MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

UNIVERSITE DE DOUALA

MINISTRY OF HIGHER EDUCATION

UNIVERSITY OF DOUALA

ECOLE NATIONALE SUPERIEURE POLYTECHNIQUE DE DOUALA

BP : 2701 DOUALA

TEL: (237) 697 542 240

Division des Affaires Académiques, de la Recherche et de la Coopération

Mémoire de fin d'études en vue de l'obtention du Diplôme d'Ingénieur à l'École Nationale Supérieure Polytechnique de Douala (ENSPD) de l'Université de Douala

Filière : Génie Informatique et Télécommunication

Option :Génie Informatique (GI)

SUJET : MISE EN OEUVRE DE L'ARCHITECTURE CLOUD NATIVE DANS UN ENVIRONNEMENT .NET ET MICROSOFT AZURE : CAS D'UNE PLATEFORME WEB ET MOBILE DE CONSULTATION MEDICALE EN LIGNE (ALICE)

Effectué du 05 février au 30 juin 2024 à Infinite Solutions SARL (Douala)

Par

CHE BENE FOMEGUEU Djoufson

Encadrant industriel

M. NOUMBO Stéphane Cédric

CEO Infinite Solutions SARL

Encadrants école

Dr. ABBOUBAKAR Hamadjam

Dr. MAKA MAKA Ebenezer

Année académique 2023-2024

DEDICACE

A

La famille Djoufack

REMECIEMENTS

Au terme de ce travail, nous tenons à exprimer notre profonde gratitude envers tous ceux qui ont contribué à sa réalisation.

Nos remerciements vont particulièrement à :

- **Au Président et aux membres du jury,** d'avoir accepté d'évaluer ce travail.
- **Pr. MOUANGUE Ruben,** Directeur de l'École Nationale Supérieure Polytechnique de Douala pour son dynamisme et l'abnégation au travail bien fait;
- **M. Cédric NOUMBO,** CEO d'Infinite Solutions qui a été pour moi la figure professionnelle par excellence, un modèle et un guide depuis ma première année de stage dans son entreprise en 2022.
- **Dr. ABBOUBAKAR Hamadjam et Dr. MAKA MAKA Ebenezer,** pour leur suivi rigoureux tout au long de ce travail.
- **L'équipe de développement à Infinite Solution,** pour le soutien et la contribution à ce projet, matérielle comme intellectuelle. Ce projet n'aurait su voir le jour sans la contribution de près comme de loin de toute l'équipe **Infinite Solutions SARL.**
- **Le corps enseignant de l'École Nationale Supérieure Polytechnique de Douala,** pour l'accompagnement, et la qualité de la formation reçue depuis mon admission en son sein.

Notre gratitude à tous ceux qui nous ont soutenus de près ou de loin dans cette œuvre et donc les noms restent anonymes mais plutôt prestigieusement considérés.

AVANT-PROPOS

L'École Nationale Supérieure Polytechnique de Douala (ENSPD), né de la transformation de l'Ex-Faculté de Génie Industriel, ayant vu le jour par le décret présidentiel N2020/272 du 11 Mai 2020, est l'une des institutions de l'Université de Douala. Elle a pour mission principale de former des citoyens aptes à diriger des travaux d'art ou d'industrie, en vue de donner un nouveau souffle à son développement technologique et de lutter contre le sous-développement. Elle offre les domaines de formations suivants:

Ingénierie de la Production et de la Maîtrise de l'Énergie;

- Génie Frigorifique et Climatique ;
- Ingénierie des Systèmes Mécatroniques ;
- Ingénierie Automobile ;
- Ingénierie des Opérations - Capitaine de Pêche ;
- Électromécanique Navale ;
- Ingénierie de l'Énergie Électrique ;
- Ingénierie des Systèmes Automatisés ;
- Qualité et Normalisation ;
- Hygiène Industrielle ;
- Sécurité Industrielle ;
- Environnement Industriel ;
- Construction Mécanique et Productique ;
- Construction Métallique et Tuyauterie ;
- Bâtiments et Constructions Industrielles ;
- Travaux Publics et Ouvrages ;
- Réseaux et Télécommunications ;
- Génie Logiciel ;
- Sécurité des Systèmes d'Information ;
- Management des Systèmes d'Information ;
- Chimie Industrielle ;
- Bio-procédés Industriels ;
- Génie Pharmaceutique ;

- Capteurs, Instrumentations et Mesures ;
- Technologie Biomédicale ;
- Mécanique et Matériau ;
- Géophysique, Eau et Environnement ;
- Électronique, Électrotechnique et Automatique ;
- Énergie.

Les étudiants y sont admis par voie de concours en 1ère année et en 3ème année pour le cursus d'ingénieur et 1ere année pour le cursus des sciences de l'ingénieur et sur étude de dossier pour le master professionnel. Les enseignements y sont organisés en cours magistraux, travaux dirigés, travaux pratiques, travaux personnels, visites d'entreprise et stages techniques.

Les études sont effectuées en trois cycles: Les enseignements du 1er cycle s'étalent sur six semestres et ont pour principal objectif « d'initier les étudiants aux techniques industrielles » afin d'assister les ingénieurs. La validation de toutes les Unités d'Enseignement (UE) du 1er cycle correspondant au quota requis donne droit à une admission au 2nd cycle et à l'obtention d'une Licence en science de l'ingénieur pour le cursus science de l'ingénieur.

Le second cycle s'étend sur quatre semestres dit de « spécialisation ». Les étudiants ayant choisi leur filière en fin de premier cycle se spécialisent en choisissant un axe pour l'élaboration d'un profil particulier et personnel. En effet, l'étudiant a un quota d'unités d'enseignements obligatoires et des optionnelles au choix en fonction de son profil. Les objectifs du 2nd cycle sont :

Donner à l'étudiant les connaissances professionnelles, technologiques et managériales de pointes pour une compétence efficiente en entreprise ;

D'initier l'étudiant à la recherche. Les études du 2nd cycle sont sanctionnées par la validation de tous les stages et Unités d'Enseignement correspondant au nombre de crédits indiqués et, l'obtention du Diplôme d'Ingénieur de l'École Nationale Supérieure Polytechnique de Douala pour le cursus d'ingénieur, celui de master 2 en science de l'ingénieur pour le cursus science de l'ingénieur donnant lieu au passage au 3eme cycle et celui de master 2 professionnel pour le cursus master professionnel. A la fin de nos études, il est obligatoire de produire un mémoire qui sera présenté devant un jury compétent.

RESUME

Le Cameroun est un pays en voie de développement, et l'avènement de la digitalisation se fait à grands pas. Dans un contexte où l'intégration professionnelle des nouveaux diplômés membres de l'ordre des médecins des écoles de médecine devient de plus en plus difficile, ainsi que l'accès aux soins médicaux de qualité en un espace de temps réduit, la plateforme Alice se place comme solution à ces deux problèmes en offrant aux médecins la possibilité d'exercer leur profession en toute liberté, et aux patients d'accéder à cette expertise en temps voulu à des prix raisonnables. Une telle solution est appelée à accueillir un nombre important d'utilisateurs, et la développer requiert une planification adéquate. Nous avons dans ce travail opté pour une approche Cloud Native, mettant en avant les avantages du cloud tels que la scalabilité, la résilience et la disponibilité. Pour faciliter la prise en main de la plateforme, deux modules ont été développées et mis à leur disposition : une application mobile pour les patients, via laquelle ils ont un large choix de médecins vers qui se tourner ; et un tableau de bord web pour permettre aux médecins de gérer leurs consultations et de consulter leurs statistiques. Ainsi les médecins nouvellement diplômés ont une plateforme sur laquelle exercer leurs compétences, et les patients accèdent plus facilement à des consultations de qualité, contribuant ainsi à l'amélioration des conditions de vies des camerounais et à une digitalisation effective.

Mots clés : Digitalisation, médecin, patient, Cloud Native

ABSTRACT

Cameroon is a developing country, and the advent of digitalization is fast approaching. In a context where the professional integration of new medical school graduates is becoming increasingly difficult, as is access to quality medical care in a short space of time, the Alice platform offers a solution to both these problems, giving doctors the opportunity to practice their profession in complete freedom, and patients access to this expertise in a timely fashion at reasonable prices. Such a solution is bound to attract many users, and developing it requires careful planning. In this work, we opted for a Cloud Native approach, highlighting the advantages of the cloud such as scalability, resilience, and availability. To make the platform easier to use, two modules have been developed and made available: a mobile application for patients, via which they have a wide choice of doctors to turn to; and a web dashboard to enable doctors to manage their consultations and consult their statistics. In this way, newly qualified doctors have a platform on which to exercise their skills, and patients have easier access to quality consultations, thus contributing to improving the living conditions of Cameroonians and to effective digitalization.

Keywords: Digitalization, doctor, patient, Cloud Native

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABREVIATIONS

AMQP : Advanced Message Queuing Protocol

API : Application Programming Interface

AWS : Amazon Web Services

CA : Clean Architecture

CD : Continuous Deployment

CEO : Chief Executive Officer

CI : Continuous Integration

CI/CD : Continuous Integration & Continuous Deployment

CNCF : Cloud Native Computing Foundation

COCOMO : Constructive Cost Model

DDD : Domain-Driven Design

DevOps : Development Operations

ENSPD : Ecole Nationale Supérieure Polytechnique de Douala

FIFO : First In First Out

gRPC: Google Remote Procedure Call

HTTP : Hypertext Transfer Protocol

IaC : Infrastructure As Code

JWT : Json Web Tokens

MAUI : Multiplatform App UI

OAuth : Open Authorization

OMS : Organisation Mondiale de la Santé

PDF : Portable Document Format

PDG : Président Directeur Général

RPC : Remote Procedure Call

SPA : Single Page Application

SQL : Structured Query Language

UI : User Interface

UML : Unified Modeling language

WIP : Work In Progress

INTRODUCTION GENERALE

La santé publique dans notre pays reste un sujet délicat. Avec la fin de l'intégration des médecins depuis la promotion 2016-2017 issus des établissements de formation publiques, on observe de plus en plus de médecins se tournant vers le secteur privé. « Le secteur privé ne peut pas absorber toutes ces promotions » [1], en partie à cause des prix exorbitants des consultations au privé qui régis par la réglementation en vigueur varient entre 6 000 Fcfa et 15 000 Fcfa [2]. D'autre part, l' automédication est un fléau réel, qui associé aux médicaments de la rue tue. Selon les statistiques de l'Organisation Mondiale de la Santé (OMS), plus de 55% de la population mondiale vit en zone urbaine, une proportion qui pourrait atteindre les 68% en 2050, engendrant une demande accrue en services de santé [3], ce qui augmente le besoin en service de santé. Les causes de l'automédication sont généralement le manque d'argent ou la difficulté d'accéder à des médecins [4] ceci à cause de l'indisponibilité de ceux-ci ou alors le fait qu'il n'en existe tout juste pas aux alentours. Ce sont des maux étroitement liés au faible taux de disponibilité et d'employabilité des médecins par les centres hospitaliers du Cameroun. Comment d'une part faciliter aux patients l'accès à des médecins compétents et à des coûts raisonnables ? Et par la même occasion, comment permettre aux jeunes médecins nouvellement diplômés de mettre leurs compétences au profit des citoyens, compte tenu de la situation non favorable à leur intégration en centres hospitaliers?

Avec la digitalisation et l'avènement du numérique, de nombreuses perspectives s'offrent à nous. Une étude récente publiée dans le *Journal of Medical Internet Research* révèle que des régressions logistiques multiples ont montré que la probabilité de recourir à des consultations médicales en ligne était positivement associée à un niveau d'études supérieur, au fait de vivre avec un partenaire dans le même ménage, à une moins bonne santé auto-évaluée, à une solitude accrue et à une satisfaction accrue dans la vie [5]. A la lumière de ces données, il est légitime de considérer dans le contexte camerounais, des consultations en ligne comme solution valable aux deux problèmes énoncés plus haut. C'est dans cette lancée que le projet Alice voit le jour. Un tel projet se veut être robuste face aux nombreux utilisateurs camerounais, et pour se faire nous adopterons une approche Cloud Native pour son implémentation.

Il sera question dans le chapitre 1 de faire un état de l'art sur cette architecture ainsi que des piliers qui la régissent. Ensuite dans le chapitre 2 nous présenterons les étapes suivies lors du développement de cette solution à travers les matériels et méthodes. Enfin dans le chapitre 3 nous étalerons les résultats obtenus ainsi que les différentes perspectives à envisager pour un projet d'une telle envergure en considérant le stade que l'on aura pu atteindre au terme de notre cycle de développement. Ces résultats incluront une estimation des coûts de développement de cette solution, ainsi qu'un comparatif avec des alternatives existantes et déjà utilisées sur le territoire Camerounais en particulier.

CHAPITRE 1 : ETAT DE L'ART SUR L'ARCHITECTURE CLOUD NATIVE

Les applications dites "Cloud Native" sont conçues et développées spécifiquement pour être déployées et exécutées dans des environnements cloud. Elles sont construites en utilisant des architectures et des technologies qui exploitent pleinement les avantages offerts par le *Cloud Computing*, comme l'élasticité, la scalabilité, la résilience et la disponibilité. Nous présenterons ce qu'on appelle application Cloud Native ainsi que les principes clé de cette pratique.

1.1. Introduction aux applications Cloud Native

L'évolution rapide des technologies numériques a transformé la manière dont les entreprises conçoivent, développent et déploient leurs applications. Dans ce paysage en constante évolution, l'architecture Cloud Native a émergé comme une approche fondamentalement nouvelle pour la conception et la construction d'applications logicielles modernes. Selon la *Cloud Native Computing Foundation* (CNCF), les systèmes cloud native présentent les propriétés suivantes :

- Les applications ou les processus sont exécutés dans des conteneurs logiciels en tant qu'unités isolées.
- Les processus sont gérés à l'aide de processus d'orchestration centraux afin d'améliorer l'utilisation des ressources et de réduire les coûts de maintenance.
- Les applications ou les services (microservices) sont faiblement couplés avec des dépendances explicitement décrites [6, p. 15].

A tout ceci s'ajoutent des patterns particuliers que nous énoncerons ci-dessous.

1.1.1. Patterns de communication

Dans les systèmes dits Cloud Native, un pattern de communication efficace est essentiel. Il façonne la manière dont les clients front-end interagissent avec les microservices back-end et la manière dont ces microservices communiquent en interne. Comprendre les principes, les modèles et les meilleures pratiques de communication est crucial pour une intégration et une évolutivité transparente.

Ces patterns peuvent se décliner sous deux grandes formes, les communications **synchrones** et les communications **asynchrones**.

Modèles de communication synchrones

Les modèles de communication synchrones impliquent une interaction en temps réelle dans laquelle chaque échange est étroitement lié entre les deux parties concernées.

Request-Response pattern :

En informatique, la requête-réponse ou la requête-réponse est l'une des méthodes de base utilisées par les ordinateurs pour communiquer entre eux dans un réseau, dans laquelle le premier ordinateur envoie une requête pour certaines données et le second répond à la requête. Plus précisément, il s'agit d'un modèle d'échange de messages dans lequel un demandeur envoie un message de requête à un système répondeur, qui reçoit et traite la requête, renvoyant finalement un message en réponse. C'est analogue à un appel téléphonique, dans lequel l'appelant doit attendre que le destinataire décroche avant de pouvoir discuter de quoi que ce soit. Il s'agit d'un modèle de messagerie simple mais puissant qui permet à deux applications d'avoir une conversation bidirectionnelle sur un canal ; c'est particulièrement courant dans les architectures client-serveur[7, p. 184].

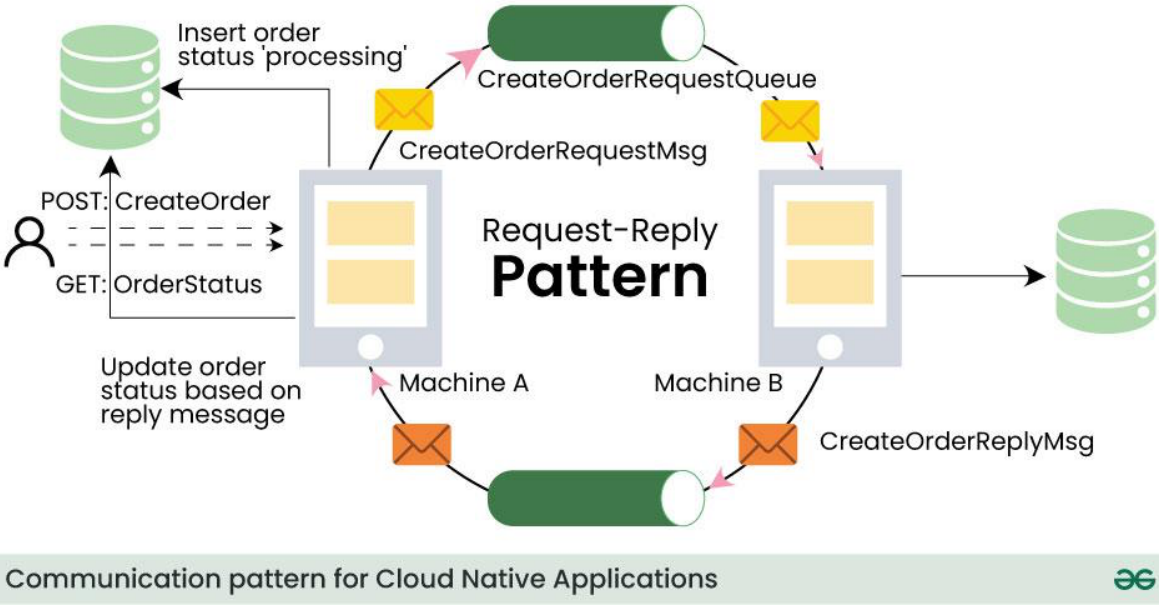


Figure 1.1: Request-Response communication pattern [8]

gRPC:

gRPC est un cadre moderne et performant qui fait évoluer le protocole RPC (Remote Procedure Call). Au niveau de l'application, gRPC rationalise la messagerie entre les clients et les services dorsaux. Créé par Google, gRPC est un logiciel libre qui fait partie de l'écosystème de la CNCF. La CNCF considère gRPC comme un projet en cours d'incubation. Incubation signifie que les utilisateurs finaux utilisent la technologie dans des applications de production et que le projet compte un nombre important de contributeurs [9, p. 90].

Voici comment cela fonctionne généralement :

- **Le client appelle une procédure distante** : Le programme client (ou l'application cliente) appelle une procédure ou une fonction qui est située sur un autre ordinateur, souvent appelé serveur.
- **Le client envoie une requête RPC** : Lorsque le client appelle la procédure distante, il envoie une requête RPC contenant des informations sur la procédure à appeler, ainsi que les paramètres nécessaires à son exécution.
- **Le serveur reçoit la requête RPC** : Le serveur reçoit la requête RPC et identifie la procédure à exécuter en fonction des informations contenues dans la requête.
- **Le serveur exécute la procédure** : Une fois que le serveur a identifié la procédure à exécuter et a reçu les paramètres nécessaires, il exécute la procédure comme demandé.
- **Le serveur renvoie le résultat au client** : Une fois que la procédure est exécutée, le serveur envoie le résultat de la procédure (s'il y en a un) au client via une réponse RPC.
- **Le client reçoit le résultat** : Le client reçoit le résultat de la procédure distante et peut l'utiliser comme nécessaire dans son programme.

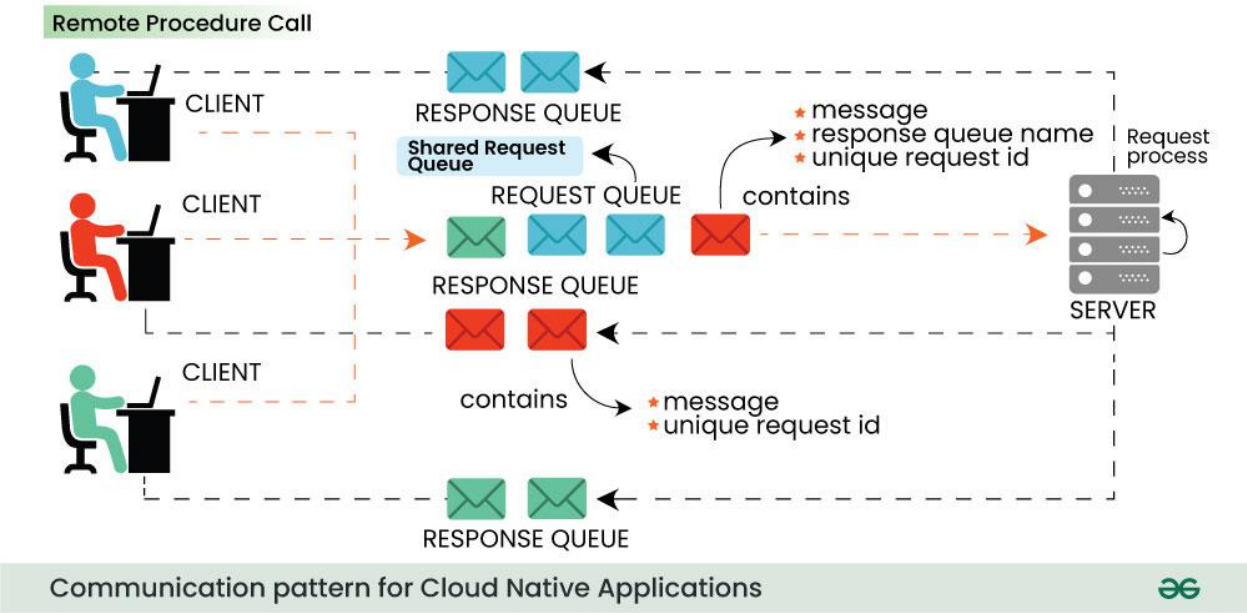


Figure 1.2: Remote Procedure Call [8]

Modèles de communication asynchrones

Contrairement aux patterns synchrones où lorsque le client effectue une requête il s'attend directement à une réponse, en asynchrone en revanche la réponse peut être retardée dans le temps, ce qui permet notamment aux différents intervenants de manager de façon plus efficientes les ressources à mettre à disposition pour l'exécution de la tâche en question.

Publish-Subscribe pattern :

Ce pattern implique généralement au moins trois composants :

- **Un Publisher**: qui dans un cas d'utilisation est celui qui produit le message ou l'évènement.
- **Un message broker :** qui distribue ledit message à tous les concernés. Ils peuvent aller de 0 à n dépendamment des ressources mises à disposition.
- **Un Subscriber :** celui-ci s'abonne auprès du message broker en tant que listener pour un évènement particulier. Ainsi le message broker chargé de la distribution des messages fera le dispatch en tenant en compte ce nouvel abonné.

Cette approche a l'avantage d'être *decoupled*, ce qui signifie que les composants ne sont pas directement liés les uns aux autres, mais juste au message broker.

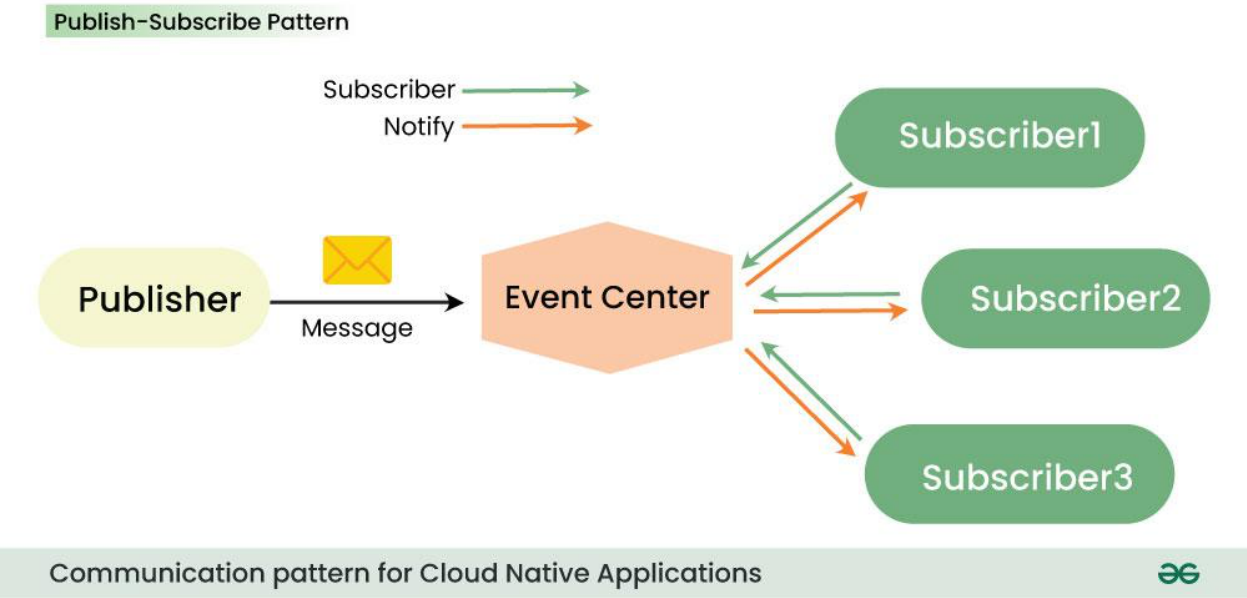


Figure 1.3: Publish-subscribe pattern [8]

Ceci permet entre autres d'avoir une structure dont les éléments peuvent évoluer de façon indépendante, ainsi que d'avoir des traitements asynchrones donc « non bloquants ».

