UNIVERSITÉ ASSANE SECK-ZIGUINCHOR

UFR Sciences Économiques et Sociales

Département d'Économie-Gestion

Mémoire pour l'obtention du Master Management des Systèmes d'Informations Automatisés

Spécialité : Management Informatisé des Organisations

Vers la mise en place d'un « Guichet Médical Unique » pour le Sénégal

Présenté par :

David Jérémy .D. DIEME

Sous la direction de :

Dr Papa Alioune CISSE

Sous la supervision de :

Pr Mélyan MENDY

Jury :

Président : Pr Mélyan MENDY, Professeur Assimilé, UASZ

Examinateur : Dr Lamine FATY, Enseignant Associé, UASZ

Examinateur : Dr Mouhamadou GAYE, M^{e} de Conférences Titulaire, UASZ

Dire de mémoire : Dr Papa Alioune CISSE, M^{e} de Conférence Titulaire, UASZ

ANNÉE 2022-2023

DEDICACE

Je dédie ce mémoire à toute ma famille et à tous ceux qui ont participé à ma formation. Mention spéciale à :

Ma chère mère Emma SAGNA
Mon père Alexis DIEME
Mes sœurs Vianney et Louise
Ma tante Margot DIEME et toute la famille DIEDHIOU
Mon binôme de guerre Namory DIAWARA
Ya Rouhi
Lybro et Modou
Daouda MBAYE

REMERCIEMENTS

La réalisation de ce mémoire a été possible grâce à plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

Je commence par mon encadreur le Dr Papa Alioune CISSE, je tiens à vous exprimer mes sincères remerciements car vous avez su être un encadreur plus que disponible malgré vos nombreuses charges. Votre rigueur, votre sens de l'engagement et votre clairvoyance m'ont motivé à arriver au bout du tunnel.

J'exprime tous mes remerciements à l'ensemble des membres du jury.

Je remercie tout le personnel de l'Université Assane Seck de Ziguinchor.

Un grand merci à mon Directeur des études Monsieur BALDE et à tous les professeurs de ma formation. Ils m'ont fourni les outils nécessaires à la réussite de mes études universitaires.

A mes ami(e)s de la promotion MSIA 2021-2022.

RÉSUMÉ

Une application de gestion des données médicales est une solution numérique conçue pour faciliter le stockage, la gestion et l'accès sécurisé aux informations médicales des patients. Elle offre une plateforme centralisée permettant aux professionnels de la santé de consigner, mettre à jour et partager efficacement les données médicales, améliorant ainsi la coordination des soins.

Dans ce mémoire, nous présentons un projet visant à mettre en place un « Guichet Médical Unique » pour le Sénégal dans lequel chaque patient et son dossier médical existent en un seul et unique exemplaire ; des espaces de types « Family », « Health Structure » et « Non Health Structure » sont ajoutés pour permettre respectivement aux patients (dans un cadre familial), aux structures de santé et aux structures non sanitaires (banques, assurances, etc.) d'accéder aux dossiers médicaux des patients. Cette plateforme propose également des fonctionnalités telles que la création de dossiers médicaux électroniques, l'enregistrement des antécédents médicaux, des diagnostics, des traitements et des résultats d'examens. Elle intègre également des outils de suivi des prescriptions, des rappels de rendez-vous et des alertes pour optimiser la prise en charge des patients.

L'accent est mis sur la sécurité et la confidentialité des données, avec des protocoles de cryptage avancés et des contrôles d'accès stricts pour garantir le respect des normes réglementaires en matière de protection des informations médicales. En facilitant la communication entre les professionnels de la santé, la plateforme contribue à une meilleure coordination des soins, réduisant les erreurs médicales et améliorant l'efficacité des traitements. Les patients bénéficient également d'un accès sécurisé à leurs propres données, renforçant ainsi l'autonomie et la transparence dans leur parcours de soins.

Dans le cadre de ce mémoire, notre contribution principale à ce projet est de proposer un "modèle" de « dossier médical » pour les patients, de développer le module de gestion des autorisations d'accès aux dossiers médicaux et de mettre en place l'espace d'administration globale du guichet.

