Université Assane Seck de Ziguinchor
UFR Sciences et Technologie
Département Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master

Mention : Informatique

Spécialité : Génie Logiciel

Conception et implémentation du module de gestion de dossier médical de la Direction de la Médecine du Travail de l'Université Assane Seck de Ziguinchor

Soutenu le : 28/12/2023

Présenté et soutenue par :
M Makane MBAYE

Sous la direction de :
Dr Gorgoumack SAMBE
M Alousseynou FALL

Sous la supervision de :
Pr Khadim DRAME

Membre du jury

Pr Khadim DRAME	Professeur assimilé	Président	UASZ
Pr Ibrahima DIOP	Professeur assimilé	Rapporteur	UASZ
Dr Papa Alioune CISSE	Maitre de conférences titulaire	Examinateur	UASZ
M Alousseynou FALL	Chef de Service à la DISI	Encadrant	UASZ
Dr Gorgoumack SAMBE	Maitre de conférences titulaire	Encadrant	UASZ

Résumé

Ce travail fait partie du Projet de Gestion Intégrée (PGI) de la Direction de l'Informatique et des Systèmes d'Information (DISI) de l'Université Assane Seck de Ziguinchor (UASZ). Nous nous intéressons essentiellement, dans le cadre de notre mémoire de master, à la mise en place du module de gestion de dossier médical (DM) pour la Direction de la Médecine du Travail (DMT) de l'Université Assane Seck de Ziguinchor.

La DMT a toujours utilisé le DM en format papier qui présente de nombreuses limites comme la difficulté de conservation de l'identité du patient, la perte de données médicales, la difficulté d'avoir des statistiques, ... C'est dans ce contexte que nous avons développé une application de gestion de dossier médical en travaillant sur l'aspect de la conception et la mise en œuvre.

Pour réaliser ce travail de manière efficace, nous avons utilisé une approche de conception AGILE, plus précisément SCRUM, pour faciliter notre collaboration étroite avec les clients impliqués dans ce module.

Dans la phase de spécification des besoins, nous avons identifié les utilisateurs et divisé notre module en sous-modules avant de sortir les cas d'utilisations de chaque sous-module. Durant la phase analyse des besoins et conception du système, nous avons proposé des diagrammes d'activités et de séquences en analyse des besoins et les architectures physiques, logiques et nos diagrammes de classes en conception. Pour le développement du front-end, nous avons utilisé l'architecture MVC, et pour le back-end, nous avons utilisé le modèle en couche. Nous avons employé la notation UML avec ses différents diagrammes pour les différentes étapes de conception et l'outil PowerAMC pour la conception de ces diagrammes. Nous avons développé de manière collaborative en utilisant l'outil Git et aussi pour la gestion de versions.

L'application a été développée avec deux frameworks : Angular pour le front-end et Spring Boot pour le back-end et les technologies qui gravitent autour d'eux. Nous avons aussi utilisé le système de gestion de base de données PostgreSQL pour les données, l'API REST pour la création des services et Postman pour leur test. Le template (template angular) de la Direction de l'Informatique et des Systèmes d'Information (DISI) nous a permis de faire le design.

Abstract

This work is part of the Integrated Management Project (PGI) of the Direction de l'Informatique et des Systèmes d'Information (DISI) of the Université Assane Seck de Ziguinchor (UASZ). As part of our master's thesis, we are mainly interested in the implementation of the Medical Record (MR) Management Module for the Occupational Medicine Department (DMT) of the Assane Seck University of Ziguinchor.

DMT has always used paper DMs, which have a number of limitations, such as the difficulty of retaining patient identity, the loss of medical data, the difficulty of obtaining statistics, and so on. It was against this backdrop that we developed a medical record management application, working on the design and implementation aspects.

To carry out this work efficiently, we used an AGILE design approach, more specifically SCRUM, to facilitate our close collaboration with the customers involved in this module.

In the requirements specification phase, we identified the users and divided our module into sub-modules before drawing up the use cases for each sub-module. During the requirements analysis and system design phases, we proposed activity and sequence diagrams in the requirements analysis phase and the physical and logical architectures and our class diagrams in the design phase. For front-end development, we used the MVC architecture, and for the back-end, we used the layered model. We used the UML notation with its various diagrams for the different design stages and the PowerAMC tool to design these diagrams. We developed in a collaborative way, using the Git tool and also for version management.

The application was developed using two frameworks: Angular for the front-end and Spring Boot for the back-end, and the technologies that revolve around them. We also used the PostgreSQL database management system for the data, the REST API for service creation, and Postman for service testing. The template (an angular template) from the Direction de l'Informatique et des Systèmes d'Information (DISI) was used for the design.

Dédicaces

***P**ar la grâce d'Allah, le tout puissant, je dédie ce travail :*

À mon père et ma mère

***C**'est grâce à vous que nous sommes présents en ce jour béni. Vous n'avez épargné aucun effort afin de nous voir évoluer dans des conditions exemplaires. Vous êtes et demeurez les plus remarquables des parents. Je souhaite sincèrement que ce travail témoigne de mon amour filial incommensurable à votre égard.*

À mes frères et sœurs

***P**uissiez-vous demeurer solidaires et unis dans l'existence, tout en restant fidèles à l'instruction bienveillante inculquée par nos chers parents.*
Que Dieu continue d'éclairer vos chemins.

À tous mes amis et camarades de promo

***V**os soutiens, conseils et encouragements me tiennent à cœur.*

À particulièrement mon grand frère Ibrahima MBAYE

Pour tout ce qu'il endure pour nous, que Dieu continue de guider tes pas, frère.

Remerciements

***T**out d'abord, j'exprime ma profonde reconnaissance envers **Allah** le Tout-Puissant pour m'avoir octroyé la force, la patience et le courage nécessaire afin de parvenir jusqu'à ce stade et accomplir cette tâche.*

***J**e souhaite exprimer toute ma gratitude envers mon encadreur **Monsieur Gorgoumack SAMBE** et mon Maître Stage **Monsieur Alousseynou FALL** pour vos soutiens, vos disponibilités, encouragements et vos remarques extrêmement précieuses qui ont contribué à l'amélioration de la qualité de ce travail.*

***Je** tiens également à exprimer mes sincères remerciements aux membres du jury, à savoir **Monsieur Khadim DRAME**, **Monsieur Ibrahima DIOP** et **Monsieur Papa Alioune CISSE**, enseignants-chercheurs distingués de l'Université Assane Seck de Ziguinchor. Je suis très reconnaissant de l'intérêt qu'ils ont manifesté à mon travail, ainsi que du temps précieux qu'ils ont bien voulu consacrer à son évaluation. Leurs suggestions précieuses sont grandement appréciées.*

***Je** tiens à exprimer mes sincères remerciements à vous, **Monsieur Albert PREIRA**, **Monsieur Malick FAYE**, et à tous les autres travailleurs de la **D**irection de la **M**édecine du **T**ravail de l'Université. Je souhaite témoigner ma gratitude pour votre aide et votre disponibilité tout au long de mon stage. Vous avez fait preuve d'une générosité sans faille en me fournissant toutes les informations essentielles qui m'ont permis d'accomplir ce travail avec succès.*

***Je** souhaite également vous exprimer mes sincères remerciements à mes chers amis pour vos encouragements et votre soutien indéfectible.*

***Mes** vifs remerciements s'adressent également à tous les enseignants du département informatique de l'Université Assane Seck de Ziguinchor pour la formation qu'ils ont eu le soin de nous donner.*

***Je** vous exprime ma gratitude la plus sincère, chers camarades de promotion, compagnons de chambre et toutes les personnes qui ont apporté leur contribution à l'élaboration méticuleuse de ce travail.*

Je *ne pourrais finir sans remercier l'équipe de la Direction de l'Informatique et des Systèmes d'Information de l'Université, plus particulièrement* ***Diariatou MANGA***, ***Ana BAKHOUM***, ***Ndèye Astou DIATTA***, ***Mohamed Chérif DIALLO*** *et* ***Dame BA*** *pour la relecture, les soutiens et les encouragements.*

À ma famille d'accueil depuis Bignona

S*ans vous, mes tontons* ***Maguette NDIAYE***, ***Modou NDIAYE***, ***Modou Mbaye NDIAYE,*** ***Abdou NDIAYE*** *et* ***Alé SALL***, *mes cousins* ***Abdou WADE*** *et* ***Bou WADE***, *je ne saurais arriver là où je suis. Merci pour vos soutiens et votre considération inconditionnelle.*

Aux familles Mané et Camara de Néma II

Sans vous, il me serait plus difficile d'arriver là où je suis. Merci pour vos soutiens et votre considération.

Table des matières

Liste des figures

Liste des tableaux

Abréviations et sigles

CD : Livraison Continue.

CI : Intégration Continu.

COEN : Conseil d'Orientation et d'Evaluation du Numérique.

CRI : Centre des Ressources Informatiques.

DAS : Division de l'Animation Sociale.

DISI : Direction de l'Informatique et des Systèmes d'Informations.

DM : Dossier Médical.

DMT : Direction de la Médecine du Travail.

DP : Division de Pharmacie.

DRH : Direction des Ressources Humaines.

DSSI : Division de Supervision des Soins Infirmiers.

ENT : Environnement Numérique de Travail.

HTTP : HyperText Transfer Protocol (ou protocole de transfert hypertexte en français).

IaC : Infrastructure as Code.

IDE : Environnement de Développement Intégré.

PATS : Personnel Administratif, Techniques et de Service.

PER : Personnel d'Enseignement et de Recherche.

PGI : Projet de Gestion Intégrée.

SCS : Service de Calculs et Simulations.

SGBD : Système de Gestion de Base de Données.

SRP : Service des Ressources Pédagogique.

SST : Service de Support et Technique.

TDD : Test Driven Development (ou Développement Piloté par des Tests en français).

TIC : Technologies de l'Information et de la Communication.

UASZ : Université Assane Seck de Ziguinchor.

UML : Unified Modeling Language, ou langage de modélisation unifié en français.

XP : Programmation Extrême.

Introduction générale

Nous avons effectué notre stage de fin d'études à la Direction de l'Informatique et des Systèmes d'Information (DISI) de l'Université Assane Seck de Ziguinchor (UASZ) dans le cadre du Projet de Gestion Intégrée (PGI) qui est composé de plusieurs modules : le module de gestion des ressources humaines, le module de la gestion de la répartition des cours, le module de la gestion des évaluations, le module de la gestion des emplois du temps, le module de la gestion des cahiers de texte, le module de la gestion des dossiers médicaux**,** etc.

Durant notre stage, nous avons pour objectif de concevoir et d'intégrer le module de gestion de dossier médical du PGI. À notre arrivée, le PGI était déjà en marche et l'équipe de développement de la DISI trouvée sur place avait déjà fait une étude et défini les architectures et les technologies jugées plus adaptées pour atteindre leur objectif.

Le dossier médical est un recueil d'informations d'ordre médical pour un patient au cours de son séjour dans un centre hospitalier. Ce document regroupe la trace de toutes les actions d'ordre médical (les visites, les consultations, les prescriptions, les préventives, etc.), ainsi que la réflexion et la relation médecin-patient.

Cet outil est utilisé pour la réflexion, la synthèse, la planification et la traçabilité des soins, mais aussi pour la recherche et l'enseignement. Il permet également de rassembler les actions de tous les acteurs de la santé.

Notre objectif est de réaliser une application de gestion de dossiers médicaux des patients pour faciliter l'échange des informations, réaliser la meilleure prise en charge mais aussi faciliter la gestion des données des patients pour la Direction de la Médecine du Travail de l'Université Assane Seck de Ziguinchor.

Notre mémoire est structuré comme suit :

- Au chapitre 1 nous présentons la structure d'accueil, la DISI, le projet dans le nous avons travaillé (le PGI), la structure à laquelle l'application est destinée, la DMT, le contexte et la problématique du stage et enfin les objectifs du stage.
- Au chapitre 2, nous présentons quelques processus de conceptions et précisons celui que nous avons adopté pour notre stage et ce qui justifie ce choix.

- Dans le chapitre 3, nous présentons la spécification des besoins. Nous allons ressortir les différents acteurs, déterminer les besoins fonctionnels du système à travers les cas d'utilisation.
- Dans le chapitre 4, nous faisons une analyse des besoins et présentons les résultats de la conception générale et de la conception détaillée. Nous exposons les architectures et les modèles choisis.
- Et enfin, au chapitre 5, nous présentons les différents outils utilisés pour l'implémentation de l'application, suivi des tests du déploiement et de la présentation de quelques interfaces de l'application.

Chapitre I : Contexte et problématique

Introduction

Dans ce chapitre, nous allons d'abord présenter la structure d'accueil qui est la Direction de l'Informatique et des Systèmes d'Information (DISI) de l'Université Assane Seck de Ziguinchor. Une structure où nous avons effectué notre stage dans le Service Support et Technique (SST). Ensuite, nous décrirons la Direction de la Médecine du Travail (DMT) qui représente le client et les futurs utilisateurs de l'application. Et enfin, nous dégageons le contexte du stage, la problématique traitée et les objectifs visés dans ce stage.

I.1. Présentation des structures

I.1.1. La Direction de l'Informatique et des Systèmes d'Information

La Direction de l'Informatique et des Systèmes d'Information (DISI) de l'Université Assane Seck de Ziguinchor a été créée le 12 février 2021 en remplacement du Centre des Ressources Informatiques (CRI). Elle est dirigée par un directeur sous la supervision d'un Conseil d'Orientation et d'Évaluation du Numérique (COEN).

I.1.1.1. Missions

Sous l'autorité du Recteur, la DISI est chargée de définir et de mettre en œuvre la politique numérique de l'Université.

Elle a donc pour mission :

- d'élaborer les plans directeurs de l'informatique de l'Université et en assurer la mise en œuvre et le suivi ;
- de participer à la réalisation des tableaux de bord et au pilotage de l'institution et de ses différents composants ;
- de stimuler, structurer, accompagner et soutenir les projets dans le domaine du calcul scientifique et de la simulation ;
- de développer un système informatique intégré et cohérent ;
- d'assurer une veille technologique.

I.1.1.2. Organisation

La DISI est structurée autour de trois services. Chaque service est constitué de divisions à sa tête un chef de division, Ils sont tous sous l'autorité du directeur de la DISI.

Ces trois services sont :

- Le Service des Ressources Pédagogiques (SRP) ;
- Le Service de Support et Technique (SST) ;
- Le Service de Calculs et Simulations (SCS).

La figure ci-dessous (***figure 1***) représente l'organigramme de la DISI.

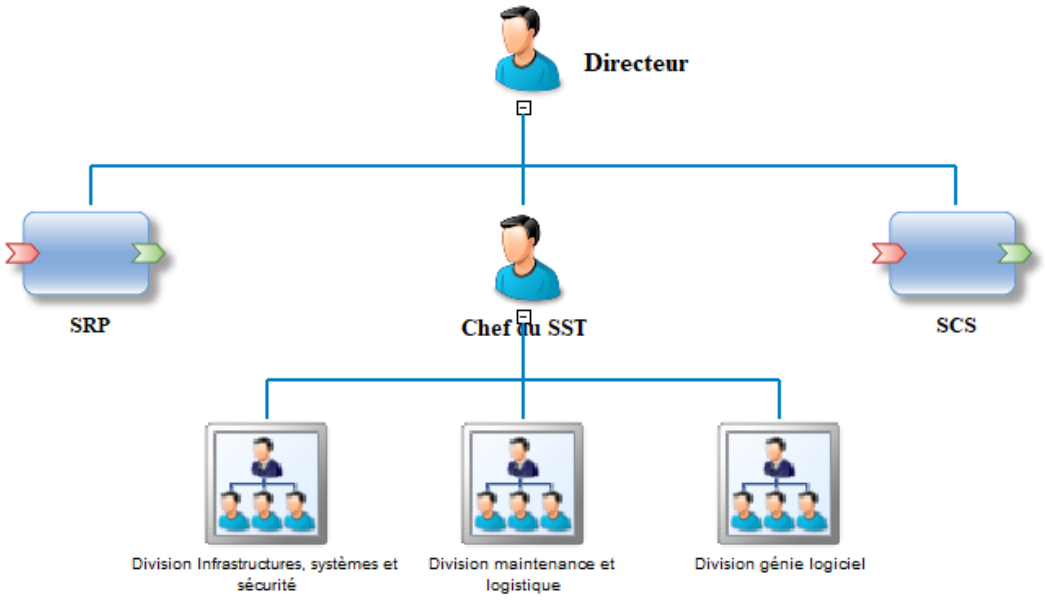

Figure 1: Organisation de la DISI

I.1.1.2.1. Le SRP

Il est chargé :

- de participer à la professionnalisation des étudiants inscrits dans les formations classiques à l'Université ;
- de former et de renforcer les capacités en Technologies de l'Information et de la Communication (TIC) du Personnel Administratif, Techniques et de Service (PATS) ;
- de contribuer à la promotion et au développement de la formation en informatique et télécommunication et leurs applications ;
- de contribuer à la promotion de l'utilisation des logiciels libres en particulier sur le plan pédagogique.

I.1.1.2.2. Le SST

Il est chargé :

- d'assurer la disponibilité des infrastructures réseaux, systèmes et des services ;
- de développer un système d'information intégré et cohérent ;
- de gérer le portail internet de l'Université en collaboration avec la direction de la communication et du marketing de l'Université ;
- de développer, déployer et maintenir un Environnement Numérique de Travail (ENT) ;
- de dématérialiser les procédures administratives de l'Université de concert avec le SRP ;
- de contribuer au respect de la charte graphique de l'institution en relation avec les directions centrales et les différentes composantes de l'Université ;
- d'assurer le respect des directives institutionnelles liées à la politique informatique d'achat de matériels, de logiciels et de gestion du parc informatique ;
- de gérer l'activité d'assistance de proximité en assurant la remontée des besoins liés au système d'information et une assistance technique auprès des utilisateurs ;
- d'administrer le parc informatique en assurant la maintenance et le renouvellement ;
- de mettre à la disposition des structures de formation et de recherche les infrastructures informatiques et les outils nécessaires au bon déroulement de leurs activités.

I.1.1.2.3. Le SCS

Il est chargé :

- de mettre à la disposition de la communauté des plateformes de test, de simulation, de stockage et de calcul (grilles, cloud, …) ;
- d'assister la communauté scientifique dans le déploiement d'outils ;
- d'assurer la prise en charge des besoins en calcul numérique de la communauté scientifique ;
- d'assurer le renforcement de capacité des chercheurs sur les outils de recherches scientifiques ;
- d'accompagner la mise en place de formations par les structures habilitées dans les domaines du calcul scientifique et du calcul haute performance ;
- de mettre en place les outils nécessaires et accompagner les enseignants-chercheurs et étudiants dans l'utilisation des plateformes distantes (clusters).

I.1.1.3. Le Projet de Gestion Intégrée

Le projet de gestion intégrée (PGI) est initié par la DISI en collaboration avec la DRH, la DMT et les UFR. Ce projet d'automatisation des tâches administratives dans la gestion du personnel vise à moderniser et à simplifier les processus administratifs au sein de l'Université. Cette initiative est motivée par le besoin d'optimiser l'efficacité opérationnelle, de réduire les erreurs humaines et d'accélérer les délais dans le traitement des données liées au personnel.

Ce projet a été lancé à la fin de l'année 2022 et est débuté avec quatre (4) modules :

- le module de gestion des ressources humaines : ce module gère certaines fonctionnalités comme le suivi des dossiers des employées notamment la gestion des candidatures, la gestion des contrats, la gestion des affectations, la gestion de promotion etc. ;
- le module de gestion des emplois du temps : ce module gère la gestion des emplois du temps pour les UFR ;
- le module de gestion de la répartition des enseignements : ce module gère la répartition des enseignements (cours) pour toutes les formations;
- le module de gestion de dossier médical : c'est le module sur lequel nous avons travaillé. Il est conçu pour stocker, organiser et gérer les informations médicales des patients de la DMT de manière électronique.

Comme tous les autres modules, le module de gestion de dossier médical a une forte dépendance avec le module de gestion des ressources humaines. En effet, notre module obtient les informations personnelles des patients internes de ce module. Mais aussi, le module de gestion des ressources humaines a besoin des informations de notre module, comme pour le cas du résultat (réponse) après une visite d'embauche.

I.1.2. La Direction de la Médecine du Travail (DMT) de UASZ

I.1.2.1. Présentation et missions de la DMT

La Direction de la Médecine du Travail de l'Université Assane Seck de Ziguinchor a été mise en place le 18 février 2019. Elle est sous l'autorité du Secrétaire Général de l'Université.

Elle est chargée :

- d’effectuer des visites d'embauche et périodiques du personnel de l’Université ;
- d’effectuer les consultations médicales au profit du personnel, de leurs conjoints et de leurs enfants en charge conformément à l'article 7 du Code de sécurité sociale ;

- d'organiser et de rationaliser les références aux structures hospitalières agréées ;
- de procéder au dépistage des maladies liées à l'environnement de travail ;
- de délivrer des soins aux personnels victimes d'accidents de travail ;
- de veiller en rapport avec la Direction de l'Environnement et du Cadre de Vie au respect des normes d'hygiène et de salubrité (protection contre les nuisances, les risques d'accidents de travail ou d'utilisation de produit dangereux) ;
- de donner des conseils, des techniques et des rythmes de travail adaptés à la physiologie humaine ;
- de promouvoir la prévention et l'hygiène sanitaire au sein de l'Institution en rapport avec l'activité professionnelle ;
- d'assurer des prestations médicales pour les populations environnantes.

I.1.2.2. Organisation

La DMT est structurée autour de trois divisions. Chaque division est dirigée par un chef de division :

- La Division de Supervision des Soins Infirmiers (DSSI) ;
- La Division de Pharmacie (DP) ;
- La Division de l'Animation Sociale (DAS).

La figure ci-dessous (***figure 2***) est une représentation de l'organigramme de la DMT

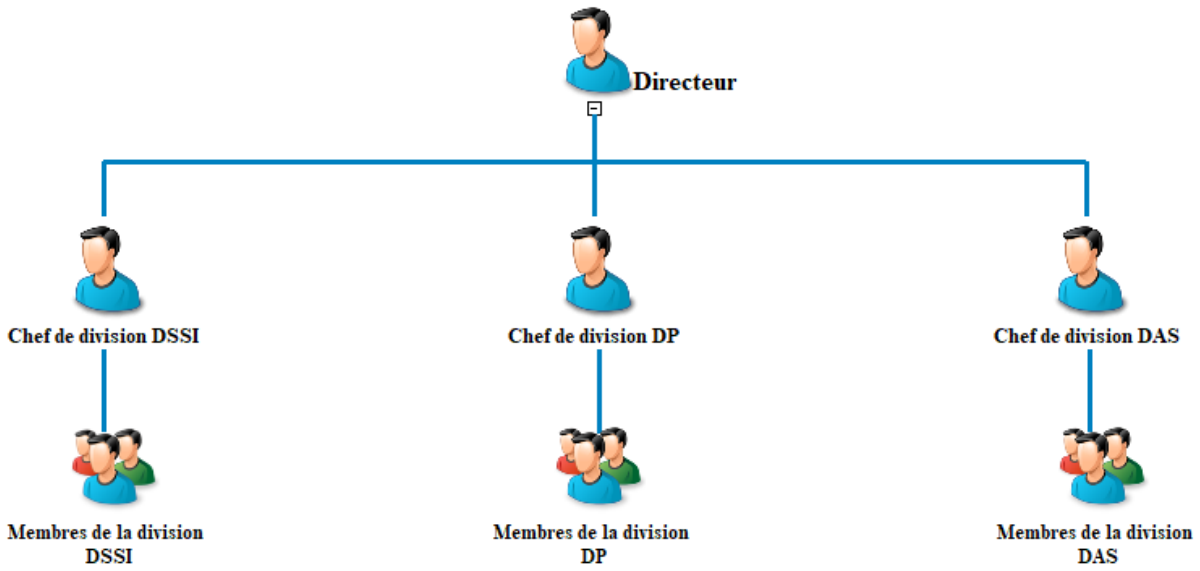

Figure 2 : Organisation de la DMT

I.1.2.2.1. La DSSI

Elle est sous l'autorité du Directeur et est chargée d'assurer le tri des patients, les soins infirmiers, la couverture médicale des manifestants, la gestion du matériel de soin, de prendre en charge les urgences sur prescription médicale et de réaliser la planification des visites médicales des travailleurs.

I.1.2.2.2. La DP

Sous l'autorité du Directeur, elle est chargée d'approvisionner la direction en médicaments, en produits de soins et en matériels médicaux et d'assurer la gestion et le stockage des médicaments, produits de soins et matériels médicaux.

I.1.2.2.3. La DAS

Elle est sous l'autorité du Directeur et est chargée d'exécuter la politique de l'institution en application des programmes d'actions en matière d'assistance et d'activités destinés aux personnels de l'Université, d'assurer l'accompagnement psychosocial du personnel et de leur famille à la demande, d'organiser les cérémonies de rapprochement (arbre de Noël, fêtes de fin d'année, colonies de vacances,) et de représenter l'institution lors des cérémonies sociales (décès, mariages, ...).

I.1.2.3. Fonctionnement de la DMT

Dans le cadre de son fonctionnement, la DMT accueille différents patients. Ces patients peuvent être le personnel de l'Université, leurs conjoint(e)s, leurs enfant (s), les étudiants, les vacataires et les externes (les patients qui ne sont ni personnel, ni conjoint(e) ni enfant du personnel, ni étudiant, ni vacataire). La DMT offre beaucoup de services à ces différents patients comme :

- Les consultations générales (avec le médecin généraliste) : tous les patients sont pris en compte, quels que soient leur âge et leurs pathologies. Le médecin et les infirmiers ont un champ d'intervention étendu : ils aident les patients à identifier leurs problèmes de santé, réalisent un examen clinique général (mesurent de la tension artérielle, testent des réflexes, auscultation cardiaque, etc.) ou spécifique, établissent un diagnostic, prescrivent des médicaments et assurent le suivi des maladies chroniques. Si nécessaire, ils peuvent également prescrire des examens complémentaires tels que des analyses de sang ou des imageries médicales, et orientent leurs patients vers des médecins spécialistes et/ou des professionnels paramédicaux tels que des infirmiers, des kinésithérapeutes ou des orthophonistes.

- Les consultations spécialisées (avec les médecins spécialistes) : contrairement à la consultation générale, la consultation spécialisée est une consultation qui est organisée en fonction de la demande des patients sur une spécialité de la médecine. Cette dernière pourrait être :
  - Médecine : anesthésiologie, cardiologie, dermatologie, endocrinologie, gastro-entérologie, génétique médicale, gériatrie, hématologie, immunologie clinique et allergie, médecine interne, médecine d'urgence, néphrologie, neurologie, oncologie médicale, pédiatrie, pneumologie, psychiatrie, rhumatologie.
  - Chirurgie : chirurgie cardiaque, chirurgie générale, chirurgie orthopédique, chirurgie plastique, neurochirurgie, obstétrique-gynécologie, ophtalmologie, oto-rhino-laryngologie, urologie.
  - Radiologie : médecine nucléaire, radiologie diagnostique, radio-oncologie
  - Laboratoire : anatomopathologie, biochimie médicale, microbiologie médicale et infectiologie.
- Les consultations sociales (avec la médecine sociale) : elle vise à fournir des conseils et une orientation en matière sociale aux patients qui en ont besoin, en mettant l'accent sur ceux qui se trouvent dans une situation sociale précaire.

Ils peuvent également signer des contrats de vacation avec les enseignants de l'UFR Santé qui passeront une ou deux fois dans la semaine (ou au besoin) assurer les consultations au niveau du Centre Médico-social.

En plus de ces services qui sont offerts à tous les patients, elle offre aussi d'autres services qui ne concernent que le personnel de l'Université. Ces services sont :

- Visite d'embauche : tout salarié fait l'objet d'un examen médical avant l'embauche ou, au plus tard, avant l'expiration de la période d'essai qui suit son embauche. L'examen médical a pour but :
  - de s'assurer que le travailleur soit médicalement apte au poste de travail auquel le chef d'établissement envisage de l'affecter ;
  - de rechercher si le salarié n'est pas atteint d'une affection dangereuse pour les autres travailleurs ;
  - de proposer éventuellement les adaptations du poste ou l'affecter à d'autres postes.