Message Queue pattern :

En ingénierie logicielle, une queue est une structure de données respectant un principe très simple du **FIFO** (*First In First Out)* dans laquelle les éléments rentrent et sortent les uns à la suite des autres respectant leur ordre d'entrée.

Une queue peut être utilisée pour multiples raisons, pour des échanges asynchrones entre différents composantes d'un système. Les raisons principales de son utilisation sont les suivantes :

- **Besoin de garder un ordre entre les messages et leur traitement dans le temps :** dans un système où un composant doit traiter des commandes en chaîne, la notion d'ordre est primordiale et doit être préservée.
- **Besoin d'optimisation du flux et de l'utilisation des ressources :** un système à ressources limitées ne permettant d'effectuer qu'un certain nombre d'opérations de façon simultanée se retrouvera très vite en épuisement de ressources lorsque le flux devient trop important. Une solution est de traiter en paquet un certain nombre d'opérations simultanées, tout en enregistrant les nouvelles en file d'attente pour les traiter une fois celles en cours achevées

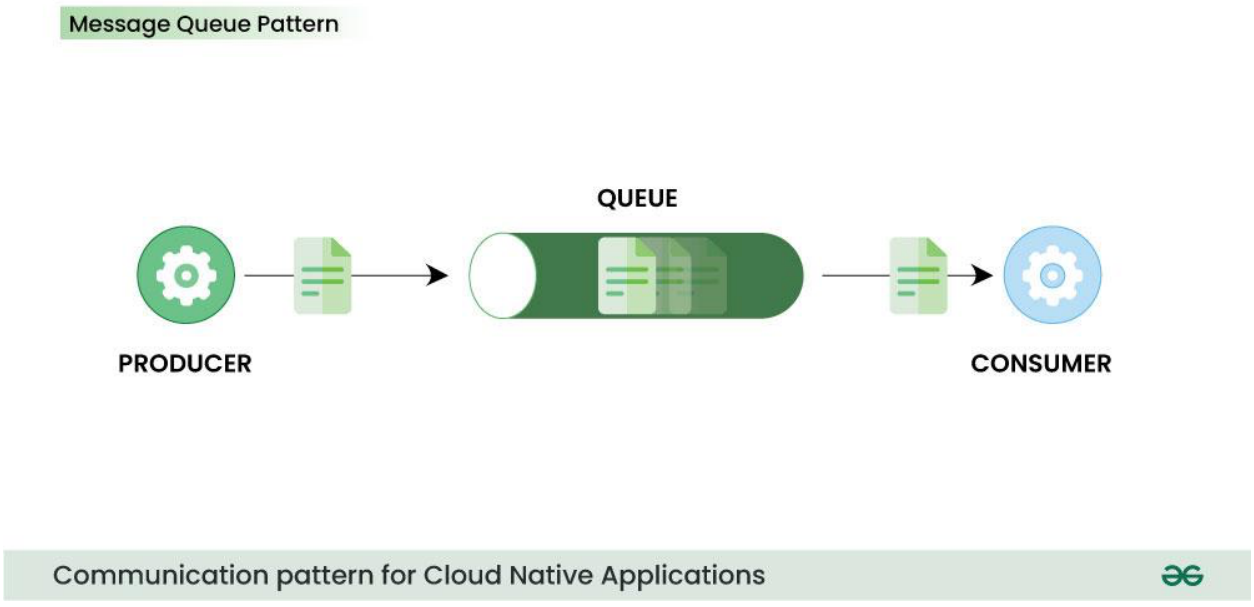


Figure 1.4 : Queue communication pattern [8]

1.1.2. Patterns de gestion de données

Une autre des challenges de l'architecture cloud native est la gestion des données. Les deux modèles sont en vigueur dans l'industrie : **Monolithe** & **Microservices.** Lorsque dans un monolithe on a une seule unité de déploiement et une seule base de données, en microservices en revanche chaque service isolé du reste a une base de données dédiée.

En cloud native, la gestion de la base de données est dirigée par plusieurs aspects, qui peuvent inclure le budget global du projet, les contraintes fonctionnelles et d'architecture définies en amont (le choix d'isoler les données ou pas), et bien sûr les compétences globales des équipes de développement.

Une séparation des bases de données est bénéfique dans un contexte de système à large échelle hautement scalable et indépendants. Il en va de même pour le choix d'un microservices ou d'un monolithe. Plusieurs bases de données impliquent un plus grand besoin de maintenance, des coûts d'infrastructure et des soucis de redondance et d'incohérence des données si cela est mal mis en place. Cependant cela garantit une scalabilité horizontale, des performances locales (locales à un service) plus élevées.

Le choix d'une seule base données peut sembler alors idéal, mais reste néanmoins sujet à discussions. Le principale avantage d'un tel choix est la simplicité dans le déploiement et la maintenance, au détriment des performances, de la sécurité et des possibilités de scalabilité. Ce choix se justifie généralement dans les applications de petite et moyenne taille, où les soucis liés à la disponibilité des données sous haute charge ne se font pas ressentir. La Figure 1.5 illustre la façon dont les données sont gérées dans les applications Cloud Native.

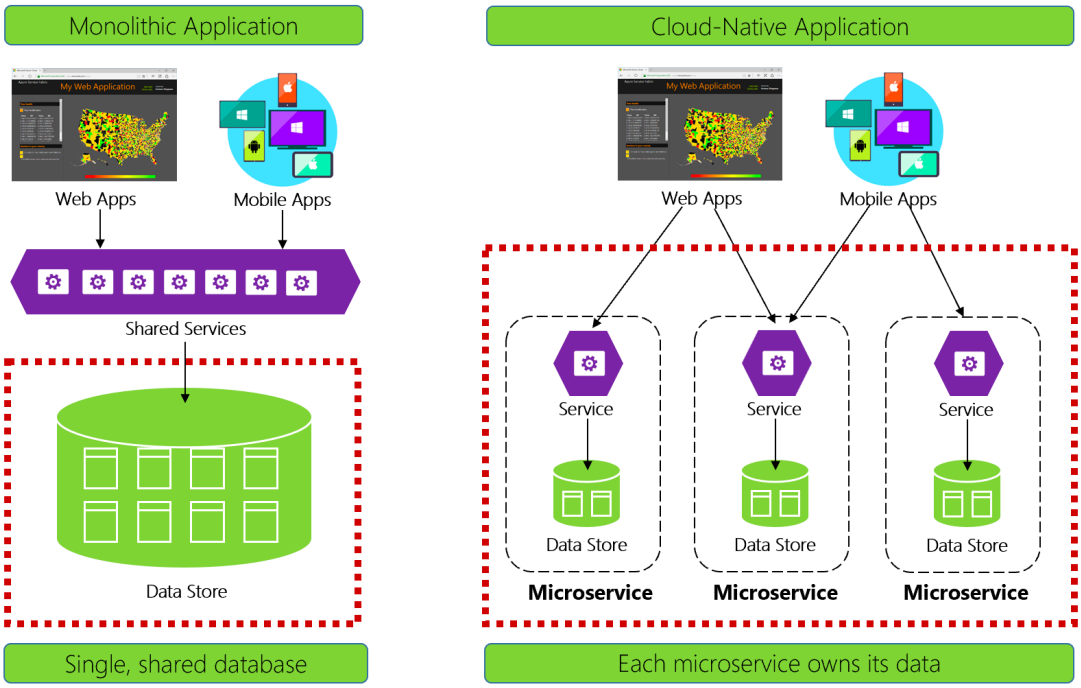


Figure 1.5 : Data management in cloud-native applications [9, p. 98]

1.1.3. Automatisation et DevOps

Avec l'évolution des technologies et des pratiques, un mouvement émerge, le **DevOps.** D'après AWS (Amazon Web Services), le DevOps peut de définir comme suit :

> *"DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market [10]."*

Cette définition dite simplement, représente l'ensemble des pratiques qu'une entreprise ou une organisation peut mettre en place pour un développement plus rapide et plus sûr des applications et service.

Ceci implique des pratiques telles que le **CI/CD** (*Continuous Integration & Continuous Deployment*) qui se traduisent littéralement en **Intégration en Continu & Déploiement en Continu.**

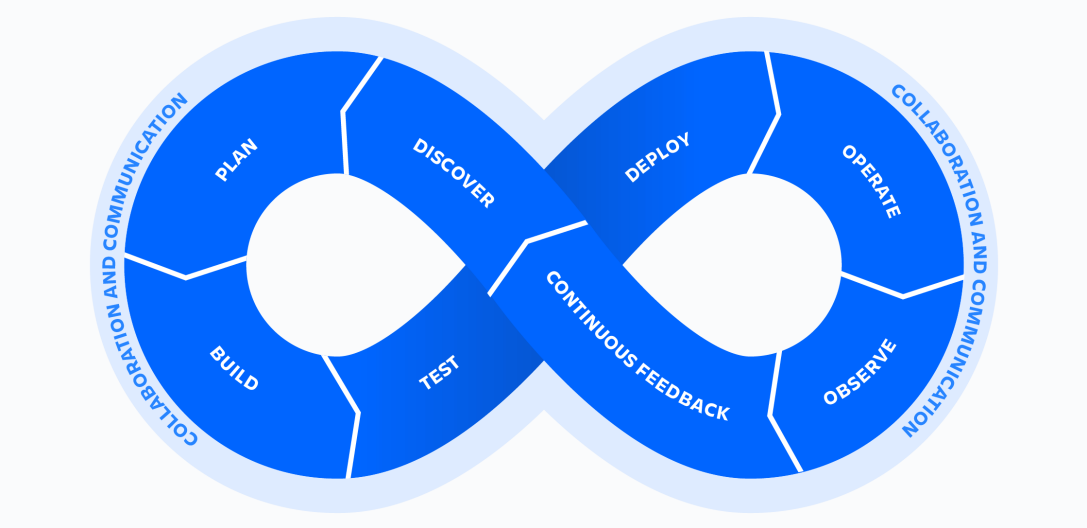


Figure 1.6 : Cycle du DevOps [11]

Ce cycle constitue un ensemble d'étapes sur lesquelles l'on itère pour au fur et à mesure peaufiner le produit, le service ou la solution que l'on propose.

On peut expliquer chaque étape comme suit :

Découverte / Discover

La phase de découverte consiste à identifier les besoins et les exigences du projet. Dans les approches traditionnelles, cette étape peut être fragmentée et manquer de clarté. Le DevOps améliore cette phase en encourageant une communication continue et une collaboration étroite entre les parties prenantes, garantissant une compréhension commune des objectifs et des attentes.

Planification / Plan

La planification traditionnelle peut être rigide et créer des blocages. Le DevOps adopte une approche agile pour planifier les tâches de manière itérative, permettant une adaptation rapide aux changements et une meilleure allocation des ressources. Cela assure que les projets avancent de manière fluide et coordonnée.

Construction / Build

Construire des applications de manière efficace peut être entravé par des processus manuels et des incohérences. Le DevOps automatise le processus de construction avec des outils de gestion de versions et de compilation, garantissant des builds reproductibles et cohérents, ce qui accélère le développement et réduit les erreurs.

Tester / Test

Les tests manuels sont souvent longs et sujets à des erreurs humaines. Le DevOps intègre des tests automatisés tout au long du pipeline CI/CD, assurant une validation continue du code à chaque modification. Cela permet de détecter les problèmes rapidement et d'améliorer la qualité du logiciel.

Déployer / Deploy

Le déploiement traditionnel peut être complexe et risqué. Le DevOps utilise des techniques comme le déploiement continu (CD) pour automatiser et standardiser ce processus, permettant des mises en production fréquentes et fiables sans interruption majeure du service.

Opérer / Operate

Les opérations manuelles peuvent être inefficaces et sujettes aux erreurs. DevOps adopte l'infrastructure en tant que code (IaC) pour automatiser la gestion des ressources, assurant une infrastructure cohérente, évolutive et facile à gérer, même dans des environnements complexes.

Observation / Observe

Le manque de visibilité sur les performances et les problèmes peut entraver la résolution rapide des incidents. DevOps met en place des outils de surveillance et de journalisation avancés pour observer en temps réel les performances et les comportements des applications, permettant une détection proactive des anomalies.

Retour continu / Continuous feedback 

Les retours traditionnels sont souvent tardifs et peu fréquents. DevOps instaure un cycle de feedback continu, recueillant des commentaires après chaque déploiement et intégrant ces retours dans le processus de développement. Cela favorise une amélioration constante et une adaptation rapide aux besoins des utilisateurs.

1.2. Étude de l'existant

L'architecture Cloud Native est une pratique assez récente qui regroupe l'évolution de plusieurs patterns utilisés au paravent. Ceux-ci sont principalement les applications Monolithiques traditionnelles, ensuite l'approche Service Oriented qui a évolué en Microservices pour aujourd'hui être l'approche Cloud Native que nous étudions.

1.2.1. Monolithes traditionnels

L'architecture monolithique dans l'écosystème .NET est une approche robuste qui centralise toutes les responsabilités du système au sein d'une seule application. Approfondissons ses caractéristiques, ses avantages, ses inconvénients, ainsi que les stratégies de mise à l'échelle et les exemples pratiques [12].

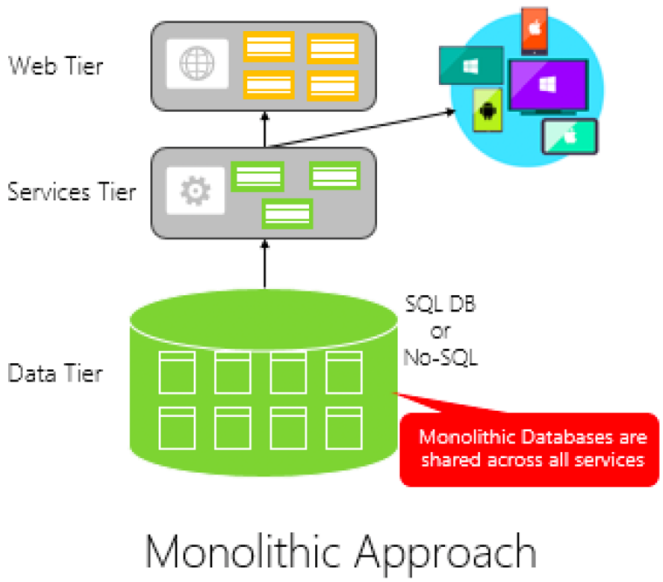


Figure 1.7: Approche monolithique [9, p. 20]

Cette approche présente les avantages suivants :

Simplicité : L'architecture monolithique simplifie le développement et les tests en regroupant tous les éléments en un seul endroit, ce qui élimine la nécessité de mettre en place et de maintenir la communication entre les services.

Performance : L'absence de communication sur le réseau offre des performances supérieures à celles des architectures distribuées.

Évolutivité : La mise à l'échelle d'une application monolithique peut s'avérer plus difficile, car elle implique la mise à l'échelle de l'ensemble du système plutôt que de composants individuels.

Couplage : Si un couplage étroit facilite la compréhension du système, il peut également compliquer la mise en œuvre des changements et l'ajout de nouvelles fonctionnalités [12].

Cette approche a prouvé sa robustesse dans le temps, du fait que plusieurs applications l'adoptent. Néanmoins, elle a également des limitations. Les inconvénients d'une application monolithique sont les suivants :

Ralentissement de la vitesse de développement pour les larges applications : Une grande application monolithique rend le développement plus complexe et plus lent.

Évolutivité : Il n'est pas possible de faire évoluer les composants individuels.

Fiabilité : Une erreur dans l'un des modules peut affecter la disponibilité de l'ensemble de l'application.

Obstacle à l'adoption de technologies : Toute modification du cadre ou du langage affecte l'ensemble de l'application, ce qui rend les changements souvent coûteux et fastidieux.

Manque de flexibilité : Un monolithe est limité par les technologies déjà utilisées dans le monolithe.

Déploiement : Une petite modification apportée à une application monolithique nécessite le redéploiement de l'ensemble du monolithe [13].

Ce sont ces limitations qui conduisent à l'adoption de la méthodologie Cloud Native et Microservices.

1.2.2. Microservices et Cloud Native

Construits comme un ensemble distribué de petits services indépendants qui interagissent par le biais d'une structure partagée, les microservices partagent les caractéristiques suivantes:

- Chaque Microservice met en œuvre une capacité commerciale spécifique dans un contexte de domaine plus large.
- Chaque Microservice est développé de manière autonome et peut être déployé indépendamment.
- Chaque Microservice est autonome et encapsule sa propre technologie de stockage de données, ses dépendances et sa plateforme de programmation.
- Chaque Microservice s'exécute dans son propre processus et communique avec les autres à l'aide de protocoles de communication standard tels que HTTP/HTTPS, gRPC, Websockets ou AMQP.
- Chaque Microservice se compose pour former une application [9, p. 20].

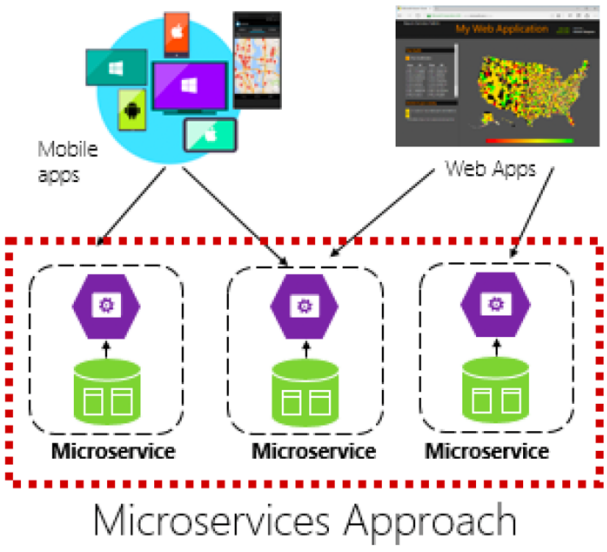


Figure 1.8: Approche Microservices [9, p. 20]

Nous parlerons de Cloud Native lorsque ces services sont déployés dans un environnement Cloud en respectant les normes énoncées à la section 1.1.

1.2.3. Applications dans le domaine des consultations en ligne au Cameroun

Le problème d'accès aux médecins compétents et à des consultations de qualité en temps et en heure au Cameroun existe depuis longtemps et a déjà été abordé de plusieurs solutions. L'on s'attardera sur une solution en particulier : **Waspito.**

Waspito est une plateforme et une application de soins de santé qui offre des services tels que la consultation en ligne, la livraison de médicaments, des médecins spécialistes, etc. Basée à Douala, au Cameroun, et fondée en 2020 par son PDG, Jean Lobe Lobe, Waspito Inc. opère en tant que société d'e-santé qui met en relation les utilisateurs avec des médecins via des consultations vidéo à partir de leurs smartphones. En novembre 2023, Waspito Inc. a levé 2,5 millions USD en financement d'amorçage auprès d'Axian Group, Asia Africa Investment & Consulting, et Health54, avec la participation de Newtown Partners et Saviu Ventures. Waspito propose une application mobile de télésanté qui permet aux utilisateurs de choisir des médecins de différentes spécialités et d'avoir une visite par appel vidéo pour obtenir des soins, recevoir une ordonnance au format PDF et obtenir des conseils de santé à partir de leurs articles sur les soins de santé. L'entreprise propose également des services de laboratoire avec un technicien de laboratoire qui prélève des échantillons au domicile de l'utilisateur et fournit des résultats numériques. La société prévoit d'utiliser les fonds levés en novembre 2023 pour soutenir sa croissance sur les marchés francophones [14].

Une étude comparative de Waspito et Alice est détaillée dans le tableau suivant :

Tableau 1.1: Comparatif entre les applications Waspito et Alice

Fonctionnalités	Waspito	Alice
Classement de médecins	✅	✅
Tri par catégorie	✅	✅
Encryptage des données	✅	✅
Notation des médecins	❌	✅
Messagerie instantanée	❌	✅
Appels audio vidéo	✅	❌

Ordonnance en ligne	✅	✅

La différence notable entre ces deux plateformes est l'expérience utilisateur. Alice vise à permettre aux utilisateurs un moyen rapide d'accéder à une consultation de qualité. Pour ce faire, aucune autre fonctionnalité ne pollue la principale, contrairement à Waspito qui au fil des années s'est plus orienté vers un réseau social où les médecins exposent leur expertise. Alice se veut de rester orienté vers la seule fonctionnalité dont l'utilisateur a besoin, qui est d'accéder à une consultation médicale.

Les manquements notés par rapport à Waspito tels que les appels audio et vidéo sont néanmoins très pertinents, puisqu'ils facilitent l'échange entre les deux partis prenants d'une consultation. Ils seront implémentés progressivement lors des différentes versions de la solution.

1.3. Avantages de l'approche Cloud Native

Suivre une approche Cloud Native présente de nombreux avantages, parmi lesquels :

Un développement plus rapide

Les développeurs utilisent l'approche cloud-native pour réduire le temps de développement et obtenir des applications de meilleure qualité. Au lieu de s'appuyer sur une infrastructure matérielle spécifique, les développeurs construisent des applications conteneurisées prêtes à être déployées avec des pratiques DevOps. Cela permet aux développeurs de répondre rapidement aux changements. Par exemple, ils peuvent effectuer plusieurs mises à jour quotidiennes sans arrêter l'application [15].

Rapport coût-efficacité

En adoptant l'approche "cloud-native", les entreprises n'ont pas besoin d'investir dans l'acquisition et la maintenance d'une infrastructure physique coûteuse. Cela se traduit par des économies à long terme sur les dépenses opérationnelles. Les économies réalisées grâce à la mise en place de solutions "cloud-native" pourraient également profiter à vos clients [15].

Stabilité et haute disponibilité des applications

Un environnement de développement sur site est toujours vulnérable aux temps d'arrêt. Le cloud, en revanche, est toujours disponible, de sorte qu'il n'y a pas de perturbation du processus de

développement ou des applications tant que le cloud est accessible. La résilience des microservices garantit également que l'application continue de fonctionner même si l'un de ses composants tombe en panne [16].

De plus, en intégrant les principes du DevOps nous bénéficions des avantages suivants :

Collaboration améliorée

La séparation entre les équipes de développement et d'exploitation crée des silos et des malentendus. Le DevOps encourage la collaboration en brisant ces silos, en favorisant une communication ouverte et une responsabilité partagée tout au long du cycle de vie du logiciel. Couplée aux méthodes agiles énoncées à la section 2.4.1, nous obtenons un développement

Sécurité

L'approche DevSecOps vise à automatiser l'intégration de la sécurité à chaque phase du cycle de vie du développement logiciel, de la conception initiale à la livraison du logiciel, en passant par l'intégration, les tests et le déploiement [17]. Contrairement aux approches traditionnelles où la sécurité est ajoutée en fin de processus, le DevSecOps assure que les pratiques de sécurité sont intégrées tout au long du développement, des tests et du déploiement. Cela permet de détecter et de corriger les vulnérabilités plus tôt, réduisant ainsi les risques et les coûts associés aux correctifs tardifs. En automatisant les contrôles de sécurité et en les intégrant aux pipelines CI/CD, DevSecOps garantit que la sécurité est une responsabilité partagée entre toutes les équipes, favorisant une culture de sécurité continue et proactive.

1.4. Inconvénients de l'approche Cloud Native

L'approche Cloud Native tire profit des avantages du cloud pour offrir des services scalables et hautement disponibles. A ceci s'ajoutent les méthodes agiles et le DevOps. Ces pratiques misent en commun contribuent à un développement efficace des applications à large échelle, mais présentent néanmoins des limites. Ces inconvénients s'illustrent principalement comme suit :

Complexité

La mise en place de ce type d'architecture n'est pertinente que sur des projets d'une certaine taille. Il s'agit généralement de grands projets impliquant plusieurs fonctionnalités et appelés à évoluer avec le temps. Ces concepts appliqués à des applications de taille relativement petite s'avèrent être

contre productifs. Ces applications s'appuient sur des services cloud tels que les conteneurs, Kubernetes et l'informatique sans serveur, dont la gestion peut s'avérer délicate [18].

Compétences requises

Le développement d'applications cloud-natives nécessite un bon travail d'équipe entre les équipes de développement, d'exploitation et de sécurité. Cela peut s'avérer délicat si l'on ne dispose pas des bons outils et processus. Les développeurs doivent donc gérer la complexité, s'attaquer aux problèmes de sécurité et investir des ressources et de l'expertise pour faire fonctionner les applications natives du cloud [18].