ABSTRACT

A medical data management application is a digital solution designed to facilitate the storage, management, and secure access to patient medical information. It provides a centralized platform for healthcare professionals to efficiently record, update, and share medical data, improving care coordination.

In this thesis, we present a project to set up a "Single Medical Window" for Senegal in which each patient and his or her medical file exists in a single copy; "Family", "Health Structure" and "Non-Health Structure" spaces have been added to allow patients (in a family setting), health structures and non-health structures (banks, insurance companies, etc.) to access patients' medical records, respectively. This platform also offers features such as the creation of electronic medical records, recording of medical history, diagnoses, treatments, and test results. It also includes prescription tracking tools, appointment reminders and alerts to optimize patient care.

The focus is on data security and privacy, with advanced encryption protocols and strict access controls to ensure compliance with regulatory standards for the protection of medical information. By facilitating communication between healthcare professionals, the platform contributes to better coordination of care, reducing medical errors and improving the efficiency of treatments. Patients also benefit from access to their own data, thus enhancing autonomy and transparency in their care pathway.

In the framework of this thesis, our main contribution to this project is to propose a "model" of "medical record" for patients, to develop the module for managing access authorizations to medical records and to set up the global administration space of the counter.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABREVIATIONS

- HTML : HyperText Markup Language
- CSS : Cascading Style Sheets
- MVT : Modèle-vue-template
- ORM : object-relational mapping
- URL : Uniform Resource Locator
- CRUD : create, read, update, delete
- SQL : Structured Query Language
- SGBD : Système de Gestion de Bases de Données
- SGBDR : Système de Gestion de Bases de Données Relationnelles
- UML : Unified Modeling Language
- VSCode : Visual Studio Code

INTRODUCTION GENERALE

1. Contexte Général

En faisant le tour d'un bon nombre de structures de santé locales (hôpitaux de niveaux 1, 2, 3 ; centres de santé, cabinets médicaux privés, etc.), nous avons constaté principalement deux situations : certaines structures sont dotées de systèmes informatisés de gestion tandis que d'autres restent encore totalement dans la gestion manuelle (avec papiers et stylos).

Ces systèmes ont encore de l'importance dans la production de la donnée médicale, mais ne sont pas adaptés face aux nouveaux défis contemporains de gestion, d'utilisation et de traitement de la donnée sanitaire. En effet, aujourd'hui, avec les nouveaux outils d'intelligence artificielle, de Business Intelligence (BI), … les données médicales, sanitaires et extra sanitaires sont utilisés pour prévenir et diagnostiquer plus rapidement des maladies, prédire des épidémies, mieux comprendre certaines pathologies, améliorer et personnaliser les traitements chez les patients, etc.

2. Problématiques de notre étude

Lors des échanges que nous avons eus avec des médecins et des directeurs de structures de santé, certains nous ont fait comprendre que cela ne les intéressait pas d'utiliser un système informatique qui ne répondrait pas à leurs attentes. Nous avons essayé de lister ci-dessous quelques attentes qu'ils ont exprimées.

- Un médecin nous a proposé un sujet de réflexion scientifique à savoir si pour une pathologie P donnée, on peut proposer 3 traitements (T1, T2, T3). Comment savoir quel traitement parmi les trois est mieux adapté pour traiter la pathologie P chez un patient donné, en fonction de critères d'ordre démographique, culturel, professionnel, géographique, etc. C'est dans cette perspective qu'il demande toujours à ses patients de revenir même s'ils sont guéris de leurs maladies afin de déterminer la durée et l'efficacité des traitements prescrits. Pour cela, il dit disposer d'une quantité importante de données sur papiers dont il ne sait pas quoi en faire. En effet, un traitement manuel de ces données est quasi impossible et les outils informatiques qu'on lui présente jusqu'ici ne permettent pas de répondre à son type de questionnements.