- Visite médicale annuelle : tout salarié doit obligatoirement bénéficier d'un examen médical au moins une fois par an, en vue de s'assurer du maintien de son aptitude au poste de travail occupé.
- Surveillance médicale spéciale (Visite périodique spécialisée) : surveillance médicale particulière (les handicapés, les femmes enceintes, les mères d'enfants de moins de deux ans, les travailleurs âgés de moins de dix-huit ans, les salariés affectés à certains travaux comportant des exigences ou des risques spéciaux…).
- Visite de reprise : après une absence pour raison de maladie professionnelle ou d'accident du travail, après un congé de maternité, après une absence d'au moins vingt et un jours pour cause de maladie ou d'accident non professionnel ou en cas d'absences répétées pour raisons médicales, le salarié est soumis à un examen par le médecin.
- Prévention et hygiène : pour promouvoir la prévention et l'hygiène sanitaire au sein de l'institution, en rapport avec l'activité professionnelle ;

I.1.2.4. Le Dossier Médical

Dans ces activés (de la DMT), pour chaque patient rencontré pour l'un ou les services cités ci-dessus (***section I.1.2.3. Fonctionnement de le DMT***), toutes les informations formelles qui ont joué un rôle dans l'élaboration et le suivi du diagnostic et du traitement ou de l'action de prévention, ainsi que des résultats d'examens, des comptes rendus de consultations ou de visites, d'intervention, d'exploration ou d'hospitalisation, des protocoles et prescriptions thérapeutiques mis en place, des feuilles de surveillance, des dossiers de suivi et des échanges écrits entre professionnels de santé sont consignés sur des fiches en **papier** qui sont appelés communément le **Dossier médical (DM)** du patient. Il est créé dès le premier passage du patient à la DMT et est rempli au fur et à mesure des autres passages.

En effet tout médecin hospitalier est concerné par la tenue de ce dossier, il doit y consigner toutes ses observations (traitements, notes de suite, avis de spécialistes, différents examens, …), ses interventions et les hypothèses qu'il formule en conclusion. Il est un élément très important du domaine des soins de santé. Il permet aux professionnels de santé de gérer et d'évaluer les soins à fournir à leurs patients.

Cependant, l'accès au dossier médical peut être demandé auprès du professionnel de santé ou de l'établissement de santé, par la personne concernée, son ayant droit en cas de décès de cette personne, le titulaire de l'autorité parentale, le tuteur ou le médecin désigné comme intermédiaire. Autrement dit, l'accès à un dossier médical n'est autorisé que pour le patient, le représentant légal et les médecins concernés [1].

I.2. Description du sujet

I.2.1. Problématique

La DMT de l'Université Assane Seck de Ziguinchor a comme principal objectif la prévention de l'intégrité physique des travailleurs (le personnel) contre toutes sortes d'atteinte et l'adaptation de leurs conditions de travail mais aussi d'assurer des prestations médicales pour les populations environnantes (les patients externes).

Avec la croissance rapide de l'Université et des populations environnantes qui influe sur l'augmentation des patients, la gestion des dossiers médicaux sous format papier devient de plus en plus difficile.

Cependant, la DMT a toujours utilisé le DM en la format classique (papier) pour la collecte d'informations d'ordre personnel, professionnel, médical et social. En conséquence, le DM, sous cette forme, présente de nombreuses limites qui peuvent entraver la mission de la DMT. Voici quelques-unes des limites associées aux dossiers médicaux papier :

- Accessibilité limitée : les dossiers médicaux papier peuvent être conservés dans des lieux physiques éloignés, ce qui rend leur accès difficile. Le partage rapide d'informations médicales entre différents services ou professionnels de la santé peut être compliqué ;
- Perte de Données : les dossiers médicaux papier présentent des vulnérabilités face à la perte, aux dommages ou au vol. En cas de catastrophe naturelle, d'incendie ou d'autres incidents, les informations médicales risquent d'être définitivement perdues ;
- Temps de recherche élevé : la recherche d'informations spécifiques dans un dossier médical papier peut prendre beaucoup de temps. Les professionnels de la santé peuvent passer un temps considérable à parcourir les pages afin de trouver les données pertinentes ;
- Espace de stockage physique : les dossiers médicaux papier prennent beaucoup de place physique, ce qui peut poser des problèmes d'espace dans les établissements de santé. De plus, stocker de grandes quantités de dossiers peut être coûteux ;
- Difficulté de Mise à Jour : les dossiers médicaux papier peuvent être difficiles à mettre à jour. Les entrées manuelles peuvent causer des erreurs ou des omissions lorsqu'il s'agit d'ajouter, de modifier ou d'annoter quelque chose ;

- Confidentialité : il existe toujours un risque de violation de la confidentialité des dossiers médicaux papier, malgré le fait que ces dossiers sont généralement stockés dans des endroits sécurisés. Les personnes non autorisées peuvent avoir accès à des données sensibles ;
- Difficulté de partage interinstitutionnel : le partage de leurs dossiers médicaux peut être difficile lorsque les patients doivent consulter des professionnels de santé dans différents endroits ou établissements ;
- Inefficacité des processus : le traitement des rendez-vous, des ordonnances et d'autres tâches administratives peut prendre plus de temps ;
- Le manque d'efficacité des procédures : les dossiers médicaux papier manquent souvent de ces fonctionnalités, ce qui peut entraîner un suivi préventif inadéquat ;
- Difficulté de faire les bilans annuels : En effet, avec les dossiers médicaux sous format papier, il est difficile d'avoir les statistiques annuelles telles que le nombre de patients consultés par année, les différentes maladies diagnostiqués dans l'année, le personnel ayant fait la visite annuelle, etc ;
- Il est difficile voire impossible d'avoir des données à destination pédagogique qui pourraient aider les chercheurs dans ce domaine sous cette forme.

Par conséquent, une direction comme celle-ci a besoin de s'acquérir de moyens plus performants et rapides améliorant pleinement sa gestion dans sa globalité. Aujourd'hui l'outil informatique permet de mettre en place une solution qui peut résoudre l'ensemble des problèmes cités plus haut

I.2.2 Objectifs

L'objectif général de ce stage était de mettre en place une application de gestion de dossier médical des patients pour le compte de la **DMT** de l'**UASZ**. Ce système devrait permettre de centraliser et de coordonner les activités au sein de la DMT pour une meilleure gestion de la santé des patients. Il permettra aux utilisateurs d'accéder aux fonctionnalités suivantes :

- enregistrer, afficher et modifier des patients externes : cette fonctionnalité leur permettra de gérer les patients externes. Pour les autres patients c'est la Direction des Ressources Humaines (DRH) qui s'occupe de leurs enregistrements et de leurs modifications ;

- visualiser pour tous les patients ;
- enregistrer des données médicaux : Aujourd'hui, avec l'explosion des connaissances scientifiques, cette fonctionnalité leur permettra d'enregistrer les nouvelles découvertes comme de nouvelles formes d'examens médicaux, d'examens d'appareils, d'analyses, de médicaments ou de nouvelles maladies, … ;
- gérer de dossier médical : Avec ce module ils pourront ajouter, modifier ou archiver des données médicales des patients ;
- Simplifier l'accès à un dossier médical en cherchant simplement à partir d'une de ses caractéristiques (comme par exemple un numéro de téléphone, un email, un CNI, etc.) ;
- avoir un tableau de bord : ce module devrait permettre d'accéder aux statistiques telles que :
  - Le nombre de patients consultés par mois, par trimestre, par année ;
  - Le nombre de patients internes consultés par service, par spécialité ;
  - Le nombre de visites d'embauche faites par année ;
  - Le nombre de personnels ayant fait leur visite annuelle durant une année ;
  - Le nombre de personnels par service ayant fait leur visite annuelle durant une année ;
  - ….

Conclusion

Dans ce premier chapitre, nous avons présenté notre structure d'accueil qui est la DISI, le projet de gestion intégrée (PGI) dans lequel nous avons travaillé et la DMT à laquelle l'application est destinée. Nous avons aussi exposé la problématique et les objectifs de ce travail.

Atteindre ces objectifs nécessite une méthodologie de développement appropriée. Dans le chapitre qui suit, nous allons revenir sur les processus de développement et ensuite décrire le processus de développement que nous avons adopté dans ce travail.

Chapitre II : Processus de conception et développement

Introduction

Le processus de développement d'une application est une série organisée d'étapes permettant de concevoir, créer, tester et déployer des applications logicielles.

Chaque étape du processus de développement d'une application peut être complexe et nécessité des compétences techniques spécifiques. En fonction des besoins du projet et de l'équipe de développement, il est important de suivre une méthodologie de développement appropriée telle qu'**Agile**, **DevOps**, **Lean**, **en Cascade** ou autres.

Ces méthodologies de développement logiciel correspondent le mieux à toute entreprise qui permet en substance de réduire au mieux les risques et les coûts. Il existe à ce titre, une série de méthodologies dont l'approche de chacune se distingue de l'autre à plus d'un égard. Toutefois, toutes les méthodologies déclinent un dénominateur commun à savoir : aider les équipes de développeurs à créer des **logiciels** d'une qualité élevée dans un délai bref et un coût réduit [2].

Dans ce chapitre nous allons essayer de faire une brève présentation de quelques méthodologies de développement de logiciel avant de venir sur les critères qui ont justifié notre choix de méthode.

II.1. Méthode en cascade

II.1.1. Définition

La méthode cascade, également connue sous le nom de méthode **waterfall** en anglais, est une technique de gestion de projet connue sous le nom de méthodologie linéaire séquentielle. Cette méthode divise le projet en plusieurs étapes distinctes, successives et interdépendantes les unes des autres.

En effet, une étape ne peut commencer qu'après la conclusion et l'approbation de l'étape précédente.

II.1.2. Les phases clés de la méthode en cascade

Le cycle de vie d'un projet en cascade comprend sept (07) phases : planification, analyse, conception, mise en œuvre, tests, déploiement et maintenance.

1. La phase planification :
   La phase de planification est la phase où toutes les parties prenantes du projet doivent déterminer leurs besoins. À l'issue de cette phase, est conçu un document appelé « document des exigences du projet » qui récapitule les étapes du projet, les personnes attitrées à chaque étape, les ressources et le temps nécessaire à chacune d'elles, etc.
2. La phase d'analyse :
   Cette phase consiste à examiner la liste des exigences et des besoins précédemment spécifiés afin d'élaborer un cahier de charges.
3. La phase de conception :
   Elle correspond au stade de conception du projet. Un modèle du produit est conçu en considérant l'analyse produite lors de l'étape deux (02).
4. La phase de mise en œuvre :
   Dans cette phase, c'est à l'équipe de développeurs de jouer en passant à la réalisation concrète du produit. Dans le cadre du développement logiciel, nous pouvons parler communément de codage ou de programmation.
5. La phase de tests :
   Elle correspond à la phase de vérification (test) et de correction, à la validation de sa conformité aux exigences initiales.
6. La phase de déploiement :
   Il s'agit de l'étape où le produit est enfin mis en service auprès de l'utilisateur final ou livré auprès du client final.
7. La phase de maintenance :
   Il s'agit des réparations ou des améliorations du produit. C'est pourquoi une phase de maintenance s'avère essentielle tout au long de la vie d'un produit.

La figure ci-après (***figure 3***) représente les phases de la méthode en cascade.

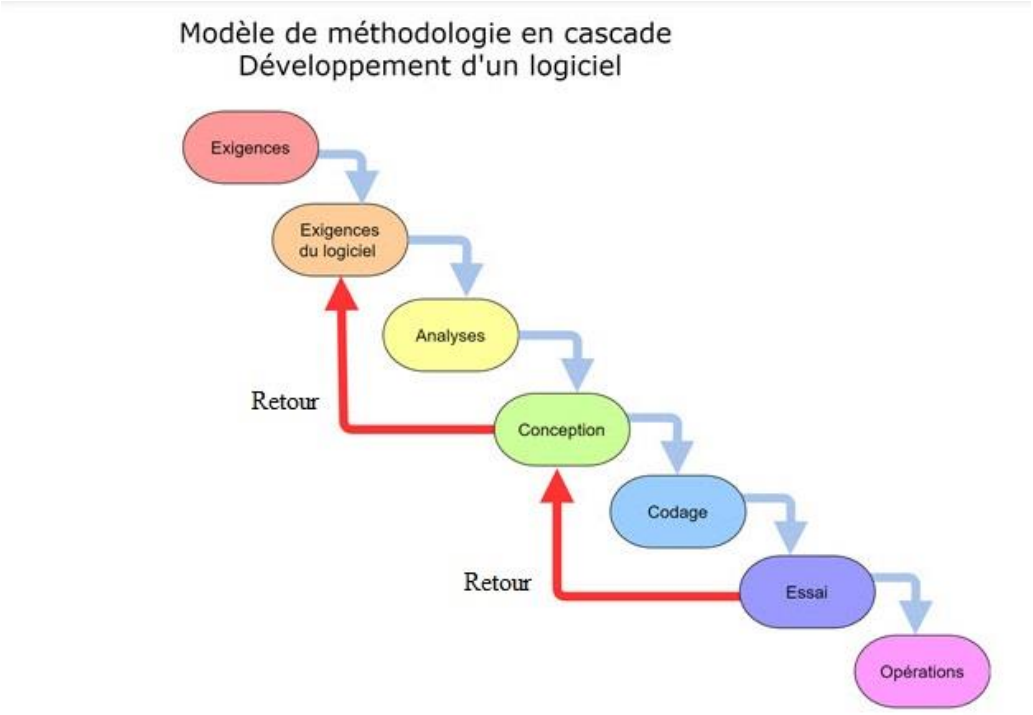

Figure 3 : Exemple de modèle en cascade [https://gitmind.com/fr/methodologie-en-cascade.html]

II.1.3. Avantages et Inconvénients

Comme beaucoup d'autres méthodes, la méthode cascade présente des avantages et des inconvénients [3].

Parmi les avantages nous pouvons citer :

- une planification structurée ;
- une documentation détaillée ;
- un suivi facile ;
- …

Et pour les inconvénients :

- absence de place pour les imprévus ;
- impossibilité de retours en arrière ;
- contrôle qualité tardif ;
- …

En résumé, les projets dont l'objectif final est bien ciblé dès le départ, les exigences stables avec une maîtrise parfaite de la mise en œuvre sont particulièrement adaptées au modèle de développement de projets en cascade.

II.2. Méthode ou concept Agile

II.2.1. Présentation de la méthode Agile

C'est une méthode ou concept de gestion de projets et de développement qui est très utilisée au sein des agences de marketing et des entreprises technologiques. L'objectif de la méthode agile est de développer des produits plus rapidement, à moindre coût, et avec un taux de réussite et de satisfaction plus important. Sa spécialité est son approche par itération. Ce sont des cycles de développement courts et très ciblés qui impliquent le client et favorisent la collaboration entre des équipes de compétences différentes [4].

II.2.2. Le principe de la méthode Agile

La méthode Agile peut prendre diverses formes et utiliser un vocabulaire spécifique. Nous présentons ici certaines variantes de méthode AGILE.

II.2.2.1. Test driven development (TDD)

Le test driven development ou développement piloté par des tests est une méthode de développement qui associe l'écriture des tests unitaires, la programmation et la revue du code.

Avant de commencer à écrire le code, des tests unitaires sont rédigés pour décrire les différentes fonctionnalités. Le développeur écrit ensuite le code le plus simple possible pour que les tests passent, mais il est prévisible que le code échoue initialement. Au fur et à mesure de l'avancement du développement, le code source doit être simplifié autant que nécessaire tout en continuant à passer les tests. Pendant tout le processus de développement, de nouveaux tests sont ajoutés et exécutés automatiquement de manière très fréquente.

Le TDD est toujours associé à des outils d'automatisation de tests unitaires [3] liés au langage de programmation utilisé.

II.2.2.2. Processus Unifié Agile (Agile UP ou AUP)

Processus Unifié Agile (Agile Unified Process en anglais) est une variante simplifiée du Rational Unified Process, également connu sous le nom de RUP [5]. Il s'agit d'une méthode de développement d'applications pour les entreprises qui utilisent les techniques agiles du TDD (développement piloté par les tests), du MDD (développement piloté par le modèle) et de la gestion du changement. Elle est divisée en quatre phases :

- lancement : délimitation du projet, définir les architectures potentielles du système, implication des acteurs et obtention du coût ;
- conception : définition de l'architecture du système ;
- réalisation : développement du logiciel lors d'un processus incrémental dans l'ordre de priorité des fonctionnalités ;
- livraison : validation et déploiement du système.

II.2.2.3. Scrum

"Scrum" est actuellement la méthode agile la plus populaire. Au rugby, ce mot signifie « mêlée ». La méthode scrum utilise des "sprint", qui sont des intervalles de temps relativement courts pouvant aller de quelques heures à un mois. Un sprint s'étend généralement sur deux semaines, de préférence. À la fin de chaque sprint, l'équipe montre ce qu'elle a fait pour le produit. Scrum regroupe trois (03) principaux acteurs [6] :

1. Le **Product Owner** ou « Directeur de produit » : pour assurer une cohérence, il communique les objectifs premiers des clients et des utilisateurs finaux, coordonne l'implication des utilisateurs et des parties prenantes et se coordonne lui-même avec les autres propriétaires de produits.
2. Le **Scrum Master :** membre de l'équipe, il cherche à maximiser la capacité de production de l'équipe. Pour ce faire, le scrum master aide l'équipe à travailler de manière autonome tout en s'améliorant.
3. L'**équipe opérationnelle** (qui regroupe idéalement moins de dix (10) personnes) : La particularité d'une équipe scrum est qu'elle n'a aucune hiérarchie interne. Une équipe de scrum est autonome.

On retrouve d'autres termes importants à connaître pour comprendre la méthode scrum :

- **Le product backlog** (carnet du produit) : Ce document contient les exigences initiales qui ont été dressées avec le client au début du projet et classées. Cependant, il évolue en fonction des différents besoins du client tout au long du projet.
- **Le sprint backlog** (carnet de sprint) : L'équipe fixe un but à chaque début de sprint. Ensuite, lors de la réunion de sprint, l'équipe de développement choisit les éléments du carnet à réaliser. Le sprint backlog est alors constitué par l'ensemble de ces éléments.
- **User story** : ceux sont les fonctionnalités décrites par le client.

- **La mêlée** (scrum) : c'est une réunion d'avancement organisée de manière quotidienne durant le sprint.

La figure ci-dessous (***figure 4***) présent les acteurs de la méthode Scrum

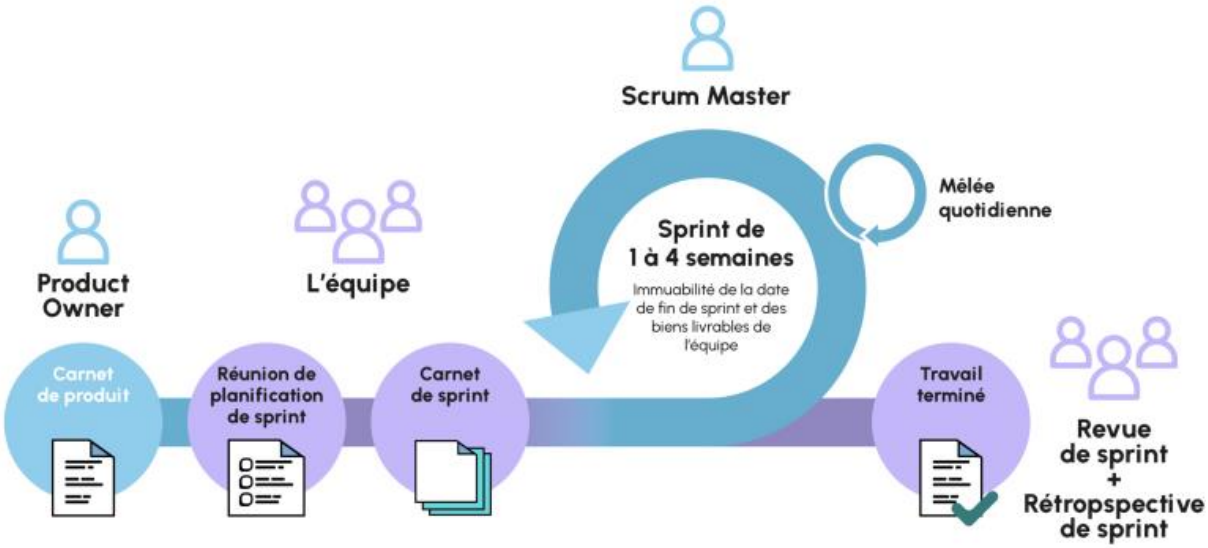

Figure 4 : Méthode Scrum Agile [7]

II.2.2.4. Programmation Extrême (XP)

La méthode Programmation Extrême (Extreme Programming en Anglais) utilise des itérations de développement très courtes et une collaboration étroite avec tous les acteurs du projet. La planification des tâches est souple et l'estimation des charges est simple. Les fonctionnalités sont livrées régulièrement pour être testées et validées à l'aide de prototypes opérationnels, garantissant ainsi la correspondance entre les attentes du client et les réalisations.

II.2.2.5. Lean Software Development

Le Lean Software Development ou le développement de logiciel Lean est basé sur sept (07) grands principes.

- éliminer les gaspillages (finition partielle, processus inutiles, fonctionnalités non nécessaires, modification de l'équipe, retards, …) ;
- favoriser l'apprentissage (multiplication des sources d'apprentissage, synchronisation des équipes , …) ;
- reporter les décisions (jusqu'au dernier moment raisonnable, pour éviter de longues discussions sources de pertes de temps et les décisions irrévocables) ;

- livrer vite (livraisons rapides et régulières de façon à avoir un retour client rapide également) ;
- responsabiliser l'équipe (favoriser l'autonomie et le leadership des équipes),
- construire la qualité (elle doit être placée au cœur du projet de la conception à la réalisation) ;
- optimiser le système dans son ensemble (mise en place de mesures de performances complètes, et gérer les différentes interactions et dépendances).

Avec la méthode Lean [8], la qualité est réellement placée au cœur de la gestion du projet, en optimisant notamment l'ensemble des processus d'apprentissage, de décision, de livraison et de mesure de performances.

II.2.3. Avantages et inconvénients de la méthodologie

Même si la méthode Agile fait partie des méthodes les plus utilisées aujourd'hui et qu'elle a beaucoup d'avantages, elle présente aussi certains inconvénients [9].

II.2.3.1. Les avantages

- L'avantage majeur de l'approche Agile relève de sa flexibilité. L'équipe projet réagit rapidement aux changements du client et aux imprévus.
- Autre atout : la collaboration et la communication fréquente avec le client, ainsi que sa forte implication dans le projet. Une relation de confiance se tisse. Le client dispose ainsi d'une meilleure visibilité sur l'avancement du projet. Il peut donc l'ajuster en fonction de ses besoins. Le contrôle qualité est permanent.
- Enfin, vous contrôlez mieux les coûts du projet. À la fin de chaque étape, vous connaissez le budget déjà dépensé et celui restant. Vous pouvez ainsi décider de le poursuivre, ou de l'arrêter si les fonds sont insuffisants.

II.2.3.2. Les inconvénients

- Comme le dialogue est privilégié, la méthode Agile laisse peu de place à la documentation. Ça peut poser problème en cas de changement d'équipe projet par exemple.
- Le client doit par ailleurs rester disponible et s'intéresser à son projet afin de s'assurer qu'il répond parfaitement à ses besoins. Tous n'ont pas le temps, ni l'envie de s'impliquer pleinement dans la réalisation d'un projet.

- La méthode Agile n'est en outre pas adaptée pour les entreprises avec une structure hiérarchique très forte, à cause de son fonctionnement collaboratif.
- Par ailleurs, si cette approche permet un bon contrôle des coûts, elle rend très difficile la vision d'un budget pour la totalité du projet. La flexibilité a un coût, que le client doit être prêt à payer.

En résumé, l'approche Agile offre une plus grande flexibilité et une meilleure visibilité dans la gestion du projet. À notre époque où la personnalisation est importante, cette méthodologie fait de plus en plus d'adeptes.

C'est aussi une approche qui se plie bien aux besoins des projets qui doivent aboutir rapidement, ou dont les domaines de déploiement évoluent vite. Un projet qui a de grandes chances d'évoluer en chemin trouve la structure dont il a besoin avec cette démarche souple. Elle autorise les ajustements et les optimisations, sans freiner l'organisation des équipes.

II.3. L'approche DevOps

II.3.1. Définition

DevOps est une méthodologie de développement logiciel qui vise à combler le fossé traditionnel entre les équipes de développement (Dev) et les équipes d'exploitation (Ops). L'objectif principal de DevOps est d'augmenter l'efficacité, la collaboration et la rapidité du développement et du déploiement de logiciels tout en garantissant la stabilité, qualité et la fiabilité des applications en production [10].

II.3.2. Les principes de l'approche :

Pour exploiter le plein potentiel de DevOps, les équipes doivent suivre les principes clés du DevOps. Parmi ces principes nous avons [11] :

- Automatisation : c'est au cœur de DevOps. Cela comprend l'automatisation des processus de développement, de tests, de déploiement et de gestion de l'infrastructure. Les outils d'automatisation aident à réduire les erreurs humaines et à accélérer la livraison des logiciels ;
- Collaboration : cela favorise une meilleure compréhension des besoins de chacun et permet de résoudre les problèmes plus rapidement ;
- Intégration continue (CI) : à chaque intégration, des tests automatisés sont exécutés pour garantir que les modifications intégrées fréquemment du code restent fonctionnelles ;

- Livraison continue (CD) : la collaboration de la CD et la CI permet de publier plus fréquemment des mises à jour de l'application ;
- Infrastructure as Code (IaC) : l'IaC consiste à gérer l'infrastructure de manière programmable, ce qui facilite la reproductibilité, la mise à l'échelle et la gestion de l'infrastructure ;
- Surveillance et retour d'information : DevOps met l'accent sur la surveillance en temps réel des applications en production. Les données collectées aident à identifier les problèmes rapidement et à améliorer les performances ;
- Sécurité : les équipes travaillent ensemble pour identifier et résoudre les vulnérabilités dès le stade du développement, dès le début du processus ;
- Evolutivité : DevOps permet de créer des applications évolutives, capables de s'adapter aux besoins changeants de l'entreprise.

La figure ci-dessous (**figure 5**) que présente les phases de l'approche DevOps

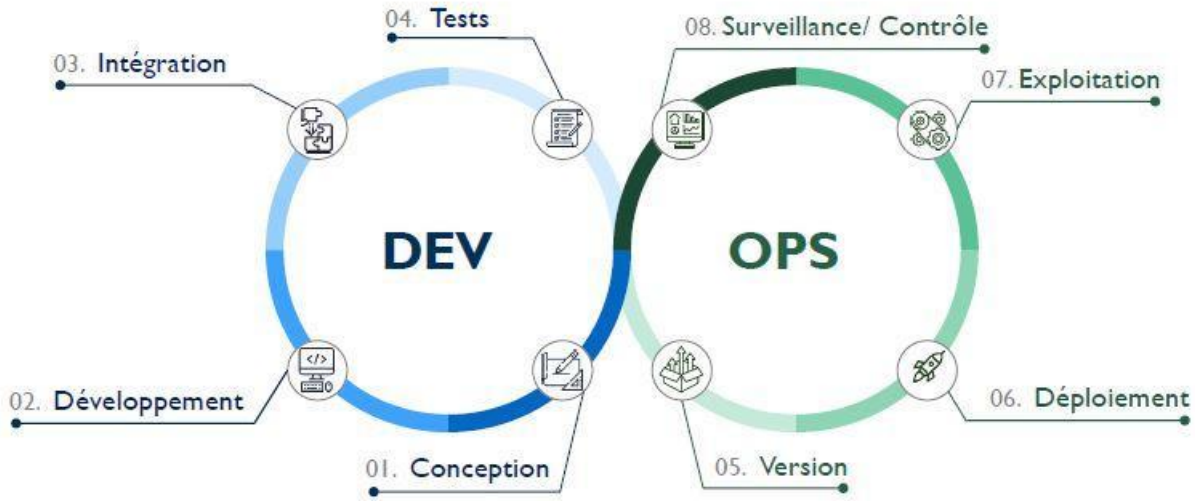

Figure 5 : Méthode DevOps [12]

II.3.3. Outils et technologies qu'utilise le DevOps

Le choix des outils dépend quasiment des préférences de l'entreprise ou de l'organisation.

Parmi les outils les plus utilisés nous pouvons citer :

- Jenkins : c'est un serveur d'automatisation open source autonome qui peut être utilisé pour automatiser toutes sortes de tâches liées à la création, au test et à la livraison ou au déploiement de logiciels [13].