Dans ce premier chapitre nous avons présenté de façon générale la terminologie ‘‘Cloud Native’’ ainsi que les règles qui régissent cette pratique. L’on a abordé les différents aspect et avantages subséquents, et détaillé comment l’implémenter en fonction du cadre spécifique dans lequel se situé notre application. Nous poursuivons par une exposition de la méthodologie utilisée lors du développement, qui s’illustre par les matériel et méthodes.

CHAPITRE 2 : MATERIEL ET METHODES

Dans ce chapitre, nous allons détailler les matériels et méthodes utilisés pour la conception et le développement de notre plateforme de consultations médicales en ligne. Nous explorerons l'environnement de développement, les technologies adoptées, ainsi que les méthodologies de gestion de projet qui ont guidé notre travail. Nous expliquons également comment l'architecture Cloud Native, combinée à des principes de Clean Architecture et de Domain-Driven Design, a été mise en œuvre pour garantir une solution robuste, scalable et maintenable. Enfin, nous décrivons les processus d'intégration, de déploiement continu et de tests qui ont été essentiels pour assurer la qualité et la performance de l'application.

2.1. Environnement de Développement

Pour mener à bien ce projet, beaucoup d'outils ont joué un rôle crucial dans le processus de développement. On peut les catégoriser en outils et technologies. Le choix de ces différents a été orienté par deux facteurs, premièrement les standards de l'entreprise qui nous a accueilli, ensuite les compétences globales des équipes à disposition.

2.1.1. Outils de développement

Les outils de développement regroupent tous les outils logiciels et matériel ayant servi lors du développement. En commençant par les outils matériels, nous pouvons citer :

- **Laptop Apple MacBook Pro 2018:** qui est l'ordinateur utilisé tout au long du développement, offrant des performances suffisantes pour la tâche qui nous incombe. Ce laptop par sa légèreté permet également de se déplacer et de travailler à tout moment peu importe le lieu où l'on se trouve.
- **Moniteur Samsung :** travailler avec un écran externe permet d'élargir le poste de travail, d'ouvrir plus de fenêtres simultanément et d'aérer les fenêtres du poste de travail. Ceci joue un rôle important dans la productivité du développeur.

Par la suite, des outils logiciels ont été ajoutés à cette équation, nous pouvons en citer les plus importants :

- **Visual Studio Code :** Constitue l'éditeur de code principal qui a été utilisé pour écrire, compiler et débugger le code source de la plateforme.
- **JetBrains Rider :** Qui est l'éditeur de code secondaire qui a été utilisé.
- **Xcode :** l'application étant cross platform Android et iOS, le logiciel Xcode est requis pour développer des applications compatibles avec les logiciels Apple. Dans le cas d'espèce elle a surtout servi à afficher le rendu sur des téléphones iPhone virtuels (Simulateurs).
- **Android Studio :** à l'instar de Xcode pour iOS, Android Studio permet une meilleure prise en main des plateformes Android pour les développeurs mobiles. Ce logiciel nous a permis d'avoir un rendu sur téléphones Android virtuels (émulateurs).
- **Docker :** étant sur un environnement macOS, développer une plateforme utilisant une base de données Microsoft SqlServer revient assez compliqué du fait que cette base de données n'est disponible en local que sous les systèmes Windows. Pour permettre le développement local, nous avons conteneurisé une base de données SqlServer pour ensuite l'utiliser sur Docker avec macOS.
- **Figma :** utilisé pour établir les maquettes de l'applications.

2.1.2. Technologies utilisées

Ainsi que mentionné dans le thème, le développement de cette plateforme est axé sur les technologies **.NET** et **Microsoft Azure**. Pour une meilleure clarté dans l'énumération des technologies utilisées, nous les regrouperons suivant le module dans lequel elles se font sentir.

2.1.2.1. Service backend API

Pour ce service l'on retrouve les technologies suivantes

- **ASP.NET Core 8 :** Framework .NET servant au développement d'applications web d'entreprise, robustes et suivant les bonnes pratiques d'industrie. Il offre un système de routing et de middlewares flexible et adaptable à divers scénarios.
- **SignalR :** est une technologie développée à base du protocole Websockets qui permet d'ajouter des fonctionnalités de *real-time connectivity* à des applications à la volée. Cette technologie a été utilisée pour implémenter la messagerie instantanée et les appels via la plateforme.

- **Microsoft SqlServer :** la base de données utilisée en production, qui s'intègre facilement dans l'écosystème .NET. Ainsi le développement et le déploiement en production restent cohérents du fait de l'uniformité de la base de données utilisée dans ces deux environnements.

2.1.2.2. Application mobile cross Platform

Aujourd'hui, s'assurer de toucher la cible souhaitée pour une application mobile c'est miser sur le plus de plateformes possibles. Les deux principales étant iOS et Android, nous avons opté pour un framework dit cross-platform du fait qu'il permette avec un seul et même code source, de générer une application compatible avec plusieurs plateformes différentes.

Il s'agit de **.NET MAUI** (Multi-platform App UI) qui est un framework créé par Microsoft en 2022, avec pour but de permettre le développement d'applications natives depuis un seul code source. Il permet le développement et le déploiement rapide du fait que l'on se concentre sur un seul code et l'on arrive à un résultat universel.

2.1.2.3. Dashboard web

L'interface des médecins est accessible via un Dashboard web interactif qui leur permet facilement et en tous lieux d'accéder à la plateforme. Il est développé en utilisant le framework **Blazor WebAssembly 8** qui est une technologie SPA (Single Page Application) développé par Microsoft en concurrence avec les plus courants tels que React JS, Angular ou Vue JS. Il permet d'avoir des interfaces riches et interactives, en utilisant comme technologies de développement C# et .NET.

2.2. Analyse Préliminaire

Avant l'implémentation de tout projet logiciel sur le long terme, des analyses et études sont primordiales pour s'assurer du bon déroulement de ce projet-là. La plateforme Alice fait intervenir deux acteurs principaux tel que l'illustre la Figure 2.1 suivante.

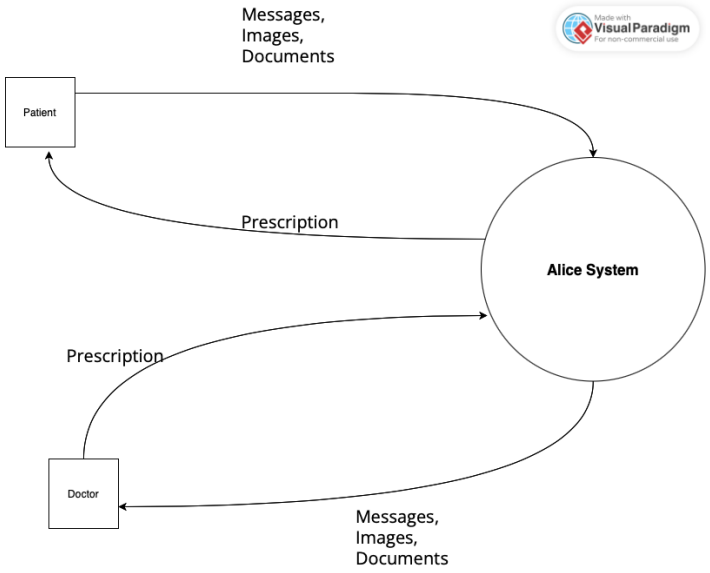


Figure 2.1: Diagramme de contexte de la plateforme Alice

Le développement de cette solution peut être subdivisé en 4 principaux groupes de fonctionnalités :

- **La gestion de l'authentification :** afin de s'assurer de l'identité des utilisateurs de l'application, et de la fiabilité des informations échangées ;
- **La gestion des consultations :** il s'agit du module principal de cette plateforme, qui regroupe les fonctionnalités liées à l'interaction entre les patients et les médecins ;
- **La gestion des profils :** il s'agit des fonctionnalités de gestion des profils de médecins, leur permettant de modifier leurs informations personnelles ;
- **L'administration générale de la plateforme :** ce module est destiné aux administrateurs de la plateforme, il facilite l'administration de la plateforme en validant les requêtes d'enregistrement des médecins sur la plateforme, ainsi que toutes les configurations relatives aux paiements.

La figure ci-dessous illustre comment ces différents modules sont représentés dans un diagramme de packages.

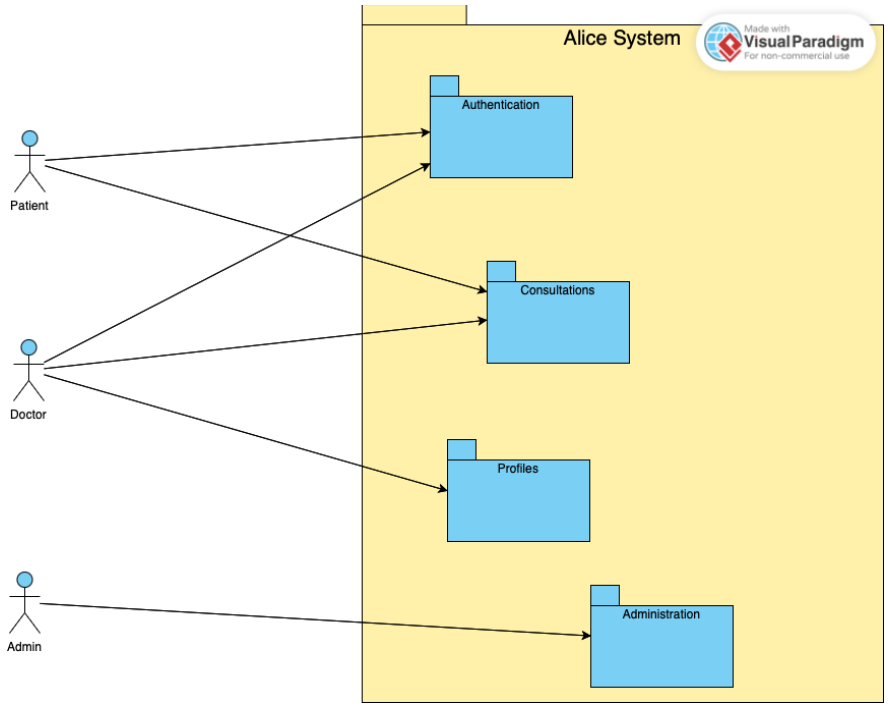


Figure 2.2: Diagramme de packages d'Alice

2.2.1. Analyse des cas d'utilisations

L'analyse des cas d'utilisation est une méthode de modélisation du système qui décrit les fonctionnalités d'un système à travers les interactions entre les utilisateurs et le système. Elle se base sur le cahier des charges établi en amont et sur une discussion avec les partis prenantes.

Un diagramme des cas d'utilisations met principalement en évidence les différents acteurs intervenants dans la plateforme ainsi que les cas d'utilisations qui leur sont associés.

2.2.1.1. Package d'authentification

Tout accès à l'application est mandaté par une authentification rigoureuse. Les utilisateurs devront s'identifier en s'enregistrant sur la plateforme, ensuite en se connectant pour avoir accès. Des modes d'authentification modernes ont été utilisés, notamment OAuth(Google, Microsoft, Apple), JWT et Cookies.

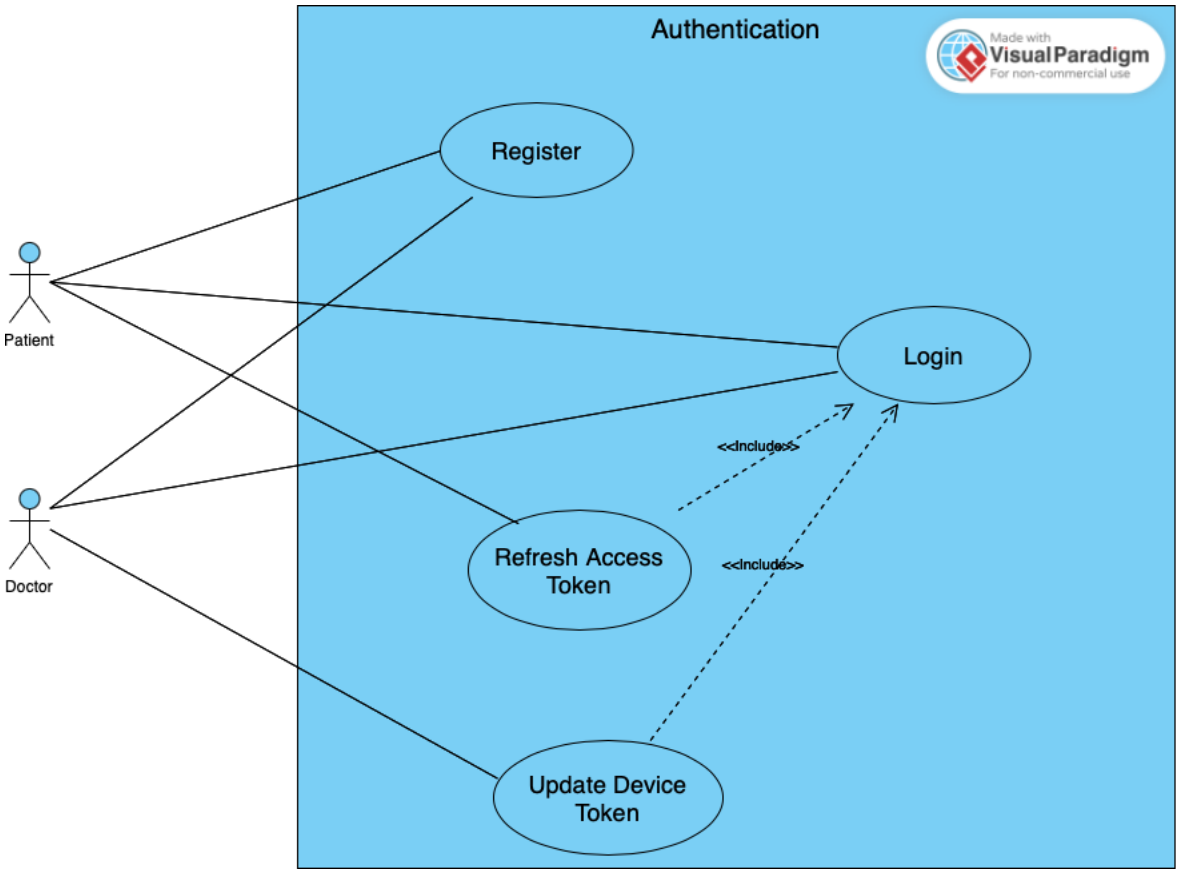


Figure 2.3: Diagramme de cas d'utilisations pour l'authentification

2.2.1.2. Package des consultations

Le module des consultations représente le module principal de cette application, car regroupe les fonctionnalités les plus importantes. La figure suivante détaille les cas d'utilisations de ce package.

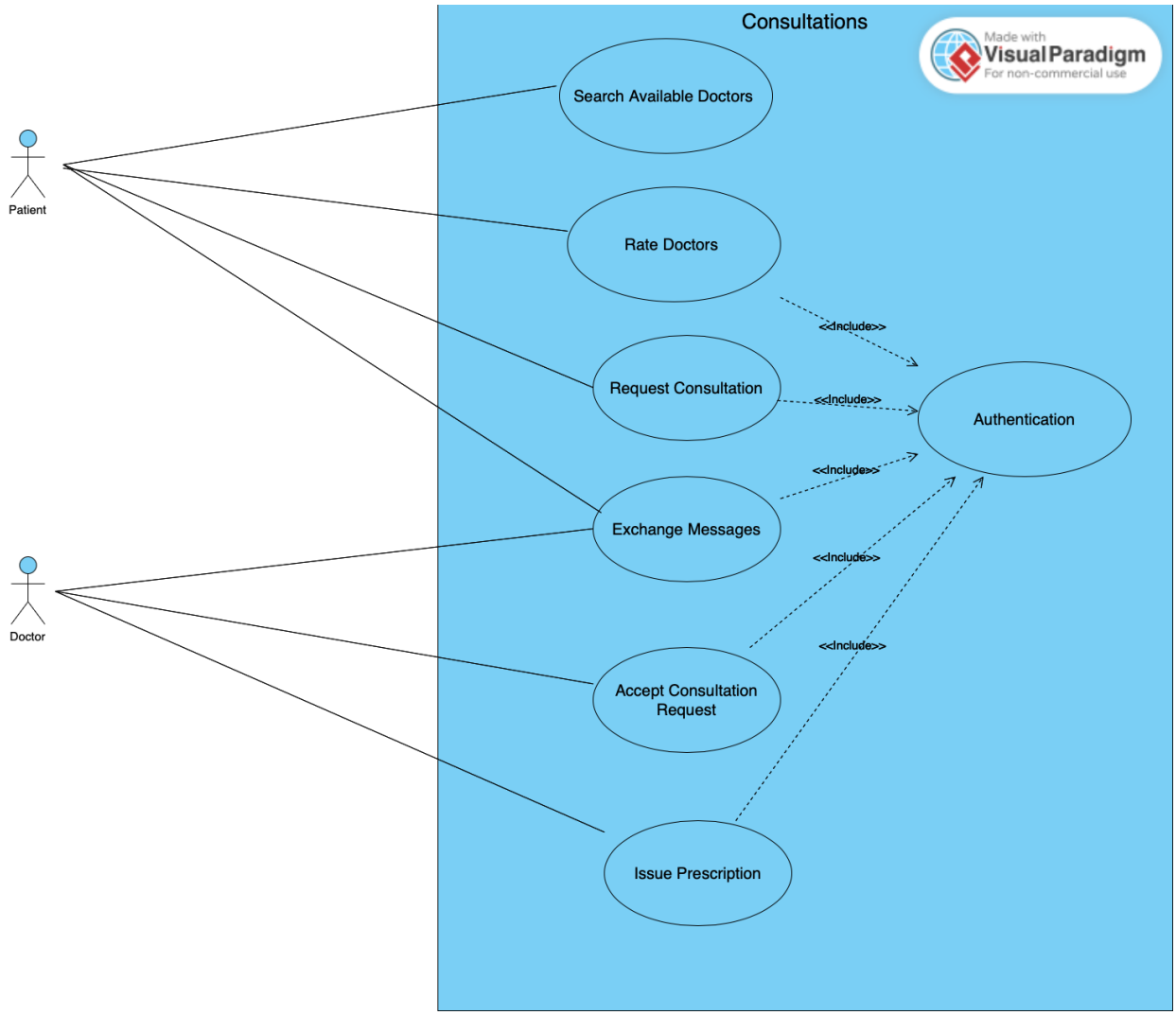


Figure 2.4: Diagramme de cas d'utilisations pour la gestion des consultations

i. Patient

Un patient sur la plateforme représente cet acteur qui a un souci de santé et se sert de l'application pour rentrer en contact avec un médecin pour se procurer des soins.

Ses cas d'utilisations s'orientent de ce fait autours de la consultation médicale.

- **Recherche et tris de médecins:**

  Les patients sur la plateforme peuvent consulter une liste exhaustive des médecins disponibles et connectés à cet instant précis. Leur permettant ainsi d'effectuer des recherches plus précises et d'avoir un classement à jours.

- **Requête de consultation**

La partie cruciale de l'application est le processus de consultation en ligne. En effet il se base sur une requête émise par le patient, qui est acheminée instantanément aux médecins pour validation. Une fois les deux partis prenant accord, la consultation se lance automatiquement et ils ont accès à un chat. Le chat permet un échange interactif, ainsi que des fonctionnalités supplémentaires telles que des appels audios et vidéos.

- **Noter un médecin**

  Pour une meilleure recommandation des médecins sur la plateforme, un mécanisme de classements est mis en place, basé sur un score calculé de façon incrémentale à chaque fois qu'un utilisateur note un médecin. Pour l'implémentation d'un classement fiable et rigoureux, plusieurs solutions sont envisageables. La première était de faire une simple moyenne de la valeur de chaque vote par le nombre de votants, qui peut sembler juste mais qui finalement ne l'est pas. Un médecin ayant un seul vote d'une valeur 5 aura une moyenne de 5 et sera mieux classé qu'un médecin ayant 30 votes chacun d'une valeur de 4 puisque sa moyenne vaudra 4, donc moins que la moyenne 5 du premier médecin. Ce qui conduit à un classement erroné.

  Notre choix s'est porté sur un algorithme en particulier : ***The Wilson Score Interval***.

$$\left(\hat{p} + \frac{z_{\propto/2}^2}{2n} \pm z_{\propto/2}\sqrt{\left[\hat{p}(1-\hat{p}) + z_{\propto/2}^2/4n\right]/n}\right)/\left(1 + z_{\propto/2}^2/n\right)$$

  Où

  $\hat{p} =$ (note moyenne d'étoiles)/(nombre maximal d'étoiles)

  $n =$ nombre total d'évaluations

  $z_{\propto/2} = 1.96 =$ Quantile de la distribution normale standard

  L'idée ici est de traiter l'ensemble existant d'évaluations d'utilisateurs comme un échantillonnage statistique d'un ensemble hypothétique d'évaluations d'utilisateurs de tous les utilisateurs, puis d'utiliser ce score. En d'autres termes, que penserait la communauté d'utilisateurs du vote positif pour un produit avec un niveau de confiance de 95 % étant donné que nous disposons d'une évaluation existante pour ce produit avec un échantillon (sous-ensemble de l'ensemble de la communauté) d'évaluations d'utilisateurs.[19]

Ainsi donc le score ne se base pas que sur la moyenne de votes, mais aussi une proportion équitable par rapport aux votes précédents et au nombre de votants.

Dans notre application, l'implémentation de cet algorithme a été menée comme le décrit la capture suivante.

```
1 reference
40  private static double CalculateWilsonScore(double averageRating, int numberOfRatings)
41  {
42      double phat = averageRating / 5.0; // Convert rating to a proportion
43      double n = numberOfRatings;
44      double score = (phat + Z * Z / (2 * n) - Z * Math.Sqrt((phat * (1 - phat) + Z * Z / (4 * n)) / n)) / (1 + Z * Z / n);
45      return score;
46  }
47
```

Figure 2.5: Implémentation de l'algorithme Wilson Score Interval

ii. Médecin

Les médecins sont les deuxièmes acteurs intervenants dans ce package. Leurs cas d'utilisations far peuvent s'énumérer comme suit :

- **Procéder à une consultation:**

  Le processus de validation et de consultation suit la même logique établie pour les patients, une fois le patient ayant fait la requête, le médecin la valide et débute la consultation. Une notification est envoyée au patient une fois la requête validée.

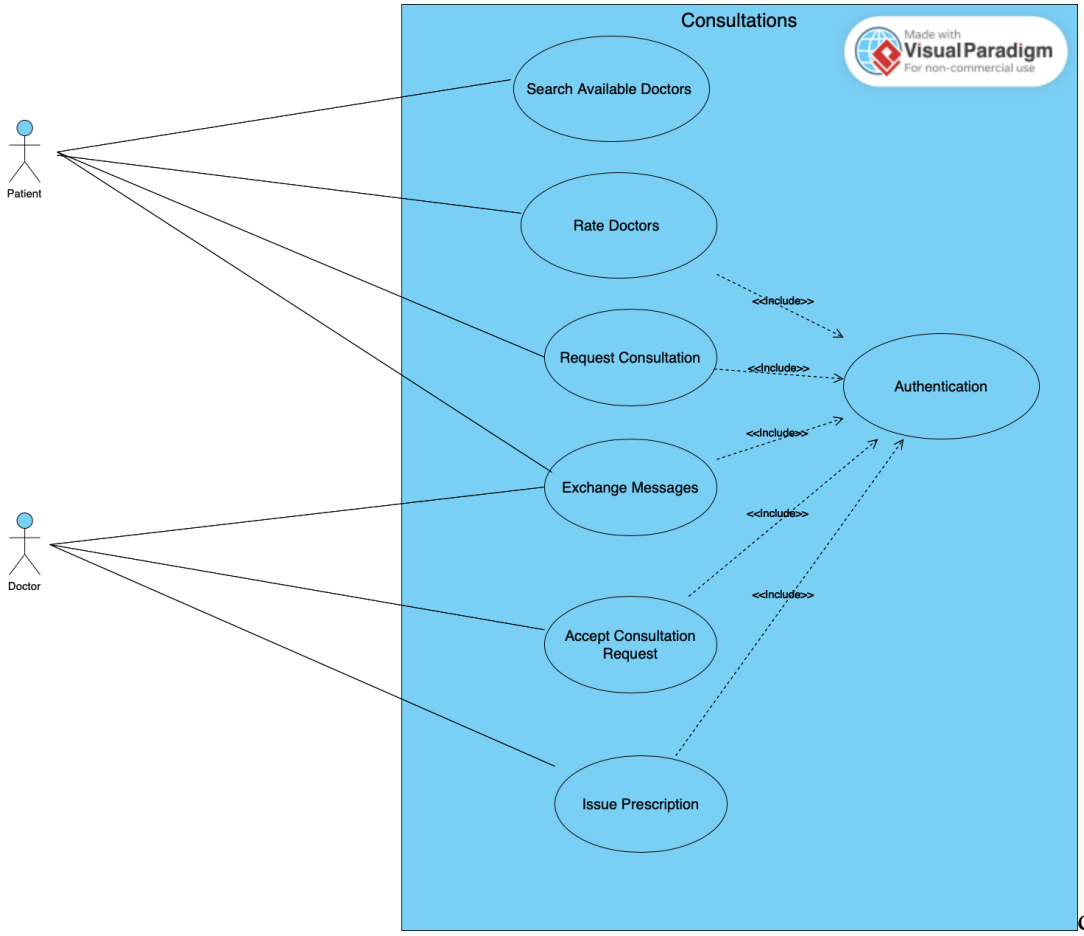
- **Délivrer une ordonnance médicale:**

  Le but d'une consultation c'est qu'au terme de celle-ci le patient ait une meilleure connaissance du mal qui le ronge, et qu'une prescription soit délivrée pour l'orienter vers un traitement approprié. Lorsque le chat permet d'éduquer le patient sur son mal, le médecin peut élaborer une ordonnance signée qui sera ensuite téléchargée sous format PDF par le patient. Il pourra donc se diriger vers son pharmacien pour se procurer les traitements mentionnés.