- Un chirurgien pédiatre nous dit qu'il intervient beaucoup pour opérer des hernies ombilicales et c'est une situation anormale dont on devrait avoir une explication sur les causes et les origines.
- Un autre médecin dit qu'un patient qui était suivi pour une pathologie donnée dans un cabinet d'une ville donnée, a été affecté récemment dans sa ville. Il doit le suivre mais n'a pas accès à son historique médical et le patient non plus n'y a pas accès. Il se demande quelles solutions face à ce problème.
- Un autre médecin nous a transmis une requête d'un patient qui est chef de famille et qui a en charge la santé de tous les membres mais parfois quand il est en voyage c'est compliqué pour lui de suivre à distance. Il aimerait avoir accès aux dossiers médicaux à distance, prendre des rendez-vous et les payer à distance.
- Un directeur d'une structure de santé nous rapporte aussi les préoccupations d'un de ses homologues banquiers qui veut pouvoir accéder aux données médicales pour savoir si un client est solvable ou pas.

3. Objectif et intérêt de notre étude

L'objectif principal de ce travail est donc de proposer une application de gestion de données médicales qui prendra en compte les préoccupations établies ci-dessus. Il s'agit entre autres de :

- Permettre aux patients d'accéder eux-mêmes à leurs propres données médicales.
- Permettre aux patients d'accorder l'accès à certaines parties de leurs données médicales à des structures de santé et à d'autres structures telles que les assurances, mutuelles de santé, banques, etc.
- Permettre aux patients de gérer aussi les données médicales des membres de leur famille.
- Fournir des services d'ordre médical et administrative tels que : la réservation à distance de rendez-vous médicaux, le paiement en ligne des frais médicaux, le suivi médical à distance, la gestion des files d'attente dans les structures de santé, l'aide à la décision avec l'utilisation d'outils BI (Business Intelligence) et IA (Intelligence Artificielle) : Tableaux de bord et apprentissage automatique pour l'aide au diagnostic et aux traitements.

4. Plan du mémoire

Ce document est réparti en trois principaux chapitres :

- Le premier chapitre, intitulé « état de l'art et généralités du projet », présente les généralités du projet dans le cadre duquel se situent nos travaux de mémoire.
- Le deuxième chapitre intitulé « Analyse et conception » donne un aperçu sur UML qui est l'outil d'analyse et de conception utilisé dans ce travail.
- Le troisième et dernier chapitre décrit, d'abord, les technologies utilisées pour implémenter notre solution. Ensuite, nous présentons les détails d'implémentation de la solution. Enfin, nous présentons le module « gestionnaire des simulations » développé, ses fonctionnalités et ses interfaces.

Enfin, nous terminerons par une conclusion générale, sans oublier d'évoquer d'éventuelles perspectives.

Chapitre 1. État de l'art et généralités du projet global

1. Définitions de quelques concepts

Dans cette partie, nous allons clarifier les différents concepts énumérés dans notre document pour qu'ils soient plus explicites.

1.1. Conception

La conception est la phase créative d'un projet d'ingénierie. Le but premier de la conception est de permettre de créer un système ou un processus répondant à un besoin en tenant compte des contraintes. Le système doit être suffisamment défini pour pouvoir être installé, fabriqué, construit et être fonctionnel pour répondre aux besoins du client [1].

1.2. Application

Une « application » est un programme directement utilisé pour réaliser une tâche et/ou assurer un ensemble de fonctions précises. Notons qu'un programme est tout simplement un ensemble d'instructions destiné à être exécuter par un ordinateur. Une instruction peut être par exemple une information communiquée. Cette information peut être à la fois une commande et/ou une explication pour décrire une action, un comportement, ou une tâche qui devra s'exécuter [2].

Au fil du temps, trois (3) grandes familles d'applications ont vu le jour, chacun avec sa particularité. Nous avons les applications mobiles, les applications web, les applications desktops.

- **Application mobile**

Une application mobile c'est en premier lieu un logiciel. Un programme téléchargeable sur smartphone ou tablette qui comporte un fichier qui est installé puis exécuté par le système d'exploitation de votre mobile. Ce fichier est codé dans un langage de développement spécifique à votre appareil :

- Java ou Kotlin pour Android (smartphones et tablettes Samsung par exemple)
- Objective C ou Swift pour IOS (appareils Apple).