- Docker : c'est une plateforme ouverte pour développer, expédier et exécuter des applications. Docker vous permet de séparer vos applications de votre infrastructure afin que vous puissiez fournir des logiciels rapidement. Avec Docker, vous pouvez gérer votre infrastructure de la même manière que vous gérez vos applications [14].
- Kubernetes [15] : c'est un moteur d'orchestration de conteneur open source permettant d'automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Le projet open source est hébergé par la Cloud Native Computing Foundation [16].
- Git Il s'agit d'un outil de développement collaboratif qui aide une équipe de développeurs à gérer les changements apportés au code source au fil du temps. Les logiciels de contrôle de version gardent une trace de chaque changement apporté au code dans un type spécial de base de données [17].
- Etc.

II.3.4. Méthodes du DevOps

Pour accélérer et améliorer le développement et le lancement de leurs produits, les entreprises peuvent utiliser diverses méthodes et pratiques de développement de logiciels DevOps [20]. Les méthodes Scrum, Kanban et Agile sont les plus couramment utilisées.

La méthode Scrum définit la manière dont les membres de l'équipe doivent collaborer pour accélérer les projets de développement et d'assurance qualité. Les pratiques Scrum utilisent des workflows clés, une terminologie spécifique et des rôles désignés.

III.3.5. Avantages de DevOps

Pour ceux qui suivent les pratiques DevOps, les avantages commerciaux et techniques sont évidents et la plupart contribuent à améliorer la satisfaction des clients :

- Accélération et amélioration la livraison des produits ;
- Résolution des problèmes plus rapidement et réduction de la complexité ;
- Production d'une plus grande évolutivité et une disponibilité sans précédent ;
- Augmentation la stabilité de l'environnement d'exploitation ;
- Meilleure utilisation des ressources ;
- Automatisation accrue ;
- Meilleure visibilité sur les résultats du système ;

- Innovation renforcée.

III.3.5. Inconvénients de DevOps

Bien que DevOps présente de nombreux avantages en termes d'efficacité, de collaboration et de rapidité de déploiement, il présente également des défis et des inconvénients potentiels. Voici quelques-uns des inconvénients fréquemment évoqués :

- Complexité initiale : DevOps peut être difficile à mettre en place, surtout pour les organisations qui passent d'un modèle de développement conventionnel à celui de DevOps. Il peut prendre du temps et de la formation pour intégrer de nouvelles pratiques, outils et processus ;
- Investissement en ressources : La mise en place d'une infrastructure et de processus DevOps peut nécessiter un investissement initial considérable en temps, en ressources humaines et en ressources financières ;
- Changement culturel : DevOps nécessite fréquemment un changement culturel dans l'organisation. Les équipes de développement et d'exploitation doivent travailler ensemble, sinon des obstacles au changement peuvent survenir.

En résumé, de nombreuses méthodes DevOps destinées à rationaliser le développement et le déploiement des logiciels reposent sur des modèles agiles tels que la programmation Lean.

L'évolution du DevOps trouve son origine dans plusieurs mouvements de la volonté d'harmoniser les activités des développeurs et les équipes chargées des opérations.

II.4. Adoption de la méthode Scrum Agile

La méthode agile nécessite une forte implication du client et des échanges fréquents avec les utilisateurs finaux. Nous avons trouvé que la collaboration étroite avec le client était nécessaire dès le début du projet. Les profils des utilisateurs de la Direction de la Médecine du Travail de l'Université étaient différents, et il était nécessaire de prendre en compte des données à traiter dans cette application. En raison de la diversité et des perspectives qui ont émergé lors des premières rencontres, la nécessité d'utiliser l'approche AGILE est apparue très tôt. C'est dans ce cadre que nous avons décidé de choisir SCRUM Agile comme méthode de travail.

Le chef du SST et le directur du DMT ont joué le rôle de Product Owner. L'équipe de développement à la DISI était formée de quatre membres : le chef d'équipe jouait le rôle de Scrum Master et moi le développeur.

Conclusion

Dans ce chapitre, nous avons présenté un certain nombre de méthodes et/ou processus de développement. Nous avons opté pour les méthodologies agiles en particulier Scrum (qui sera adopté tout au long de ce travail) en raison de la forte implication du client dans le développement.

Même si le choix d'une méthode adéquate est important, la communication et la collaboration entre les membres de l'équipe sont également essentielles à la réussite d'un projet.

Dans le prochain chapitre, nous aborderons la spécification des besoins.

Chapitre III : Spécification des besoins

Introduction

Ce chapitre couvre la spécification des besoins. Tout d'abord, nous mettons en avant les utilisateurs que nous avons identifiés en divisant le système en sous-modules pour faciliter notre travail. Ensuite, nous présentons l'outil et le langage de modélisation et enfin, nous attaquons aux exigences fonctionnelles de l'application identifiées et exprimées à l'aide de diagrammes de cas d'utilisation. Ce dernier (le diagramme de cas d'utilisation) fournit un aperçu complet du comportement fonctionnel de l'application.

III.1. Identification des acteurs du système

III.1.1. Définition d'un acteur de système

Un acteur dans un système, qu'il s'agisse d'un système informatique, social, économique, ou tout autre type de système, est une entité ou une composante qui interagit avec ce système d'une manière ou d'une autre. Les acteurs peuvent être des individus, des groupes, des organisations, des entités ou des éléments qui ont un rôle ou une influence sur le fonctionnement, les performances ou les résultats du système.

Pour un système informatique, les acteurs peuvent être des utilisateurs, des administrateurs, des applications externes, ou même des périphériques matériels, car ils ont des interactions avec le système.

Cependant, comprendre les acteurs d'un système est crucial pour analyser son fonctionnement, identifier les relations et les dépendances, et concevoir des solutions efficaces. Les acteurs peuvent avoir des rôles variés, des besoins différents, et des niveaux d'influence distincts sur le système, ce qui les rend essentiels à la compréhension globale du système.

III.1.2. Les utilisateurs du système

Notre système dispose de plusieurs acteurs qui peuvent avoir des profils différents :

- Le profil administrateur : c'est le Directeur de la DMT qui se charge de cette responsabilité ;
- Le profile assistant social ;
- Le profile médecin ;

- Le profile infirmier ;
- Le profile secrétaire.

III.2. Identification des modules du système

Nous avons divisé l'application en huit sous-modules afin de rendre au plus clair ses besoins fonctionnels :

Sous-module	**Descriptions**
Authentification	C'est le sous-module principal et le plus important, car elle permet au système de vérifier l'identité de la personne qui souhaite y accéder et ces autorisations d'accès. Sans ce dernier, nous ne pouvons pas accéder aux autres modules.
Paramétrage	Ce sous-module permet à l'administrateur de gérer les comptes, les analyses et leurs types, les examens, les médicaments et les maladies et leurs types.
Patient	Ce sous-module est composé de six (6) sous-modules, personnel, conjoint, enfant, étudiant, vacataire et externe. Ils devraient permettre d'afficher la liste de tous les patients, mais aussi d'ajouter et de modifier pour le dernier module (externe).
Visite	Ce sous-module est aussi composé de quatre (4) sous-modules, embauche, annuelle, de reprise et périodique spécialisée. Ils ne concernent que les patients internes (le personnel) et permettent de gérer les visites médicales lors de l'embauche du personnel, les visites annuelles, les visites de reprise et les visites périodiques spécialisées.
Consultation	Ce sous-module est composé de deux (2) sous-modules, médical et social et concerne tous les patients et permet de gérer les consultations médicales et les consultations sociales.
Gestion des fichiers	Dans ce sous-module, nous allons gérer la génération des bulletins d'analyse, d'examen, le certificat d'arrêt de travail et des prescriptions.
Dossier médical	Ce sous-module devrait permettre de regrouper toutes les informations personnelles et médicales de chaque patient de la DMT.
Statistique	Dans ce sous-module, nous allons gérer les statistiques comme : le nombre de consultations annuelles, le nombre de personnel ayant fait la visite annuelle.

Tableau 1 : Liste des modules et leurs descriptions

III.3. L'outil de modélisation PowerAMC

PowerAMC est l'outil de modélisation que nous avons utilisé pour la construction de nos organigrammes et nos diagrammes de cas d'utilisation ci-dessous et de tous les autres diagrammes (de classes, d'activités, de séquences, etc.) contenus dans ce document.

Dans les lignes qui suivent, nous allons le présenter avant de justifier notre choix.

III.3.1. Présentation de Power AMC

Son nom d'origine AMCDesignor, Power AMC a été créé pour compenser le manque d'outils de modélisation graphique d'Oracle. Il reposait sur la méthode Merise. Lorsque l'application a été acquise par Powersoft, elle a été traduite en anglais et le module modèle conceptuel de données a été modifié pour correspondre à la méthodologie d'ingénierie de l'information (IE, Information engineering), plus précisément selon la méthode Yourdon-DeMarco [18].

III.3.2. Justification du choix de PowerAMC

PowerAMC est un outil de modélisation qui existe sous deux versions qui ont vu le jour simultanément : PowerAMC, méthodologie Merise, application en français ;

PowerDesigner, méthodologie IE, application en anglais.

En plus de ces deux versions, PowerAMC permet de générer les diagrammes en images, de générer des modèles (voir Annexe 1) et des codes (voir Annexe 2). Il est simple à utiliser.

En Résumé, d'un point de vue technique, SAP PowerDesigner offre une très large gamme de modélisations standards et graphiques avec lesquels travailler comme la version très plébiscitée Merise mais aussi MCD, MOO, MPM, MSX, MPM, MPD, MFI, MGX, MTM et MAE, disponible en français.

III.4. Langage de modélisation UML

III.4.1. Définition

Le langage UML (Unified Modeling Language, ou langage de modélisation unifié en français) est un moyen de représenter visuellement l'architecture, la conception et la mise en œuvre de systèmes logiciels complexes.

Les diagrammes UML résolvent le problème de milliers de codes qui rendent difficile la gestion des interactions et des hiérarchies en divisant un système logiciel en composants et sous-composants [19].

III.4.2. Les types de diagramme UML

Le nombre de diagrammes d'UML est environ 13 qui sont divisés en deux groupes [19].

- Les diagrammes UML structurels :

Comme leur nom l'indique, les diagrammes UML structurels représentent la structure d'un système et les relations entre les éléments. Ils sont au nombre de six (6) : le diagramme de classes, le diagramme de composants, le diagramme de déploiement, le diagramme de structure composite, le diagramme d'objets et le diagramme de paquetages

- Les diagrammes UML comportementaux :
  Ces diagrammes UML montrent comment le système interagit et se comporte avec lui-même, les utilisateurs, les autres systèmes. Ils sont au nombre de sept (7) : le diagramme de cas d'utilisation, le diagramme de temps, le diagramme d'aperçu des interactions, le diagramme de communication, le diagramme de séquence, le diagramme d'activités et le modèle de diagramme états-transitions
- Le diagramme de profil : Les diagrammes de profil, qui ont récemment été ajoutés à UML 2.0, sont uniques et rarement utilisés dans les spécifications. Un diagramme de profil est plus largement utilisé comme un mécanisme d'extensibilité pour adapter les modèles UML à des domaines et des plateformes particuliers.

En résumé, Les diagrammes UML sont des outils de communication qui permettent de simplifier la conception et les fonctionnalités d'une application. Cependant, il est important que ces diagrammes soient compris de la même manière par tous.

III.5. Les diagrammes de cas d'utilisations

III.5.1. Définition

Un diagramme de cas d'utilisation est un outil de modélisation qui permet de représenter graphiquement les interactions entre les acteurs (utilisateurs externes) et les systèmes informatiques. Il est l'un des outils les plus utilisés dans le développement de logiciels pour définir les fonctionnalités des applications et les scénarios d'utilisation.

Dans cette section, nous allons présenter quelques diagrammes de cas d'utilisation et leurs descriptions.

III.5.2. Gestion de comptes

La figure ci-dessous (***figure 6***) est le diagramme de cas d'utilisation pour la gestion des comptes. Elle représente les interactions entre l'administrateur et le système lors de la gestion des

comptes. Le diagramme permet de visualiser les responsabilités de l'administrateur pour la gestion des comptes.

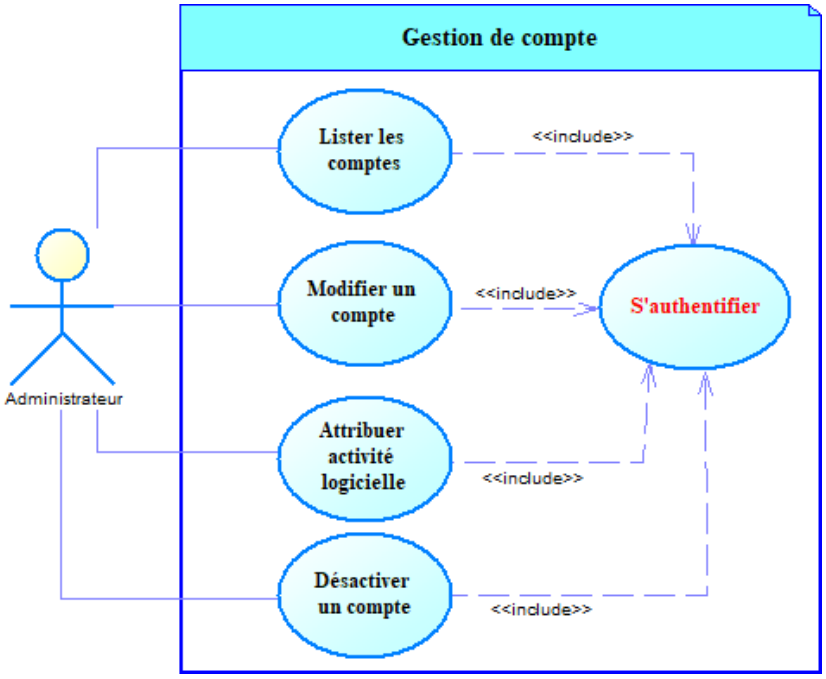

Figure 6 : Diagramme de cas d'utilisation pour la gestion des comptes

Description des cas d'utilisations de la *figure 6* :

- **Lister les comptes** : L'administrateur peut consulter la liste des comptes des utilisateur du système.
- **Modifier un compte** : L'administrateur peut modifier un compte
- **Attribuer activité logicielle** : L'administrateur peut attribuer un ou des activité(s) logiciel(s) à un compte qui permettront à l'utilisateur d'accéder à certaine fonctionnalité du système.
- **Désactiver compte** : Permet à l'administrateur de désactiver un compte à un utilisateur
- **Authentification** : Permet à l'administrateur (et les autres utilisateurs) de s'authentifier pour accéder au système

III.5.3. Gestion des patients internes et externes

La figure ci-dessous (***figure 7***) est le diagramme de cas d'utilisation pour la gestion des patients. Elle représente les interactions entre l'administrateur et le système pour la gestion des patients.

Le diagramme permet de visualiser les responsabilités de l'administrateur pour la gestion des patients.

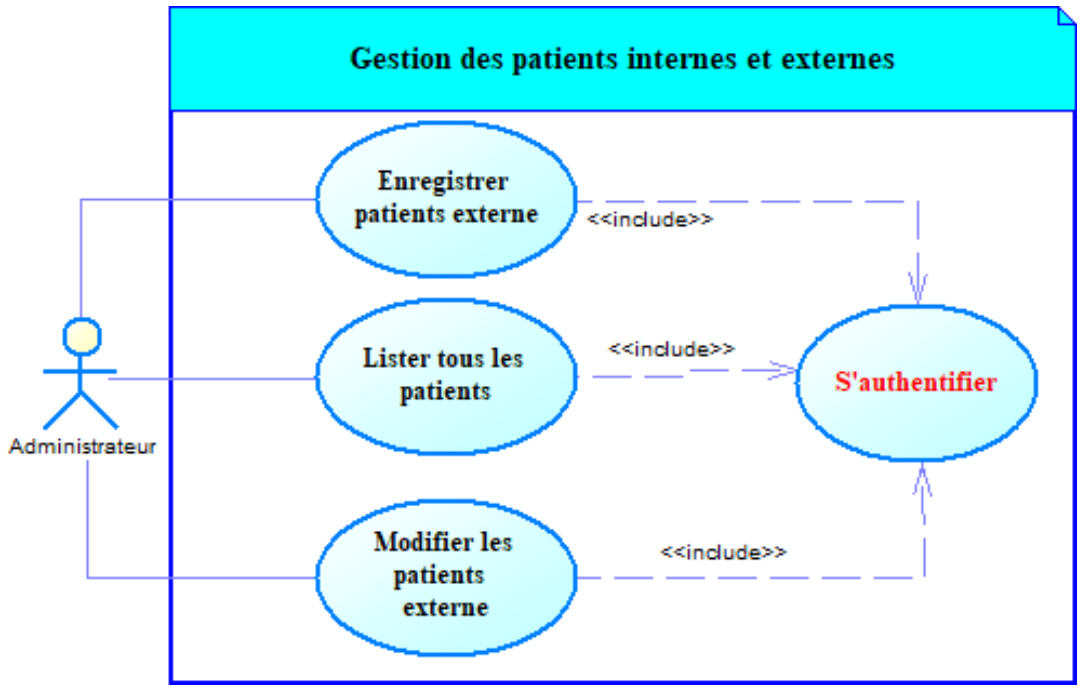

Figure 7 : Diagramme de cas d'utilisation pour la gestion des patients

Description des cas d'utilisations de la *figure 7* :

- **Enregistrer des patients externes** : L'administrateur ou personne chargé (secrétaire par exemple) d'enregistrer un patient externe (patients qui ne sont pas du personnel de l'Université ni de leur famille.
- **Lister tous les patients** : L'administrateur peut lister tous les patients (interne et externe).
- **Modifier les infos des patients externe** : L'administrateur peut modifier les informations personnelles des patients externes.

III.5.4. Gestion des consultations

La figure ci-dessous (***figure 8***) est le diagramme de cas d'utilisation pour la gestion des consultations. Elle représente les interactions entre un médecin et le système lors de la gestion des consultations. Le diagramme permet de visualiser les activités d'un médecin pour la gestion des consultations.

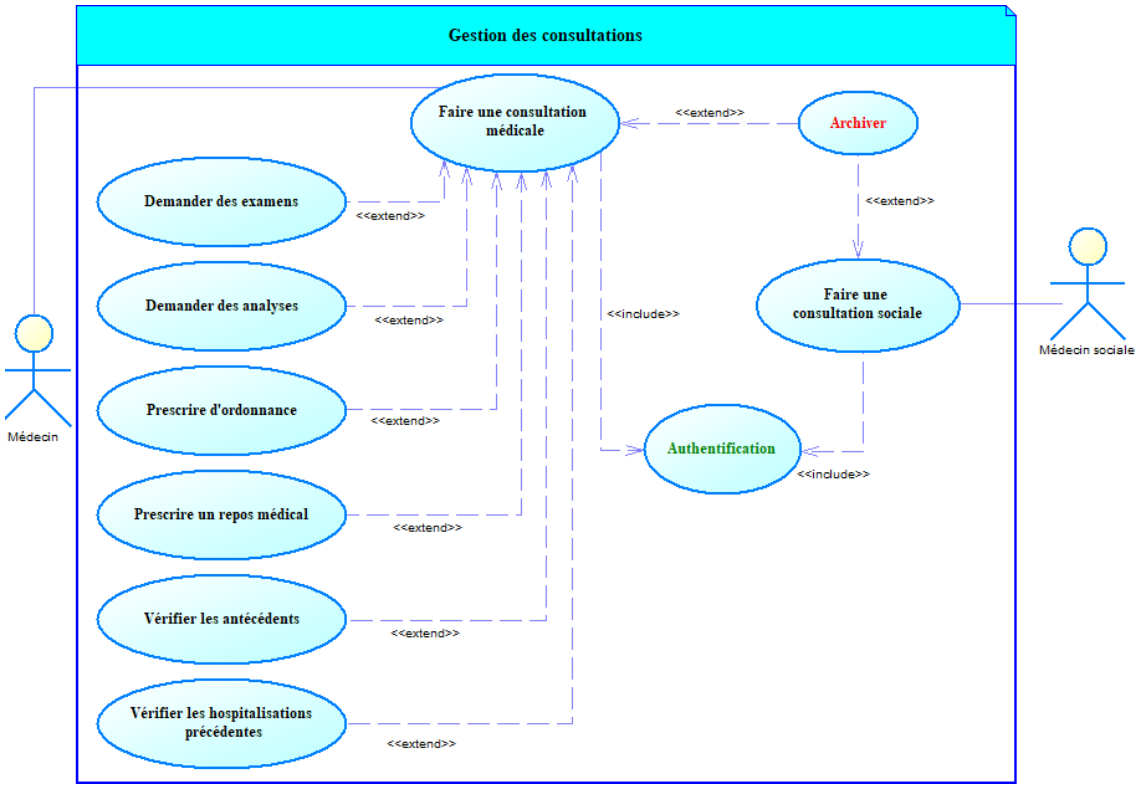

Figure 8 : Diagramme de cas d'utilisation pour la gestion des consultations

Description des cas d'utilisations de la *figure 8* :

- **Faire une consultation médicale** : Un médecin peut créer une consultation dans laquelle il peut :
  - ✓ **Demander des examens**
  - ✓ **Demander des analyses**
  - ✓ **Prescrire d'ordonnance**
  - ✓ **Prescrire un repos médical**
  - ✓ **Vérifier les antécédents** : Le médecin vérifier si le patient a des antécédents personnels et ou familiaux.
  - ✓ **Vérifier les hospitalisations précédentes** : Le médecin vérifier si le patient a été précédemment hospitaliser.
- **Faire une consultation sociale** : Un médecin social peut créer une consultation sociale.
- **Archiver** : Le médecin peut archiver une consultation.

III.5.5. Gestion des visites d'embauches

La figure ci-dessous (***figure 9***) est le diagramme de cas d'utilisation pour la gestion des visites (embauche, annuelle, de reprise et périodique spécialisée). Elle représente les interactions entre

un médecin et le système lors de la gestion des visites. Le diagramme permet de visualiser les activités d'un médecin pour la gestion des consultations.

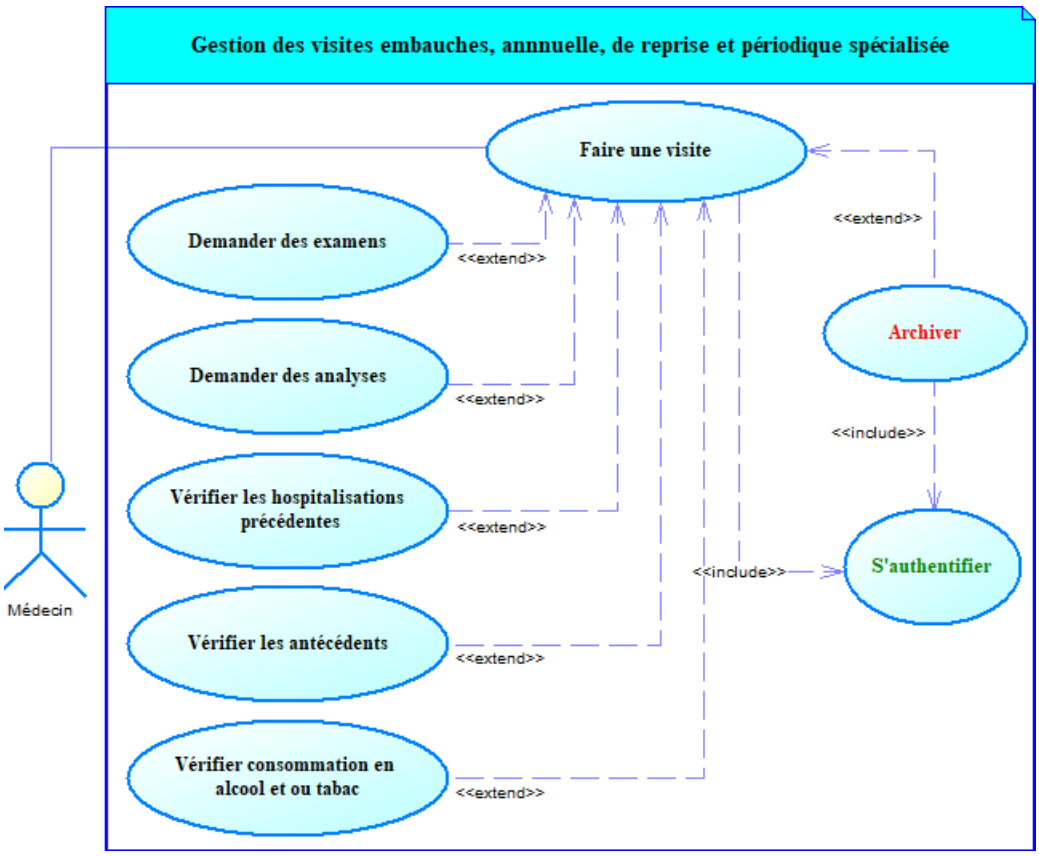

Figure 9 : Diagramme de cas d'utilisation pour la gestion des visites

Description des cas d'utilisations de la *figure 9* :

- **Faire une visite** : Un médecin peut créer une visite dans laquelle il peut :
  - ✓ **Demander des examens**
  - ✓ **Demander des analyses**
  - ✓ **Vérifier les antécédents** : Le médecin vérifier si le patient a des antécédents personnels et ou familiaux.
  - ✓ **Vérifier les hospitalisations précédentes** : Le médecin vérifier si le patient a été précédemment hospitaliser.
  - ✓ **Vérifier consommation en tabac et ou alcool** : Le médecin peut vérifier la fréquence de consommation de tabac et ou d'alcool du patient.
- **Archiver** : Le médecin peut archiver une visite.

NB : L'administrateur a la pouvoir d'attribuer certaines taches (comme par exemple la gestion des patients externes à un autre utilisateur du système.

III.6. Les modules et leurs cas d'utilisations

Le tableau suivant répertorie les cas d'utilisation de chaque sous-module.

Sous-module	**Cas d'utilisation**
Authentification	Authentification
Paramétrage	Gestion compte, Gestion analyse, Gestion type d'analyse, Gestion maladie, Gestion type maladie, Gestion médicament et Gestion d'examen
Patient	**Patient externe** : Enregistrer patient, Lister patients, Modifier les informations des patients et Créer dossier patient.
	Patient interne : Lister patients et Créer dossier patient.
Visite	Visite, Archiver, Demande d'analyse, Demande d'examen, Vérification d'antécédent et Vérification d'hospitalisation précédent.
Consultation	Consultation sociale, Consultation médicale, Archiver, Demande d'analyse, Demande d'examen, Vérification d'antécédent, Vérification d'hospitalisation précédent, Prescription d'ordonnance et Prescription de repos médical.
Gestion des fichiers	Bulletin d'analyse, Bulletin d'examen, Repos médical, Prescription
Dossier médical	**Dossier médical du personnel :** Information personnel du patient, Visite d'embauche, Visite annuelle, Visite de reprise, Visite périodique spécialisée, Consultation médicale et Consultation sociale
	Dossier médical des autres patients : Information personnel du patient, Consultation médicale et Consultation sociale
Statistique	Visualisation des statistiques

Tableau 2 : Les sous-modules et leurs cas d'utilisations

Conclusion

Dans cette section, nous avons divisé en deux groupes les acteurs de notre système, qui sont le groupe des administrateurs et le groupe des autres travailleurs du DMT. Puis nous avons regroupé les besoins fonctionnels de notre futur système en modules et sous-modules que nous avons modélisés à l'aide de diagrammes de cas d'utilisation. Enfin, nous avons fourni un tableau récapitulatif de cas d'utilisation pour chaque module et sous-module avant de présenter l'outil de modélisation et le langage de modélisation. Après avoir identifié les acteurs et leur cas d'utilisation, nous passerons à l'analyse et à la conception dans le prochain chapitre.

Chapitre IV : Analyse des besoins et Conception

Introduction

La phase d'analyse et de conception est un processus crucial qui permet de formaliser les étapes préliminaires du développement d'un système afin de le rendre plus adapté aux besoins du client.

Ce chapitre aborde l'analyse et les différents éléments de la conception, que sont la conception générale et la conception détaillée. Nous présenterons les diagrammes d'activités et de certaines diagrammes séquences dans la partie analyse, l'architecture physique que nous avons choisi pour notre application dans la conception générale et certains diagrammes de classes seront présentés dans la conception détaillée. Le diagramme de séquence décrivant le processus de sécurité pour l'accès aux données, les diagrammes de classes pour les informations personnelles et professionnelles et le dictionnaire des données sont présentés en annexe (respectivement Annexes 3, 4, 5 et 6).