Figure 2.6: Diagramme des cas d'utilisations pour la gestion des consultations

2.2.1.3. Package de gestion du profil des médecins

Les médecins sur la plateforme ont un ensemble de fonctionnalités qui leurs sont exclusives, liées à la gestion de leur profil. La plateforme est développée de façon à laisser les utilisateurs décider



du médecin à consulter en fonction de l'impression qu'ils ont du médecin sur la plateforme. Cette impression se définit par le classement du médecin (Page 25). La figure suivante décrit les différentes fonctionnalités de ce package.

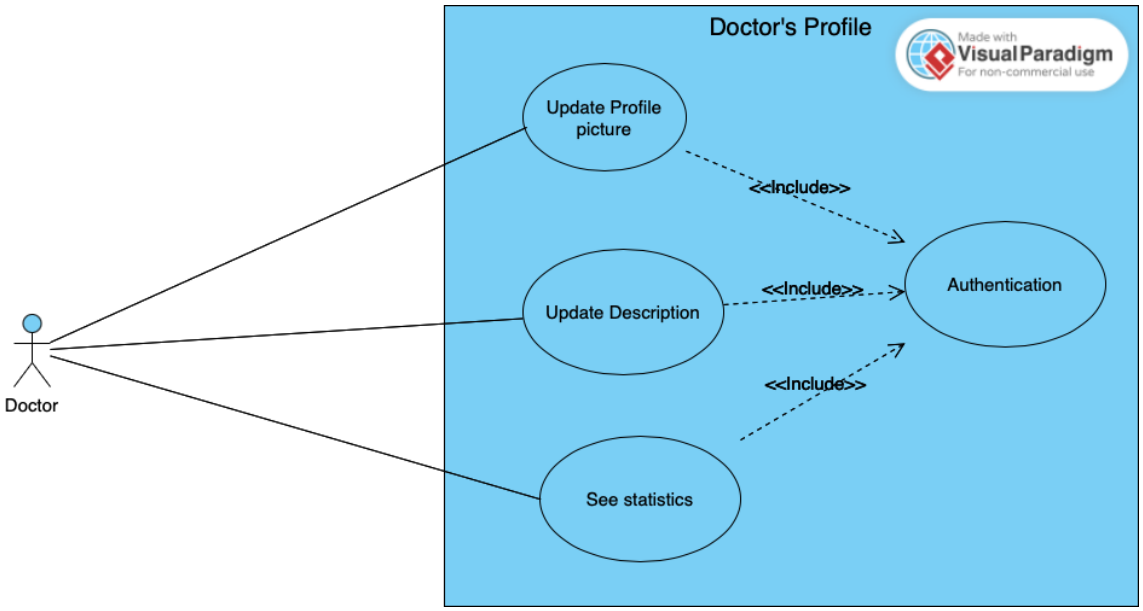


Figure 2.7: Diagramme de cas d'utilisations pour le package du profil des médecins

2.2.1.4. Package d'administration

Pour s'assurer du bon fonctionnement de la plateforme, un administrateur est chargé de manager toutes les configurations de l'application. Il pourra donc effectuer les différents cas d'utilisations illustrés ci-dessous.

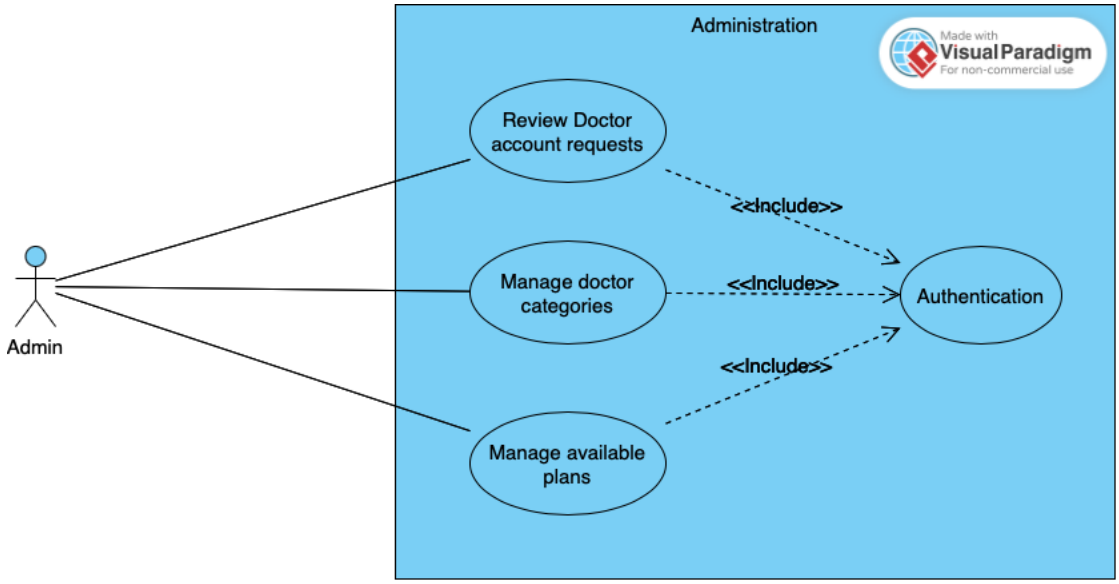


Figure 2.8: Diagramme des cas d'utilisation pour l'administration

L'analyse des cas d'utilisation est une technique essentielle pour le développement de logiciels et d'autres systèmes complexes, garantissant que le système répond aux besoins des utilisateurs et est facile à utiliser.

2.2.2. Analyse du diagramme de classes

Les diagrammes de classes sont essentiels pour modéliser la structure et le comportement d'un système logiciel, permettant de visualiser les classes, identifier leurs responsabilités et communiquer la conception du système aux parties prenantes.

Le diagramme qui ressort de cette analyse est le suivant :

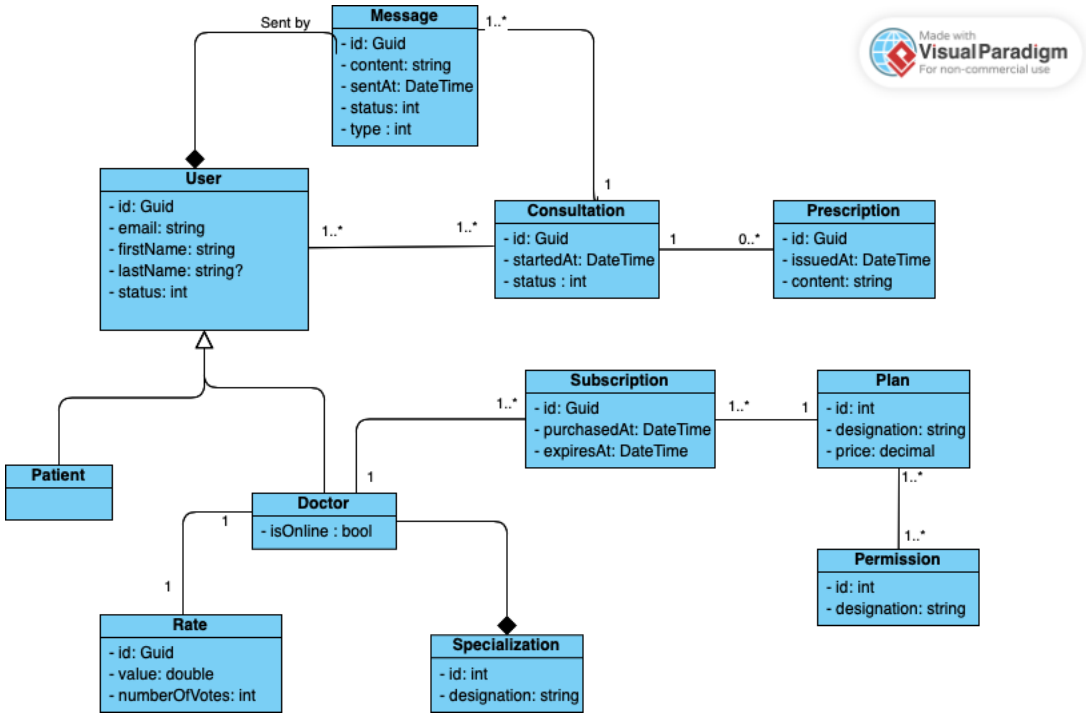


Figure 2.9: Diagramme des classes

Le diagramme de classes ci-dessus illustre les différentes entités ainsi que leurs relations et comment celles-ci interagissent entre elles.

2.3. Architecture de la Solution

2.3.1. Domain Driven Design (DDD)

Le Domain-Driven Design (DDD) est une approche agile de création de logiciels qui met l'accent sur le domaine métier. Il met fortement l'accent sur la communication et l'interaction avec des experts du domaine métier qui peuvent expliquer aux développeurs comment fonctionne le système du monde réel. Par exemple, si vous créez un système qui gère les transactions boursières, votre expert en domaine peut être un courtier en valeurs mobilières expérimenté. DDD est conçu pour résoudre des problèmes commerciaux importants et complexes et n'est souvent pas approprié pour des applications plus petites et plus simples, car l'investissement dans la compréhension et la modélisation du domaine n'en vaut pas la peine.

L'approche DDD comprend des principes à respecter, ces principes se déclinent sous plusieurs aspects. Celui sur lequel nous avons porté un accent particulier est le fait d'intégrer tout au long du développement un *"Ubiquitous language"* en français un langage omniprésent de façon à faire correspondre exactement les termes employés dans le code source aux termes du jargon technique du système que l'on souhaite modéliser.

Ainsi, nous avons procédé en commençant par un découpage des Agrégats.

Un Agrégat représente un groupe d'objets qui doivent être conservés en tant qu'unité. Ces objets entretiennent une forte cohésion entre eux, et une faible dépendance aux objets externes à cet agrégat.

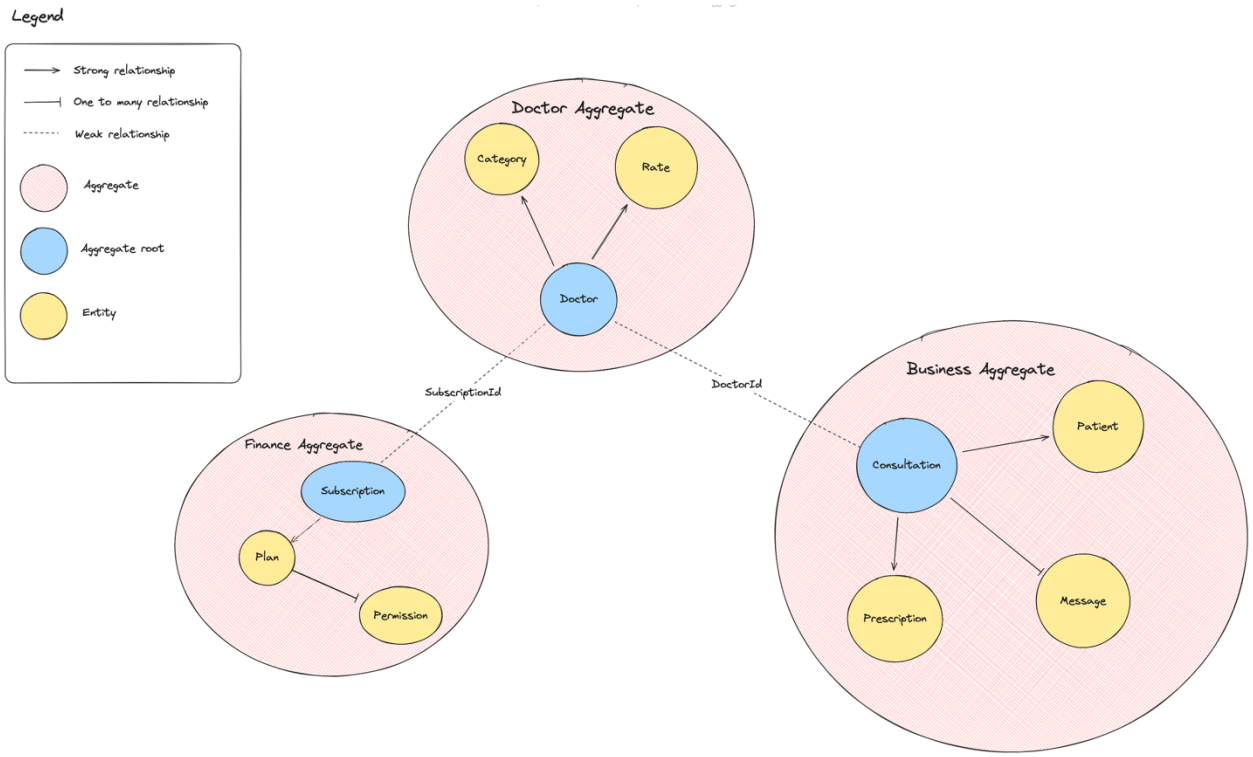


Figure 2.10: Structure du Domain

Avec un diagramme de classes bien établi, il revient plus facile de définir les différentes entités et de mettre en évidence leurs dépendances mutuelles. Ainsi découper les agrégats en fonction des dépendances observées revient également plus évident. En règle générale, les agrégats ne sont pas liés les uns aux autres directement mais plutôt par le biais d'identifiants uniques et immuables, vu que ces différents agrégats ne changent pas pour les mêmes raisons. De plus, le monde externe n'interagit avec un agrégat que par deux façons, soit en accédant aux méthodes exposées par l'agrégat racine, soit par le biais d'évènements.

Notre domaine se divise ainsi en trois (03) agrégats :

Doctor Aggregate

Cet agrégat regroupe la logique métier liée directement aux médecins et à leur profil sur la plateforme. On aura comme racine la classe Doctor, et comme entités liées une catégorie associée et une note.

En se penchant un peu plus sur la classe Doctor qui est la classe principale de cet agrégat. Elle encapsule les fonctionnalités s'appliquant aux médecins ainsi que les relations avec les autres entités.

Finance Aggregate

Cet agrégat se focalise sur les fonctionnalités liées aux finances et aux paiements. Il regroupe donc les règles liées aux différentes offres disponibles et un historique des souscriptions effectuées par les médecins. De même, chaque plan ayant un ensemble de permissions associées, ces règles sont également encapsulées dans cet agrégat.

La classe Subscription sera ainsi définie dans notre code comme suit :

Business Aggregate

Cet agrégat est le plus complexe, vu qu'il encapsule le plus de fonctionnalités, pour la plupart liées aux consultations et aux patients, ainsi que les messages échangés lors des consultations.

2.3.2. Clean Architecture (CA)

La Clean Architecture place la logique métier et le modèle d'application au centre de l'application. Au lieu que la logique métier dépende de l'accès aux données ou d'autres problèmes d'infrastructure, cette dépendance est inversée : les détails de l'infrastructure et de la mise en œuvre dépendent de l'Application Core. Cette fonctionnalité est obtenue en définissant des abstractions, ou interfaces, dans l'Application Core, qui sont ensuite implémentées par des types définis dans la couche Infrastructure. Une manière courante de visualiser cette architecture consiste à utiliser une série de cercles concentriques, semblables à un oignon [20].

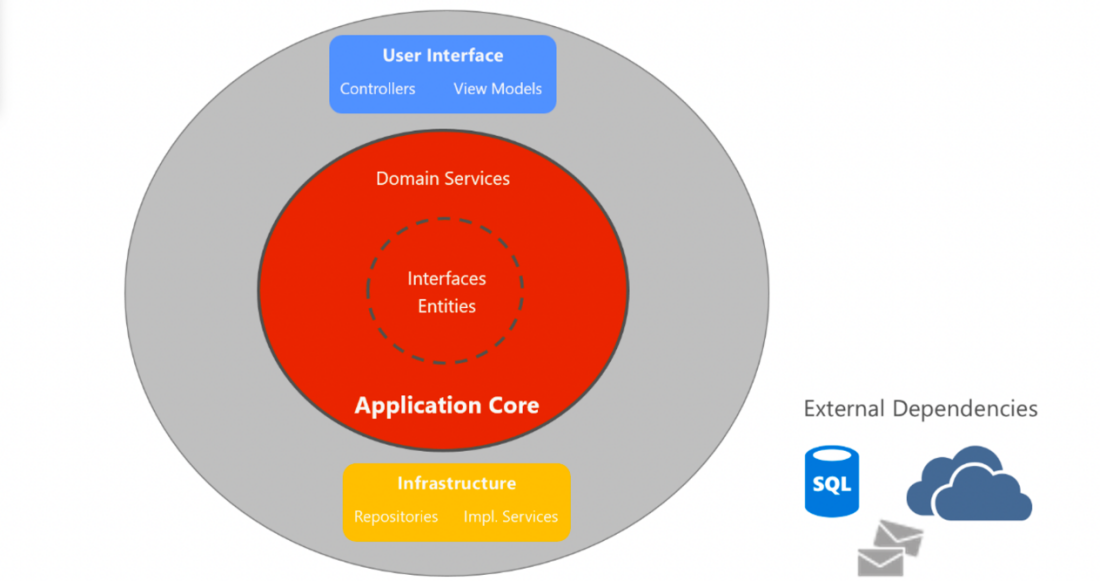


Figure 2.11: Clean Architecture; onion view [20, p. 32]

Dans ce diagramme, les dépendances se dirigent vers le cercle le plus intérieur. L'Application Core tire son nom de sa position au cœur de ce diagramme. Et vous pouvez voir sur le diagramme que l'Application Core n'a aucune dépendance sur d'autres couches d'application. Les entités et interfaces de l'application sont au centre même. Juste à l'extérieur, mais toujours dans l'Application Core, se trouvent les services de domaine, qui implémentent généralement les interfaces définies dans le cercle intérieur. En dehors du cœur d'application, les couches d'interface utilisateur et d'infrastructure dépendent du cœur d'application, mais pas les unes des autres (nécessairement).

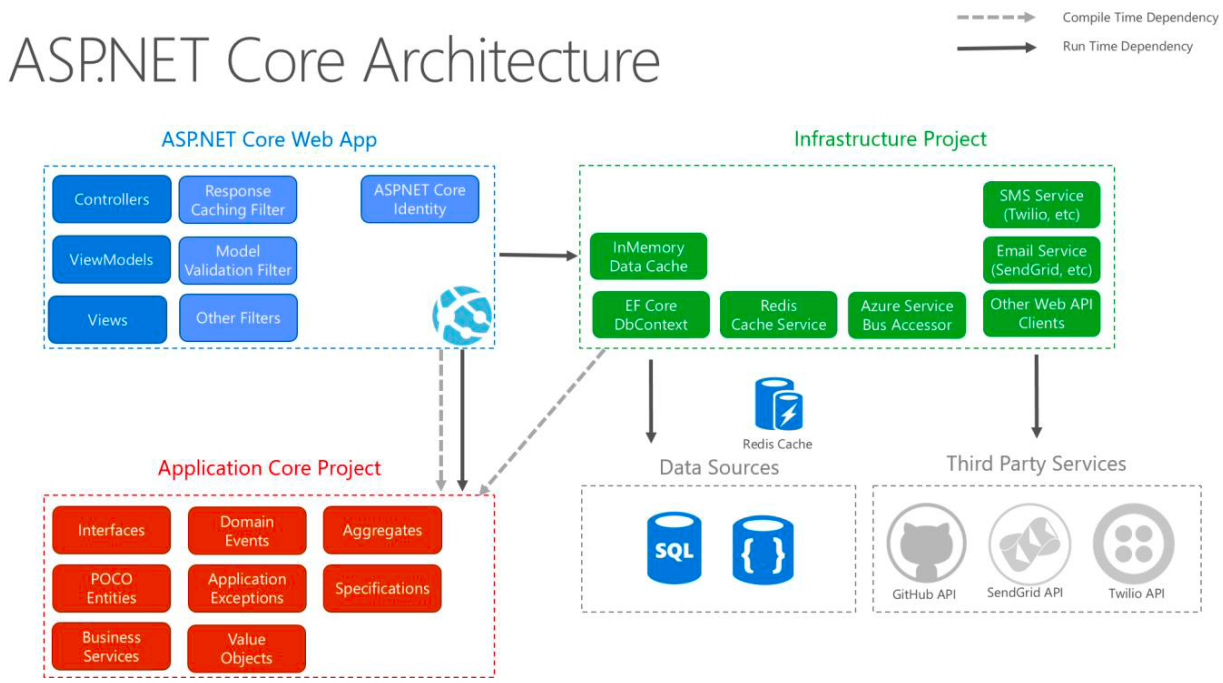


Figure 2.12: ASP.NET Core architecture diagram following Clean Architecture [20, p. 34]

Notez que les flèches pleines représentent les dépendances au moment de la compilation, tandis que la flèche en pointillés représente une dépendance au moment de l'exécution uniquement. Avec l'architecture propre, la couche UI fonctionne avec les interfaces définies dans l'Application Core au moment de la compilation et, idéalement, ne devrait pas connaître les types d'implémentation définis dans la couche Infrastructure. Au moment de l'exécution, cependant, ces types d'implémentation sont requis pour que l'application s'exécute. Ils doivent donc être présents et connectés aux interfaces Application Core via l'injection de dépendances.

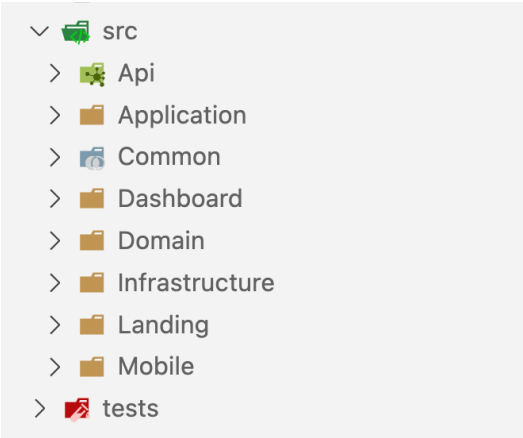


Figure 2.13: Organisation du code source

2.4. Méthode de Développement

2.4.1. Méthodologie utilisée

Dans le développement logiciel, de nombreuses méthodes de gestion de projet sont disponibles, chacune offrant des avantages spécifiques en fonction des besoins et des contraintes du projet. Parmi les méthodologies les plus courantes, on trouve Agile, Scrum, Kanban, Lean et Waterfall. Chacune de ces approches présente des caractéristiques distinctes en termes de flexibilité, de structure et de gestion du flux de travail. Pour notre projet, nous avons opté pour Scrumban, une méthodologie hybride qui combine les meilleures pratiques de Scrum et de Kanban. Ce choix a été motivé par notre besoin de bénéficier à la fois de la rigueur et de la cadence des sprints de Scrum et de la flexibilité et de la visualisation continue des tâches de Kanban. En adoptant Scrumban, nous avons pu optimiser notre gestion de projet, favoriser une meilleure adaptation aux changements et garantir une livraison régulière et de haute qualité de notre plateforme de consultation en ligne.

