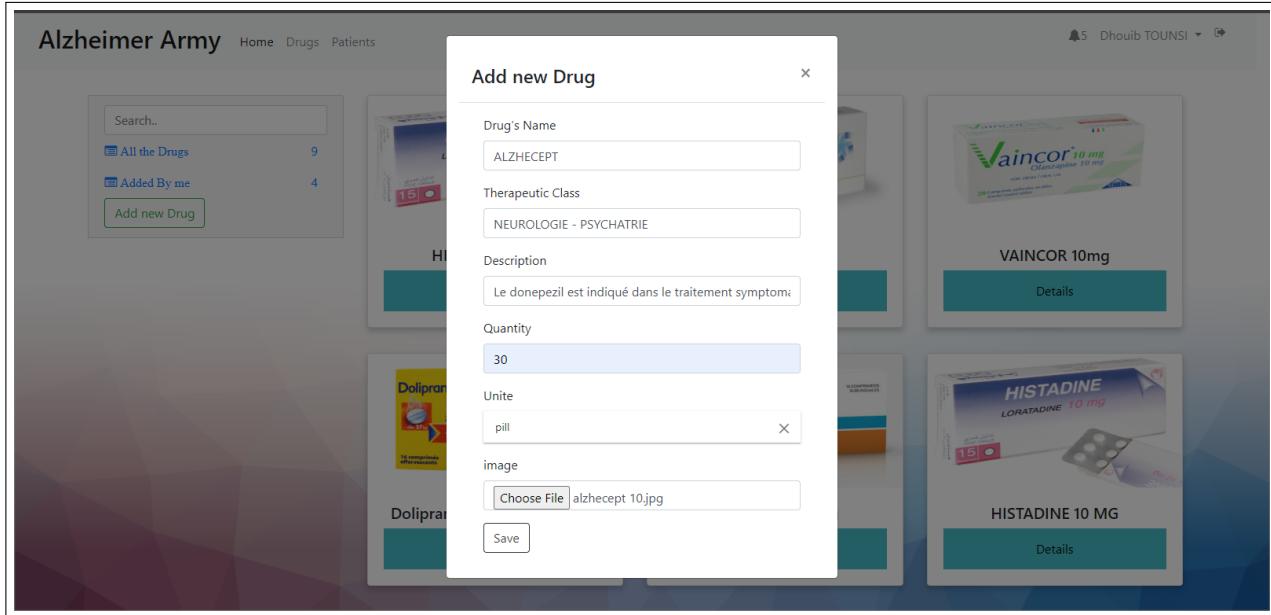


Figure 6.13: Interface d'ajout d'un nouveau médicament

Tout patient a la possibilité de rechercher un médecin et de lui envoyer une demande de suivi ou de consulter la liste de ses médecins ou de ses proches aidants.

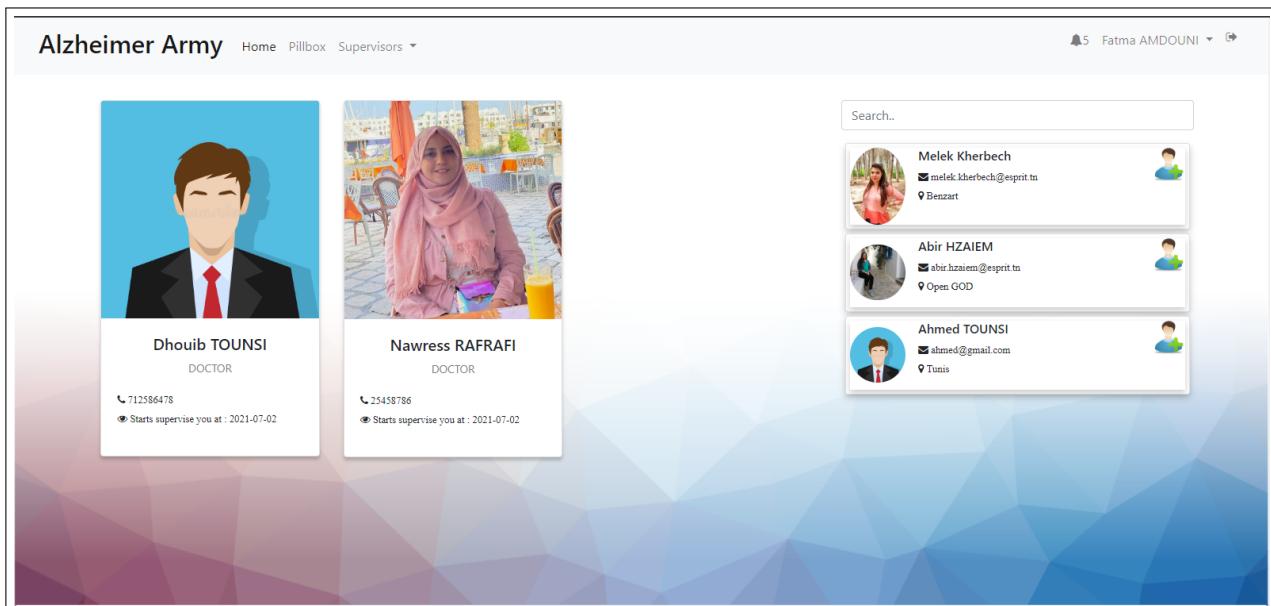


Figure 6.14: Consulter ses médecins

Chapitre 6. Réalisation

The screenshot shows a user profile for 'Fatma AMDOUNI'. On the left, there are three cards for caregivers: 'Atef MADDOURI' (CAREGIVER), 'Salah AMDOUNI' (CAREGIVER), and 'Zhayra AMDOUNI' (CAREGIVER). Each card includes a photo, name, title, phone number, and a note about starting supervision. On the right, there is a sidebar with a search bar and a list of other users: 'Skander SAADAOUI', 'Aladin MADDOURI', and 'Malek Ben Yakhlef', each with their profile picture, email, and location.