IV.1. Analyse des besoins

Dans cette section, nous allons nous limiter à vous présenter deux diagrammes d'activités, l'un dédié à la gestion des comptes et l'autre à la gestion des patients externes. De plus, nous aurons le privilège de vous exposer trois diagrammes de séquences captivants : celui dédié à l'authentification, puis ceux consacrés respectivement à la gestion des consultations et celle des visites. Le diagramme de séquence décrivant le processus de sécurité pour l'accès aux données est présenté en annexe (Annexes 3).

IV.1.1 Les diagrammes d'activités

IV.1.1.1. Définition d'un diagramme d'activité

Un diagramme d'activités, sous le langage UML, décrit la séquence d'actions d'un processus pour fournir une vue du comportement d'un système. Les diagrammes d'activités sont similaires aux organigrammes de traitement de l'information, car ils montrent les flux entre les actions dans une activité. Cependant, les diagrammes d'activités peuvent également montrer des flux parallèles simultanés et des flux de remplacement. Les nœuds d'activité et les bords d'activité sont utilisés dans les diagrammes d'activités pour simuler le flux de commande et de données entre les actions [20].

IV.1.1.2. Diagramme d'activité de gestion des comptes

Pour gérer les comptes, l'utilisateur doit s'authentifier en tant qu'administrateur pour avoir accès à cette interface. La figure ci-dessous (***figure 10***) renseigne les étapes à suivre.

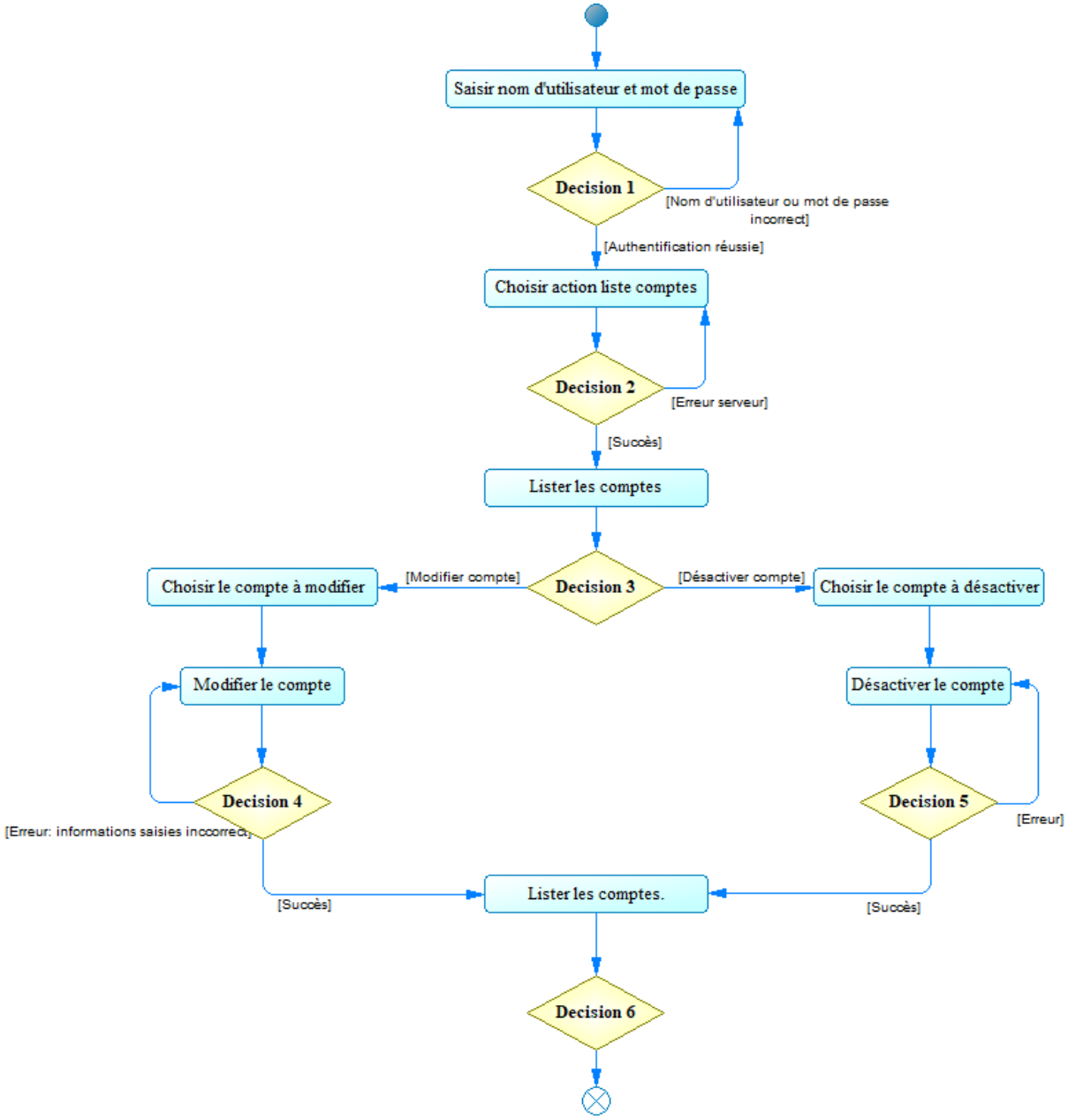

Figure 10 : Diagramme d'activités pour la gestion des comptes

La figure ci-dessus représente le diagramme d'activités permettant de visualiser les actions faites pour la gestion des comptes.

IV.1.1.3. Diagramme d'activité de gestion des patients externes

Pour la gestion des patients externes, l'utilisateur doit s'authentifier en tant qu'administrateur ou avoir les autorisations nécessaires pour avoir accès à l'interface avec les boutons d'ajout, de mise à jour et d'archivage. La figure ci-dessous (***figure 11***) renseigne les étapes à suivre.

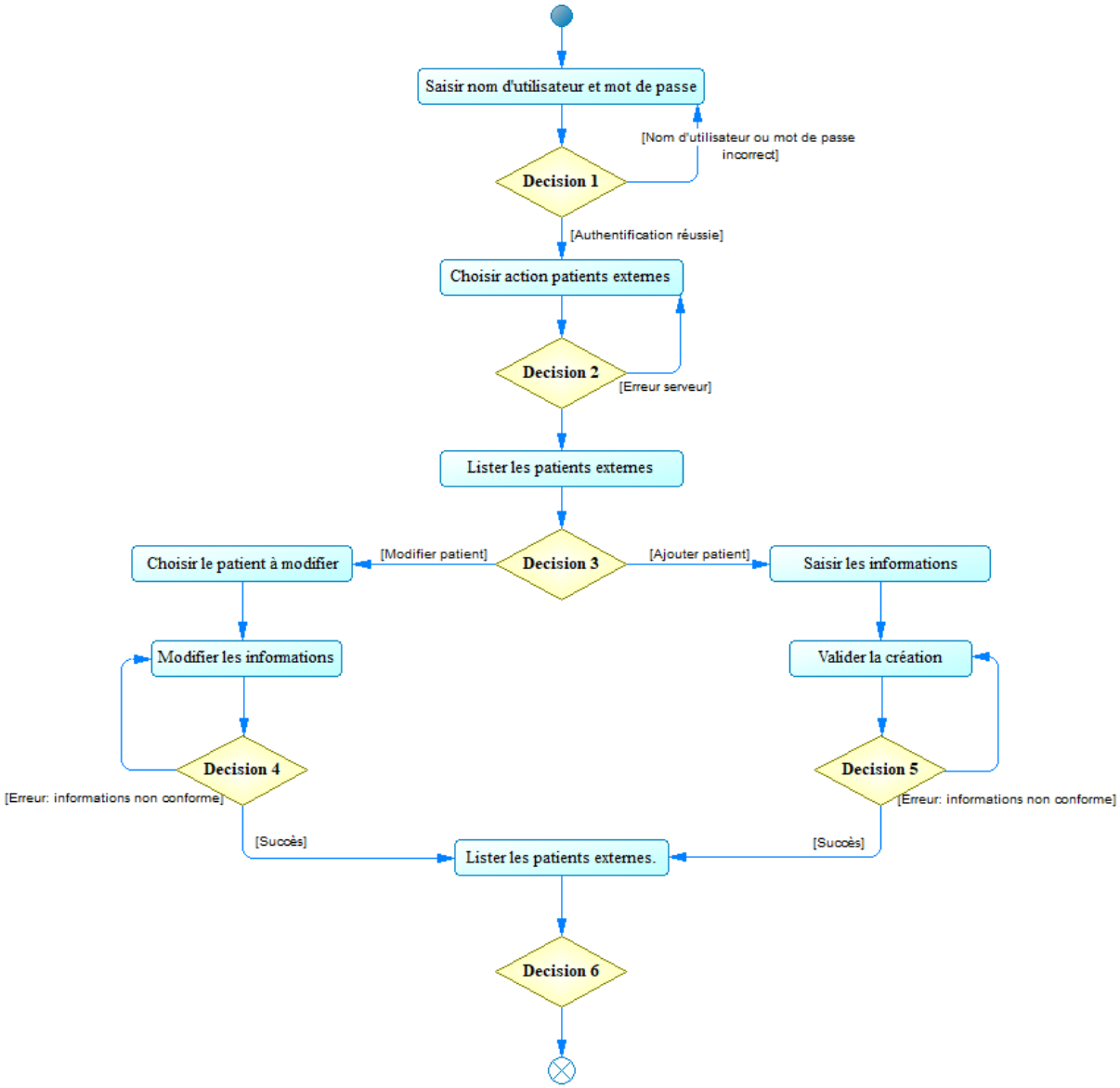

Figure 11 : Diagramme d'activités pour la gestion des patients externes

La figure ci-dessus représente le diagramme d'activités permettant de visualiser les actions faites pour la gestion des patients externes.

IV.1.2 Les diagrammes de séquences

IV.1.2.1. Définition d'un diagramme de séquences

Un diagramme de séquences représente la séquence de messages entre les objets pendant une interaction. Un diagramme de séquences représente un groupe d'objets avec des lignes de vie et des messages qu'ils échangent lors de l'interaction. Les diagrammes de séquence sont utilisés pour représenter la séquence de messages qui sont transmis entre des objets. Ils sont également capables de représenter les structures de contrôle interdépendantes [20].

IV.1.2.2. Diagramme de séquence d'authentification

La figure ci-dessous (***figure 12***) est le diagramme de séquences du processus d'authentification.

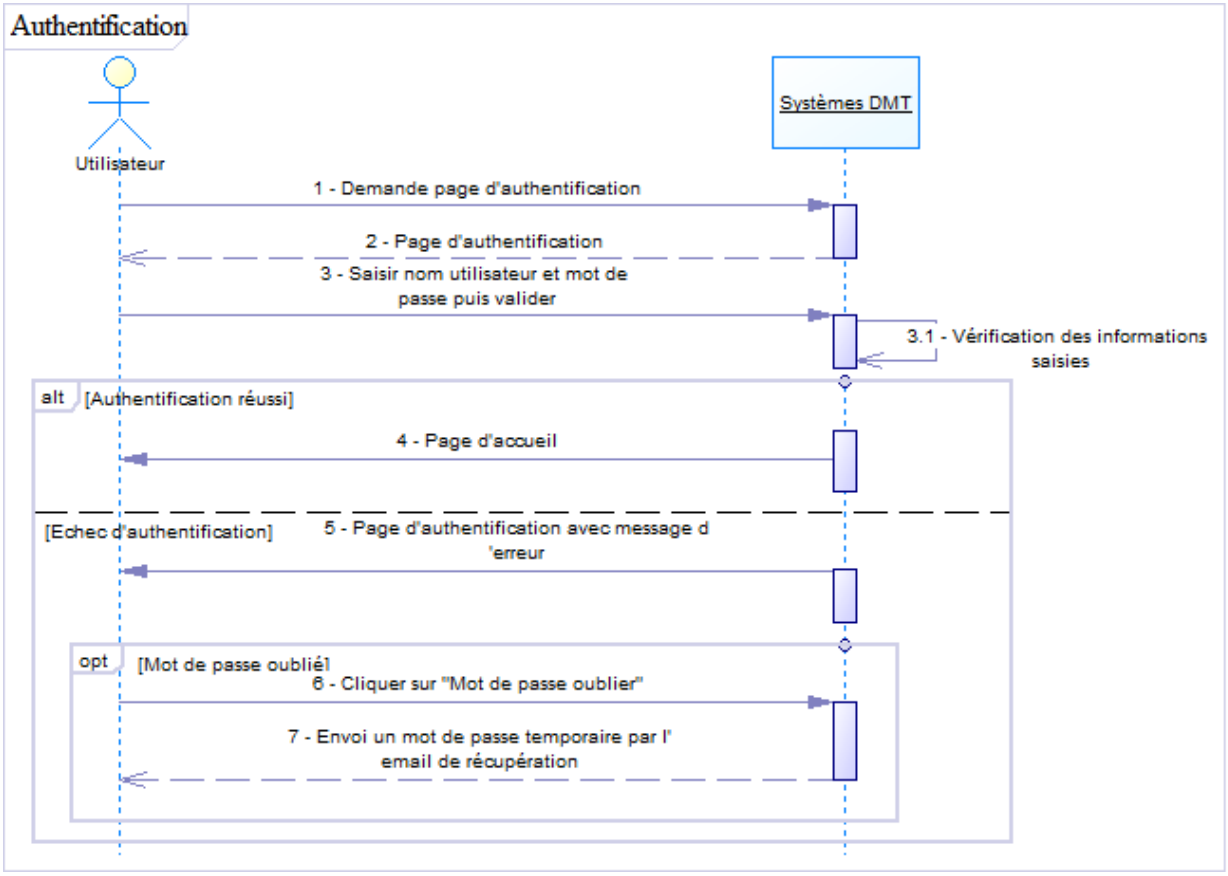

Figure 12 : Diagramme de séquences d'une authentification

Le tableau ci-dessous (***tableau 3***) représente la description détaillée qui accompagne le schéma ci-dessus :

Scénario principal	
1-Demande page d'authentification	Le système affiche la page de connexion

2-Saisir nom utilisateur et mot de passe puis valider	3.1-Le système vérifie les informations saisies par le médecin.
Si l'authentification réussit	4-Le système affiche la page d'accueil
Scénario secondaire	
Si le mot de passe ou le nom d'utilisateur est incorrect	5-Le système affiche : la page d'authentification avec le message d'erreur « Nom d'utilisateur ou mot de passe incorrect
Si l'utilisateur oublie son mot de passe. 6- Cliquez sur "Mot de passe oublié "	7-Le système envoie un nouveau mot de passe temporaire dans l'email de récupération de l'utilisateur

Tableau 3 : Description détaillée du diagramme de séquences d'authentification

IV.1.2.3. Diagramme de séquence pour création d'une consultation

La figure ci-dessous (***figure 13***) est le diagramme de séquences du processus de création d'une nouvelle consultation.

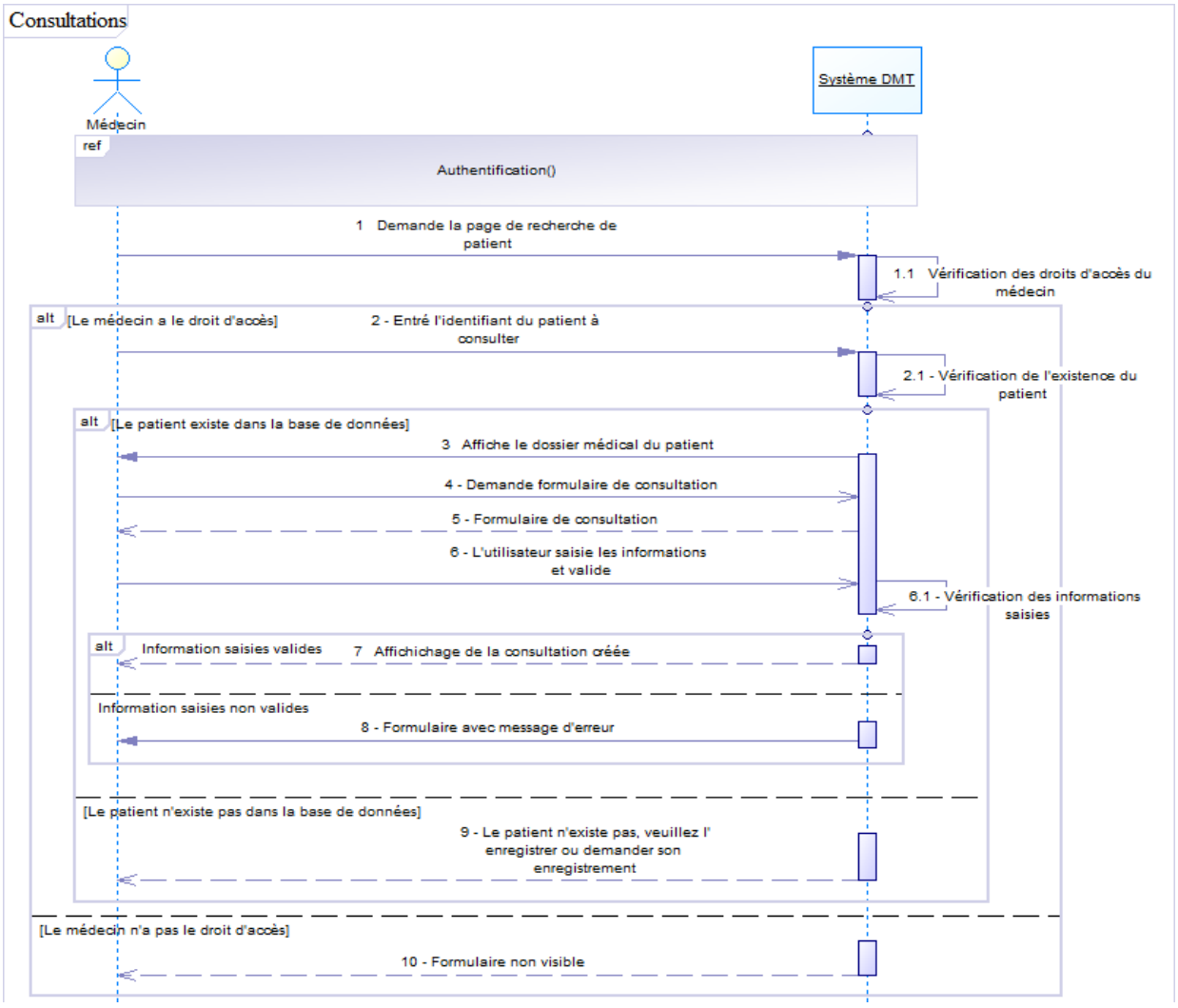

Figure 13 : Diagramme de séquences pour la création d'une consultation

Le tableau ci-après (***tableau 4***) représente la description détaillée qui accompagne le schéma ci-dessus :

Scénario principal	
0-Authentification	
1-Demande la page de recherche de patient	1.1-Le système vérifie les droits d'accès du médecin
Scénario secondaire 1 – 1	
Si le médecin a le droit d'accès	2- Entrez l'identifiant du patient à consulter.
Scénario secondaire 2 – 1	
	2.1-Le système vérifie l'existence du patient dans la base de données.
Si le patient existe dans la base de données	3-Le système affiche son dossier médical
4- Demande de formulaire de consultation	5-Le système affiche : le formulaire de consultation.
6-L'utilisateur saisit les informations et valide	6.1-Le système vérifie les informations saisies
Scénario secondaire 3 – 1	
Si les informations saisies sont valides	7-Le système affiche : la consultation créée
Scénario secondaire 3 – 2	
Si les informations saisies ne sont pas valides	8-Le système affiche : le formulaire avec les messages d'erreur
Scénario secondaire 2 – 2	
Si le patient n'existe pas dans la base de données	9-Le système affiche : le patient n'existe pas, veuillez l'enregistrer ou demander son enregistrement
Scénario secondaire 1 – 2	
Si le médecin n'a pas le droit d'accès	8- Le système n'affiche pas le formulaire de recherche d'un dossier médical

Tableau 4 : Description détaillée du diagramme de séquences d'ajout d'une nouvelle consultation

IV.1.2.4. Diagramme de séquence pour la création d'une visite

La figure ci-dessous (***figure 14***) est le diagramme de séquences du processus de création d'une nouvelle visite.

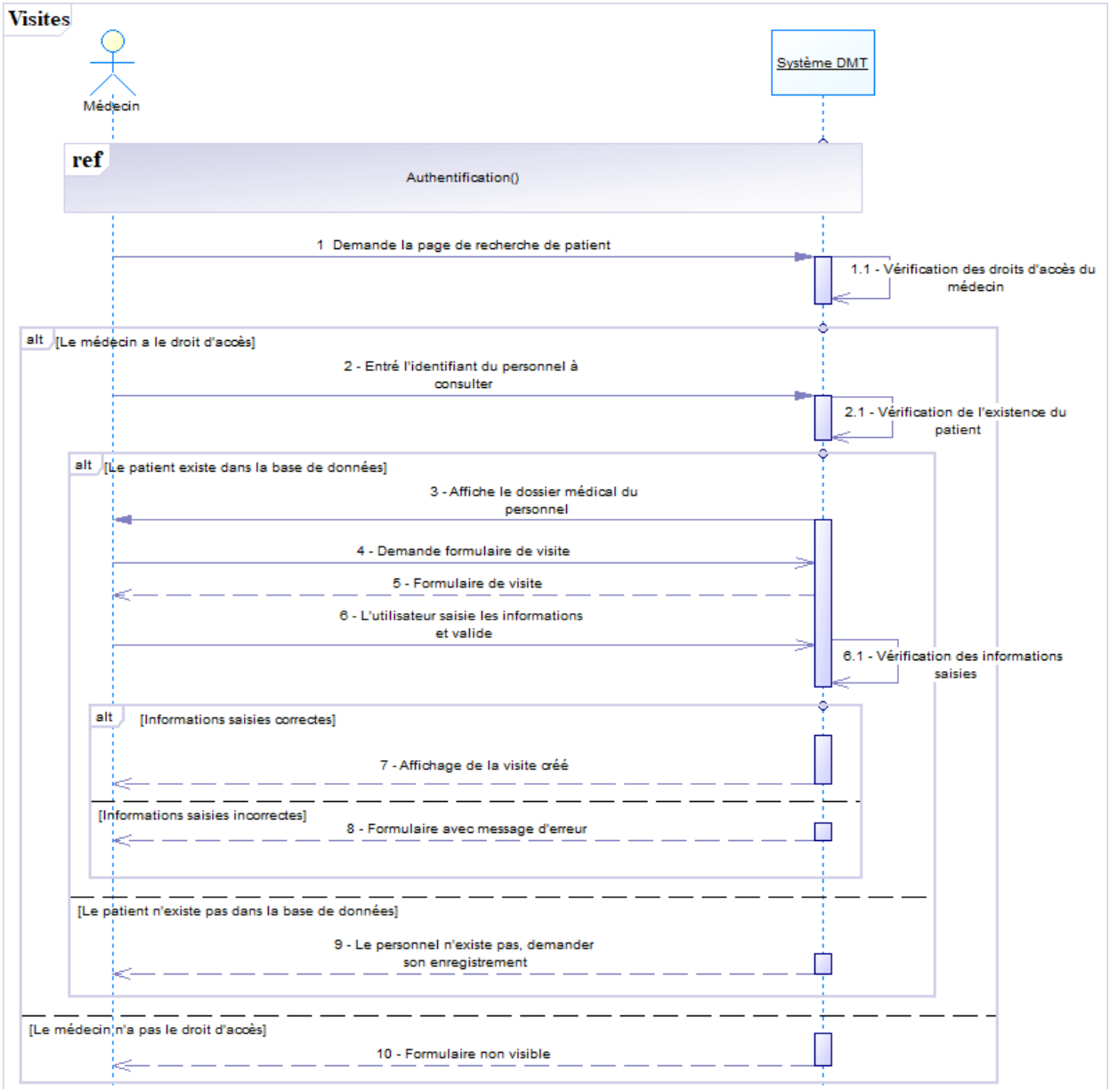

Figure 14 : Diagramme de séquence pour la création d'une visite

Le tableau ci-après représente la description détaillée qui accompagne le schéma ci-dessus :

Scénario principal	
0-Authentification	
……	1.1-Le système de vérification des droits d'accès du médecin
Scénario secondaire 1 – 1	
Si le médecin a le droit d'accès	2- Le système affiche la barre de recherche
Scénario secondaire 2 – 1	
Le médecin saisi l'identifiant du patient	2.1-Le système vérifie l'existence du patient dans la base de données.
Si le patient existe dans la base de données	3-Le système affiche son dossier médical
4- Demande de formulaire de visite	5-Le système affiche : le formulaire de visite.

6-L'utilisateur saisit les informations et valide	6.1-Le système vérifie les informations saisies
Scénario secondaire 3 – 1	
Si les informations saisies sont valides	7-Le système affiche : la visite créée
Scénario secondaire 3 – 2	
Si les informations saisies ne sont pas valides	14-Le système affiche : le formulaire avec les messages d'erreur
Scénario secondaire 2 – 2	
Si le patient n'existe pas dans la base de données	15-Le système affiche le message : le patient n'existe pas, veuillez demander son enregistrement au DRH
Scénario secondaire 1 – 2	
Si le médecin n'a pas le droit d'accès	16- Le système n'affiche pas le formulaire de recherche d'un dossier médical

Tableau 5 : Description détaillée du diagramme de séquence d'ajout d'une nouvelle visite

Nous pouvons noter que seuls les membres du personnel peuvent faire les visites.

IV.2. Conception générale

La conception de l'architecture est une étape cruciale du développement d'un logiciel. Elle détermine son efficacité, sa stabilité et sa pérennité. Cependant, il est possible que certaines applications présentent des lacunes en raison d'une architecture mal conçue ou qui n'est pas appropriée pour le contexte.

IV.2.1. Architecture physique de l'application

Il existe beaucoup de modèles d'architectures physiques pour le développement d'une application. Nous pouvons citer l'architecture 2-tiers, également appelée architecture client-serveur à deux niveaux, qui est un modèle d'architecture logicielle dans lequel la fonctionnalité de l'application est répartie entre deux couches principales : le client et le serveur. Chacune de ces couches effectue une tâche particulière dans le traitement des données [21].

La ***figure 15*** illustre l'architecture 2-tiers

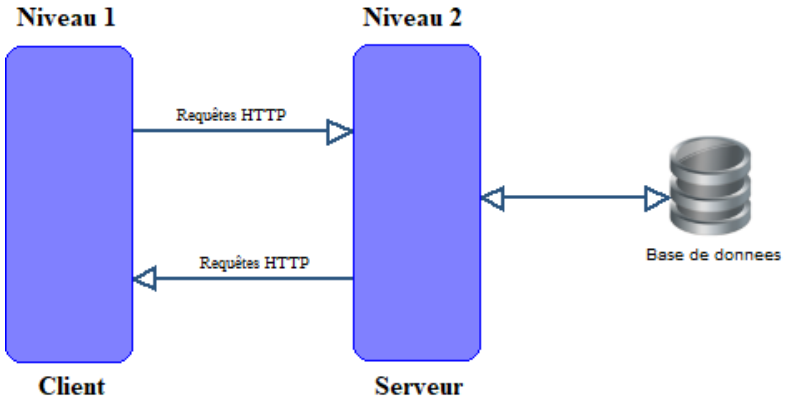

Figure 15 : Architecture 2-tiers

Il y'a aussi l'architecture 3-tiers, aussi appelée architecture à trois niveaux. Elle divise une application en trois principales composantes ou couches logiques, la couche client (présentation), la couche métier (logique métier) et la couche de données (base d données). Chacune de ces couches remplit une fonction unique dans le traitement des données [21].