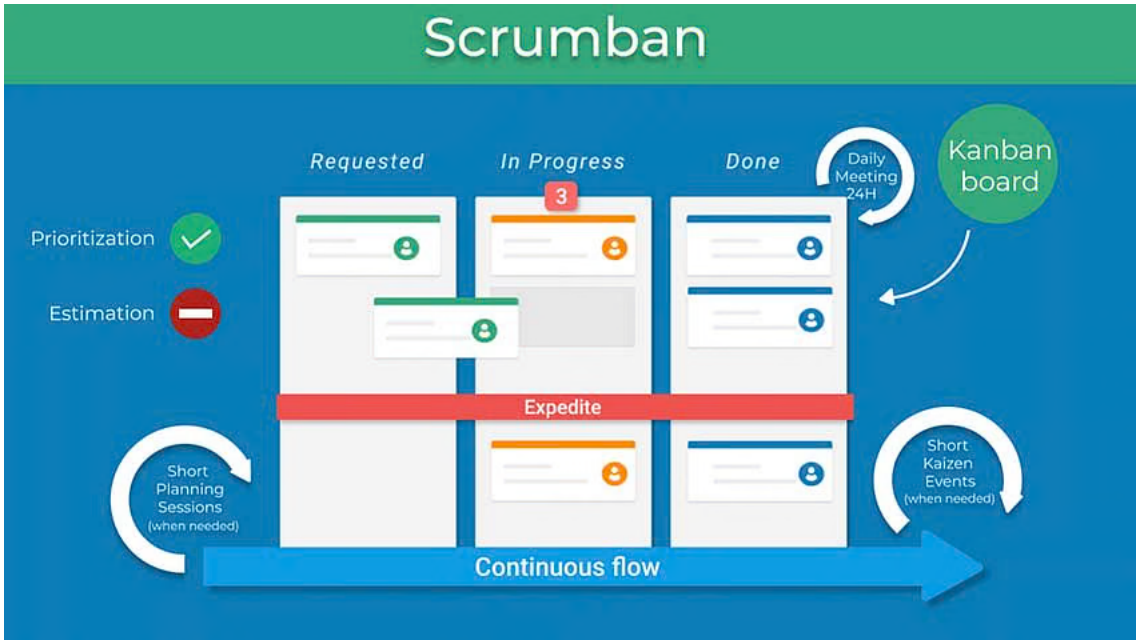


Figure 2.14: Cycle de la méthodologie Scrumban [21]

Scrumban est une méthodologie hybride de gestion de projet qui combine les principes de Scrum et de Kanban pour offrir une approche flexible et efficace de la gestion des processus de développement [11]. Scrumban utilise la structure stable de Scrum de sprints, standups et

rétrospectives. Ensuite, il ajoute le flux de travail visuel de Kanban et les limitations des travaux en cours.

2.4.1.1. Piliers de la méthodologie Scrumban

La méthodologie Scrumban est axée sur plusieurs aspects principaux qui constituent ses piliers et sont incontournables dans cette pratique. On pourra citer entre autres :

- **Sprints :**

Les sprints sont des périodes fixes, généralement de deux à quatre semaines, durant lesquelles une quantité de travail définie doit être accomplie. Les sprints permettent de structurer le travail en cycles itératifs, favorisant une cadence régulière et prévisible.

- **Tableaux Kanban :**

Les tableaux Kanban sont utilisés pour visualiser le flux de travail en représentant les tâches à travers différentes colonnes (à faire, en cours, terminé). Cette visualisation aide à gérer les tâches de manière efficace.

- **Épic :**

Un épic est une grande unité de travail qui peut être décomposée en plusieurs user stories plus petites et gérables. Les épics aident à organiser et à hiérarchiser les grands objectifs et fonctionnalités du projet.

- **User Stories :**

Les user stories sont des descriptions courtes et simples d'une fonctionnalité du point de vue de l'utilisateur final. Elles servent de base pour définir les tâches à réaliser et assurer que le développement répond aux besoins des utilisateurs.

- **Limitation du Travail en Cours (WIP) :**

La mise en place de limites WIP (Work In Progress) pour chaque colonne du tableau Kanban permet de prévenir la surcharge de travail et de s'assurer que les tâches en cours sont terminées avant d'en commencer de nouvelles.

- **Réunions de Planification :**

Les réunions de planification sont tenues au début de chaque sprint pour définir les objectifs et le travail à accomplir. Cela permet d'aligner l'équipe sur les priorités et de planifier efficacement les tâches à réaliser.

2.4.1.2. Avantages de la méthodologie Scrumban

Cette méthodologie a plusieurs avantages sur les autres, vu qu'elle combine deux des plus populaires utilisées dans l'industrie. On pourra citer :

- **Flexibilité et Adaptabilité :**

Scrumban permet une grande flexibilité grâce à l'absence de sprints rigides. Les équipes peuvent ajuster leur charge de travail en continu, ce qui facilite l'adaptation aux changements de priorités ou aux nouvelles exigences du projet.

- **Amélioration Continue :**

Grâce à l'approche Kanban, Scrumban encourage l'amélioration continue des processus. Les équipes sont constamment à la recherche de moyens pour optimiser leur flux de travail et éliminer les inefficacités.

- **Visualisation du Travail :**

L'utilisation de tableaux Kanban pour visualiser les tâches en cours, à faire et terminées améliore la transparence et la clarté du processus de travail. Cela permet à tous les membres de l'équipe de voir l'état d'avancement du projet en un coup d'œil.

- **Réduction des Goulets d'Étranglement :**

En visualisant le flux de travail, les équipes peuvent identifier et traiter rapidement les goulets d'étranglement, ce qui améliore l'efficacité et la productivité.

- **Cadence Prévisible :**

La combinaison des sprints de Scrum avec les principes de Kanban permet de maintenir une cadence de travail prévisible tout en offrant la flexibilité nécessaire pour ajuster les priorités en cours de route.

- **Encouragement de la Collaboration :**

Scrumban favorise une communication ouverte et régulière entre les membres de l'équipe, ce qui renforce la collaboration et assure que tout le monde est aligné sur les objectifs du projet.

Ces différents points ont influencé la décision portant sur le choix de la méthodologie à appliquer.

2.4.2. Intégration avec le logiciel Jira

Jira est un produit propriétaire développé par Atlassian qui permet le suivi des bogues, le suivi des problèmes et la gestion de projets agiles. Jira est utilisé par un grand nombre de clients et d'utilisateurs dans le monde entier pour la gestion des projets, du temps, des exigences, des tâches, des bogues, des changements, du code, des tests, des versions, des sprints [22]. Les équipes peuvent créer et gérer des épics, des user stories et des tâches, tout en utilisant des tableaux Kanban pour visualiser le flux de travail. Jira permet également de définir des sprints, d'assigner des limites de travail en cours (WIP), et de suivre les progrès via des rapports et des tableaux de bord personnalisables. Pour notre projet, l'on a procédé par les différentes étapes suivantes.

2.4.2.1. Définition des taches et user stories

Avant d'écrire la première ligne de code, l'on doit avoir un ordre clair des modules que l'on implémentera ainsi que les user stories associées. L'on a donc découpé les tâches en 4 sprints, qui sont les suivants :

Figure 2.15: Capture d'écran du Backlog sur Jira

La flexibilité de la méthodologie Scrumban permet surtout l'ajout et la modification des sprints initiaux, dans certaines circonstances bien spécifiques.

En déroulant ces sprints, on peut facilement voir les user stories contenues ainsi que leurs états d'avancement. Cela permet d'avoir rapidement une vue globale sur la progression globale du projet.

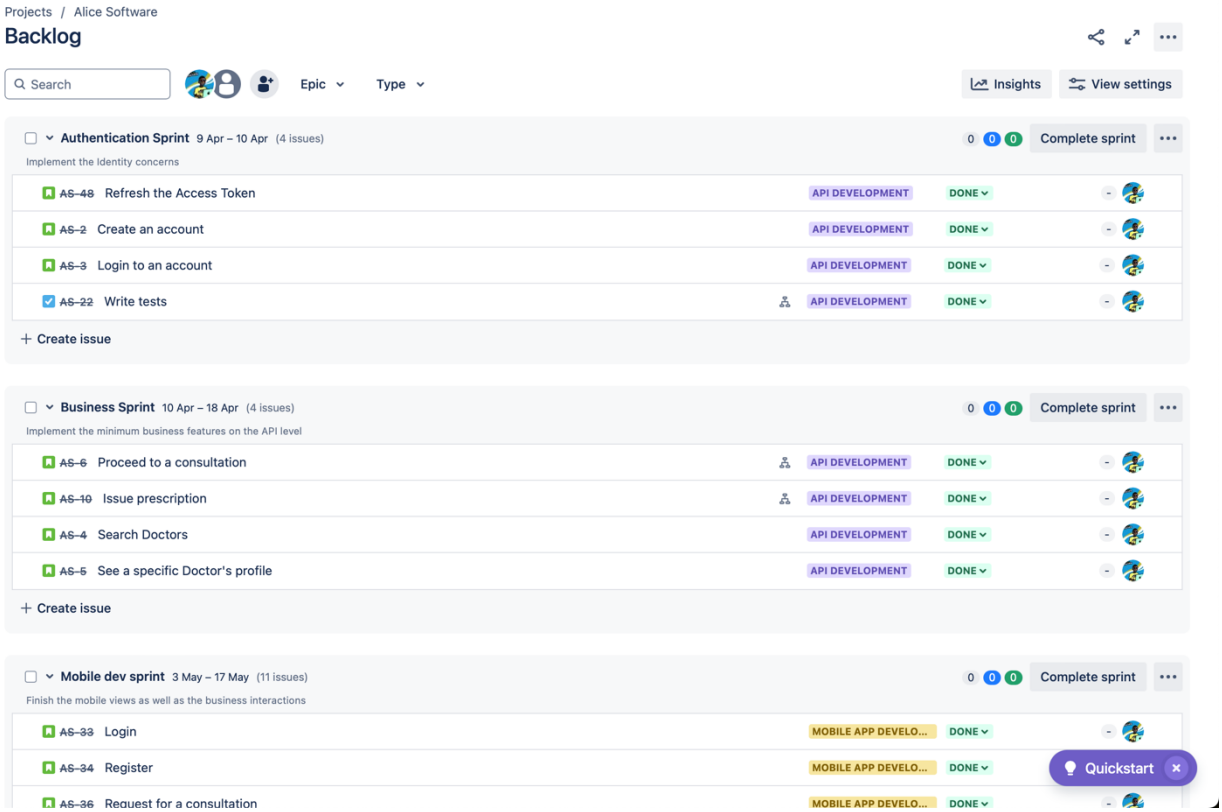


Figure 2.16: Capture d'écran des sprints sur Jira

On remarque facilement que l'on a des épiques différents, ils permettent notamment de regrouper les tâches en relation directe les unes aux autres.

Le backlog contient les tâches qui ne sont pas encore planifiées, ni assignées à un sprint. Elles ont déjà néanmoins un épic associé.

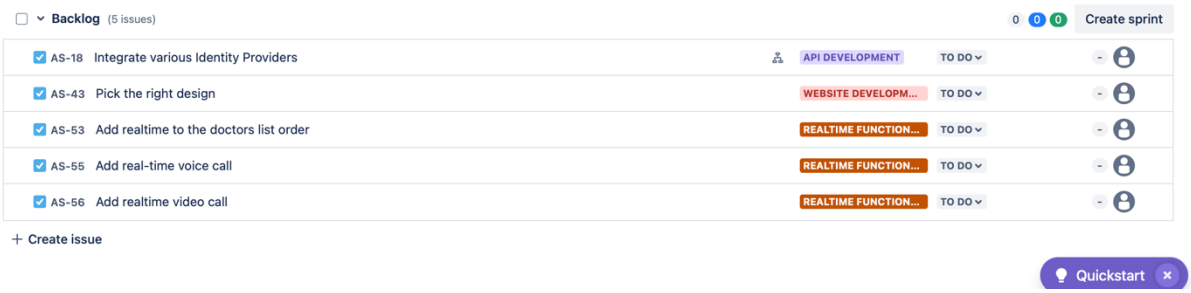


Figure 2.17: Capture d'écran du backlog non planifié sur Jira

2.4.2.2. Exploration du tableau Kanban

Un tableau Kanban est un outil visuel utilisé dans la gestion de projet pour représenter le flux de travail et suivre l'avancement des tâches. Il est généralement divisé en colonnes, chacune représentant une étape spécifique du processus de production. Les tâches ou les éléments de travail sont représentés par des cartes qui sont déplacées de colonne en colonne à mesure qu'elles progressent à travers les différentes étapes.

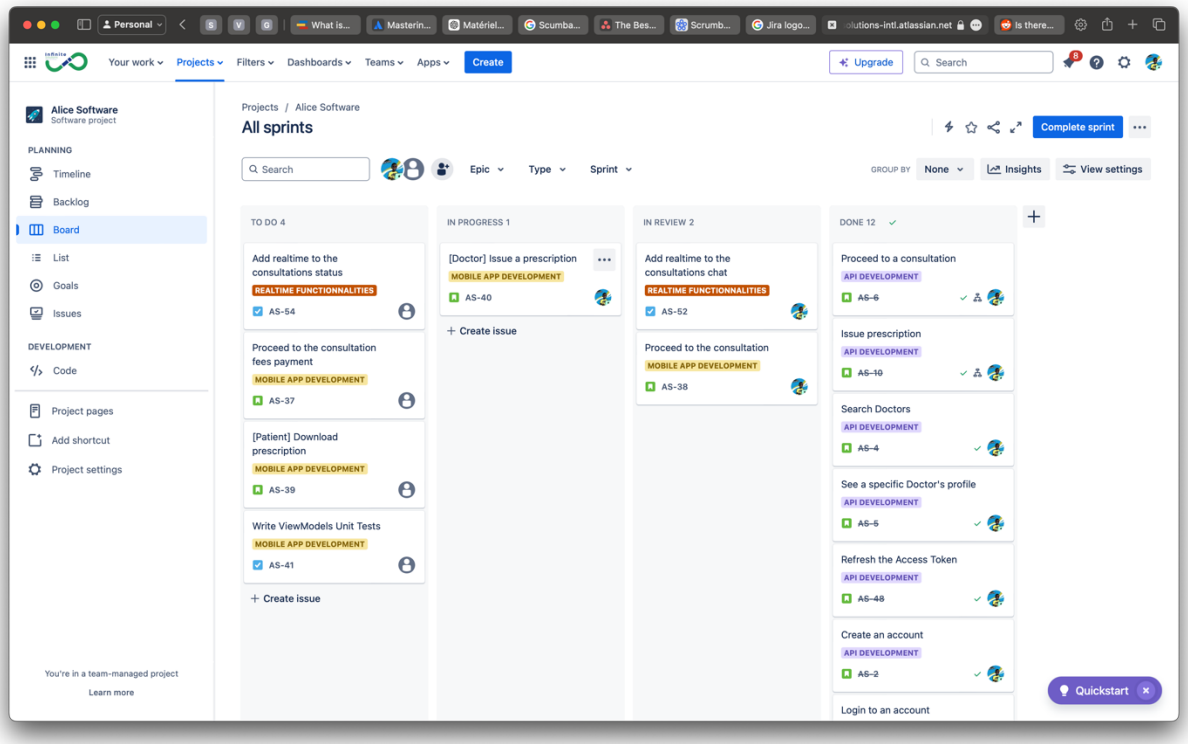


Figure 2.18: Capture d'écran du tableau Kanban sur Jira

Ce tableau est appelé à être modifié à chaque avancée du processus. L'on note également des sous-tâches associées à chaque tâche au besoin, et dans ce cas l'état de complétion d'une tâche dépend de celui de ses sous-tâches associées.

On prend l'exemple de la tâche globale *'Proceed to a consultation'* ci-dessous, qui se subdivise en quatre sous-tâches :

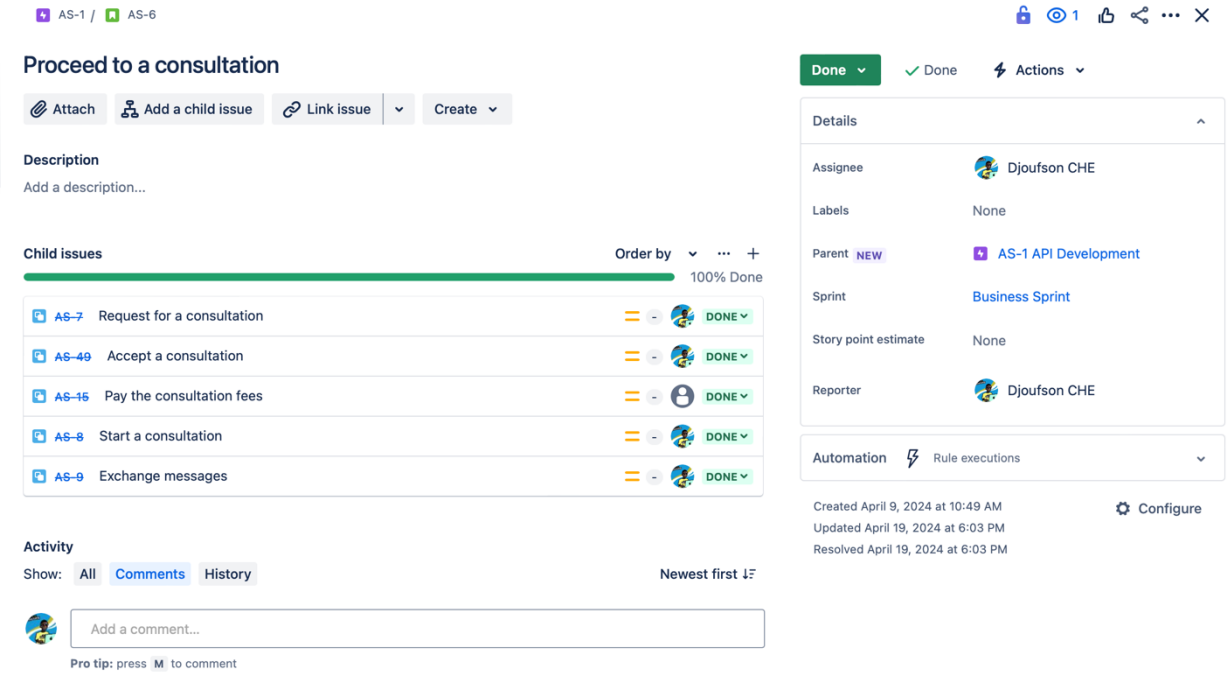


Figure 2.19: Capture d'écran des sous-taches sur Jira

2.5. Intégration Cloud avec Microsoft Azure

2.5.1. Présentation de l'infrastructure déployée sur Microsoft Azure

Microsoft Azure, ou simplement Azure, est la plateforme de cloud computing développée par Microsoft. Elle offre la gestion, l'accès et le développement d'applications et de services aux particuliers, aux entreprises et aux gouvernements via son infrastructure mondiale.

Dans le cadre de notre application, Azure a été utilisé à plusieurs niveaux différents, et son intégration débute par une claire compréhension des besoins et des composantes qui seront mis en jeu.

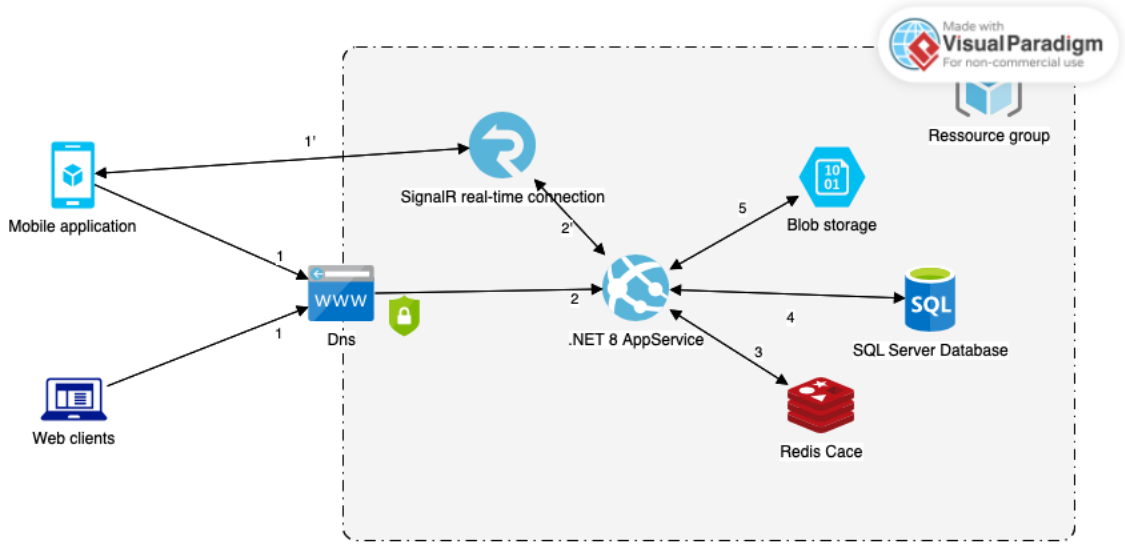


Figure 2.20: Architecture de déploiement sur Azure

L'infrastructure qui a été mise en place comprend les différents composants suivants :

Clients

- Application mobile : l'application est utilisée par les patients, pour interagir avec la plateforme.
- Clients Web : les tableaux de bord administrateur et médecin, qui affichent des statistiques et des informations pertinentes pour l'utilisateur, et constitue l'interface principale des médecins.

Prestations des services

- SignalR : une instance SignalR sera utilisée pour activer les fonctionnalités en temps réel
- AppService : nous aurons un seul AppService monolithique qui hébergera la logique de base
- Redis cache : un service de cache sera mis à disposition afin d'accélérer les requêtes récurrentes, et d'alléger la charge côté serveur pour les requêtes similaires

- SQL Server database : la base de données que nous utiliserons est une base de données SQL Server
- Blob Storage : pour stocker des données non structurées telles que des fichiers multimédias et des documents.

Flux de connexions

Les différentes connexions et communications dans le système sont représentées par des flèches et des indices associés.

1. Appels HTTP de divers clients

2. Les appels sont transmis au service interne

3. Si nous devons interroger certaines données, nous vérifions d'abord le cache

4. Sinon on interroge la base de données SQL Server

5. Pour toutes les données non structurées telles que les images/fichiers, nous interrogeons le stockage blob

1'. Connexion socket Web pour une interactivité en temps réel à l'aide de SignalR

2'. Chaque événement est traité via le service interne

Ci-dessous une capture de l'environnement Azure mis en place :

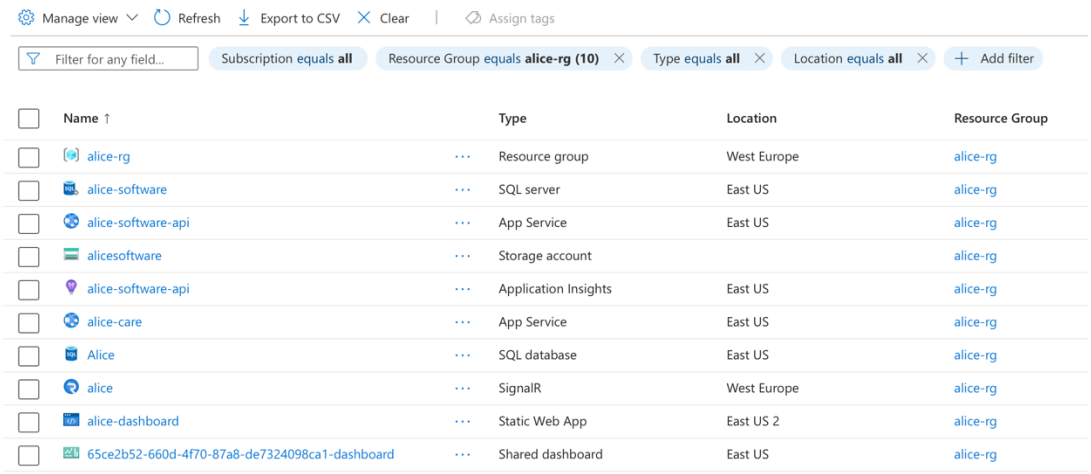


Figure 2.21: Vue d'ensemble des services déployés

2.5.2. Développement et Déploiement en continu

Le DevOps énoncé plus haut implique certaines pratiques telles que l'automatisation des processus. Ces processus incluent les tests, la génération des artéfacts, la publication de ces artéfacts et le déploiement en ligne des modifications. Cette automatisation a été effectuée dans le cadre de ce projet grâce à un outil d'automatisation appelé ***GitHub Actions.***

GitHub Actions est un outil intégré de CI/CD proposé par GitHub, conçu pour automatiser les workflows et les processus de développement logiciel directement depuis les dépôts GitHub. Les GitHub Actions permettent aux développeurs d'automatiser des tâches telles que les tests unitaires, la compilation de code, le déploiement d'applications, et bien plus encore, en réponse à divers événements déclenchés sur GitHub, comme des pull requests, des push de code, des créations de tags, etc.

L'implémentation d'une GitHub Action se fait en écrivant un fichier de déploiement .yml que l'on place dans un répertoire spécifique de notre code base.