Figure 6.15: Consulter ses proches aidants

D'un simple clic sur le bouton, le patient a le pouvoir d'arrêter cette surveillance.

The screenshot shows a doctor's profile for 'Nawress RAFRAFI'. It includes a photo, title ('DOCTOR'), a message ('Hello! Glad to have you'), and a button to 'cancel this supervision'. To the right, a section titled 'Current Treatment' displays a table of medications with their details: ALLERGICA 10mg, HISTADINE 10 MG, OROZA 5, and Doliprane Vitamine C 500 mg. The table columns are 'Drug's Name', 'Dose', 'Interval', 'Starts At', and 'Ends At'.

Figure 6.16: Consulter ses traitements données par le médecin

Conclusion

A travers ce dernier chapitre, nous avons fait le tour de l'environnement logiciel utilisé lors du développement de notre application. Ensuite, nous avons illustré notre travail par des impressions d'écran comprenant la majorité des fonctionnalités.

Conclusion générale

A la fin de ce rapport, nous pouvons dire que ce stage qui s'est déroulé au sein d'ESPRIT TECH m'a offert l'opportunité d'améliorer mes connaissances théoriques et pratiques. C'est là que réside l'intérêt d'un tel projet de fin d'études.

Ce travail de conception et de développement d'une solution e-Heath dédié aux malades d'alzheimer pendant le stage a été très enrichissant sur le plan des thèmes abordés et des technologies utilisées : Il nous a permis de nous familiariser avec l'outil Spring Cloud, un outil très intéressant et innovant. Il nous a également fait connaître de nouvelles solutions de sécurité avancées pour les systèmes informatiques distribués. En outre, il nous a permis d'améliorer nos savoirs en matière de développement web en utilisant le framework Spring Boot et le framework Angular afin d'offrir à l'utilisateur une interface de navigation agréable et ergonomique. Et surtout, Il nous a fait découvrir un monde que j'ai toujours voulu connaître, celui des microservices, de la conteneurisation et du DevOps.

A la suite de ce projet, nous pouvons envisager plusieurs perspectives. En effet, grâce à l'intelligence artificielle, nous pouvons améliorer l'ajout d'un nouveau traitement en prévenant le médecin au cas où le nouveau traitement qu'il veut ajouter n'interfère pas avec le reste des traitements du patient. En outre, nous pouvons utiliser les outils que l'Internet of Things (IoT) peut fournir. En effet, nous pouvons installer différents types d'objets connectés et d'actionneurs chez ces patients afin de suivre leurs mouvements en temps réel et de prévenir les soignants en cas de danger dû au gaz etc.

Avec toutes ces améliorations possibles, nous sommes ambitieux et très motivés pour poursuivre ce travail très intéressant au sein de l'équipe Esprit-tech.

Bibliographie

- [1] (). « Les Valeurs De L'École, » Esprit, adresse : <https://esprit.tn/esprit/valeurs>.
- [2] (). « Esprit-Tech, » Esprit, adresse : <https://esprit.tn/esprit-tech>.
- [3] (). « GitHub, » GitHub, adresse : <https://guides.github.com/activities/hello-world/>.
- [4] (). « SpringSource Tool Suite, » APPENDIXA, adresse : <https://link.springer.com/content/pdf/bbm%3A978-1-4302-4108-9%2F1.pdf> ..
- [5] (). « WebStorm, » adresse : <https://fr.wikipedia.org/wiki/WebStorm>.
- [6] (). « Docker Desktop - Le moyen le plus simple de conteneuriser les applications, » adresse : <https://geekflare.com/fr/docker-desktop>.
- [7] (). « microsoft-azure, » adresse : <https://www.lebigdata.fr/microsoft-azure-tout-savoir>.
- [8] N. LAMOUCHI. (2020-10-21). « Playing with java microservices with quarkus and kubernetes. »
- [9] ——, (2020-01-14). « Playing with java microservices on kubernetes and openshift, » adresse : <http://leanpub.com/playing-with-java-microservices-on-k8s-and-ocp>.
- [10] A. PAI. (). « Samsung releases Backup Memory, » adresse : <https://www.mobihealthnews.com/43023/samsung-releases-backup-memory-app-or-alzheimers-patients>.
- [11] (). « Healthcare Support System for Alzheimer's Patients, » adresse : <https://www.hindawi.com/journals/wcmc/2020/8822598/>.
- [12] (). « Monolithique vs microservices, » adresse : <https://www.talend.com/fr/resources/monolithic-architecture/>.
- [13] D. RAY. (). « An Introduction to Spring Cloud Gateway, » adresse : <https://dev.to/only2dhir/an-introduction-to-spring-cloud-gateway-3o89>.
- [14] P. YOUSSEFI. (). « Sources de l'Informaticien avec Pr.Mohamed YOUSSEFI, » adresse : https://www.youtube.com/channel/UCCwIYNpQVHZTd3Vx_krnmdA.
- [15] NETFLIX. (). « Netflix Technology Blog, » adresse : <https://netflixtechblog.medium.com/>.