Voici une illustration (***figure 16***) de l'architecture 3-tiers

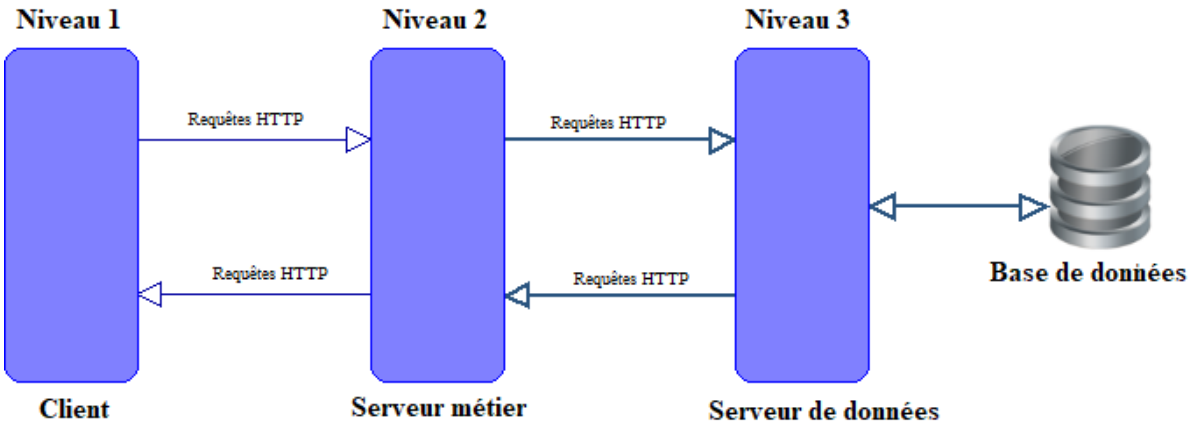

Figure 16 : Architecture 3-tiers

Couche de Présentation (Niveau 1 - Interface Utilisateur) :

- C'est le niveau avec lequel l'utilisateur interagit directement.
- Responsable de la présentation des données et de la collecte des entrées utilisateurs.
- Souvent appelée la couche "client" dans le contexte de l'architecture 3-tiers.

Couche métier (Niveau 2 - Logique Métier) :

- Contient la logique métier de l'application.

- Responsable du traitement des données, des calculs et de la mise en œuvre des règles métiers.
- Peut être appelée la couche "serveur d'application" ou "middleware".

Couche de données (Accès aux Données) :

- Gère l'accès et la manipulation des données.
- Stocke et récupère les données à partir de sources de données telles que des bases de données.
- Souvent appelé la couche "base de données".

IV.2.2. Justification du choix de l'architecture physique

Le choix de l'architecture est sans aucun doute l'une des parties les plus importantes pour bénéficier de nombreux avantages lorsqu'il s'agit de maintenir et de développer une application. Il dépend souvent des besoins spécifiques de l'application et des contraintes du projet.

Nous avons porté notre choix sur l'architecture physique 3-tiers pour les raisons suivantes :

- **Séparation des responsabilités :** chaque couche a des responsabilités spécifiques, facilitant la maintenance et les mises à jour ;
- **Réutilisabilité :** la logique métier peut être réutilisée par différentes interfaces utilisateurs ;
- **Évolutivité :** chaque couche peut être mise à l'échelle indépendamment des autres.
- **Flexibilité :** les technologies utilisées dans chaque couche peuvent être choisies indépendamment des autres, tant qu'elles respectent les interfaces définies ;
- **Performances** : les clients peuvent se concentrer sur l'interface utilisateur et la réactivité en confiant des tâches lourdes en terme de calcul ou de traitement de données au serveur.

IV.2.3. Architecture logique de l'application

Il existe plusieurs architectures logicielles, chacune ayant ses propres avantages, inconvénients et cas d'utilisation approprié. Le choix de l'architecture d'un logiciel est une décision cruciale qui affecte la conception, le développement, la maintenance et l'évolutivité de l'application. Parmi les architectures logicielles, nous avons :

- L'architecture monolithique : c'est un modèle de développement logiciel traditionnel qui utilise une base de code unique pour exécuter plusieurs fonctions métiers. En raison

des mécanismes d'échange de données au sein d'un système monolithique, tous ces composants logiciels sont interconnectés [22].

La figure ci-après (***figure 17***) est une illustration l'architecture monolithique.

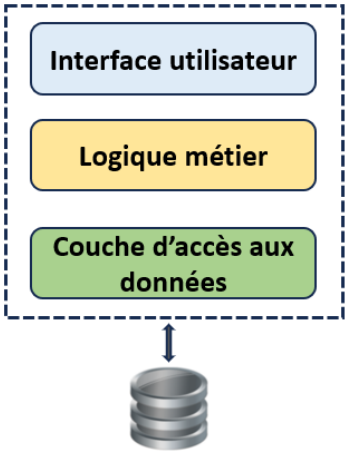

Figure 17 : Exemple architecture monolithique

- L'architecture orientée services (SOA pour Service Oriented Architecture en anglais) est un modèle de structure et un ensemble de principes de conception qui prennent en charge le couplage et la réutilisabilité de différents composants dans un système distribué. Il est devenu populaire au début des années 2000 et est devenu un des modèles d'architecture dominants aujourd'hui. En effet, alors que le SOA était uniquement lié à des applications Web spécifiques à ses débuts, ses cas possibles d'utilisation sont démultipliés et peuvent même s'adapter au cloud ou aux microservices [23].

La figure ci-dessous (***figure 18***) est une illustration l'architecture SOA.

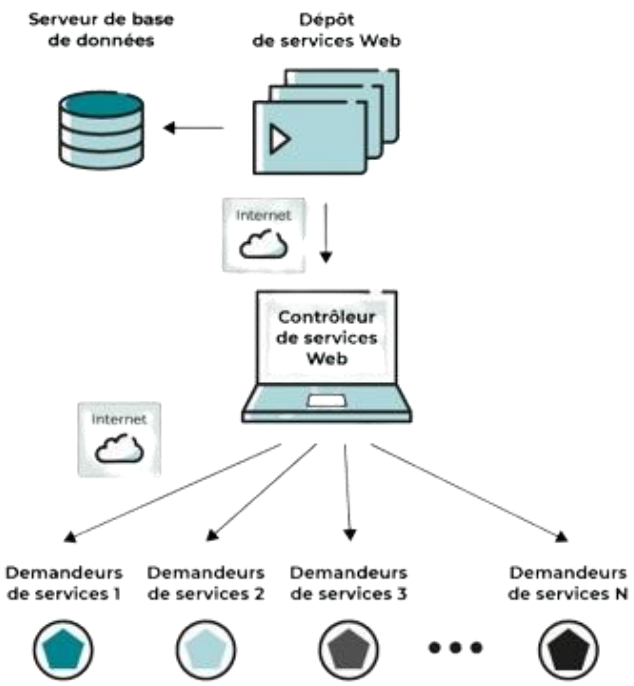

Figure 18 : Exemple architecture orienté service [https://openclassrooms.com/fr/courses]

- L'architecture microservices est un style de conception spécifique pour les applications logicielles. L'architecture microservices utilise une approche modulaire, contrairement aux styles architecturaux traditionnels qui visent à construire un logiciel comme une seule et unique unité. Par conséquent, les applications sont conçues comme des suites de plusieurs microservices qui peuvent être gérés et déployés indépendamment [23].

La figure ci-après (***figure 19***) est une illustration l'architecture microservices.

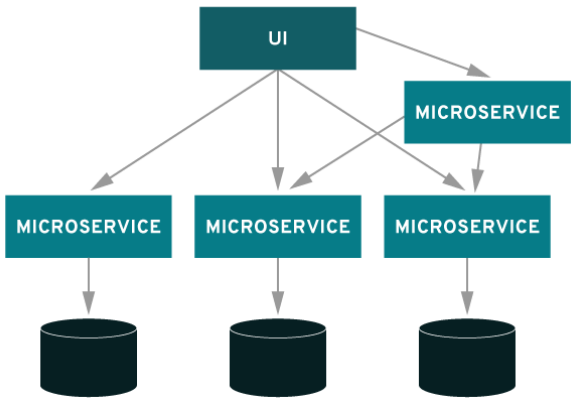

Figure 19 : Exemple d'architecture microservices

- Etc.

IV.2.4. Justification du choix de l'architecture logique

L'architecture microservices serait un excellent choix si elle n'était pas coûteuse avec le nombre de modules dans le PGI, mais vu que nous avons une application à utilisation interne dans la structure, nous avons opté de développer notre module sous forme de services et de choisir l'architecture monolithique pour chaque partie de notre application, avec le modèle en couche pour le Back-end et le modèle MVC pour le Front-end.

- L'architecture en couche permet une organisation claire du code en utilisant ses différentes couches, ce qui facilite la maintenance et l'évolution de l'application. Il est important de noter que certaines de ces couches peuvent être divisées ou combinées davantage en fonction des besoins spécifiques de l'application [24].

La figure ci-après (***figure 20***) est une représentation l'architecture en couche.

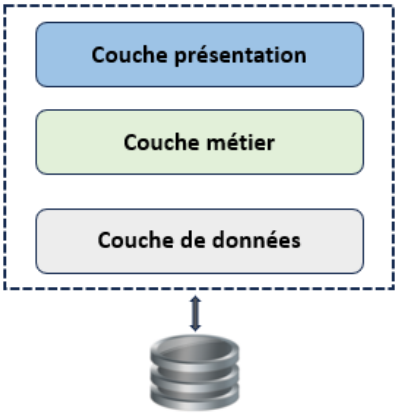

Figure 20 : Architecture en couches

- Le Model-View-Controller (MVC) présente une architecture organisée et modulaire qui favorise le développement, la maintenance et la progression des applications logicielles. Cela nous permet de travailler avec davantage d'efficacité et de concevoir des applications rigoureuses et aisément appréhendables [25].

La figure ci-après (***figure 21***) est une illustration l'architecture MVC.

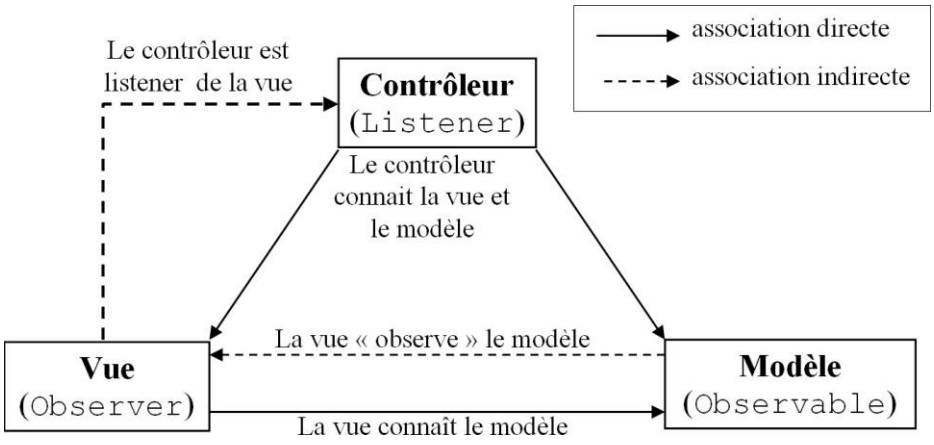

Figure 21 : Architecture MVC [26]

- **Le modèle :** Le modèle est une représentation des données de l'application et est généralement constitué d'objets.
- **La Vue :** L'interface utilisateur de l'application est illustrée dans la vue. Les vues sont définies à l'aide de modèles qui décrivent comment les données du modèle doivent être affichées.
- **Le contrôleur :** Les composants remplissent le rôle de contrôleur. La logique de présentation et la gestion des événements visuels sont définies par les composants.

La figure ci-dessous (***figure 22***) représente l'architecture logique de notre application (vue d'ensemble).

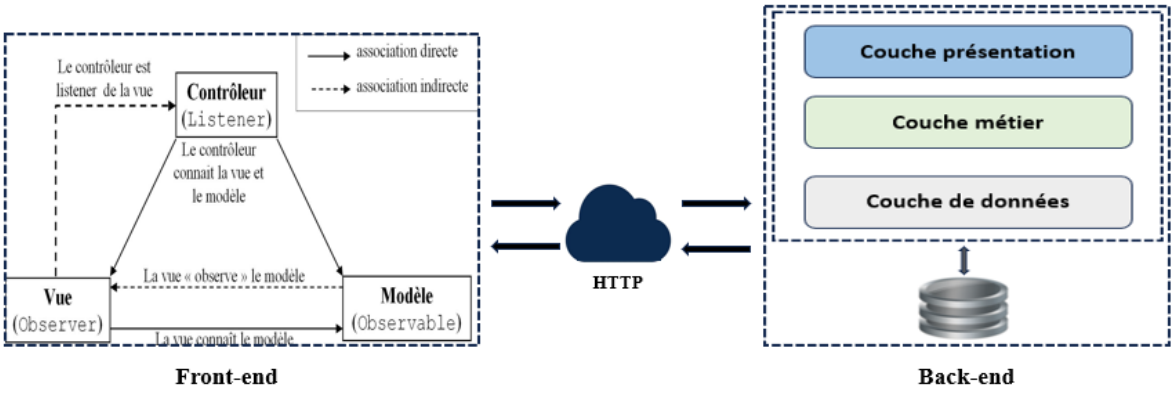

Figure 22 : Architecture de notre application

IV.3. Conception détaillée

Dans cette section, nous allons présenter quelques diagrammes de classes (patients, consultations et visites) avant de terminer par le dictionnaire des données qui sera présenté en annexe. Le reste des diagrammes de classes (informations personnelles et informations professionnelles) est consigné aussi en Annexe.

IV.3.1. Définition d'un diagramme de classes

Le diagramme de classe est un type de diagramme qui fait partie d'un langage de modélisation unifié (UML) et qui définit, fournit la vue d'ensemble et la structure d'un système en termes de classes, d'attributs et de méthodes, ainsi que les relations entre les différentes classes. Il sert de ressource de développement du système dans le cycle de vie du développement logiciel et est utilisé pour illustrer et créer un diagramme fonctionnel des classes du système [27].

IV.3.2. Diagramme de classes des patients

La figure ci-dessous (***figure 23***) présente le diagramme de classes du sous-module Patients. Nous représentons toutes les classes représentant les différents patients de la DMT.

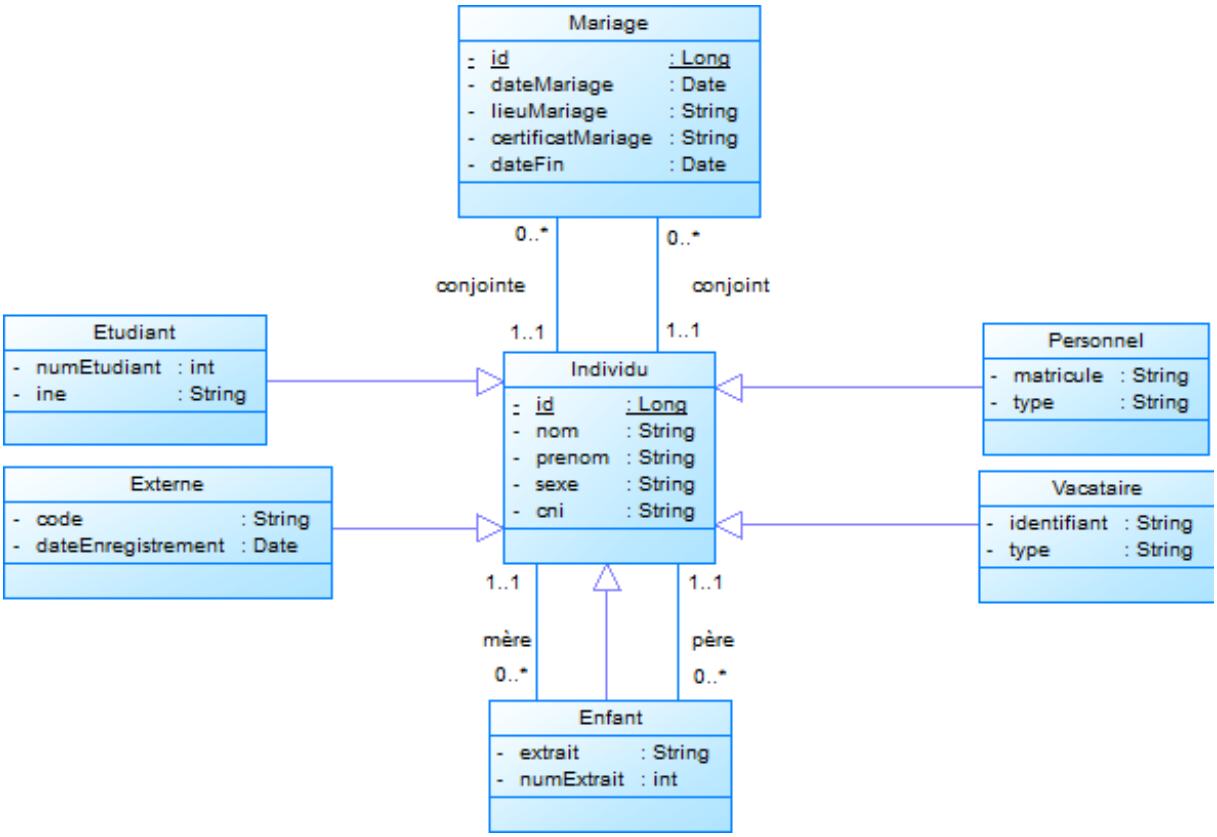

Figure 23 : Diagramme de classe des patients

Le tableau suivant nous donne la description détaillée du diagramme de classes ci-dessus.

Classe	**Description de la classe**
Individu	Cette classe regroupe les informations communes des patients
Personnel	Cette classe hérite de la classe Individu, contient les informations supplémentaires concernant le personnel de l'Université (PER et PATS).
Etudiants	Cette classe hérite de la classe Individu, contient les informations supplémentaires concernant les étudiants et représente les étudiants de l'Université.

Vacataires	Cette classe hérite de la classe Individu, contient les informations supplémentaires concernant les vacataires et représente les vacataires de l'Université.
Enfant	Cette classe hérite de la classe Individu, contient les informations supplémentaires concernant les enfants et représente les enfants des personnels de l'Université.
Externes	Cette classe hérite de la classe Individu, contient les informations supplémentaires concernant les patients externes et représente les patients habitant aux alentours de l'Université.
Mariages	Cette classe représente la liaison entre deux individus (par mariage), elle contient les informations supplémentaires concernant le mariage de ces deux individus.

Tableau 6 : Description du diagramme de classes des patients

IV.3.3. Diagramme de classes des consultations

Dans cette partie, nous avons le diagramme de classes du module Consultation (***figure 24***). Nous représentons les deux types de consultations et toutes les classes pouvant intervenir lors d'une consultation à la DMT.

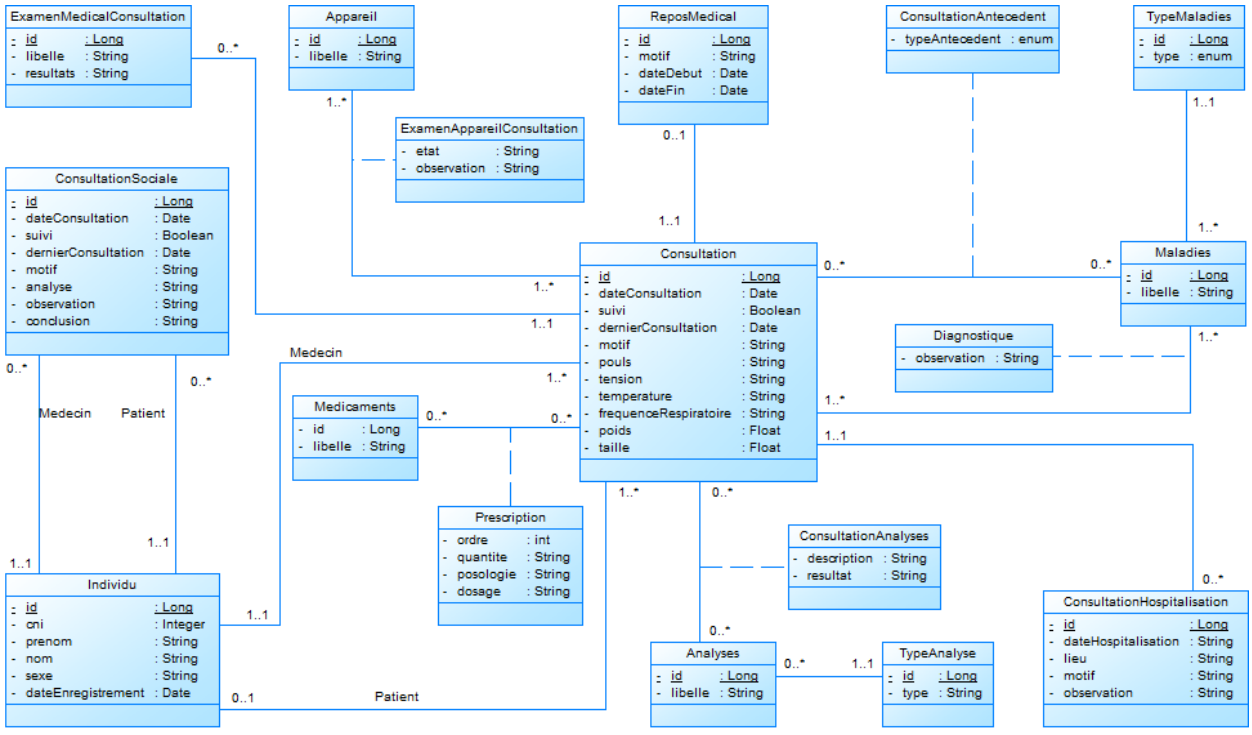

Figure 24 : Diagramme de classes des consultations

Le tableau ci-après nous donne la description détaillée du diagramme de classes ci-dessus.

Classe	**Description de la classe**

Consultation	Cette classe est la principale pour une consultation, elle regroupe les constantes physiologiques prises lors de la consultation d'un patient.
Appareil	Cette classe permet de regrouper tous les appareils de l'organisme (poumon, rein, etc.).
ExamenAppareilConsultation	Cette classe permet de regrouper les informations sur les examens d'appareils faits lors des consultations. C'est une classe d'association entre la classe consultation et la classe appareil.
ExamenMédicalConsultation	Cette classe permet de regrouper les informations sur les examens médicaux faits lors des consultations.
ReposMedical	Cette classe permet de regrouper tous les repos médicaux prescrits au patient par la DMT.
Maladie	Cette classe permet de regrouper les maladies.
Type Maladie	Cette classe permet de regrouper les types de maladies.
ConsultationAntecedent	Cette classe regroupe les antécédents de maladie des patients. C'est une classe d'association entre la classe consultation et la classe antécédent
Diagnostique	Cette classe regroupe les diagnostics émis lors des consultations.
ConsultationHospitalisation	Cette classe permet de regrouper les informations sur les hospitalisations faites ailleurs.
Analyse	Cette classe regroupe toutes les analyses médicales disponibles
TypeAnalyse	Cette classe regroupe tous les types d'analyses médicales disponibles
ConsultationAnalyse	Cette classe regroupe les analyses médicales faites lors d'une consultation d'un patient. C'est une classe d'association entre la classe consultation et la classe antécédent
Medicament	Cette classe permet de regrouper les médicaments.
Prescription	Cette classe regroupe les prescriptions faites lors d'une consultation d'un patient. C'est une classe d'association entre la classe consultation et la classe Medicament
ConsultationSociale	Cette classe concerne les consultations avec le médecin social et regroupe les informations recueillies lors de la discussion du patient avec le médecin.
Individu	Cette classe représente les patients et les médecins, elle regroupe leurs informations personnelles.

Tableau 7 : Description du diagramme de classes des consultations

IV.3.4. Diagramme de classe de visites

Dans cette partie, nous avons le diagramme de classes du module Visites (***figure 25***). Nous représentons les différents types de visites et toutes les classes pouvant intervenir lors d'une visite à la DMT.

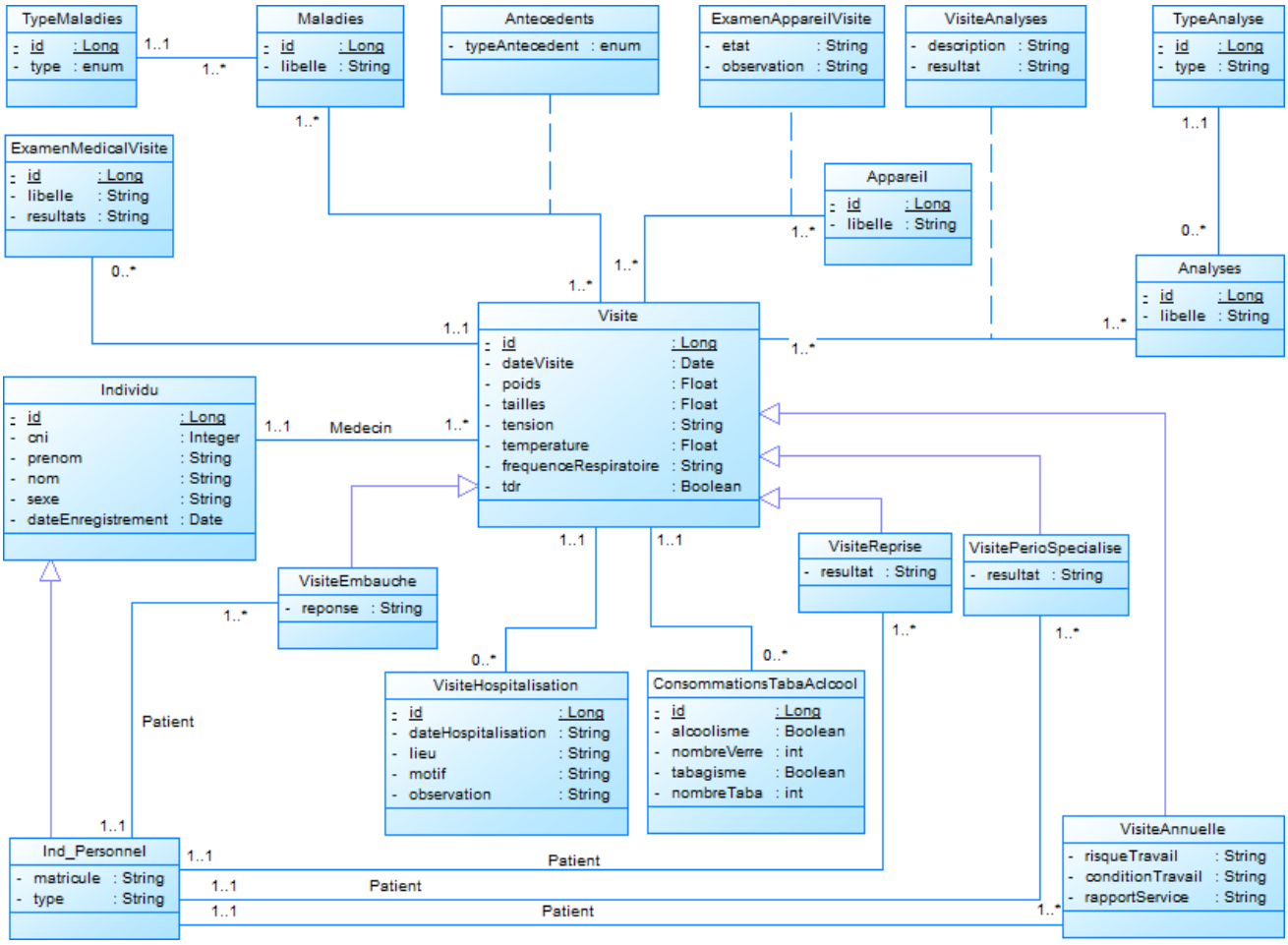

Figure 25 : Diagramme de classes des visites

Le tableau ci-dessous nous donne la description détaillée du diagramme de classe ci-dessus.

Classe	**Description de la classe**
Visite	Cette classe est la principale pour une consultation, elle regroupe les constantes physiologiques prises lors de la visite d'un patient.
Appareil	Cette classe permet de regrouper tous les appareils de l'organisme (poumon, rein, etc.).
ExamenAppareilConsultation	Cette classe permet de regrouper les informations sur les examens d'appareils faits lors des visites. C'est une classe d'association entre la classe visite et la classe Appareil.
ExamenMedicalConsultation	Cette classe permet de regrouper les informations sur les examens médicaux faits lors des visites.
Maladie	Cette classe permet de regrouper les maladies.
Type Maladie	Cette classe permet de regrouper les types de maladies.
VisiteAntecedent	Cette classe regroupe les antécédents de maladie des patients. C'est une classe d'association entre la classe visite et la classe antécédent.
VisiteHospitalisation	Cette classe permet de regrouper les informations sur les hospitalisations faites ailleurs.

Analyse	Cette classe regroupe toutes les analyses médicales disponibles
TypeAnalyse	Cette classe regroupe toutes les types analyses médicales disponibles
VisiteAnalyse	Cette classe regroupe les analyses médicales faites lors d'une visite d'un patient. C'est une classe d'association entre la classe visite et la classe Analyse.
Medicament	Cette classe permet de regrouper les médicaments.
Individu	Cette classe représente les patients et les médecins, elle regroupe leurs informations personnelles.