En fonction de chaque cas, les technologies et les langages de développement utilisés vont être différents et chaque sorte d'application mobile a ses spécificités. A savoir qu'il est possible de

développer une application qui est capable de fonctionner sur les deux systèmes d'exploitation (IOS et Android) : on appelle cela une application hybride [3].

- **Application web**

Une application Web est un logiciel applicatif qui affiche son interface homme-machine dans un navigateur web. Le logiciel est hébergé par un serveur web transformé. Il est exécuté de façon partagée par les ordinateurs serveur et client (l'usager). Cette technologie est apparue dans les années 1990.

L'usage d'un serveur de fichiers permet d'héberger sur un ordinateur donné les fichiers d'une application comme s'ils étaient hébergés par l'ordinateur de l'usager. Un serveur de fichier est typiquement utilisé pour héberger sur un ordinateur unique les fichiers d'une application utilisée par un ensemble d'usages qui ont chacun leur ordinateur personnel.

Les premières applications web avaient une ergonomie et une maniabilité moindre que les applications de bureau. Une « application web riche » (anglais *Rich Internet Application*) est par définition une application web qui a une ergonomie et une maniabilité équivalente à une application de bureau. La notion d'« application web riche » a été introduite par Macromedia en 2002 [4].

- **Une application desktop**

Une « application de bureau » (« *Desktop application* » en anglais) est un logiciel applicatif qui affiche son interface graphique dans un environnement de bureau, il est hébergé et exécuté par l'ordinateur de l'usager. Cette technologie est apparue avec les premiers environnements de bureau en 1970 [5].

1.3. Structure de santé

Une structure (ou établissement) de santé est une structure définie par un statut légal, et dont les missions sont fixées par le Code de la santé publique. Ces missions sont exécutées dans le cadre d'un système de valeurs et d'obligations de service public. La compétence d'un établissement de santé peut être de nature communale, intercommunale, départementale, régionale, ou nationale.

L'établissement de santé (hôpital, clinique…) a pour but, conformément à l'article L6111-1 du Code de la santé publique :

- D'assurer le diagnostic, le suivi et le traitement des malades, des blessés et des femmes enceintes ;
- Délivrer les soins avec hébergement, sous forme ambulatoire ou à domicile ;
- Participer à la coordination des soins en relation avec les membres des professions de santé exerçant en pratique de ville ;
- Participer à la mise en œuvre de la politique de santé publique et des mécanismes de contrôle pour assurer la sécurité sanitaire ;
- Mener, en leur sein, une réflexion sur l'éthique liée à l'accueil et la prise en charge médicalisée.

Pour l'exécution de ces missions, ces derniers peuvent dispenser, avec ou sans hébergement des patients, en fonction de leur état de santé :

- Des soins de courte durée ou courts séjours, prenant en charge des affections graves pendant leur phase aiguë en médecine, chirurgie, obstétrique, odontologie ou psychiatrie ;
- Des soins de suite et de réadaptation ou moyens séjours, qui ont pour objet la rééducation ou la réadaptation de patients qui connaissent des déficiences ou des limitations de capacité et de promouvoir leur réadaptation et leur réinsertion [6].

1.4. Gestion des rendez-vous

La gestion des rendez-vous dans une application de santé est une composante cruciale pour assurer un flux de travail efficace et offrir une expérience positive aux utilisateurs, qu'ils soient patients, professionnels de la santé ou administrateurs. Voici quelques éléments clés de la gestion des rendez-vous dans une application de santé :

- Prise de rendez-vous en ligne :

Permettre aux patients de prendre des rendez-vous en ligne offre une commodité accrue. Les utilisateurs peuvent choisir des créneaux disponibles sans avoir à appeler le cabinet médical.

Intégration de fonctionnalités de calendrier intuitives pour afficher les disponibilités des praticiens.

- Confirmation et rappels automatiques :

L'application doit envoyer des rappels automatiques aux patients pour confirmer leur rendez-vous. Cela peut réduire les rendez-vous manqués et optimiser la planification.