Figure 2.22: Différents fichiers de déploiement

Dans notre projet, l'on a défini plusieurs Workflows en fonction de nos besoins spécifiques. Attardons-nous sur celui qui permet le déploiement de notre service backend, le fichier **build-and-deploy-api.yml**

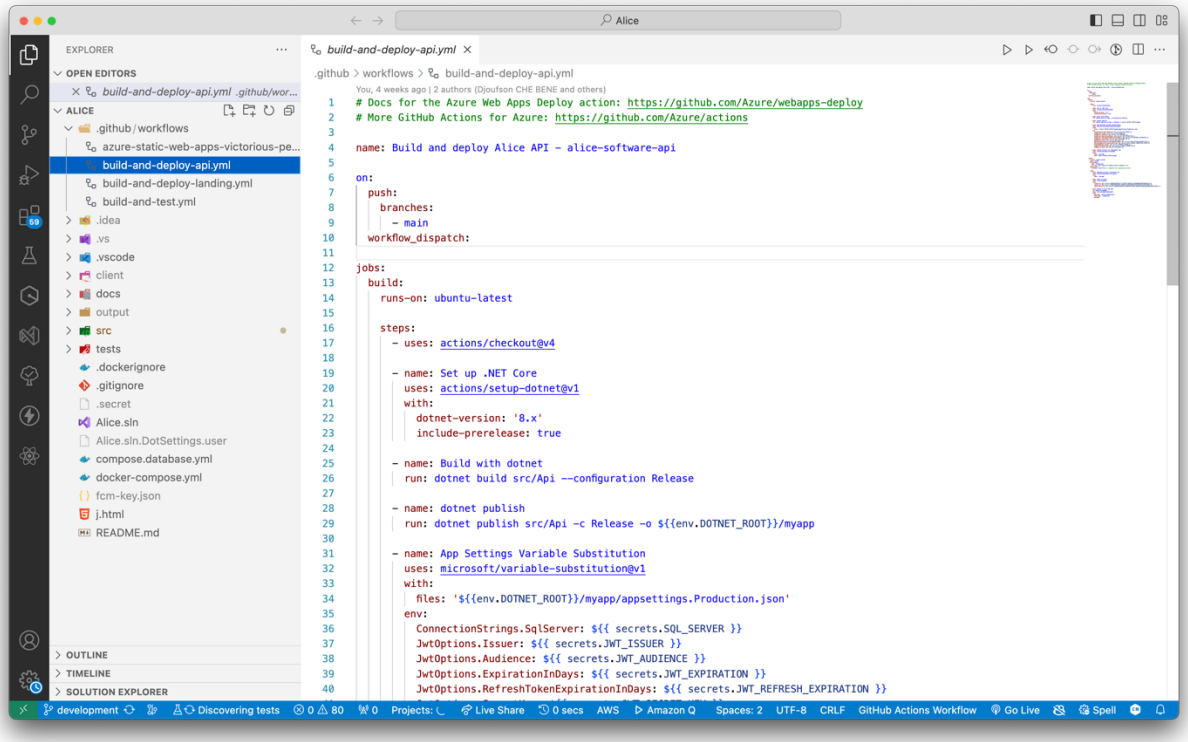


Figure 2.23: Capture d'écran de l'étape de compilation

Il définit un workflow qui s'exécutera à chaque 'push' sur la branche principale. Ceci déclenche un build du projet, et génère les artéfacts.

Ces artéfacts sont ensuite compressés et envoyés au service d'Azure qui va les appliquer et ainsi relancer l'instance de notre application en considérant les nouveaux artéfacts, comme le définit la suite du fichier de déploiement.

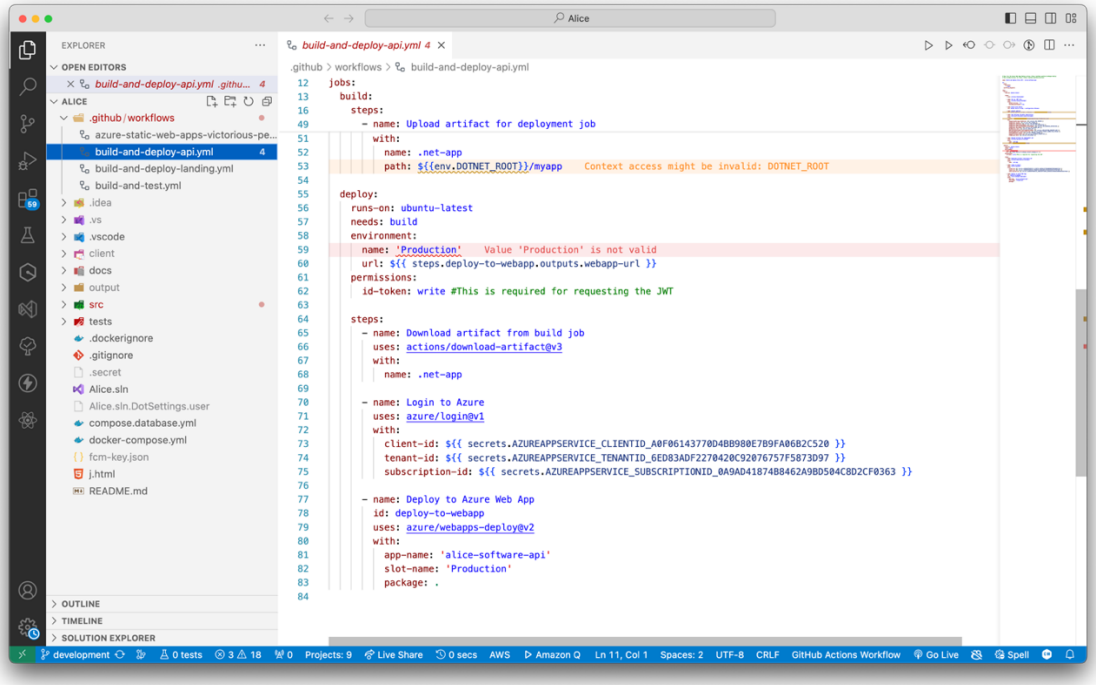


Figure 2.24: Capture d'écran de l'étape de déploiement

Une fois les workflows déclenchés, sur l'interface de GitHub ils se présentent comme suit :

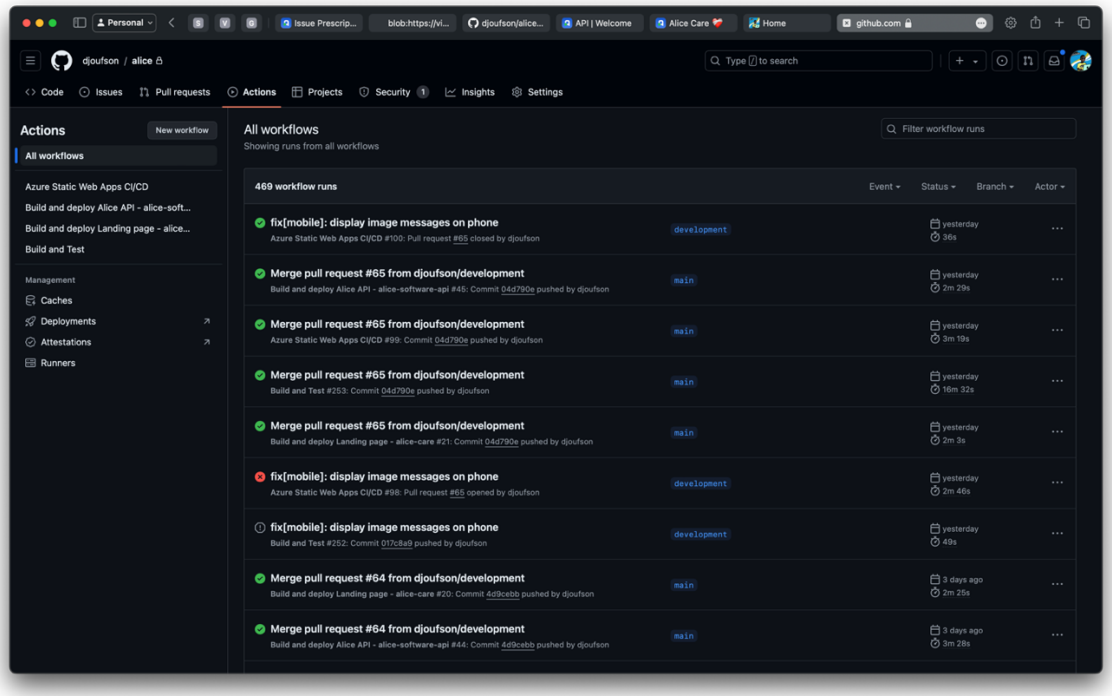


Figure 2.25: Capture d'écran du rendu sur GitHub

2.5.3. Monitoring

Un des avantages principaux d'utiliser le Cloud est le monitoring des ressources. Azure offre une interface pour le monitoring de façon générale. L'on présentera la notion de monitoring avec Azure sur deux aspects principaux, les métriques de nos services et la remontée des bugs.

2.5.3.1. Métriques

L'on a provisionné plusieurs services sur Azure, et chacun de ces services expose des données sur son fonctionnement et son cycle de vie. On a par exemple :

La base de données

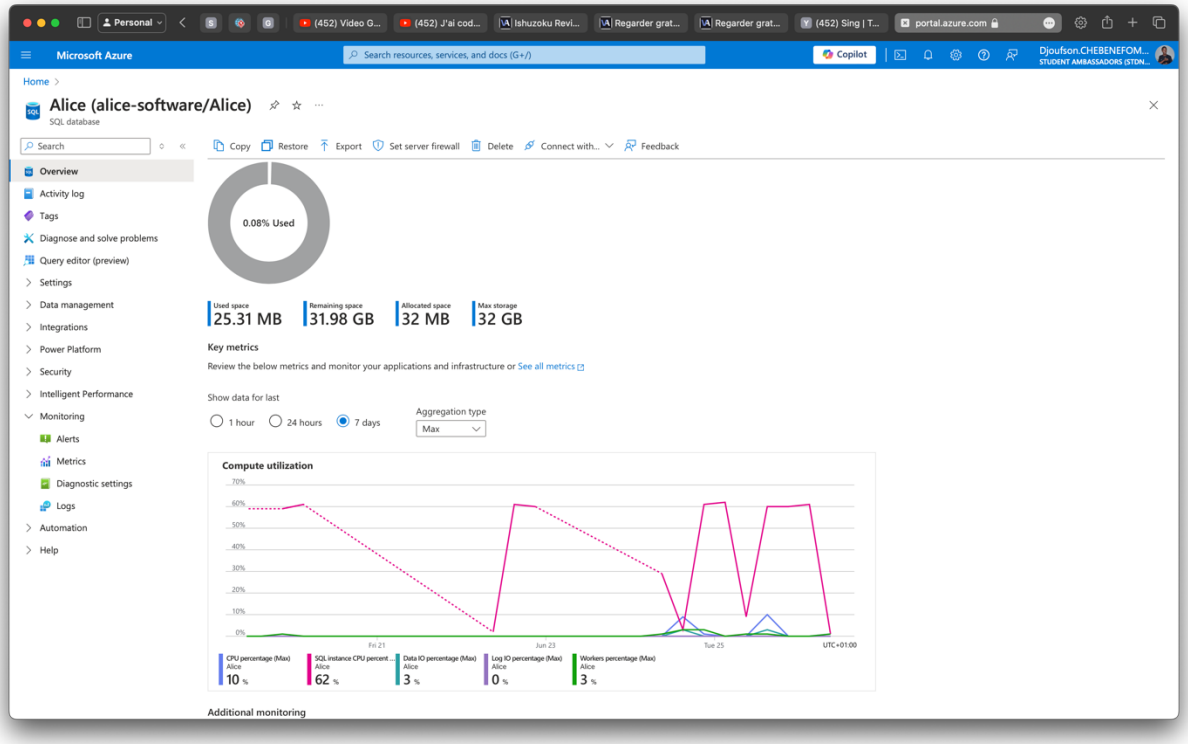


Figure 2.26: Vue d'ensemble de l'état de la base de données

On a une vue globale de l'état de la base de données, et la possibilité de faire des requêtes plus complexes avec une précision , comme sur la figure suivante :

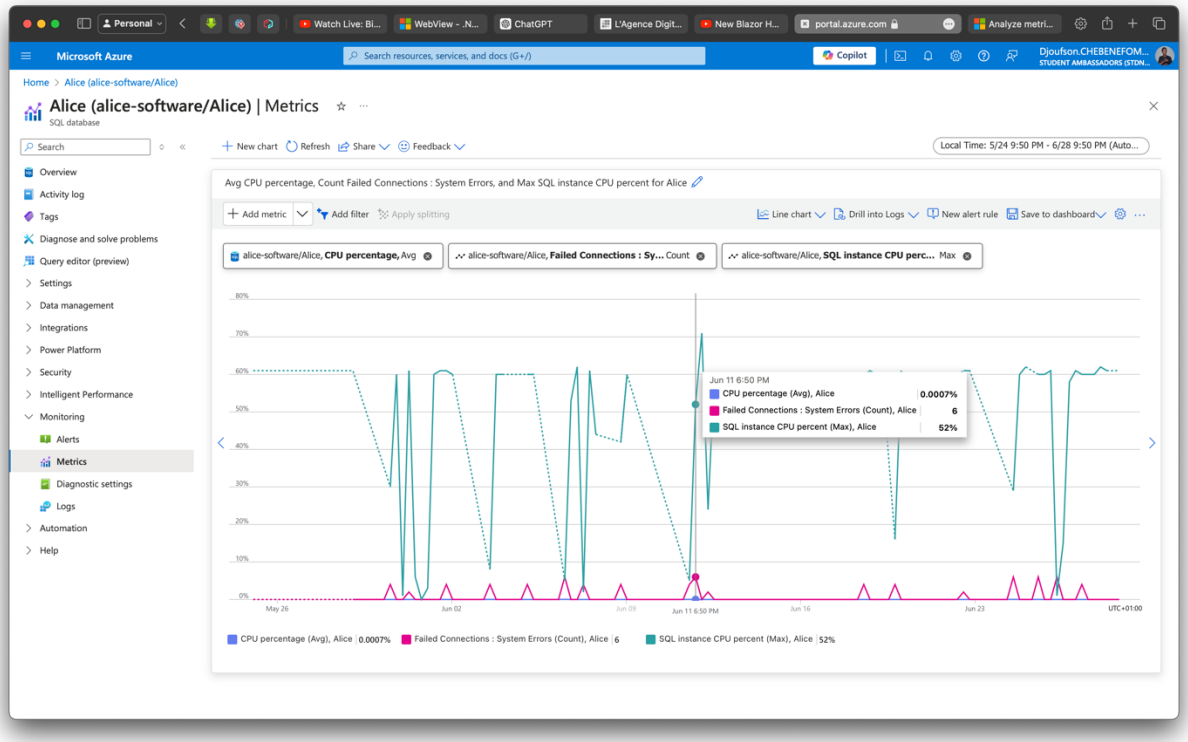


Figure 2.27: Vue détaillée de l'état de la base de données

Il en va de même pour toutes les ressources provisionnées sur Azure. Ces données peuvent être téléchargées en fichier Excel, et exploitées par un Data Scientist. L'idée étant de tirer un meilleur parti des données générées par notre système, d'anticiper les dépenses et coûts sur le long terme, et par la même occasion d'optimiser l'utilisation des ressources.

2.5.3.2. Remontée des bugs

Dans une application, plusieurs facteurs peuvent entrainer un disfonctionnement et entraîner ou non l'arrêt temporaire ou définitif des services. Une fois dans un environnement de production, il devient difficile de garder un œil sur l'état des services et de reporter manuellement chaque erreur lorsqu'elles surviennent.

Pour pallier ce soucis, l'environnement Azure met à notre disposition un service d'Insights, permettant d'avoir en temps réel des informations sur le fonctionnement interne de nos services.

L'accueil nous offre une vue d'ensemble sur les différentes erreurs sur une période donnée

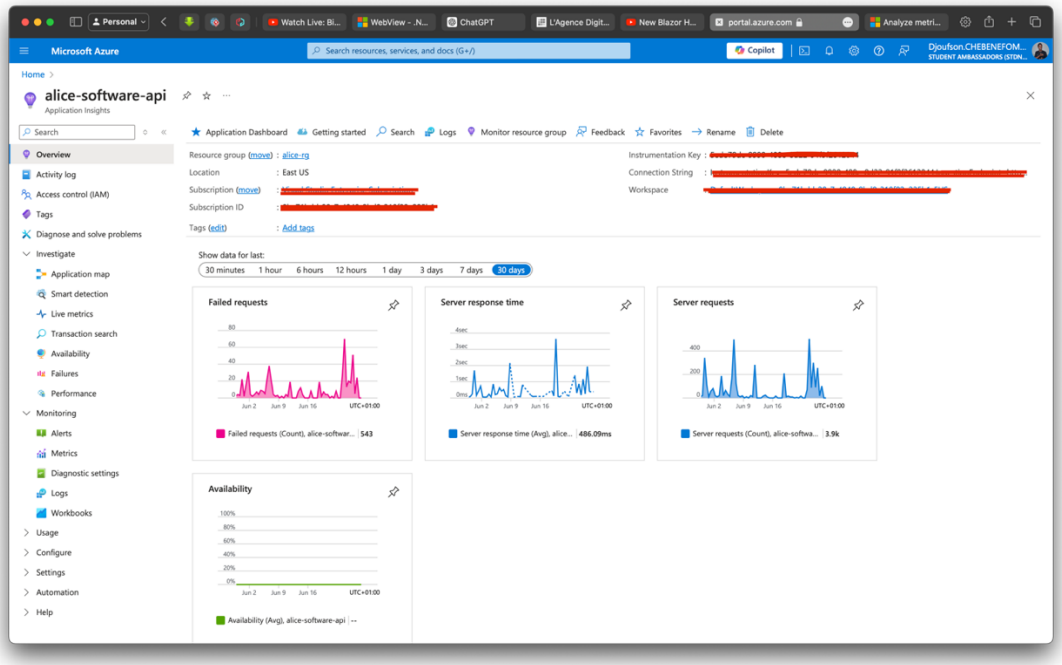


Figure 2.28: Vue d'ensemble des erreurs survenues

Les Insights d'Azure permettent d'effectuer des requêtes complexes servant à interroger sur les problèmes survenus.

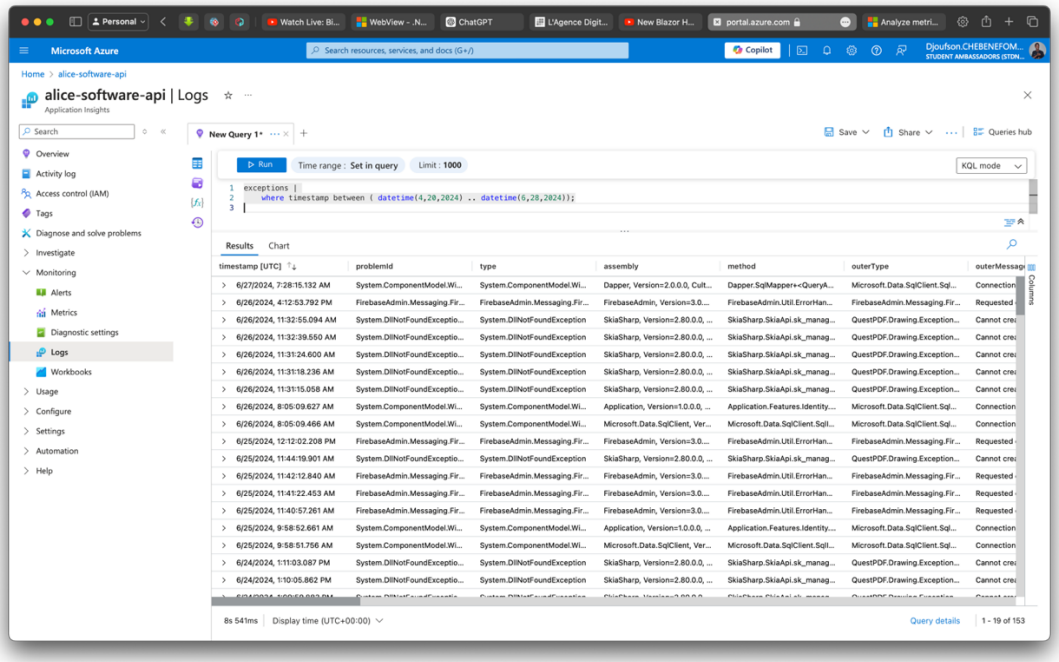


Figure 2.29: Vue détaillée des logs

Ces requêtes sont écrites d'une façon très proche du SQL classique, et offrent beaucoup de flexibilité sur l'exploration des méta données de notre système. Tout ceci contribue à une meilleure expérience de débogage de façon globale dans le système, constituant un élément clé du monitoring en environnement Cloud Native.

Ce deuxième chapitre expose la méthodologie adoptée lors du développement de cette solution ainsi que les choix effectués. Nous avons procédé suivant une méthodologie Scrumban pour le suivi du déroulement du projet, en appliquant les techniques UML pour un développement organisé. Le chapitre qui suit présente les résultats obtenus.

CHAPITRE 3 : RESULTATS ET DISCUSSION

Maintenant que nous connaissons comment la plateforme Alice a été développée, les outils qui ont été utilisés, selon ce qui doit être fait, voyons ce qui a été fait.

De façon structurelle, la plateforme est scindée en trois grandes parties à savoir:

- La partie **application mobile :** permettant à tous les patients d'accéder à la plateforme et à des services de santé de qualité;
- La partie **dashboard web :** ou encore partie **gestion** depuis laquelle les médecins administrent leurs consultations ainsi que leur profil de médecins;
- La partie **Accès aux données :** via une **API** permettant de gérer de façon centralisée le cœur de la plateforme ainsi que les traitements de façon générale.

3.1. Présentation des résultats

La plateforme comme produit final est livrée sous deux applications principales. Les sections suivantes présentent les résultats obtenus.

3.1.1. Application Mobile

L'interface avec laquelle les patients interagissent avec le système est une application mobile. Elle est relativement intuitive et facile d'utilisation, du fait qu'elle intègre des interfaces conviviales et des couleurs vives aux utilisateurs. La première ouverture de l'application expose une présentation brève du produit, avant de rediriger l'utilisateur sur la page de connexion.

L'application étant sécurisée, les utilisateurs sont mandatés de s'authentifier dans le but d'y accéder.

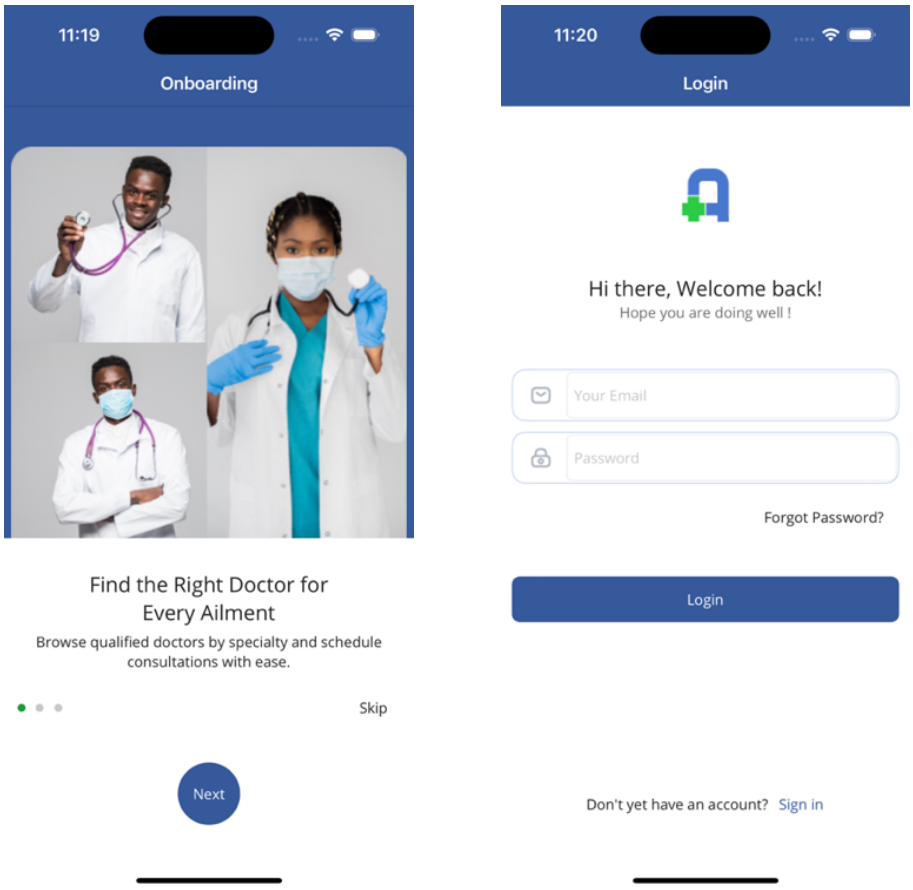


Figure 3.1: Application mobile - authentification

Une fois l'authentification effectuée avec succès, le patient a accès à son interface d'accueil, listant un top 5 des meilleurs médecins du moment, une liste des catégories et enfin la fonctionnalité de recherche.