Tableau 8 : Description du diagramme de classes des visites

Conclusion

Nous avons commencé l'analyse des besoins en montrant les diagrammes d'activités et les diagrammes de séquences. Nous avons ensuite dans la conception générale proposé des architectures physique et logique. Enfin, dans la conception détaillée, nous avons produit et décrit les différents diagrammes de classes pour les différents modules que nous avons présenté dans les chapitres précédents. Ces phases nous a permis d'avoir un modèle de base de données pour entamer le chapitre implémentation de l'application dans la partie suivante.

Chapitre V : Implémentation, test et déploiement

Introduction

Dans ce chapitre, nous allons commencer d'abord par la présentation des outils que nous avons utilisés pour mettre en œuvre notre application. Ensuite, nous passons aux implémentations, aux tests, au déploiement et à la fin, nous présentons quelques interfaces de l'application.

V.1. Présentation des outils de développement

Comme nous avons utilisé l'architecture client-serveur (avec le client comme front-end et le serveur comme back-end) pour la création de notre application, nous avons choisi de travailler avec deux Frameworks : Angular pour le Front-end et Spring Boot pour le Back-end. Pour la base de données, nous avons choisi de travailler avec PostgreSQL.

Dans cette section, nous allons présenter les Frameworks Spring Boot et Angular, l'API REST que nous avons utilisé pour mettre en place les APIs du Back-end, l'Environnement de Développement Intégré (IDE, ou Integrated Development Environment en anglais) IntelliJ et le système de gestion base de données (SGBD) PostgreSQL en justifiant notre choix pour chacun d'eux.

V.1.1. Le Framework Spring Boot

V.1.1.1. Présentation du framework Spring Boot

Spring Boot est une extension ou module construit sur le framework Spring et vise à simplifier le processus de développement d'applications Spring. Il est open source ; développé par Pivotal et basé sur le langage Java. Il est utilisé pour créer des applications Spring (applications Java) autonomes de niveau productif avec une configuration minimale et un code passe-partout. Il fournit un moyen simple et efficace pour la création d'applications Web robustes et évolutives.

Il utilise un ensemble préconfiguré de dépendances et une approche avisée pour fournir aux développeurs un moyen rapide et facile de commencer à créer des applications sans se soucier des détails de configuration [28].

V.1.1.2. Fonctionnalités du framework Spring Boot

Au-delà du framework Spring, Spring Boot permet d'installer, de configurer et d'exécuter des applications plus facilement et plus rapidement. Il élimine le lourd travail de configuration

nécessaire à la mise en place de la plupart des applications basées sur Spring. Les développeurs peuvent se lancer directement dans l'utilisation de Spring Boot sans avoir à apprendre le framework Spring sous-jacent [29].

Voici quelques-unes des principales fonctionnalités de Spring Boot permettant de simplifier les tâches de programmation :

- **Un serveur intégré** : Spring Boot est livré avec un serveur intégré préconfiguré, tel que Tomcat, Jetty ou Undertow, qui facilite l'exécution d'applications sans avoir besoin d'un serveur externe ;
- **Une configuration automatique** : Spring Boot configure automatiquement Spring et d'autres bibliothèques tierces dans la mesure du possible. Cela permet aux développeurs d'économiser du temps et des efforts pour configurer l'application ;
- **Des dépendances de démarrage** : Spring Boot fournit des dépendances de démarrage, qui sont des dépendances préconfigurées qui simplifient l'intégration de diverses bibliothèques et framework dans l'application ;
- **Un actionneur** : Spring Boot Actuator fournit des fonctionnalités de surveillance et de gestion pour l'application, telles que des vérifications de l'état, des métriques et des points de terminaison de surveillance ;
- **Des applications autonomes** : Spring Boot permet de créer des applications qui ne sont pas liées à une plateforme spécifique et qui peuvent s'exécuter localement sur un appareil sans connexion Internet ou d'autres services installés pour être fonctionnels ;
- **Une approche avec opinion** : Spring Boot simplifie les configurations de build en fournissant des dépendances de démarrage avec opinion ;
- **Des fonctionnalités prêtes pour la production** : Spring Boot fournit des fonctionnalités prêtes pour la production, telles que des métriques, des vérifications d'intégrité et une configuration externalisée.

V.1.1.3. Justification du choix de Spring Boot comme Framework pour le Back-end

Les frameworks sont aujourd'hui la quintessence du développement d'applications. Les développeurs doivent trouver le bon framework pour garantir des performances et une évolutivité optimale. Compte tenu du nombre d'options disponibles aujourd'hui comme Django, Laravel, Ruby on Rails, etc, choisir celles qui sont appropriées peut être un vrai défi.

Nous avons porté notre choix sur le Framework Spring Boot pour les raisons suivantes :

- Il utilise le langage de programmation Java qui reste un choix solide pour un large éventail d'applications, en particulier pour les projets nécessitant la portabilité, la sécurité, la gestion de la mémoire automatique, et une riche base de bibliothèques. Java prend en charge aussi le multithreading de manière native, ce qui facilite la création d'applications concurrentes. Les programmes Java ont la capacité d'exécuter simultanément plusieurs threads, ce qui est particulièrement important pour les applications nécessitant un traitement parallèle ;
- Il simplifie le développement d'applications Spring avec Java en réduisant le temps consacré aux décisions et aux tâches répétitives grâce à son approche annotée de Spring, ce qui permet d'augmenter la productivité et de se concentrer sur la création et le test des applications ;
- Il réduit la nécessité d'écrire du code réutilisable, des annotations et des configurations XML. Les développeurs n'ont pas besoin de générer du code ou de configurer du code XML, ou même d'apprendre le framework Spring, s'ils ne le souhaitent pas ;
- Il intègre des applications dans la famille de projets Spring. Il s'intègrent de manière transparente à d'autres projets de l'écosystème Spring, tels que Spring Data, Spring Cloud, Spring Security, ainsi qu'à d'autres services cloud approuvés tels que Microsoft Azure Spring Cloud ;
- Il fournit des outils de développement/test grâce à l'outil d'interface de ligne de commande (CLI) de Spring Boot et aux serveurs HTTP incorporés, il est très simple de créer des environnements pour les applications Spring dev/test ;
- Il offre des plug-ins et des outils pour faciliter le développement. Spring Boot offre des plug-ins qui peuvent fonctionner avec des bases de données en mémoire, ainsi que d'autres outils d'automatisation de build populaire tel qu'Apache Maven ;
- Il utilise la plateforme Spring et les modules Spring existants pour simplifier la configuration et la mise en place de projets Java. Il permet aux développeurs de créer des applications avec une configuration minimale, en fournissant des configurations par défaut pour de nombreuses technologies telles que JDBC, JPA, Web, etc.

Le choix de Spring Boot pour le backend à l'échelle mondiale est souvent motivé par sa simplicité, sa flexibilité, sa modularité et son support pour la création d'applications d'entreprise modernes.

V.1.2. Le Framework Angular

V.1.2.1. Présentation du framework Angular

Angular est un framework open-source développé par Google qui permet de créer des applications web dynamiques et interactives. Il est basé sur le TypeScript, un langage de programmation typé qui permet une meilleure vérification des types de données et une meilleure maintenance du code. Il inclut également un système de gestion des dépendances (NgModule) et un ensemble de directives et de composants prêts à l'emploi pour faciliter le développement [30].

V.1.2.2. Fonctionnalités du framework Angular

Angular propose un large éventail de fonctionnalités pour la création d'applications Web modernes. Voici quelques-unes des principales fonctionnalités d'Angular :

- **Architecture basée sur des composants :** Angular utilise une architecture basée sur des composants, ce qui signifie que chaque composant est une composante de l'interface utilisateur qui peut être utilisée à nouveau. Cette méthode modulaire facilite la gestion de la complexité de l'application et permet une meilleure organisation du code ;
- **Binding Bidirectionnel de Données :** Angular permet une liaison de données bidirectionnelle, ce qui signifie que les modifications du modèle sont automatiquement reflétées dans la vue et vice versa. Cela augmente la productivité des développeurs et simplifie la gestion de l'état de l'application ;
- **Modèle de Modules :** Les modules Angular organisent l'application en fonctionnalité et encapsulent les composants, services, etc, associés à chaque fonctionnalité ;
- **Services :** Les services sont des classes injectables qui fournissent des fonctionnalités partagées entre différents éléments. Ils sont fréquemment utilisés pour gérer la logique de l'entreprise, la communication avec le backend, ou d'autres tâches connexes ;
- **Injection de dépendance :** La création et la distribution d'instances de services sont gérées par un système d'injection de dépendance utilisé par Angular. La réutilisation, le test et la gestion des dépendances sont facilités par cela ;
- **Observables et RxJS :** RxJS (Reactive Extensions for JavaScript) est utilisé par Angular pour la gestion des événements asynchrones. Les observables permettent un traitement efficace des flux de données ;

- **Router :** Un module de routage est fourni par Angular pour gérer la navigation entre les différentes vues de l'application. Il est compatible avec la navigation déclarative et impérative ;
- **Forms :** Les fonctionnalités puissantes d'Angular sont disponibles pour la création et la gestion de formulaires. Il prend en charge les formulaires basés sur des modèles ainsi que les formulaires réactifs ;
- **Pipes :** Les pipes sont des outils permettant de transformer l'affichage des données dans les templates. Vous pouvez utiliser les pipes intégrées d'Angular pour formater des nombres, des dates, etc., mais vous pouvez également créer vos propres pipes ;
- **Validation :** Les mécanismes de validation côté client sont fournis par Angular pour les formulaires. Les validations intégrées sont disponibles ou vous pouvez créer vos propres validations personnalisées ;
- **Animation :** Les animations fluides et réactives peuvent être créées directement dans les composants grâce au système d'animation intégré d'Angular ;
- **HTTP Client :** Angular fournit un module HTTP qui peut envoyer des requêtes HTTP à des serveurs lointains. Il supporte les opérations CRUD (créer, lire, actualiser et supprimer) ;
- **Tests Unitaires et E2E :** Angular facilite les tests unitaires avec des outils comme Karma et Jasmine. Il propose également des outils pour les tests end-to-end (E2E) avec Protractor ;
- **Internationalisation (i18n) :** Angular permet le développement d'applications multilingues en permettant la localisation et l'internationalisation ;
- **Écosystème :** Angular permet le développement d'applications robustes et performantes grâce à son écosystème riche de bibliothèques tierces, d'outils et de modules complémentaires.

V.1.2.3. Justification du choix de Angular comme Framework pour le Front-end

Comme les frameworks backend, les frameworks front-end sont aujourd'hui la quintessence du développement d'applications. Les développeurs doivent trouver le bon Framework pour garantir des performances et une évolutivité optimale. Même s'il en existe beaucoup aujourd'hui comme Semantic UI [31], Ink [32], React [33], Angular, etc., choisir ceux qui sont appropriées peut être un vrai défi [34].

Nous avons porté notre choix sur Angular comme framework front-end parce qu'en plus de ses fonctionnalités riches, Angular présente d'autres avantages :

- Angular utilise TypeScript qui est un langage de programmation open source développé par Microsoft. Il est conçu comme un sur-ensemble (ou superset) de JavaScript, ce qui signifie que tout code JavaScript valide est également du code TypeScript, mais TypeScript offre des fonctionnalités supplémentaires comme du typage statique, des concepts de programmation orientées objet, une meilleure maintenance du code et une intégration fluide avec le développement JavaScript existant. Il est très apprécié dans le développement Web, en particulier pour la création d'applications front-end complexes ;
- Angular est un composant de l'écosystème Google et dispose d'une communauté active de développeurs. Cela facilite le développement d'applications robustes en offrant une abondance de ressources, de bibliothèques tierces et de solutions prêtes à l'emploi ;
- Les fonctionnalités de sécurité d'Angular comprennent la protection contre les attaques de type Cross-Site Scripting (XSS) et Cross-Site Request Forgery (CSRF). De plus, il favorise les meilleures pratiques de sécurité pendant le développement ;
- Angular peut être facilement intégré à de nombreux backends, tels que les serveurs REST, GraphQL et d'autres services Web ;
- Angular permet une expérience utilisateur performante et réactive, même dans des conditions de réseau défavorable, grâce à ses fonctionnalités intégrées pour le développement d'applications Web progressives (Support des Progressive Web Apps) ;
- Angular est un Framework que Google maintient activement. Les mises à jour régulières incluent de nouvelles fonctionnalités, des correctifs de sécurité et des améliorations.

En résumé, le choix d'Angular dépend des exigences particulières de notre projet, de la complexité de notre application et de la préférence du maître de stage. Il offre une solution complète pour la création d'applications modernes, en particulier pour les projets nécessitant une structure solide et une maintenance à long terme.

V.1.3. API REST (Representational State Transfer)

V.1.3.1. Présentation de l'API REST

REST (Transfert d'État Représentatif en français) appelé aussi RESTful est un modèle de conception pour la mise en œuvre de systèmes connectés. Ce n'est ni un standard ni une technologie ; c'est un type d'architecture pour l'exposition de ressources sur Internet [35].

V.1.3.2. Propriétés :

L'API REST est basée sur 7 propriétés principales :

- Performance : Interaction simple entre les composants ;
- Evolutivité : Supporte une large variété de composants ;
- Simplicité : Entre les interfaces ;
- Modification : Peut être modifié sans impacter les clients ;
- Visibilité : Communication claire entre les composantes ;
- Confiance : Reprise sur panne.

V.1.3.3. Caractéristiques

L'architecture RESTful a plusieurs caractéristiques :

- Les requêtes sont client-serveur (séparation client-serveur) ;
- Les requêtes sont stateless (sans état) ;
- Une interface uniforme est utilisée par les clients et les serveurs. Une interface générique permet l'accès à toutes les ressources. : les méthodes HTTP : GET, POST, PUT, DELETE, HEAD ou OPTION ;
- Les clients accèdent à des ressources nommées. Le système comprend des ressources nommées en utilisant des URL, comme des URL HTTP (mais ne se limitant pas à des URL HTTP). Pour faire simple, une ressource est représentée par une URL ;
- Possibilité de mise en cache ;
- Etc.

V.1.4. Le Système de gestion de base de données PostgreSQL

Une base de données est un ensemble d'informations organisées de manière à ce qu'il soit facile d'accéder, de gérer et d'améliorer. Elle est utilisée par les organisations pour stocker, gérer et récupérer des informations [36]. Il existe différentes SGBD : relationnelles (Oracle, MySQL, PostgreSQL, etc.) et non relationnelles (MongoDB, ArangoDB, etc.).

Parmi ces bases de données, nous avons choisi le SGBD relationnelles PostgreSQL que nous allons présenter avant de justifier son choix.

V.1.4.1. Présentation du SGBD PostgreSQL

PostgreSQL est un système de gestion de base de données relationnelles, qui utilise le langage SQL et offre de nombreuses fonctionnalités pour stocker et gérer en toute sécurité des charges de travail de données complexes. Il a été développé à partir du projet POSTGRES de l'Université de Californie à Berkeley en 1986 et a depuis connu plus de 35 ans de développement actif [37].

V.1.4.2. Justification du choix de PostgreSQL comme SGBD

PostgreSQL est l'une des SGBD les plus utilisées aujourd'hui, il présente beaucoup d'avantages qui peuvent justifier notre choix. Voici quelques-uns de ses avantages :

- **Extensibilité :** Les utilisateurs peuvent créer leurs propres types de données, fonctionnalités, langages de procédures stockées et opérateurs ;
- **Type de données avancés :** En plus des types de données de base (entiers, chaînes de caractères, etc.), PostgreSQL offre des types de données avancés tels que les tableaux, les stores (pour le stockage de paires clé-valeur), les géométries pour les données spatiales, etc ;
- **Performance :** PostgreSQL fonctionne bien, en particulier dans les environnements chargés. Il comprend des mécanismes de mise en cache, des optimisations et des index avancés ;
- **Intégrité des données :** Il prend en charge les contraintes d'intégrité des données telles que les clés primaires, les clés étrangères et les vérifications, assurant ainsi la cohérence des données ;
- **Extensibilité horizontale** : Le partitionnement de tables et le clustering de PostgreSQL permettent l'extensibilité horizontale ;
- **Réplication :** Il propose des solutions de réplication pour assurer la disponibilité et la tolérance aux pannes des données ;
- **Support de transaction :** Pour garantir l'intégrité des données même en cas de défaillance du système, PostgreSQL prend en charge les transactions ACID (Atomicité, Cohérence, Isolation et Durabilité) ;
- **Communauté active :** La communauté open source active de PostgreSQL soutient son développement continu et fournit une assistance technique ;

- **Compatible avec de nombreux langage de programmation :** De nombreux langages de programmation permettent l'accès à PostgreSQL, ce qui facilite son intégration dans divers environnements de développement.

En résumé, PostgreSQL est un SGBD relationnelle puissante, extensible et open source qui, en raison de sa flexibilité et de ses fonctionnalités avancées, est adaptée à une variété de cas d'utilisation.

La figure ci-dessous (***figure 26***) est une illustration de l'architecture logique globale de notre application

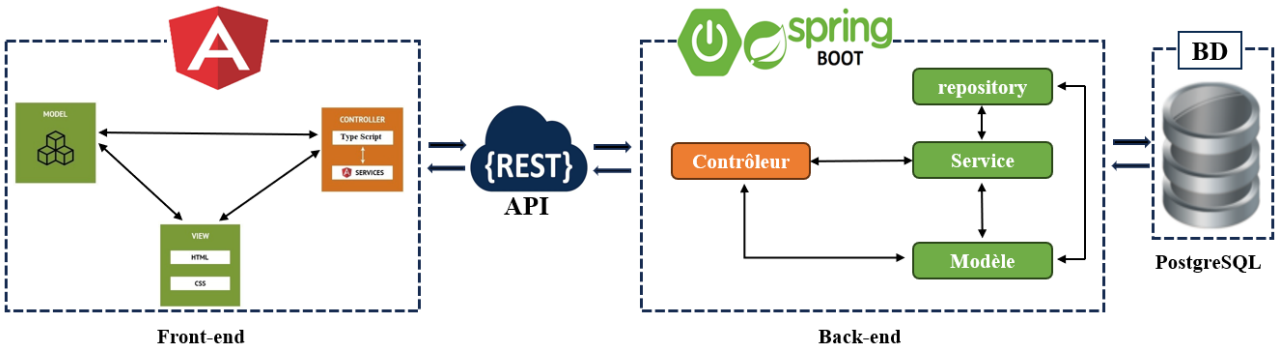

Figure 26 : Architecture logique générale de l'application

V.1.5. Environnement de développement intégré (IDE)

Les IDE se distinguent des outils d'édition de codes traditionnels grâce à leur intégration, tout comme leurs capacités de débogage, de compilation et d'automatisation. Cependant, la capacité de personnaliser un environnement avec des plugins et des intégrations est le principal avantage des IDE. Cela permet de personnaliser les procédures ou d'ajouter des fonctionnalités. Parmi les IDE, nous pouvons citer Visual Studio, IntelliJ IDEA, PyCharm, Eclipse, etc [38].

Nous avons opté pour **IntelliJ IDEA** pour des raisons que nous allons présenter après sa définition.

V.1.5.1. Présentation de IntelliJ IDEA

IntelliJ IDEA est l'un des IDE les plus puissants du marché. Grâce à l'indexation initiale, il comprend les tenants et les aboutissants de votre code. Il a été développé par JetBrains. Bien qu'il supporte également d'autres langages de programmation tels que Kotlin, Groovy, Scala, TypeScript et plus encore, c'est l'un des IDE les plus populaires pour le développement d'applications Java. Il peut détecter les erreurs à la volée, proposer des méthodes de saisie semi-

automatique du code avec une compréhension précise du contexte, initier des refactorisations sécurisées, etc.

V.1.5.2. Justification du choix de l'IDE IntelliJ IDEA

Nous avons choisi IntelliJ IDEA du fait qu'il soit un IDE qui intègre plusieurs langages de programmation tels que Java, TypeScript, HTML, CSS, JavaScript et tant d'autres. Il intègre aussi les plugins Spring Initializr (pour la création d'applications Spring Boot), Angular (pour la création d'applications Angular), etc.

Mais, il présente aussi des points clés qui font de lui l'un des IDE les plus utilisés aujourd'hui. Voici quelques points clés d'IntelliJ IDEA :

- **Éditeur intelligent :** L'IDE possède un éditeur intelligent avec des fonctionnalités avancées telles que la refactorisation du code, la navigation rapide, la vérification de la syntaxe et la complétion automatique ;
- **Intégration avec les outils de construction :** La compatibilité d'IntelliJ IDEA avec des outils de construction tels que Maven et Gradle facilite la gestion des dépendances et la construction de projets ;
- **Soutien aux frameworks :** Il prend en charge une variété de frameworks, tels que Spring, Hibernate, JavaFX et Android, et fournit des fonctionnalités spécifiques à ses frameworks pour faciliter le développement ;
- **Débogage avancé :** Des fonctionnalités de débogage puissantes, telles que le suivi des variables, la surveillance des expressions et la possibilité de déboguer des applications à distance, sont fournies par l'IDE ;
- **Test unitaire :** Il est compatible avec des frameworks comme JUnit pour l'écriture et l'exécution de tests unitaires ;
- **Contrôle de version :** La collaboration avec des systèmes de contrôle de version comme Git simplifie la gestion du code source et le suivi des modifications ;
- **Analyse de code statique :** Il comprend des outils d'analyse statique du code qui aident à identifier les erreurs potentielles, les problèmes de performance et à maintenir la qualité du code ;
- **Interface utilisateur conviviale :** L'IDE propose une interface utilisateur simple et conviviale avec des raccourcis claviers adaptables et des thèmes personnalisables ;

- **Extensions et plugins :** IntelliJ IDEA offre un large éventail d'extensions et de plugins qui permettent d'élargir les fonctionnalités de l'application pour prendre en charge d'autres langages et technologies ;
- **Intégration avec d'autres outils JetBrains :** IntelliJ IDEA fonctionne bien avec d'autres outils JetBrains, comme ReSharper, TeamCity ou Kotlin, pour une expérience de développement plus fluide.

Pour résumer, IntelliJ IDEA est un IDE polyvalent et puissant avec des fonctionnalités avancées qui facilitent le développement logiciel, en particulier dans l'écosystème Java.

Grâce à sa polyvalence, il a été utilisé dans ce travail pour le développement du Backend et du Frontend.

V.2. Implémentations

V.2.1. Implémentation du Back-end

Durant l'implémentation de notre back-end, nous avons régulièrement tenu des réunions avec certains membres de la Direction des Ressources Humaines (DRH) et la Direction de la Médecine du Travail (DMT) de l'Université pour la mise en place d'un cahier de charges. Après la mise en place du cahier de charge, nous avons commencé la conception et l'implémentation du back-end. Nous organisons des réunions d'équipe une fois par semaine pour la validation d'un sprint et le début d'un autre sprint.

En effet, comme le backend était unique pour le PGI, nous avons dû utiliser un outil de travail collaboratif pour que chaque membre de l'équipe qui travaillait sur son module puisse apporter des modifications le concernant sans faire obstruction aux autres. Pour ce travail collaboratif, nous avons choisi de travailler avec Git pour des raisons que nous allons expliquer après sa présentation avant de passer à la description de la manière dont nous avons travaillé avec Git.

V.2.1.1. Définition de Git

Git est un système de contrôle de version très populaire, créé en 2005 par Linus Torvalds, le créateur de Linux. Il est utilisé par de nombreux projets logiciels, qu'ils soient commerciaux ou open source, pour gérer les différentes versions de leur code. Git est un projet Open Source avancé et maintenu, ce qui signifie qu'il est constamment amélioré par une communauté de développeurs. Il fonctionne sur différents systèmes d'exploitation et environnements de

développement intégré (IDE), ce qui en fait une solution flexible pour les développeurs. De plus, les développeurs qui maîtrisent Git sont très recherchés sur le marché du travail.

V.2.1.2. Justification du choix de Git comme outil de travail collaboratif

Les logiciels de gestion de projet Git facilitent la collaboration entre les développeurs, que ce soit dans le domaine du développement informatique ou du Big Data. Il permet aux différentes personnes travaillant sur le même projet de partager les mêmes informations. Ainsi, chaque participant a la possibilité de voir les modifications apportées aux différentes versions du projet, ainsi que les tâches terminées et les tâches restantes [39]. Voici quelques autres avantages incontestables qui ont motivé notre choix :

- Enregistrer l'historique des modifications d'un ensemble de fichiers.
- Revenir à des versions précédentes d'un ou plusieurs fichiers.
- Rechercher les modifications qui ont pu créer des erreurs.
- Partager ses modifications et récupérer celles des autres.
- Proposer des modifications, les discuter, sans pour autant modifier la dernière version existante.
- Identifier les auteurs et la date des modifications.

V.2.1.2. Méthodologie de travail avec Git

Dans cette section, nous n'allons pas trop entrer dans les détails, nous allons juste nous limiter sur la structure. Nous avons commencé par la création d'un dépôt local dans le serveur de l'Université, puis nous avons créé une branche pour chaque membre de l'équipe.

Après ces étapes, nous avons d'abord débuté par l'implémentation des classes en commun de chacun de son côté avec son sprint. Une fois fini, chacun fait un commit dans sa branche respective et après validation de son travail, il fait le push de son travail dans sa branche avant d'attaquer un autre sprint. À la fin de l'implémentation de ces classes en commun, chacun de nous a continué le travail de son côté. Enfin, Scrum Master a regroupé le contenu des branches dans la branche principale et ensuite le déployé.

Pour la découverte de toutes les facettes de Git, vous pouvez consulter [39].

V.2.2. Implémentation du Front-end

Durant l'implémentation de notre front-end, nous avons presque procédé de la même manière à celle du back-end (avec un nouveau dépôt pour le front-end).

À la seule différence, chaque front-end a ses propres sprints et à la fin de ces sprints, nous n'avons pas besoin de fusionner les branches du fait que chaque module du PGI a son propre front-end.

En ce qui concerne le design, le maître de stage a choisi d'acheter un template au niveau du site Creative Tim [40]. Ainsi, nous avons choisi un template adéquat au Framework angular de notre choix et nous l'avons adapté à nos besoins.

V.3. Tests

V.3.1. Test du Back-end

Pour tester les services de notre back-end, nous avons utilisé Postman qui est un outil polyvalent et puissant qui facilite la gestion des API, la documentation, le développement et le test. Postman est devenu un outil essentiel dans le domaine du développement logiciel moderne pour les développeurs, les testeurs et les équipes travaillant sur des projets liés aux API.

Postman permet d'effectuer des requêtes HTTP via une interface graphique [41].

Les méthodes HTTP diffèrent suivant l'opération souhaitée et l'API sur laquelle vous faites votre opération. Voici les méthodes HTTP les plus utilisées et leurs fonctions :

- **GET** : les requêtes GET sont celles effectuées par un navigateur lorsque vous entrez une URL dans la barre de navigation. Elles ont pour but d'aller chercher une page ou de la donnée.
- **POST** : les requêtes POST ont pour but d'envoyer de l'information, contenue dans le *body* de la requête, vers le serveur.
- **PUT** : les requêtes PUT vont écraser une ressource avec de la nouvelle donnée, là aussi présente dans le *body* de la requête. Elle est utilisée pour mettre à jour de la donnée à condition qu'on soit capable de fournir la ressource mise à jour dans son intégralité.
- **PATCH** : les requêtes PATCH servent également à mettre à jour une ressource, mais en ne modifiant que l'élément envoyé en body de la requête.
- **DELETE** : comme son nom l'indique, la requête DELETE sert à effacer une ressource.

V.3.2. Test du Front-end

Pour cette étape, nous avons créé un compte administrateur pour tester les fonctionnalités de notre front-end.

V.4. Déploiement

Dans cette section, nous nous limitons à la présentation de notre diagramme de déploiement du back-end, ensuite aux ressources et aux outils que nous avons utilisés pour déployer notre backend.

V.4.1. Diagramme de déploiement

La figure ci-dessous (***figure 27***) représente le diagramme de déploiement de notre back-end. La description des éléments cités vient juste après.