- Synchronisation avec les calendriers :

Intégration avec les calendriers électroniques des professionnels de la santé pour éviter les conflits d'horaires et améliorer la visibilité sur la charge de travail.

- Gestion des annulations et des reports :

Fournir des mécanismes simples pour annuler ou reporter un rendez-vous. Cela peut inclure des politiques de gestion des annulations et des notifications pour remplir les créneaux laissés vacants [7].

1.5. Gestion du personnel

La gestion du personnel désigne l'ensemble des tâches administratives qui sont nécessaires à la bonne gestion des ressources humaines dans l'entreprise. L'efficacité et la productivité sont particulièrement recherchées par l'employeur en matière de gestion du personnel, afin d'assurer la conformité de l'entreprise à toutes ses obligations législatives et réglementaires.

Rôle central dans le développement stratégique et la bonne croissance de l'entreprise, l'optimisation des tâches relatives à la gestion du personnel concerne l'intégration de tout nouveau salarié dans l'entreprise jusqu'à la rupture de son contrat [8].

1.6. Gestion des dossiers

Les dossiers patients, également appelés dossiers médicaux, jouent un rôle essentiel dans les métiers de la santé. Ils servent à rassembler, organiser et conserver toutes les informations relatives à un patient spécifique tout au long de son parcours médical. Ces dossiers permettent aux professionnels de la santé de fournir des soins de qualité, cohérents et personnalisés à chaque individu [9].

2. Revue de quelques applications de gestion de données médicales

Une plateforme de gestion des données médicales permet une meilleure conservation des dossiers médicaux, y compris d'automatiser le processus de facturation, de prise des rendez-vous, la planification, la conformité au sein des structures de santé.

A la suite de quelques investigations, nous avons trouvé des applications à peu près similaires à ce qu'on veut réaliser, nous pouvons citer : DHIS2, SIMENS

2.1. DHSI2

DHIS2 est un logiciel libre et open-source qui permet de collecter, analyser, visualiser et partager des données. Le modèle de données DHIS2 prend en charge à la fois les données agrégées et les données individuelles – y compris les fonctions de surveillance et de suivi auprès de personnes ou d'entités individuelles au fil du temps – et la saisie de données en ligne et hors ligne via le portail web DHIS2, l'application mobile Android, les SMS ou l'importation directe.

DHIS2 est le système de gestion d'information sanitaire (SGIS) le plus utilisé au monde, et il a été déployé dans divers autres contextes, notamment dans les domaines de l'éducation, de la logistique et de l'agriculture. Voir la page DHIS2 en action pour des exemples de l'utilisation actuelle de DHIS2 [10].

2.2. SIMENS

SIMENS (Système d'Informations MEdicales National du Sénégal) est un système d'informations pour la gestion des activités et des données médicales et administratives des établissements sanitaires.

SIMENS permet d'identifier de manière unique un patient sur le plan national à travers son dossier médical informatisé et de conserver de manière structurée sur ordinateur toutes sortes d'informations utiles recueillies durant les activités médicales. Ces informations médicales pourront être partagées confidentiellement entre acteurs de la santé, et être exploitées pour des besoins d'alerte, de prévention, de suivi et de contrôle de phénomènes épidémiologiques [11].

- Tableau 1. Tableau comparatif de deux solutions hospitalières

	DHSI2	SIMENS
Cibles	Les institutions de santé mondiales, Médecins sans Frontières, Population Services International (PSI) et les programmes de santé mondiale de l'Organisation Mondiale de la Santé (OMS)	Les établissements sanitaires publics sur le plan national
Fonctionnalités	• Analyse et gestion de données • Enregistrement de données individuelles • Saisie de données pour mobile • Hébergement local ou dans le cloud	• SIMENS-SIH • SIMENS-BLOCCOP • SIMENS-ORL • SIMENS-URGENCES • SIMENS-CARDIO • SIMENS-GASTRO • SIMENS-LIH • SIMENS-MATERNITÉ
Accès	OPEN SOURCE	PAYANT

Tableau 1:Comparatif de deux solutions hospitalières

3. Besoins technologiques :

3.1.Un langage de programmation pour le Back-end :

Pour faire simple, le langage de programmation est un ensemble de commandes et d'instructions numériques qui utilisent des syntaxes spécifiques pour créer des applications informatiques. Dans le monde de la programmation, le terme « backend » fait référence aux codes informatiques qui gèrent les opérations côté serveur telles que la logique du serveur, les fonctions de base de données, et bien d'autres encore. Lorsque vous utilisez une application, la grande majorité des données que vous envoyez et recevez sont gérées par le backend de l'application. Cependant, les fonctions du backend sont totalement invisibles pour l'utilisateur de l'application.