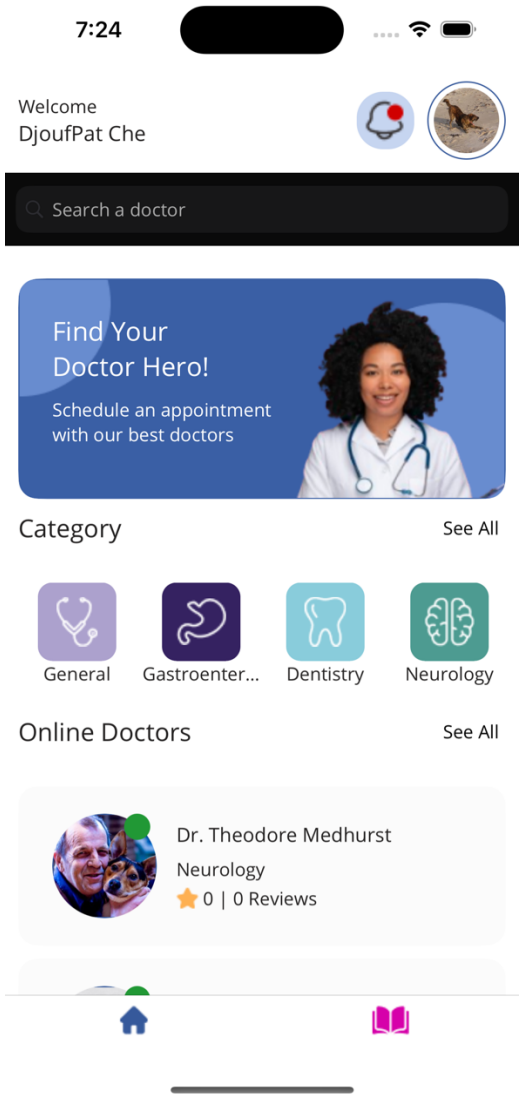


Figure 3.2: Application mobile - accueil

L'application est conçue de façon à continuellement établir un classement sur les médecins de la plateforme, affichant ainsi à chaque instant la liste des médecins classés selon leur score, avec une priorité sur les médecins en ligne et non en consultation.

La fonctionnalité de recherche s'opère ainsi sur le nom et le prénom du médecin recherché, avec en en-tête une barre de filtres pour restreindre la recherche à une catégorie en particulier.

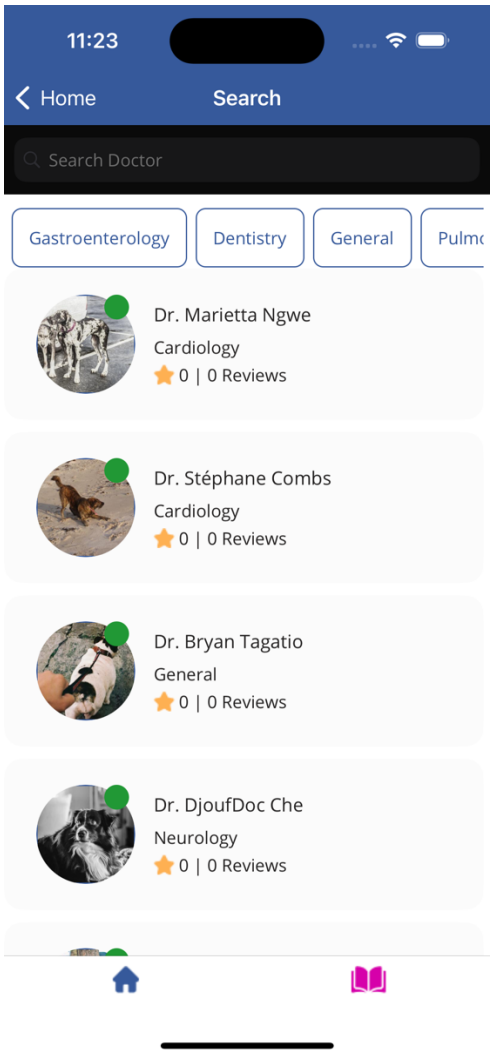


Figure 3.3: Application - Liste des médecins

Sur cette page, les patients ont la possibilité de naviguer vers le profil des médecins, sur lequel ils peuvent consulter les informations supplémentaires d'un médecin en particulier.

Sur cette interface ils peuvent soumettre une requête de consultation à un médecin, requête qui sera revue et approuvée par celui-ci avant le début de la séance de consultations.

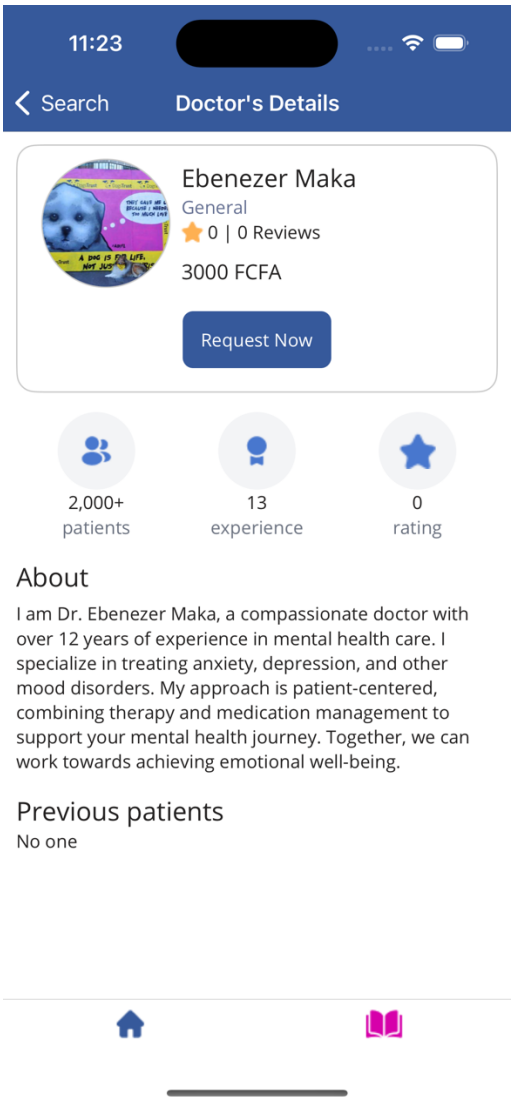


Figure 3.4: Application mobile - Détails du profil d'un médecin

La page suivante liste les consultations liées au patient authentifié, et classées en fonction du niveau de complétion, comme l'illustre la page suivante.

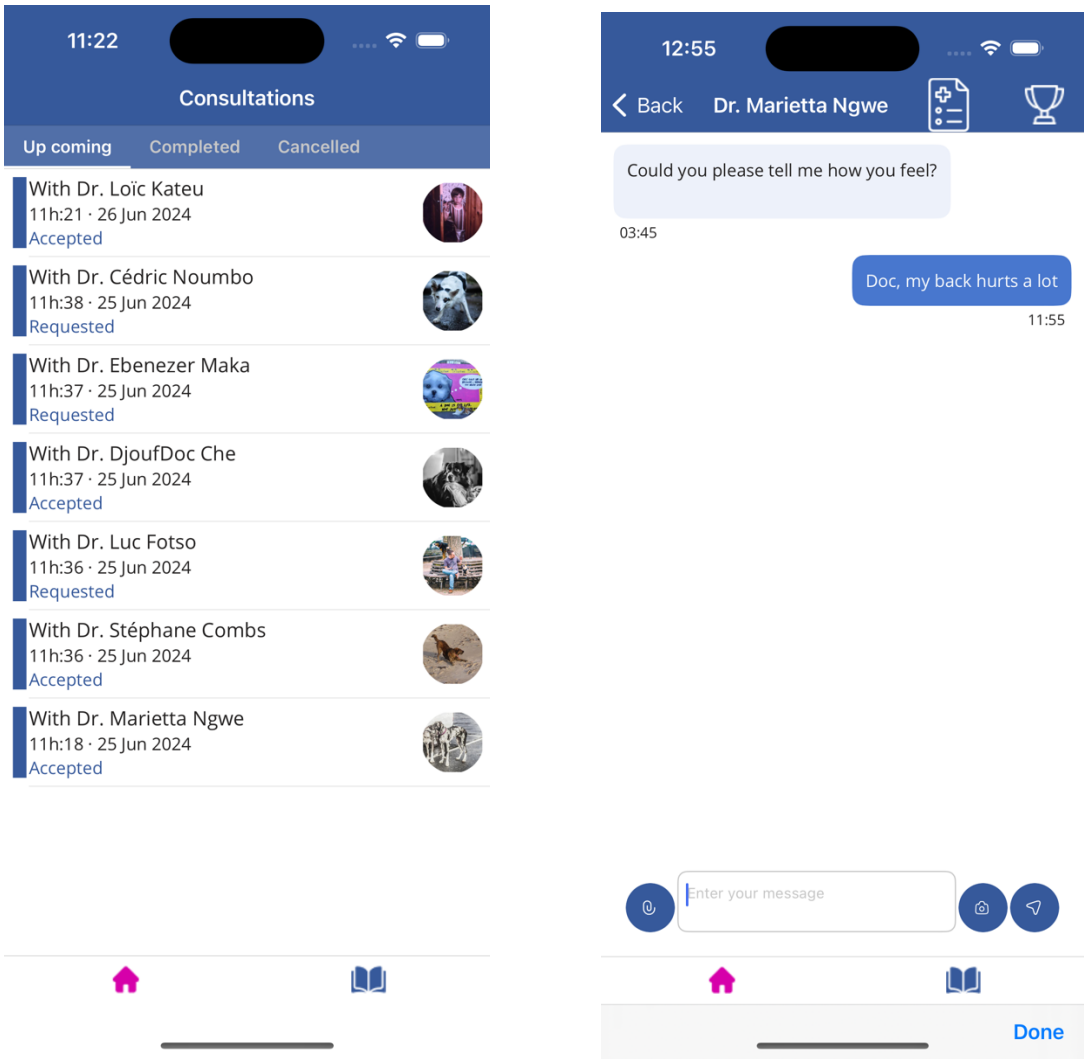


Figure 3.5: Application mobile - Processus de consultation

Au clic sur une consultation, l'utilisateur navigue vers le chat dédié avec le médecin. Il s'agit d'une messagerie instantanée et complète, permettant aux deux intervenants d'avoir une riche expérience de communication. Les fonctionnalités supportées dans le chat sont entre autres :

- Envoi de messages textes
- Envoi de photos
- Envoi de fichiers
- Appels audio
- Appels vidéo

Lorsqu'une prescription est délivrée, l'utilisateur reçoit une notification mobile et peut directement télécharger l'ordonnance en PDF sur son téléphone.

A la fin de la séance, l'utilisateur il est demandé à l'utilisateur de noter le médecin sur l'expérience acquise lors de sa consultation

3.1.2. Dashboard web des médecins

Contrairement aux patients qui utilisent les téléphones, les médecins utilisent une interface web interactive présentant les fonctionnalités far requises pour une consultation médicale.

Il devra d'abord s'authentifier à la plateforme avant d'y avoir accès.

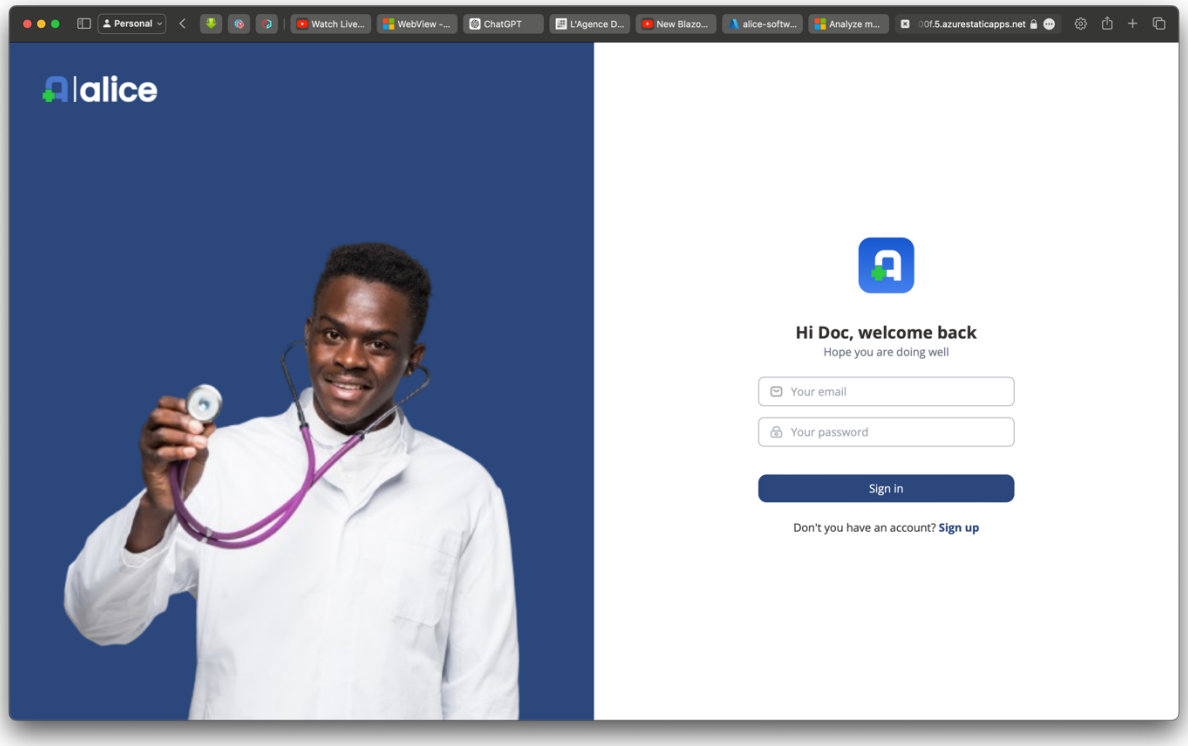


Figure 3.6: Dashboard - authentification

Une fois l'étape de connexion passée, le médecin a accès à un Dashboard présentant une vue globale de son activité sur la plateforme.

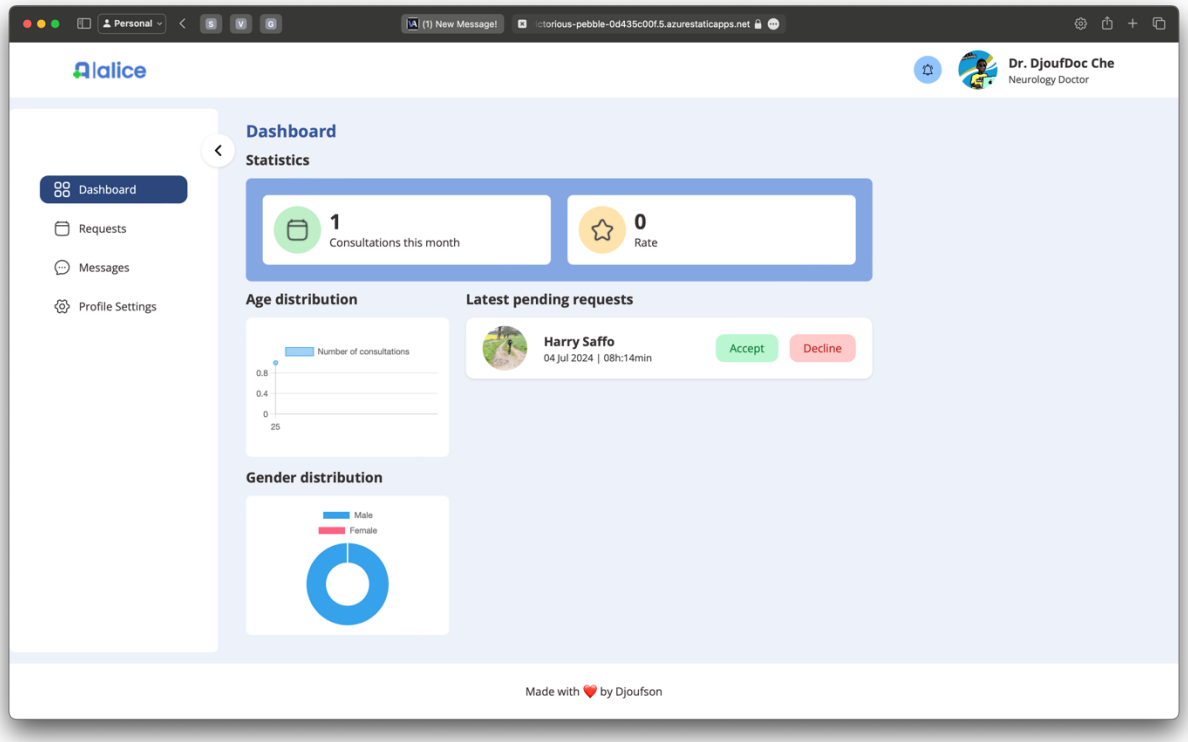


Figure 3.7: Dashboard - accueil

Le Dashboard présente au médecin un état de son activité mensuelle. Principalement :

- Le nombre de consultations effectuées durant le mois courant,
- Sa note moyenne en termes d'étoiles
- Un graphe illustrant la répartition des âges de ses patients sur le mois courant
- Un graphe illustrant la répartition des sexes de ses patients sur le mois courant
- Une liste des 5 dernières requêtes de consultations en attente de validation

La page suivante se présente comme suit :

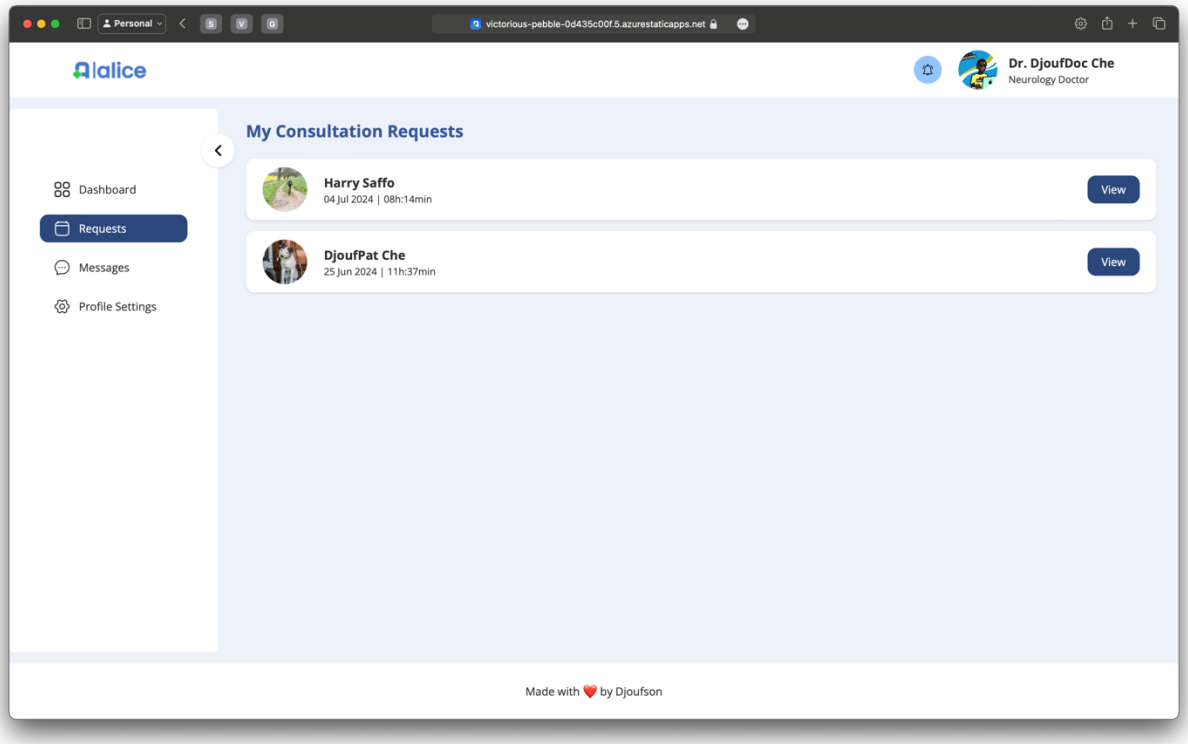


Figure 3.8: Dashboard - Liste des requêtes

Elle présente au médecin toutes les requêtes qui lui ont été soumises sur la plateforme. Elle ne sert que de page d'indication, vu que les interactions permises à ce niveau sont très réduites.

Le bouton view nous redirige vers la page des messages suivante

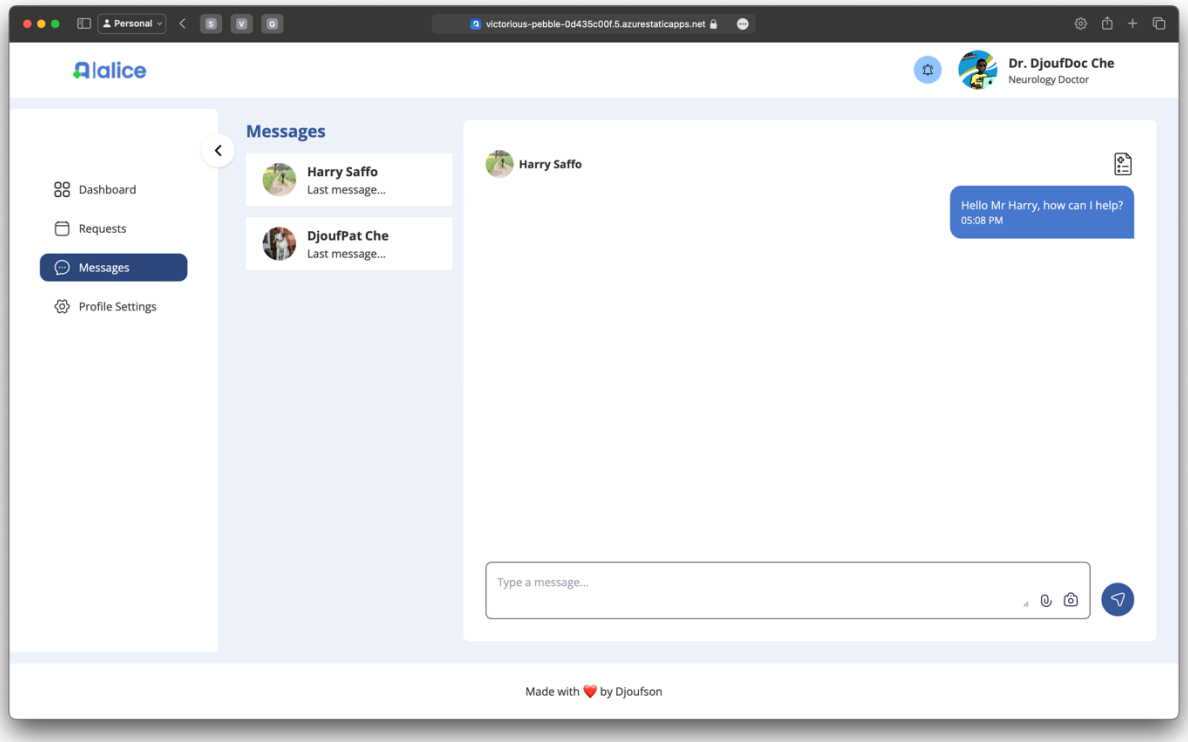


Figure 3.9: Dashboard - processus de consultation

Essentiellement il s'agit d'une messagerie instantanée entre le patient et le médecin, leur permettant d'échanger de façon sécurisée leurs différents messages. De la même façon de cette page côté mobile, elle inclut les différentes fonctionnalités suivantes :

- Envoi de message textes
- Envoi de photos
- Envoi de fichiers
- Appels audio
- Appels vidéo

Une fois l'échange terminé, le médecin peut délivrer une ordonnance médicale sous forme de fichier PDF en se servant de la page suivante.

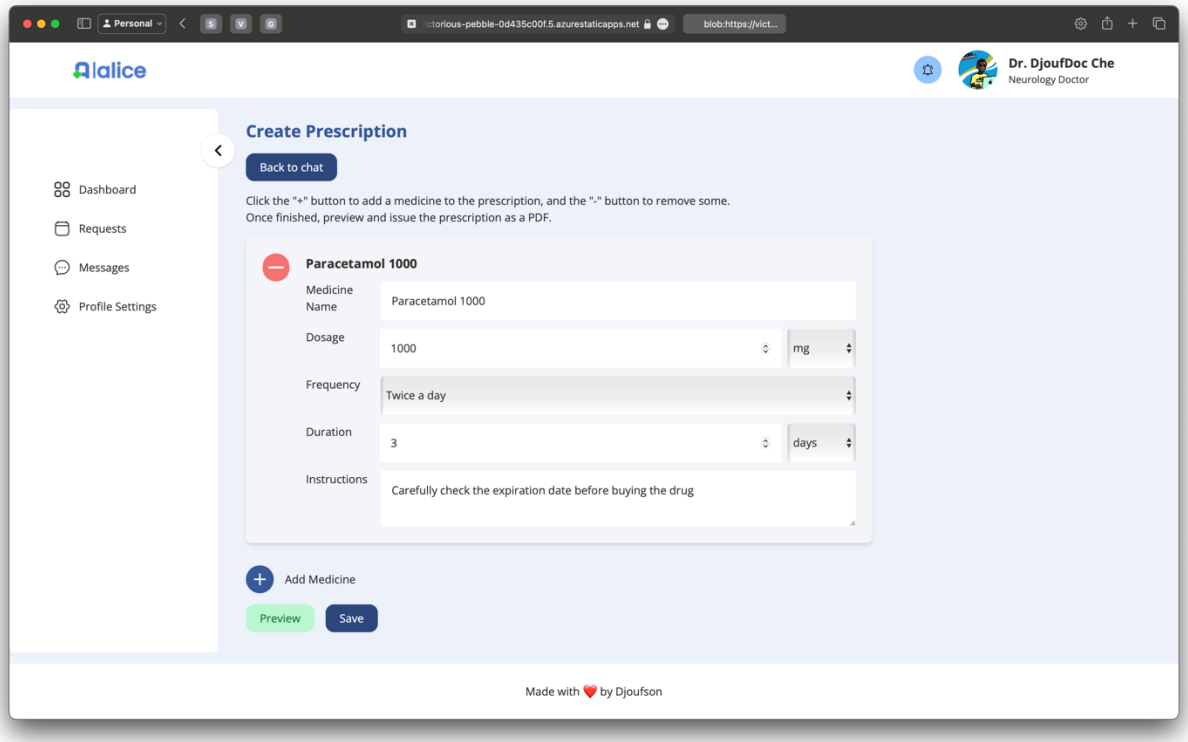


Figure 3.10: Dashboard - délivrer une prescription

L'interface fournie au médecin pour lui faciliter la génération de prescription inclut un formulaire dynamique dans lequel renseigner les différents médicaments ainsi que leur posologie.