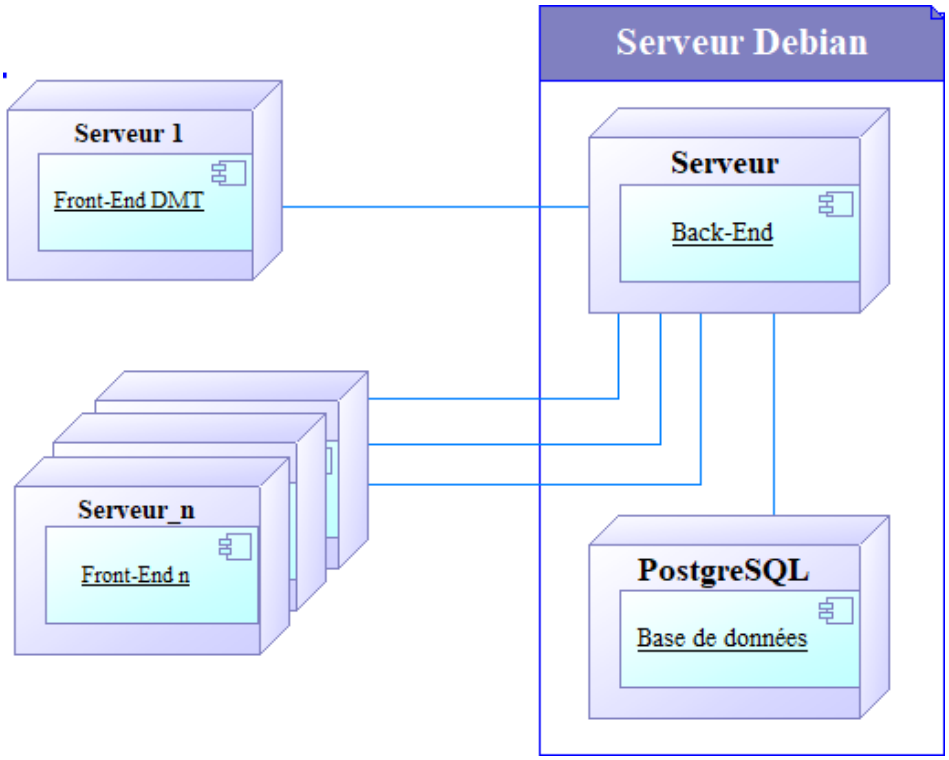

Figure 27 : Diagramme de déploiement du backend

Description des éléments représentés sur la figure ci-dessus :

- Front-end DMT : il représente notre front-end qui attaque le serveur (le back-end) à partir d'un navigateur Web.
- Front-end_n : ce sont les front-end des autres modules qui attaquent le backend.
- Serveur Debian : c'est la machine virtuelle installée dans le serveur de l'Université dans laquelle nous avons installé PostgreSQL et déployé le serveur (back-end).
- PostgreSQL : c'est le SGBD installée dans la machine virtuelle.
- Serveur : c'est le back-end déployé dans la machine virtuelle.

V.4.2. Les ressources utilisées pour le déploiement

- Dans cette partie, nous allons présenter les étapes de déploiement du back-end [42].

Le déploiement a été fait dans un serveur Debian en suivant les étapes suivantes :

- Installation JDK 15 ;
- Installation et configuration de Maven pour le déploiement ;
- Installation de PostgreSQL.

V.5. Présentation de quelques interfaces de l'application

Notre application dispose de plusieurs profils utilisateurs. Nous avons l'administrateur qui est le directeur de la DMT et les travailleurs de cette direction. Dans cette partie, nous allons présenter la page de connexion, la page d'accueil et quelques autres pages.

V.5.1. La page de connexion

La figure ci-dessous représente la page de connexion de notre application.

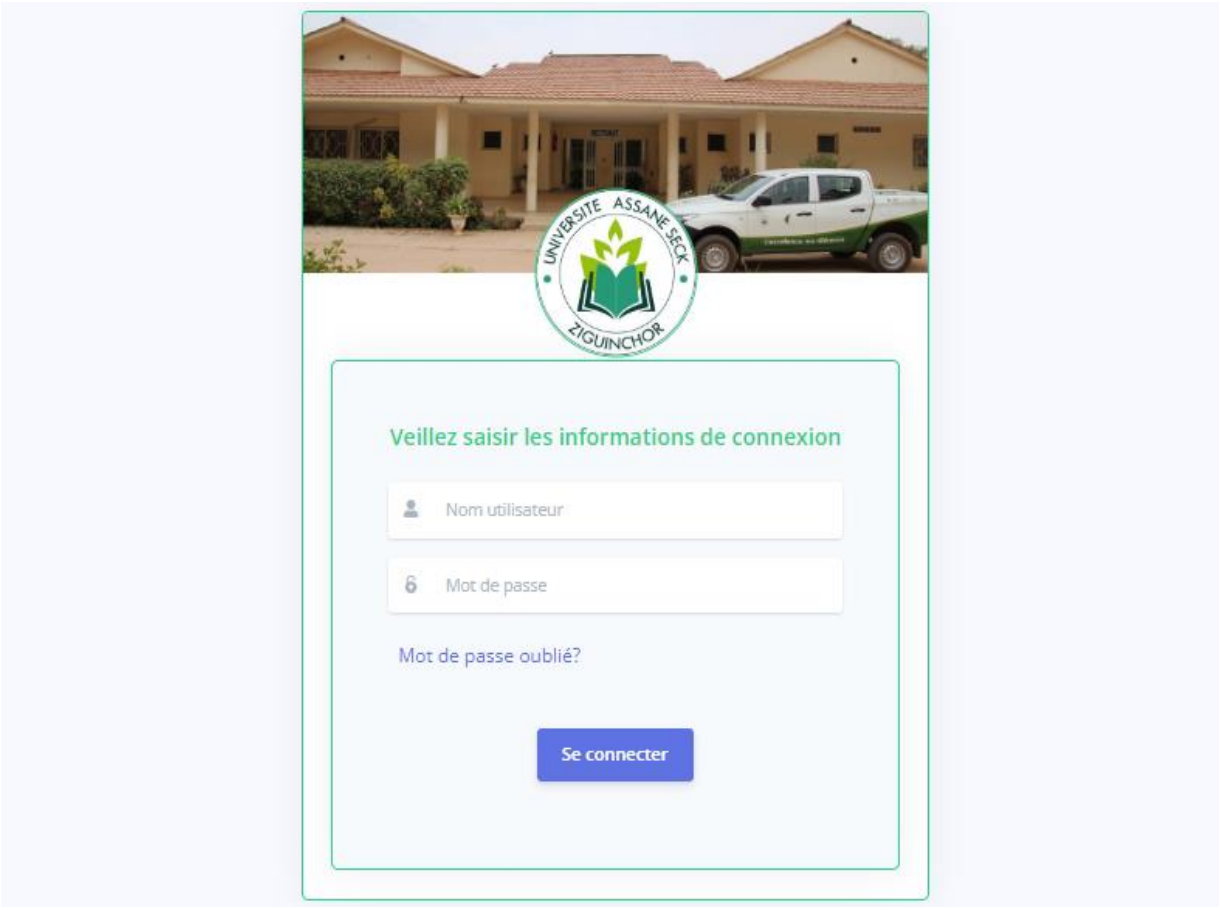

Figure 28 : Page d'accueil de l'application

V.5.2. La page d'accueil

La figure ci-dessous représente la page d'accueil de notre application.

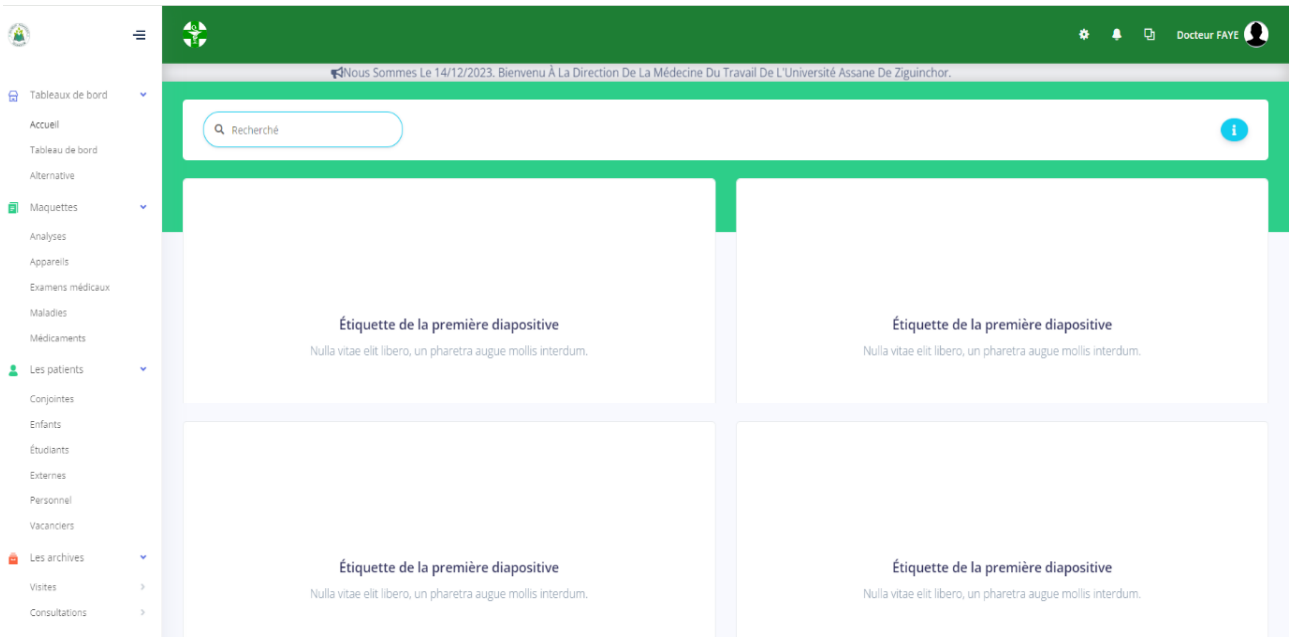

Figure 29 : Page d'accueil de l'application

V.5.3. La page de dossier médical

V.5.3.1. Exemple de vue des informations personnelles du patient

Nous allons présenter ici un exemple d'affichage des informations personnelles des patients.

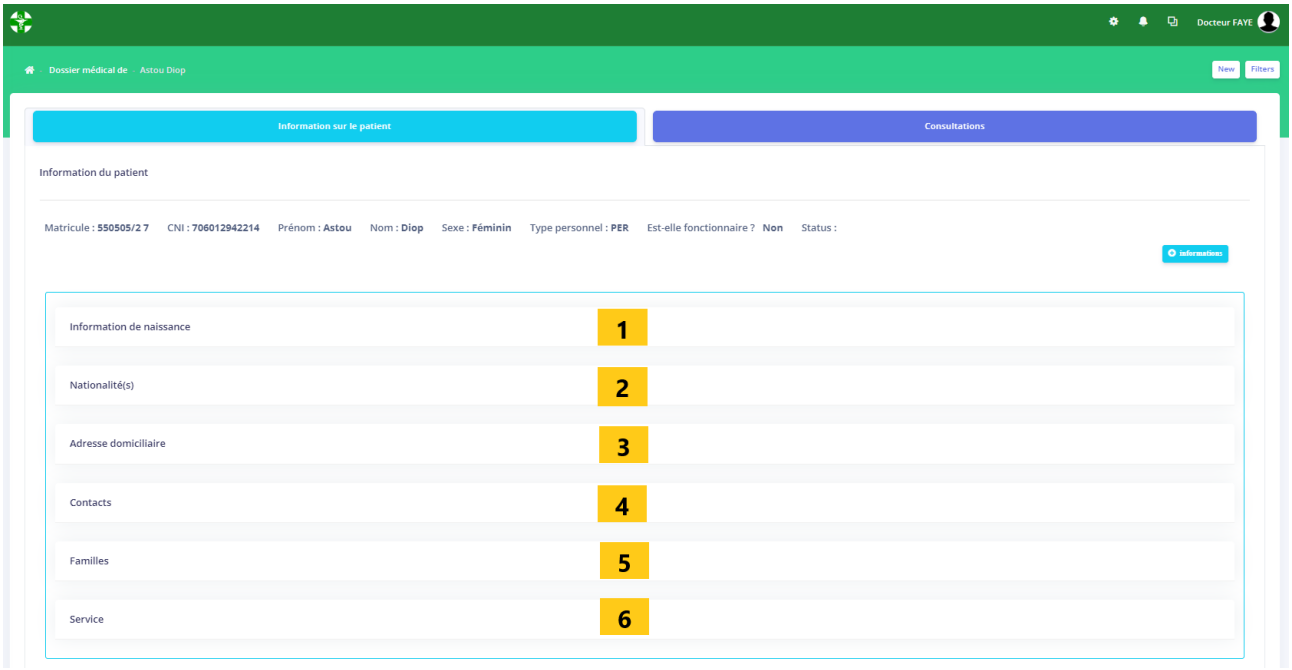

Figure 30 : Exemple d'affichage des informations personnelles des patients

La figure ci-dessus est un exemple d'affichage des informations personnelles des patients. En cliquant sur :

1 – Nous affichons les informations de naissances du patient ;

2 - Nous affichons les informations de nationalité du patient ;

3 - Nous affichons les adresses du patient ;

4 - Nous affichons les contacts du patient ;

5 - Nous affichons les informations sur la famille (époux ou épouse et enfant) du patient ;

6 - Nous affichons les informations sur son service.

NB : Les points 5 et 6 ne concernent que le personnel de l'Université.

V.5.3.2. Exemple de vue du dossier médical par l'administrateur (le médecin chef)

La figure ci-dessous est un exemple d'affichage du dossier médical d'un patient pour l'administrateur

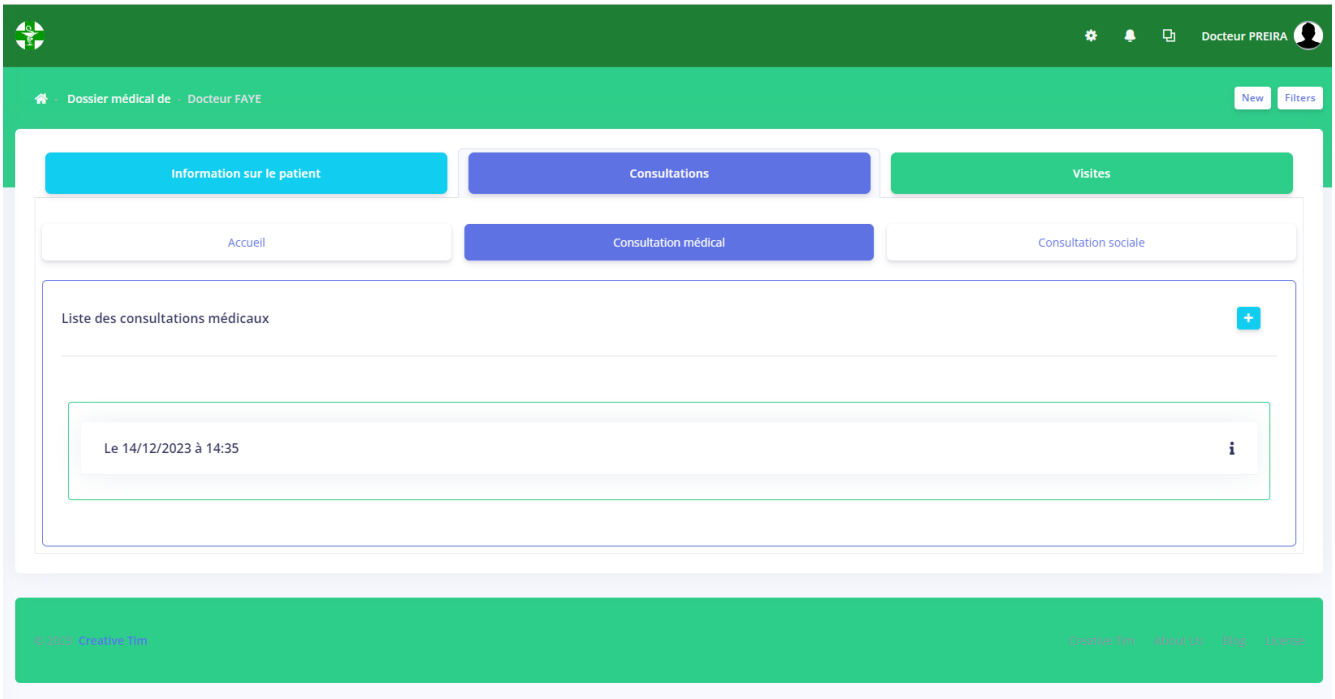

Figure 31 : Exemple d'affichage du dossier médical d'un personnel par administrateur

Dans cette interface (figure ci-dessus), on a le bouton information sur le patient qui regroupe les informations personnelles du patient, le bouton consultations qui regroupe les consultations médicales et sociales et le bouton visites qui n'apparaît que lorsqu'il s'agit d'un personnel et renferme les visites d'embauche, annuelle, de reprise et périodique spécialisée.

V.5.3.3. Exemple de vue de la dossier médical par le médecin social

La figure ci-dessous est un exemple d'affichage du dossier médical d'un patient pour le médecin social.

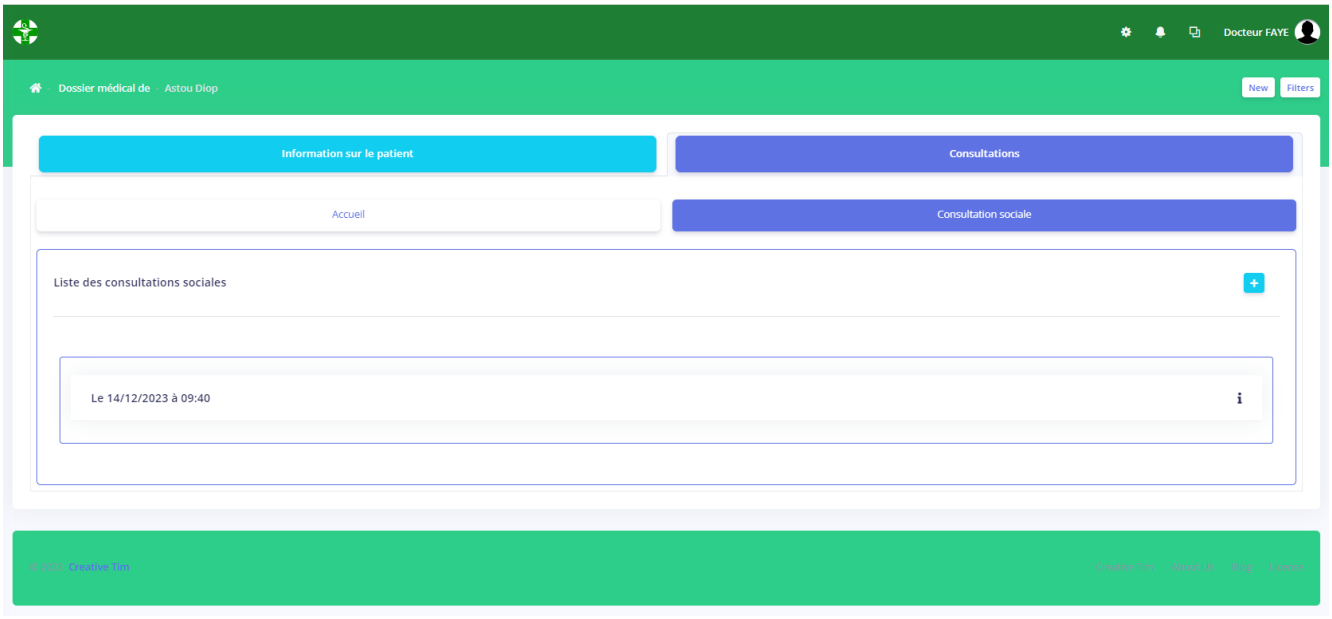

Figure 32 : Exemple d'affichage du dossier médical d'un personnel par médecin social

Dans cette interface, contrairement à celui de l'administrateur, on a le bouton information sur le patient qui regroupe les informations personnelles du patient et le bouton consultations qui ne regroupe que les consultations sociales.

V.5.4. Exemple de vue de la page de gestion de compte par l'administrateur

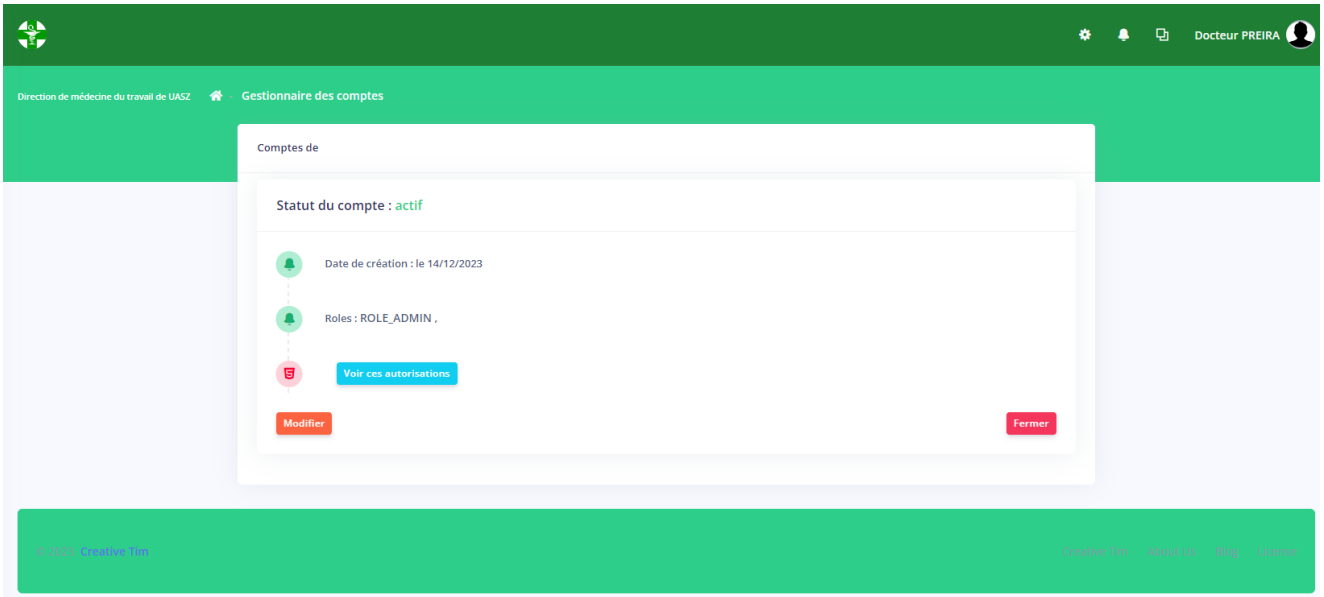

Figure 33 : Page de gestion des comptes utilisateurs.

Cette page (représentée par la figure ci-dessous) n'est accessible que par l'administrateur, elle lui permet de voir l'état du compte et peut y ajouter des modifications comme activer, désactiver ou gérer des autorisations.

Conclusion

Dans cette partie, nous avons décrit les différents outils et langages que nous avons choisis pour la réalisation de cette application. Puis nous avons parlé des tests menés avec deux utilisateurs pour vérifier les droits d'accès. Ensuite, pour tester la communication avec le module de gestion des ressources humaines, nous avons déployé le back-end dans une machine virtuelle installée dans le serveur de l'Université. Et enfin, nous avons présenté quelques interfaces avec deux acteurs différents du système.

Conclusion générale

Pour l'obtention de notre diplôme de Master en Génie logiciel, nous avons eu l'opportunité de réaliser un stage au sein de la Direction de l'Informatique et des Systèmes d'Information (DISI) de l'Université Assane Seck de Ziguinchor (UASZ) dans le cadre du Projet de Gestion Intégrée (PGI). Au cours de ce stage, nous avons eu à travailler sur le module de gestion des Dossiers Médicaux (DM) pour le compte de la Direction de la Médecine du Travail (DMT) de l'UASZ.

Le dossier médical d'un patient est un recueil exhaustif de toutes les informations nécessaires pour prendre en charge et surveiller un patient. Il est considéré comme un indicateur de la qualité des soins et est censé remplir plusieurs fonctions, dont la principale est la fonction de soins. Cependant, sous sa forme papier, il présente de nombreux inconvénients et limites qui nuisent à cette fonction, mais aussi à l'exploitation des données qui pourraient être intéressantes pour la recherche. Par conséquent, l'informatisation du dossier médical peut remédier à ces défauts en le structurant mieux et en utilisant les outils modernes des technologies de l'information.

La mise en place de l'application a débuté par la présentation des structures (DISI et DMT), ainsi que par la mise en évidence de la problématique du sujet dans lequel nous avons parlé des limites du DM sous format papier. Nous avons ensuite adopté la méthodologie Agile pour notre processus de conception, car elle était mieux adaptée à notre méthode de travail, qui nécessitait une forte implication du client et les échanges au sein des membres de l'équipe. Après cela, nous avons procédé à la modélisation et à la conception en commençant par spécifier et analyser les besoins afin d'identifier les différents acteurs du système. Ensuite, nous avons divisé notre application en sous-modules et identifié les fonctionnalités correspondantes pour chaque sous-module à l'aide de diagrammes de cas d'utilisation. Dans la partie conception, nous avons présenté les architectures choisies pour la réalisation de l'application. Enfin, nous avons exposé nos diagrammes de séquences, d'activités et de classes pour conclure cette étape.

À la fin de ce processus, nous avons débuté la mise en œuvre de l'application. Nous avons choisi d'utiliser les outils suivants : Spring Boot pour le back-end et Angular pour le front-end. Enfin, nous avons procédé à la présentation des quelques interfaces de l'application.

Pendant la réalisation de ce travail, nous avons utilisé les connaissances et compétences acquises durant notre formation sur différents processus, technologies et outils tels que 'UML, ainsi que des compétences pratiques, comme l'utilisation des frameworks Spring Boot et

Angular. De plus, nous avons utilisé des outils de développement tels que IntelliJ, ainsi que PowerAMC pour créer les différents diagrammes et PostgreSQL pour gérer la base de données. Nous avons également appris dans ce stage énormément de choses dans un contexte professionnel et le travail en équipe.

Nous pouvons affirmer que nous avons globalement atteint nos objectifs initiaux. Nous avons créé un dossier médical informatisé qui facilite le travail du personnel de la DMT. Cela concerne les visites, les consultations des patients, la sauvegarde des dossiers patients et leur sécurité.

Dans le cadre de ce travail, plusieurs perspectives peuvent être envisagées pour des travaux futurs. Comme par exemple :

- Etablir des statistiques pour faciliter la gestion des bilans du DMT ;
- Effectuer la gestion des rendez-vous ;
- Effectuer la gestion des consultations spéciales ;
- Etc.

Annexes

Annexe 1 : Tableau des modèles générés par PowerAMC

Types de modèle	Définition
Modèle d'architecture d'entreprise	Diagramme de cartographie des processus Diagramme d'organisation Diagramme de communications métier Diagramme d'urbanisation Diagramme orienté service Diagramme d'architecture d'application Diagramme d'infrastructure de technologie
Modèle processus métier	Diagramme de processus métier Diagramme de hiérarchie de processus
Modèle conceptuel de données	Merise IE
Modèle libre	Réseau Collaboration simplifiée Simplified Diagramme de flux simplifié Cas d'utilisation simplifié
Modèle de traitements merise	Diagramme de flux Diagramme conceptuel Diagramme organisationnel
Modèle orienté objet	Diagramme de classes Diagramme objet Diagramme de packages Diagramme de cas d'utilisation Diagramme de séquence Diagramme de communication Diagramme d'interactions Diagramme d'activités Diagramme d'états-transitions Diagramme de composants Diagramme de structures composites Diagramme de déploiement
Modèle physique	Diagramme physique Diagramme multidimensionnel
Modèle XML	XML Schema definition 1.0 XML-Data Reduced 1.0 Document type Definition 1.0
Modèle de fluidité de l'information	IQ Staging Mobilink Replication Server
Modèle de gestion des exigences	Vues Document des exigences Vue Matrice de traçabilité Vue Matrice des affectations des utilisateurs

Tableau 9 : Modèles générer par PowerAMC [10]

Annexe 2 : Tableau de codes générés par PowerAMC

Catégorie	Langage
Langages processus	BPEL4WS1.1 WSBPEL 2.0 ebXML BPSS v1.01 ebXML BPSS v1.04
Langages objets	C# C# 2.0 IDL - CORBA Powerbuilder Java Java 1.x Visual Basic .Net Visual Basic 2005 XML-DTD XML Schema
Bases de données	ADABAS D ALLBASE/SQL G.1 AS400 ANSI2 IBM DB2/UDB Informix Ingres InterBase MS Access MS SQL Serveur MySQL NonStop SQL ODBC 3.0 Oracle PostgreSQL Red Brick Warehouse Sybase ASE/ASA/ASIQ/Avaki Teradata
Définitions XML	Document Type Definition 1.0 XML Schema Definition 1.0 XML-Data Reduced 1.0

Tableau 10 : Codes générer par PowerAMC [10]

Annexe 3 : Diagramme de séquence du processus de sécurité pour l'accès aux données.