3.2.Pourquoi l'utilisation d'un Framework ?

Un framework est un espace de travail modulaire. C'est un ensemble de bibliothèques et de conventions permettant le développement rapide d'applications. Il fournit suffisamment de briques logicielles et impose suffisamment de rigueur pour pouvoir produire une application aboutie et facile à maintenir. Ces composants sont organisés pour être utilisés en interaction les uns avec les autres (voir urbanisation).

Un framework fournit un ensemble de fonctions facilitant la création de tout ou d'une partie d'un système logiciel, ainsi qu'un guide architectural en partitionnant le domaine visé en modules. Un framework est habituellement implémenté à l'aide d'un langage à objets, bien que cela ne soit pas strictement nécessaire : un framework objet fournit ainsi un guide architectural en partitionnant le domaine visé en classes et en définissant les responsabilités de chacune ainsi que les collaborations entre classes. Un sous-ensemble de ces classes peuvent être des classes abstraites.

Il existe plus d'une dizaine de Framework, chacun ayant ses particularités et selon le langage approprié.

3.3.PHP avec le framework Symfony :

Symfony est donc un framework PHP (langage de programmation web) utilisé afin de développer des applications ou sites web. Il est considéré comme l'un des meilleurs framework dans la création d'application web car il permet d'obtenir plus de flexibilité. Il est également assez facile à prendre en main, grâce aux nombreuses documentations disponibles en ligne. De plus, l'outil de débogage sur Symfony est puissant. Le débogage est utilisé afin d'identifier et corriger vos fautes dans le code. Cet outil vous fait donc gagner un temps considérable dans la phase de développement de votre application.

De plus, la technologie Symfony intègre aussi des mesures de sécurité innées afin de lutter contre des failles et attaques XSS, CSRF et SQL. Ce sont des failles permettant d'injecter du code HTML dans des variables mal protégées. C'est aussi un avantage, par rapport à l'utilisation du simple PHP. En effet, lorsque que vous développez un site internet en PHP, vous êtes obligé(e) de protéger chaque formulaire manuellement. Grâce à Symfony, vous n'avez plus à vous soucier de cela, il le fait systématiquement pour vous. De ce fait, votre application ou

site web est mieux protégé des pirates, par rapport aux CMS par exemple (WordPress, Wings ou Joomla) qui eux sont assez répandus et dont leur architecture est plus facilement attaquable.

3.4. Le Framework Django

Django est un framework web open source robuste et puissant qui simplifie le processus de développement d'applications web en utilisant le langage de programmation Python. L'architecture de Django suit le modèle architectural MVC (Modèle-Vue-Contrôleur), appelé MVT chez Django. Il divise l'application en trois composants principaux : le modèle, la vue et le template. Ce modèle architectural favorise la séparation des préoccupations, ce qui facilite la maintenance et l'extensibilité du code.

Voici quelques points clés sur Django :

Modèle-Vue-Template (MVT) :

- La couche Modèle de Django est responsable de la gestion des données. Les développeurs définissent les modèles, qui représentent la structure des données de l'application, et Django se charge de l'interaction avec la base de données grâce à son ORM. Cette approche facilite la manipulation des données et offre une abstraction efficace des détails spécifiques à la base de données.
- La couche Vue de Django gère la logique métier de l'application. Elle reçoit les requêtes HTTP, interagit avec le modèle pour récupérer ou manipuler des données, puis renvoie une réponse au client. Les vues peuvent être associées à des URL spécifiques, permettant ainsi une gestion claire des routes de l'application.
- La couche Template est responsable de la présentation des données et de la création des pages web. Les templates utilisent le langage de balisage Django, qui ressemble à HTML, pour générer dynamiquement le contenu affiché à l'utilisateur. Cette séparation claire entre le modèle, la vue et le template rend le code plus lisible et facilite la collaboration entre les développeurs front-end et back-end.
- ORM (Object-Relational Mapping) : Django intègre un ORM qui permet aux développeurs de travailler avec des bases de données relationnelles en utilisant des objets Python plutôt que des requêtes SQL directes. Cela simplifie le code et améliore la portabilité entre différents moteurs de base de données.