Le bouton « Preview » lui permet d'avoir un aperçu visuel de la prescription finale avant de l'enregistrer via le bouton « Save ».

L'enregistrement de la prescription notifie directement le patient, ainsi donc il pourra la télécharger sur son téléphone et la présenter en pharmacie.

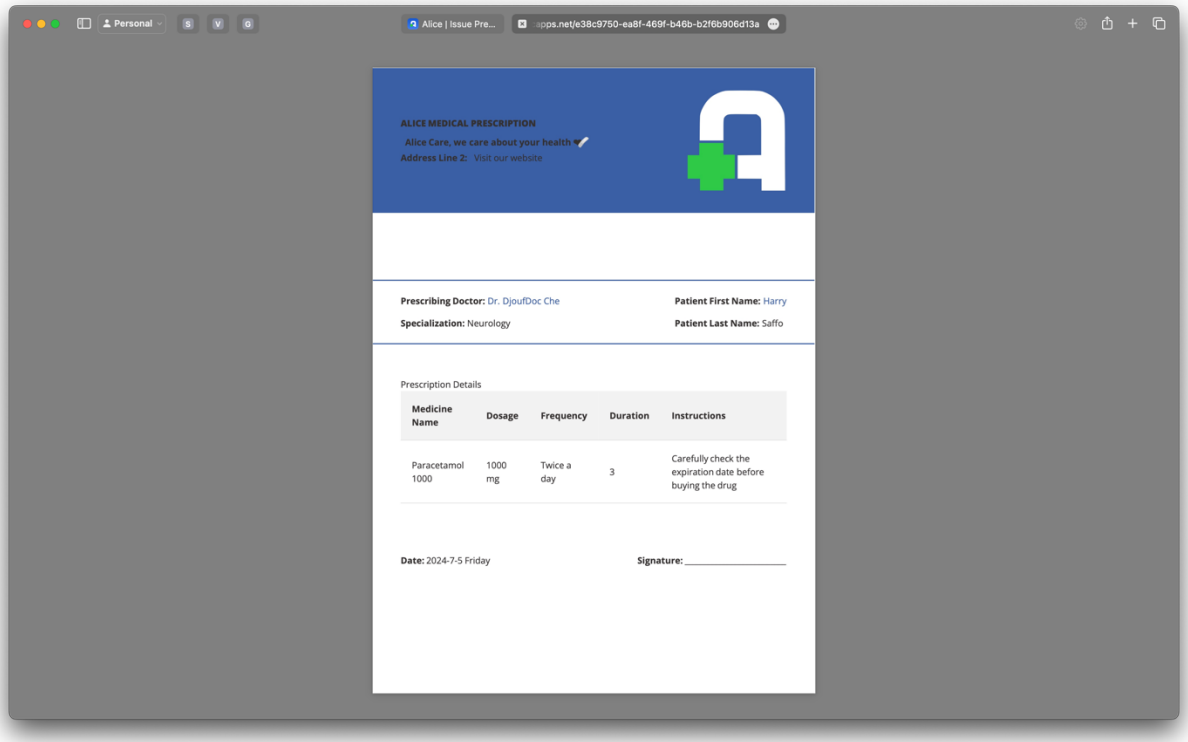


Figure 3.11: Dashboard - Prévisualisation de la prescription

Cette page s'affiche dans un nouvel onglet du navigateur, ce qui permet de s'isoler du reste de l'application. Il est à noter que certains navigateurs bloquent l'ouverture programmée de nouveaux onglets par les sites web. Dans ce cas il faudra s'adapter au comportement adopté par notre navigateur et permettre cette fonctionnalité.

La dernière page est le profil du médecin, on voit comment elle se présente sur la capture d'écran suivante :

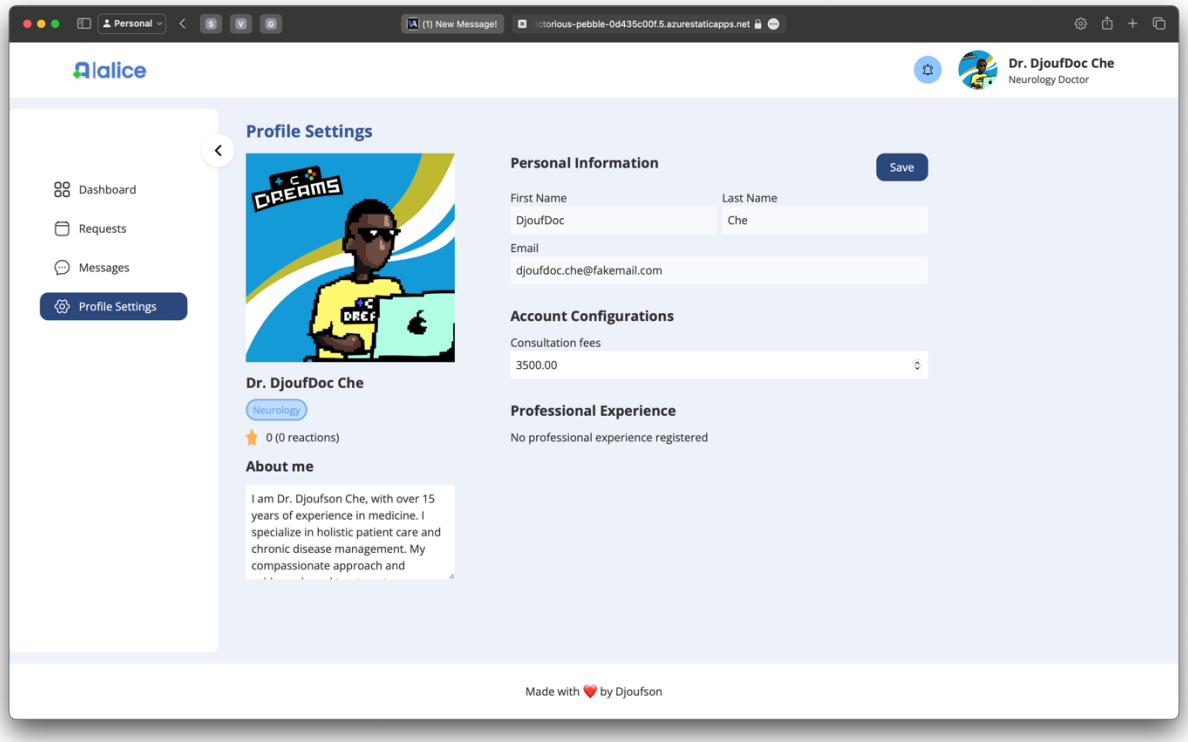


Figure 3.12: Dashboard - Profil du médecin

Sur cette page, le médecin a une vue sur ses différentes informations, et modifier en fonction de ce qui lui est autorisé de faire. Il pourra modifier entre autres :

- Les frais de consultation
- Sa photo de profil
- Sa description

3.1.3. Vitrine du produit

Tout produit se voulant compétitif sur le marché du 21e siècle se doit d'avoir une vitrine en ligne le présentant ainsi qu'exposant ses avantages et comment y avoir accès. La plateforme Alice ne fait pas exception, et à cet effet un site web vitrine a été développé dans le but d'améliorer la visibilité du produit. Le site web en question présente les différentes fonctionnalités far de l'application ainsi que comment l'acquérir.

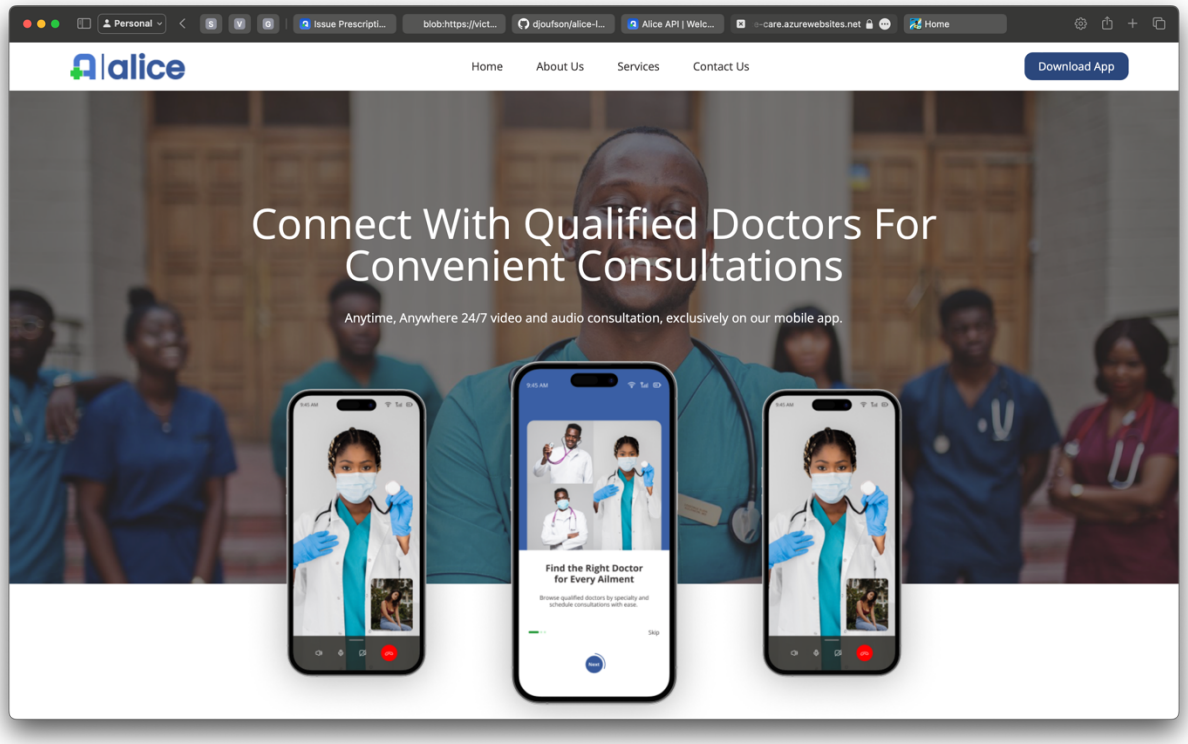


Figure 3.13: Accueil du site web

Au bout de la période de développement observée, le bilan de fonctionnalités implémentées est le suivant :

Tableau 3.1: Bilan des fonctionnalités implémentées

Fonctionnalités	État
Authentification	✅
Classement de médecins	✅
Recherche et Tri par catégorie	✅
Encryptage des données	✅
Notation des médecins	✅
Messagerie instantanée	✅
Gestion des profils de médecins	✅

Ordonnances en ligne	✅
Interface d'administration et de configuration de la plateforme	❌
Appels audio vidéo	❌

3.2. Évaluation financière du projet

L'évaluation de notre projet permet d'estimer son prix sur un marché de vente. Afin de procéder de manière plausible, nous scinderons les coûts suivant les équipements utilisés et selon la main d'œuvre.

3.2.1. Évaluation des coûts des équipements

Les équipements sont principalement matériels et logiciels, et ensuite nous comptons les coûts d'infrastructure et des services déployés gr.

Tableau 3.2: Coût total materiels et logiciels utilisés

Matériel	Quantité	Prix Unitaire	Total
MacBook Pro 2018	1	540.000 FCFA	540.000 FCFA
Moniteur Samsung 4K	1	210.000 FCFA	210.000 FCFA
License JetBrains Rider	3 (mois)	$14.90 (9015 FCFA)	27 044 FCFA
ChatGPT Plus	3 (mois)	$20 (12 100 FCFA)	36 300 FCFA
GitHub Copilot	3 (mois)	$10 (6050 FCFA)	18 150 FCFA
MTN Home Box	Promotion 3 mois	45 000 FCFA	45 000 FCFA
Forfait Internet	2 (mois)	15 000 FCFA	30 000 FCFA
			TOTAL : 906 494 FCFA

La plateforme Microsoft Azure nous permet d'avoir un rapport net des coûts de l'infrastructure déployée pendant une certaine période.

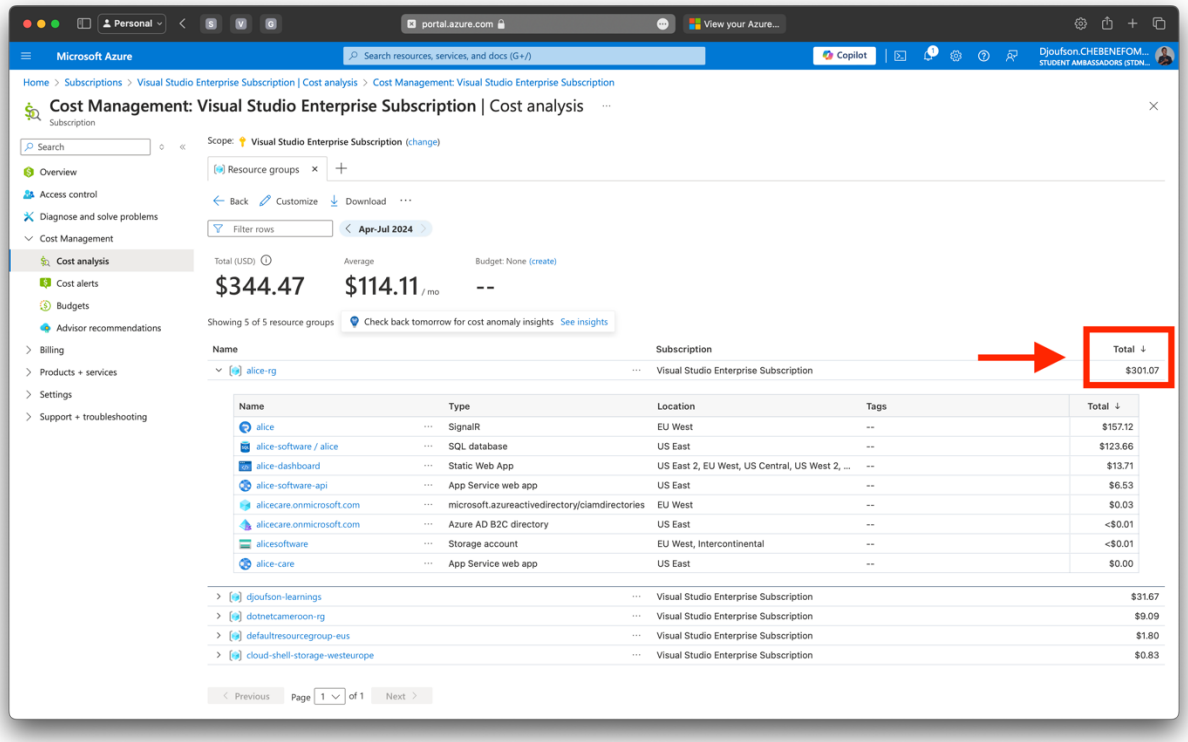


Figure 3.14: Estimation du coût d'infrastructure sur Azure

Cette capture nous permet de totaliser les frais de déploiement à $301.07 ce qui équivaut à un total de **182 155 FCFA.**

3.2.2. Évaluation du coût de main d'œuvre selon la méthode COCOMO

La charge est la quantité de travail qu'une personne peut réaliser en jour/homme (J/h), en mois/homme (M/h) ou en année/homme (A/h). La taille du projet se mesure à sa charge. L'ordre de grandeur est donné selon les normes ISO :

- Charge < 6 M/h très petit projet
- 6 M/h ≤ charge ≤12 M/h petit projet
- M/h ≤ charge ≤30 M/h projet moyen
- 30 M/h ≤ charge ≤100 M/h grand projet
- 100 M/h = charge très grand projet.[23, p. 73]

La durée du projet dépend de la charge et du nombre de personnes affectées. Par exemple 60M/h peuvent être fait par une personne sur 5 ans, 10 personnes sur 6 mois et 60 personnes sur 1

mois. Pour essayer donc de trouver une corrélation entre la taille d'un projet et la charge de travail associée, Barry W. Boehm propose en 1981 la méthode COCOMO[24].

En se basant sur les différentes formules de la méthode COCOMO ci-dessous :

Tableau 3.3: Formules de base de COCOMO

Effort = a(KLS)b *Durée = c.(Effort)d* *Participants = Effort/Durée*	Avec: • **KLS :** le nombre de ligne de codes • **a**, **b**, **c**, **d :** des paramètres fonction de la catégorie du projet

A l'aide d'un script python, il est possible d'approximer le nombre de lignes de code écrites dans le code source, et nous parvenons à un résultat de 31 000 lignes. Ce nombre ne dépassant pas les 50 000 lignes, nous utiliserons donc la méthode liée aux projets simples.

Tableau 3.4: Résultats de l'éffort, da durée et la productivité selon la méthode COCOMO

Calcul de l'effort	**Calcul de la durée**
$Effort = 2{,}4 \times (31)^{1{,}05}$ *Effort = 88M/h*	$Durée = 2{,}5 \times (88)^{0{,}38}$ *Durée = 14 mois*
Productivité = 88/ 14 ***Productivité = 6 lignes/heure***	

Déduisons-en le coût du logiciel en suivant les étapes ci-dessous.

- Quantité d'effort par an :

  88/14 = ***6.28 A/h;***
- Le nombre d'heures à fournir pour un effort sur l'année :

En ayant fixé au préalable l'équivalence *1 H/An = 1350h*, on obtient

6.28× 1350 = ***8478 heures ;***

- Le coût en euros du projet, puis en francs CFA :

  En ayant également fixé l'équivalence *1h=€10*, on obtient

  8478×10 = ***€ 84 780 = 55 609 128 FCFA***

Ceci nous permet de faire un bilan sur les différents coûts évalués pour cette solution.

Tableau 3.5: Récapitulatif des coûts liés au projet

Matériel	**Total**
Coûts matériels et logiciels	906 494 FCFA
Coûts d'infrastructure de déploiement	182 155 FCFA
Coûts de main d'œuvre	55 609 128 FCFA
	TOTAL : 56 697 777 FCFA

Dans ce chapitre, nous avons présenté les différents résultats que nous avons pu obtenir au terme de cette première période de développement, ainsi qu'une brève comparaison de notre solution face à l'existante. Enfin nous avons évalué les coûts liés à notre solution pour juger de sa faisabilité sur le long terme grâce à la méthode COCOMO, pour un coût total de **56 697 777 FCFA**.

CONCLUSION GENERALE ET PERSPECTIVES

Arrivé au terme de notre travail, il était question pour nous de concevoir, implémenter et déployer une plateforme de consultation en ligne, répondant au problème de mise en relation des médecins et patients dans un contexte camerounais.

Nous avons suivi une approche Cloud Native en intégrant les pratiques du DevOps, des méthodes Agiles Scrumban et de l'automatisation des tâches. Notre thème axé sur la mise en œuvre d'une telle architecture met en avant les avantages du Cloud Computing avec Microsoft Azure et la robustesse des technologies .NET utilisées. Couplé à cela, la méthode Scrumban nous a permis d'adopter une stratégie de développement flexible dont l'évolution est incrémentale et perceptible.

La solution proposée se décline sous deux applications, la première étant une application mobile destinée aux patients en quête de consultations de qualité à des prix abordables, la deuxième étant une application Web destinée aux médecins leur permettant d'avoir une vue globale sur l'administration de leurs consultations et les statistiques associées. Ceci offre aux médecins l'opportunité de partager leur expertise au grand public, et aux patients l'accès aux médecins qualifiés et disponibles en temps réel.

L'objectif principal étant atteint, il sera question en guise de perspectives d'une part d'intégrer un système de carnets médicaux en ligne pour faciliter le suivi des patients au cours du temps par différents médecins. Les médecins pourront ainsi consulter le parcours antérieur de leurs patients avant de leur prescrire des soins. Ils auront également un profil plus complet sur le patient à examiner. Nous notons également comme perspectives, l'intégration des services de paiement locaux et internationaux. Ceux-ci contribueront à une meilleure expérience de la part des utilisateurs locaux, et à l'extension de cette solution sur le plan international.

REFERENCES

[1] rfi, “Cameroun: le désarroi des médecins diplômés et non intégrés.” [Online]. Available: https://www.rfi.fr/fr/afrique/20211020-cameroun-le-désarroi-des-médecins-diplômés-et-non-intégrés

[2] Munutunews, “Au Cameroun, des médecins malades du chômage et de la précarité.” [Online]. Available: https://cameroun-muntunews.com/au-cameroun-des-medecins-malades-du-chomage-et-de-la-precarite/

[3] World Health Organization, “Local action for health: a repository of WHO resources.” [Online]. Available: https://urbanhealth-repository.who.int

[4] Pan Afr Med J, *Evaluation of self-medication practices and their characteristics among Uvira in Democratic Republic of Congo students*. National Library of Medicine (NLM).

[5] Journal of Medical Internet Research, “Frequency and Correlates of Online Consultations With Doctors or Therapists in Middle-Aged and Older Adults: Nationally Representative Cross-sectional Study.” [Online]. Available: https://www.jmir.org/2022/4/e29781

[6] Michael Hofmann, Erin Schnabel, and Katherine Stanley, *Microservices Best Practices for Java*. IBM International Technical Support Organization.

[7] Hohpe and Gregor, *Enterprise Integration Patterns: Designing, Buildong, and Deploying Messaging Solutions.*

[8] geeksforgeeks, “Communication Design Patterns for Cloud Native Applications.” [Online]. Available: https://www.geeksforgeeks.org/communication-design-patterns-for-cloud-native-applications/

[9] S. “ardalis” S. Rob Vettor, *Architecting Cloud Native .NET Applications for Azure*. Microsoft Developer Division, .NET, and Visual Studio product teams.

[10] Amazon Web Services, “What is DevOps?” [Online]. Available: https://aws.amazon.com/devops/what-is-devops/

[11] Atlassian, “Scrumban: Mastering two Agile methodologies.” [Online]. Available: https://www.atlassian.com/agile/project-management/scrumban

[12] Adrián Bailador, “Monolithic Architecture in .NET,” *Dev*. [Online]. Available: https://dev.to/adrianbailador/monolithic-architecture-in-net-33i2

[13] Atlassian, “Microservices vs. monolithic architecture.” [Online]. Available: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith

[14] Prequin, “Waspito Inc.” [Online]. Available: https://www.preqin.com/data/profile/asset/waspito-inc-/475652

[15] Amazon Web Services, “What is Cloud Native?” [Online]. Available: https://aws.amazon.com/what-is/cloud-native/

[16] eSecurity Planet, “10 Major Benefits of Cloud-Native Application Development.” [Online]. Available: https://www.esecurityplanet.com/cloud/cloud-native-benefits/

[17] Nishant, *Mastering DevSecOps with Devtron: a strategic approach*. Cloud Native Computing Foundation (CNCF). [Online]. Available: https://www.cncf.io/blog/2024/06/20/mastering-devsecops-with-devtron-a-strategic-approach/

[18] EXADEL, “Understanding Cloud-Native Apps: A Complete Guide.” [Online]. Available: https://exadel.com/news/cloud-native-apps-development-guide/

[19] A. Kumar, “Wilson Lower bound Score and Bayesian Approximation for K star scale rating to Rate products.” [Online]. Available: https://medium.com/tech-that-works/wilson-lower-bound-score-and-bayesian-approximation-for-k-star-scale-rating-to-rate-products-c67ec6e30060

[20] Steve “ardalis” Smith, *Architecting Modern Web Applications With ASP.NET Core and Microsoft Azure*. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure

[21] ProductPlan, “Scrumban.” [Online]. Available: https://www.productplan.com/glossary/scrumban/

[22] Wikipedia, “Jira (software).” [Online]. Available: https://en.wikipedia.org/wiki/Jira_(software)

[23] Adrien Lagrange and Henri Djouaka, *Conception et réalisation d’un système de mise en relation des acteurs de l’entrepreneuriat et de l’innovation au Cameroun*.

[24] Acervo Lima, “Génie logiciel | Modèle COCOMO.” [Online]. Available: https://fr.acervolima.com/genie logiciel modele cocomo/