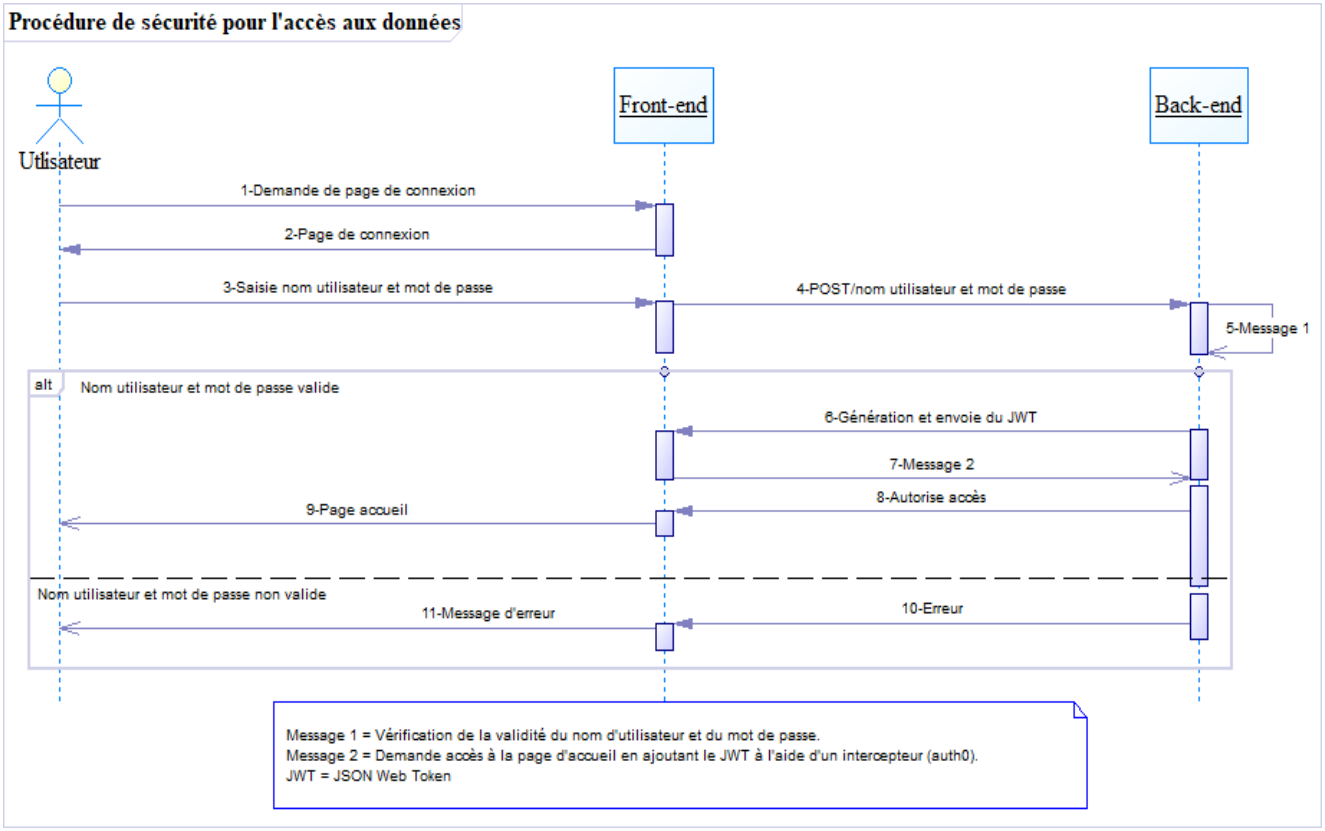

Figure 34 : Diagramme de séquence de la procédure de sécurité pour l'accès aux données de l'application

Annexe 4 : Diagramme de classe des informations personnelles des patients

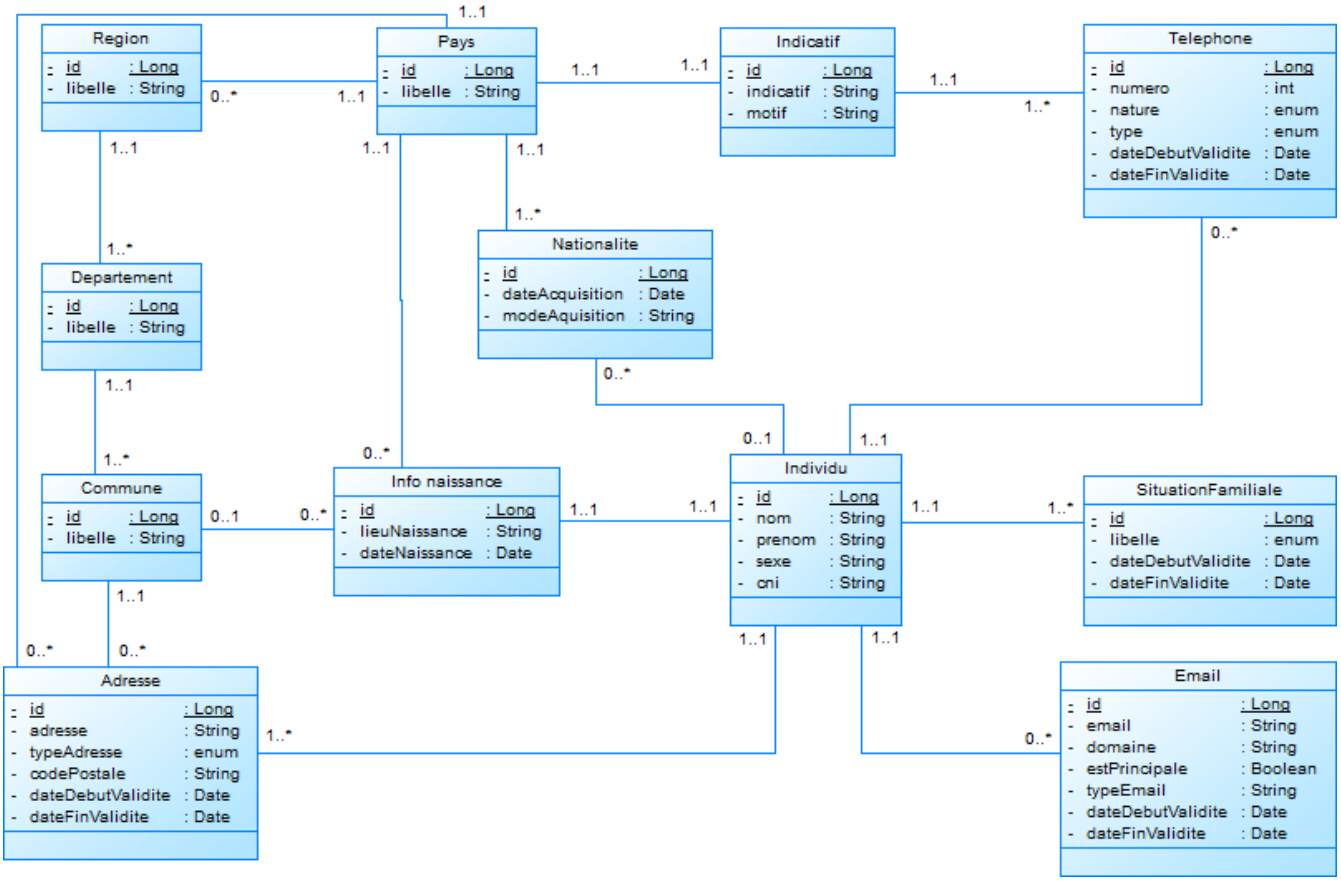

Figure 35 : Diagramme de classe des informations personnelles des patients

Annexe 5 : Diagramme de classe des informations professionnelles des patients

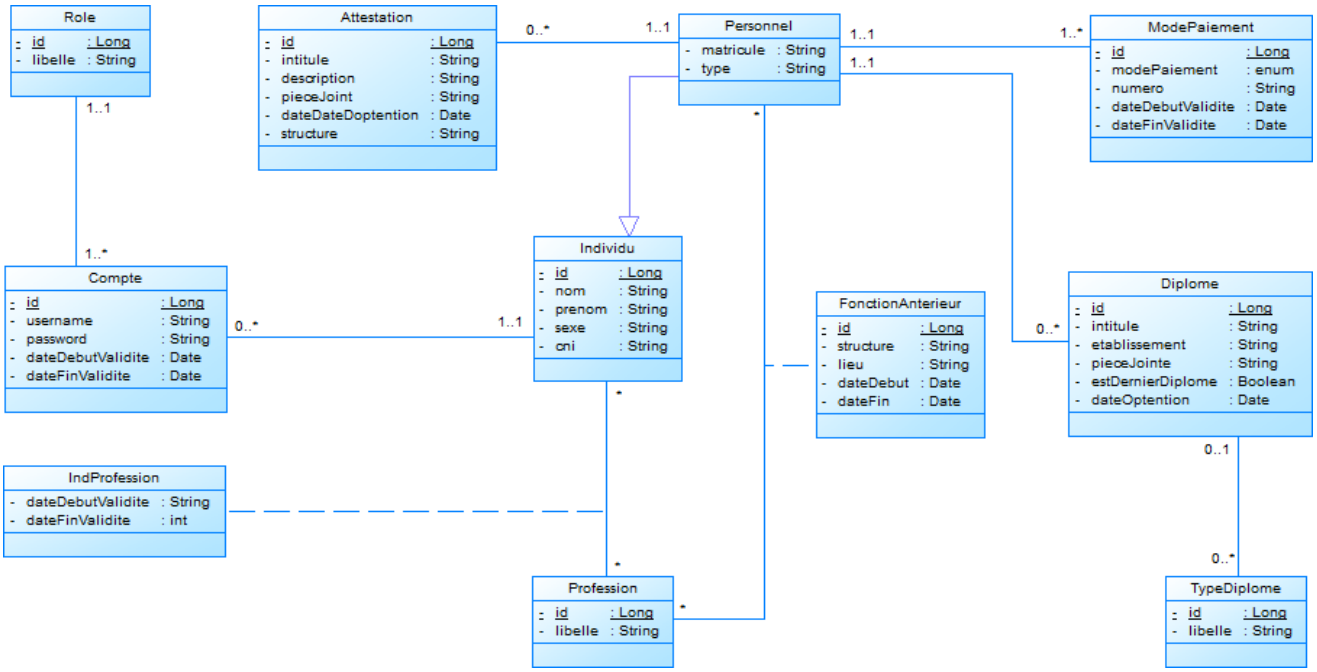

Figure 36 : Diagramme de classe des informations professionnelles des patients

Annexe 6 : Dictionnaire des données

activite_logicielle		
Colonne	**Type**	**Null**
id (Primaire)	bigint(20)	Non
date_debut_validite	datetime(6)	Oui
date_fin_validite	date	Oui
libelle	varchar(255)	Non
analyses		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
libelle	varchar(255)	Non
type_analyse_id	bigint(20)	Oui
appareil		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
libelle	varchar(255)	Non
cdiagnostique		
Colonne	**Type**	**Null**
consultation_id *(Primaire)*	bigint(20)	Non
maladies_id *(Primaire)*	bigint(20)	Non
observation	varchar(255)	Non
commune		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
Libelle	varchar(255)	Oui
departements_id	bigint(20)	Oui
compte_roles		
Colonne	**Type**	**Null**
compte_id *(Primaire)*	bigint(20)	Non
role_id *(Primaire)*	bigint(20)	Non
consomations_taba_alcool		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
alcoolisme	bit(1)	Oui
nombre_taba	int(11)	Non
nombre_verre	int(11)	Non
tabagisme	bit(1)	Oui
visite_id	bigint(20)	Oui
consultation		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
archiver	bit(1)	Oui
date_consultation	datetime(6)	Oui
dernier_consultion	datetime(6)	Oui
frequence_respiratoire	int(11)	Non
motifs	varchar(255)	Non

poids	float	Oui
pouls	varchar(255)	Non
suivi	bit(1)	Oui
taille	float	Oui
temperature	float	Oui
tension	varchar(255)	Oui
medecin_id	bigint(20)	Non
patient_id	bigint(20)	Non
consultation_analyses		
Colonne	**Type**	**Null**
analyse_id *(Primaire)*	bigint(20)	Non
consultation_id *(Primaire)*	bigint(20)	Non
description	varchar(255)	Non
resultat	varchar(255)	Non
consultation_antecedents		
Colonne	**Type**	**Null**
consultation_id *(Primaire)*	bigint(20)	Non
maladies_id *(Primaire)*	bigint(20)	Non
type_antecedent	varchar(255)	Oui
consultation_hospitalisation		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_hospitalisation	date	Oui
lieu	varchar(255)	Non
Motif	varchar(255)	Non
Observation	varchar(255)	Non
consultation_id	bigint(20)	Oui
Cprescription		
Colonne	**Type**	**Null**
consultation_id *(Primaire)*	bigint(20)	Non
medicaments_id *(Primaire)*	bigint(20)	Non
dosage	varchar(255)	Non
posologie	varchar(255)	Non
quantite	varchar(255)	Non
crepos_medical		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_debut	datetime(6)	Oui
date_fin	date	Oui
consultation_id	bigint(20)	Oui
departements		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
libelle	varchar(255)	Oui
region_id	bigint(20)	Oui
examen_appareil_consultation		
Colonne	**Type**	**Null**

appareil_id *(Primaire)*	bigint(20)	Non
consultation_id *(Primaire)*	bigint(20)	Non
etat	varchar(255)	Non
observation	varchar(255)	Non
examen_appareil_visite		
Colonne	**Type**	**Null**
appareil_id *(Primaire)*	bigint(20)	Non
visite_id *(Primaire)*	bigint(20)	Non
etat	varchar(255)	Non
observation	varchar(255)	Non
examen_medical		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
description	varchar(255)	Non
libelle	varchar(255)	Non
examen_medical_consultation		
Colonne	**Type**	**Null**
consultation_id *(Primaire)*	bigint(20)	Non
examen_medical_id *(Primaire)*	bigint(20)	Non
conclusion	varchar(255)	Non
observation	varchar(255)	Non
examen_medical_visite		
Colonne	**Type**	**Null**
examen_medical_id *(Primaire)*	bigint(20)	Non
visite_id *(Primaire)*	bigint(20)	Non
conclusion	varchar(255)	Non
observation	varchar(255)	Non
indicatif		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
indicatif	varchar(255)	Non
motif	varchar(255)	Non
pays_id	bigint(20)	Oui
individu		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
cni	varchar(255)	Oui
nom	varchar(255)	Non
prenom	varchar(255)	Non
sexe	varchar(255)	Non
ind_adresse		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
adresse	varchar(255)	Non
code_postale	varchar(255)	Oui
date_debut_validite	datetime(6)	Oui
date_fin_validite	datetime(6)	Oui

commune_id	bigint(20)	Oui
ind_type_adresse_id	bigint(20)	Oui
individu_id	bigint(20)	Oui
pays_id	bigint(20)	Oui
ind_attestation		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_obtention	date	Oui
description	varchar(255)	Oui
intitule	varchar(255)	Oui
piece_joint	varchar(255)	Oui
structure	varchar(255)	Oui
personnel_id	bigint(20)	Non
ind_compte		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
actif	bit(1)	Non
date_debut_validite	date	Oui
date_fin_validite	date	Oui
password	varchar(255)	Non
username	varchar(255)	Non
individu_id	bigint(20)	Oui
ind_contact_urgent		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
email	varchar(255)	Oui
nom	varchar(255)	Oui
prenom	varchar(255)	Oui
telephone	varchar(255)	Oui
individu_id	bigint(20)	Oui
ind_diplome		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_obtention	date	Oui
est_derniere_diplome	bit(1)	Non
etablissement	varchar(255)	Oui
intitule	varchar(255)	Oui
piece_joint	varchar(255)	Oui
personnel_id	bigint(20)	Non
type_diplome_id	bigint(20)	Non
ind_email		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_debut_validite	datetime(6)	Oui
date_fin_validite	date	Oui
domaine	varchar(255)	Non
email	varchar(255)	Non

est_email_principale	bit(1)	Oui
type_email	int(11)	Non
individu_id	bigint(20)	Oui
ind_enfant		
Colonne	**Type**	**Null**
extrait	varchar(255)	Oui
num_extrait	int(11)	Oui
id *(Primaire)*	bigint(20)	Non
individu_mere	bigint(20)	Non
individu_pere	bigint(20)	Non
ind_etudiant		
Colonne	**Type**	**Null**
ine	varchar(255)	Non
num_etudiant	int(11)	Non
id *(Primaire)*	bigint(20)	Non
ind_externe		
Colonne	**Type**	**Null**
code	varchar(255)	Oui
date_enregistrement	datetime(6)	Oui
id *(Primaire)*	bigint(20)	Non
ind_fonction_anterieur		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_debut	date	Oui
date_fin	date	Oui
lieu	varchar(255)	Oui
structure	varchar(255)	Oui
personnel_id	bigint(20)	Non
profession_id	bigint(20)	Non
ind_individu_activite_logicielle		
Colonne	**Type**	**Null**
activite_logicielle_id *(Primaire)*	bigint(20)	Non
individu_id *(Primaire)*	bigint(20)	Non
date_debut_validite	datetime(6)	Oui
date_fin_validite	date	Oui
ind_info_naissance		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_naissance	Date	Oui
lieu_naissance	varchar(255)	Oui
commune_id	bigint(20)	Oui
individu_id	bigint(20)	Oui
pays_id	bigint(20)	Oui
ind_mariage		
Colonne	**Type**	**Null**
individu_epouse *(Primaire)*	bigint(20)	Non
individu_epoux *(Primaire)*	bigint(20)	Non

certifica_mariage	varchar(255)	Oui
date_fin	date	Oui
date_mariage	date	Oui
lieu_mariage	varchar(255)	Oui
limite_mariage	int(11)	Non
regime	varchar(255)	Oui
ind_nationalite		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_acquisition	date	Oui
mode_acquisition	varchar(255)	Oui
individu_id	bigint(20)	Non
pays_id	bigint(20)	Non
ind_personnel		
Colonne	**Type**	**Null**
est_fonctionnaire	bit(1)	Non
est_informaticien_synpics	varchar(255)	Oui
matricule	varchar(8)	Oui
statut	varchar(255)	Oui
type	varchar(255)	Oui
id *(Primaire)*	bigint(20)	Non
ind_profession		
Colonne	**Type**	**Null**
individu_id *(Primaire)*	bigint(20)	Non
profession_id *(Primaire)*	bigint(20)	Non
date_debut_validite	datetime(6)	Oui
date_fin_validite	date	Oui
ind_situation_familliale		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_debut_validite	datetime(6)	Oui
date_fin_vadilite	date	Oui
libelle	varchar(255)	Oui
individu_id	bigint(20)	Non
ind_telephone		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_debut_validite	datetime(6)	Oui
date_fin_validite	date	Oui
nature	varchar(255)	Oui
numero_telephone	int(11)	Non
type	varchar(255)	Oui
indicatif_id	bigint(20)	Oui
individu_id	bigint(20)	Oui
ind_type_adresse		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non

libelle	**varchar(255)**	Non
ind_type_diplome		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
libelle	varchar(255)	Non
ind_vacataire_externe		
Colonne	**Type**	**Null**
code	varchar(255)	Oui
date_enregistrement	datetime(6)	Oui
id *(Primaire)*	bigint(20)	Non
maladies		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
libelle	varchar(255)	Non
type_maladies_id	bigint(20)	Oui
medicaments		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
libelle	varchar(255)	Non
organisation		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
abreviation	varchar(255)	Non
arrete_creation	varchar(255)	Oui
date_creation	date	Non
email	varchar(255)	Oui
libelle	varchar(255)	Non
type	varchar(255)	Non
organisation_mere_id	bigint(20)	Oui
pays		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
libelle	varchar(255)	Non
profession		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
libelle	varchar(255)	Oui
region		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
libelle	varchar(255)	Oui
pays_id	bigint(20)	Oui
responsabilites		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_debut_vadilite	datetime(6)	Oui
date_fin_vadilite	date	Oui

libelle	**varchar(255)**	Non
responsabilite_organisation		
Colonne	**Type**	**Null**
organisation_id *(Primaire)*	bigint(20)	Non
responsabilite_id *(Primaire)*	bigint(20)	Non
date_debut_validite	datetime(6)	Oui
date_fin_validite	date	Oui
roles		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
name	varchar(255)	Oui
type_analyse		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
type	varchar(255)	Non
type_maladies		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
type	varchar(255)	Non
visite		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
archiver	bit(1)	Oui
date_visite	datetime(6)	Oui
poids	float	Oui
tailles	float	Oui
tension	varchar(255)	Oui
temperature	float	
frequenceRespiratoire	varchar(255)	
tdr	bit(1)	
medecin_id	bigint(20)	Non
visite_analyses		
Colonne	**Type**	**Null**
analyse_id *(Primaire)*	bigint(20)	Non
visite_id *(Primaire)*	bigint(20)	Non
description	varchar(255)	Non
resultat	varchar(255)	Non
visite_annuelle		
Colonne	**Type**	**Null**
condition_travail	varchar(255)	Oui
rapport_service	varchar(255)	Oui
risque_travail	varchar(255)	Oui
id *(Primaire)*	bigint(20)	Non
patient_id	bigint(20)	Oui
visite_antecedents		
Colonne	Type	**Null**
maladies_id *(Primaire)*	bigint(20)	Non

visite_id *(Primaire)*	bigint(20)	Non
type_antecedent	varchar(255)	Oui
visite_embauche		
Colonne	Type	**Null**
reponse	varchar(255)	Oui
id *(Primaire)*	bigint(20)	Non
patient_id	bigint(20)	Oui
visite_hospitalisation		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_hospitalisation	date	Oui
Lieu	varchar(255)	Non
Motif	varchar(255)	Non
observation	varchar(255)	Non
visite_id	bigint(20)	Oui
id *(Primaire)*	bigint(20)	Non
patient_id	bigint(20)	Oui
visite_reprise		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
patient_id	bigint(20)	Oui
ind_mode_paiement		
Colonne	**Type**	**Null**
id *(Primaire)*	bigint(20)	Non
date_debut_validite	datetime(6)	Oui
date_fin_validite	date	Oui
name	varchar(255)	Oui
numero	varchar(255)	Oui
personnel_id	bigint(20)	Non

Tableau 11 : Dictionnaire des données

Bibliographie

[1] S. Pelletier, « Le dossier médical », *Journal du droit des jeunes*, vol. 223, n° 3, p. 15-19, 2003, doi: 10.3917/jdj.223.0015.

[2] T. Collonvillé, « Elaboration de processus de développements logiciels spécifiques et orientés modèles: application aux systèmes à évenements discrets ».

[3] V. Messager Rota, *Gestion de projet agile*, 3e éd. in Architecte logiciel. Paris: Eyrolles, 2010.

[4] V. Messager Rota, *Gestion de projet agile*, 3e éd. in Architecte logiciel. Paris: Eyrolles, 2010.

[5] K. Jridi, « Processus Unifié et Approche Agile », Disponible sur: https://www.academia.edu/31823113/Processus_Unifi%C3%A9_et_Approche_Agile

[6] G. Benefield, C. Larman, et B. Vodde, « Guide Léger de la Théorie et de la Pratique de Scrum ».

[7] « ToolApp | Méthode agile "SCRUM" », ToolApp. Disponible sur: https://toolapp.fr/methode-agile-scrum/

[8] *Les pratiques de l'équipe agile*. Disponible sur: https://www.calameo.com/read/000015856334a13229f41

[9] D. Team, « Qu'est-ce que la méthode agile ? », Formation Data Science | DataScientest.com. Disponible sur : https://datascientest.com/quest-ce-que-la-methode-agile

[10] « Livre Passez au DevOps - Votre nouvelle façon de travailler ». Disponible sur : https://www.editions-eni.fr/livre/passez-au-devops-votre-nouvelle-facon-de-travailler-9782409038228

[11] J. Wiley, « DevOps Pour LES NULS 2ème Edition Limitée IBM », 2015.

[12] Julie, « DevOps », Harwell Management. Disponible sur : https://harwell-management.com/2019/09/18/devops/

[13] « Jenkins User Documentation », Jenkins User Documentation. Disponible sur : https://www.jenkins.io/doc/

[14] « Docker overview », Docker Documentation. Disponible sur : https://docs.docker.com/get-started/overview/

[15] « Documentation de Kubernetes », Kubernetes. Disponible sur : https://kubernetes.io/fr/docs/home/

[16] « Cloud Native Computing Foundation », Cloud Native Computing Foundation. Disponible sur : https://www.cncf.io/

[17] « Qu'est-ce que Git ? » Disponible sur : https://www.next-decision.fr/wiki/qu-est-ce-que-git

[18] C. Reynaud, « La méthode S.A. "Structured Analysis" ».

[19] C. Solnon, « Modélisation UML ».

[20] B. Charroux, A. Osmani, et T.-M. Yann, *Uml 2 pratique de la modélisation*. 2015. Disponible sur : https://fr.slideshare.net/nassimamine3994/uml-2-pratique-de-la-modlisation

[21] R. Leblond, « Vers une architecture n-tiers ».

[22] « Monolithique et microservices : différence entre les architectures de développement logiciel- AWS », Amazon Web Services, Inc. Disponible sur : https://aws.amazon.com/fr/compare/the-difference-between-monolithic-and-microservices-architecture/

[23] *SOA, Microservices, API management*. Disponible sur : https://www.cantook.net/p/3430656?f=pdf

[24] « Structurez et configurez votre projet », OpenClassrooms. Disponible sur : https://openclassrooms.com/fr/courses/6900101-creez-une-application-java-avec-spring-boot/7077993-structurez-et-configurez-votre-projet
[25] A. Daoudi, « Étude des patrons architecturaux de type MVC dans les applications Android ».
[26] « Une conception sur le moèle MVC ». Disponible sur : https://perso.telecom-paristech.fr/hudry/coursJava/interSwing/boutons5.html
[27] « Qu'est-ce qu'un diagramme de classes? - définition de techopedia - Développement 2023 », Icy Science. Disponible sur: https://fr.theastrologypage.com/class-diagram
[28] « Qu'est-ce que Java Spring Boot ?-Présentation de Spring Boot | Microsoft Azure ». Disponible sur: https://azure.microsoft.com/fr-fr/resources/cloud-computing-dictionary/what-is-java-spring-boot/
[29] « Spring Boot Reference Documentation ». Disponible sur: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
[30] « Angular ». Disponible sur: https://angular.io/
[31] « Semantic UI ». Disponible sur: https://semantic-ui.com/
[32] S. W. & M. DESIGN, « Ink - Interface Kit », Ink. Disponible sur: https://ink.sapo.pt/
[33] « React – Une bibliothèque JavaScript pour créer des interfaces utilisateurs ». Disponible sur: https://fr.legacy.reactjs.org/
[34] « MakinaCorpus_LivreBlanc_FrontEnd_v1-7.pdf ». Disponible sur: https://makina-corpus.com/sites/default/files/2022-07/MakinaCorpus_LivreBlanc_FrontEnd_v1-7.pdf
[35] « REST – Architecture Orientée Ressource | Loic's Blog ». Disponible sur: https://www.loicmathieu.fr/wordpress/informatique/rest-architecture-orientee-ressource/
[36] « Qu'est-ce qu'une base de données ». Disponible sur: https://www.oracle.com/fr/database/definition-base-de-donnees/
[37] « PostgreSQL : à propos ». Disponible sur: https://www.postgresql.org/about/
[38] M. Mamou, « IDE : qu'est-ce qu'un environnement de développement intégré ? », Formation Data Science | DataScientest.com. Disponible sur: https://datascientest.com/ide
[39] L. Galiana, *Chapitre 3 Utiliser GIT avec | Travail collaboratif avec R*. Disponible sur: https://linogaliana.gitlab.io/collaboratif/git.html
[40] « Premium Bootstrap Themes and Templates ». Disponible sur: https://www.creative-tim.com/
[41] « Postman documentation overview », Postman Learning Center. Disponible sur: https://learning.postman.com/docs/introduction/overview/
[42] F. Mary, « How to deploy a Spring Boot - Angular application on Ubuntu server », The Code She Writes. Disponible sur: https://thecodeshewrites.com/2020/10/13/how-to-deploy-spring-boot-angular-ubuntu-server/