- Admin Interface : Django fournit une interface d'administration automatique générée à partir des modèles de données de l'application. Cela facilite la gestion et la manipulation des données par les administrateurs du site.
- Système de gestion des URL : Django utilise un système de routage pour gérer les URL des applications. Cela permet de définir facilement des vues et des modèles associés à des URL spécifiques.
- Sécurité intégrée : Django intègre des fonctionnalités de sécurité telles que la protection contre les attaques CSRF (Cross-Site Request Forgery), les injections SQL, et plus encore.
- Gestion des formulaires : Il fournit des outils pour simplifier la création, la validation et le traitement des formulaires HTML.
- Gestion des sessions et de l'authentification : Django prend en charge la gestion des sessions utilisateur, l'authentification et l'autorisation des utilisateurs.
- Extensibilité : Django est extensible à travers de nombreux packages et applications tierces qui peuvent être intégrés dans les projets Django.
- Déploiement facile : Le déploiement des applications Django peut être facilité grâce à des solutions telles que Gunicorn, uWSGI, et des serveurs web tels que Nginx ou Apache.
- Pour commencer avec Django, vous pouvez installer Django via pip et suivre le guide officiel Django pour créer votre première application.
- Django propose également une interface d'administration automatique, ce qui simplifie la gestion des données pour les administrateurs du site. Cette fonctionnalité génère une interface d'administration basée sur les modèles définis dans l'application, offrant ainsi un outil puissant et convivial pour gérer le contenu du site.

En résumé, Django offre une approche complète pour le développement web, en combinant la puissance de Python, la simplicité de l'ORM, la clarté de l'architecture MVT et des fonctionnalités intégrées pour rendre le processus de développement plus rapide, plus sûr et plus agréable [17].

- Tableau 2. Tableau comparatif PYTHON/PHP

PARAMETRES	PYTHON	PHP
Apprentissage	Python est plus facile à apprendre, même pour les débutants. Ce langage de script a été conçu pour être facilement lisible, avec un formatage épuré, une syntaxe plus simple et des mots en anglais simple au lieu de ponctuations.	Il faut beaucoup de temps pour apprendre à coder en PHP, surtout pour les débutants. Mais la communauté des développeurs s'efforce de rendre les choses plus faciles.
Communauté	PHP et Python ont été lancés sur le marché il y a plusieurs décennies et disposent d'une excellente communauté de développeurs pour soutenir les professionnels. Par conséquent, la compétition est au coude à coude entre PHP et Python en termes de communauté de développeurs et de support.	
Tarification	Python est disponible sur son site officiel pour être téléchargé et utilisé GRATUITEMENT. Vous pouvez également utiliser son code source disponible publiquement et le modifier en fonction de vos besoins.	Open source : comme PHP est un logiciel libre, tout le monde peut le télécharger et l'utiliser comme bon lui semble pour atteindre ses objectifs de développement web. Cela permet aux développeurs d'inspecter les codes, de signaler les erreurs et de contribuer au code et à la correction des bogues.
Vitesse	En comparaison, le processus de compilation de code de Python est conçu pour être plus rapide, même sans installer de système de cache. Lorsqu'un fichier est créé et / ou modifié, il convertit ce code en bytecode. Il était bien plus rapide que ce que PHP avait l'habitude d'être avant l'introduction de PHP 7.x. PHP l'emporte en termes de vitesse et de performances.	

Tableau 2;Comparatif framework

4. Généralités du projet global et cadrage de nos travaux de mémoire

4.1. Généralités du projet

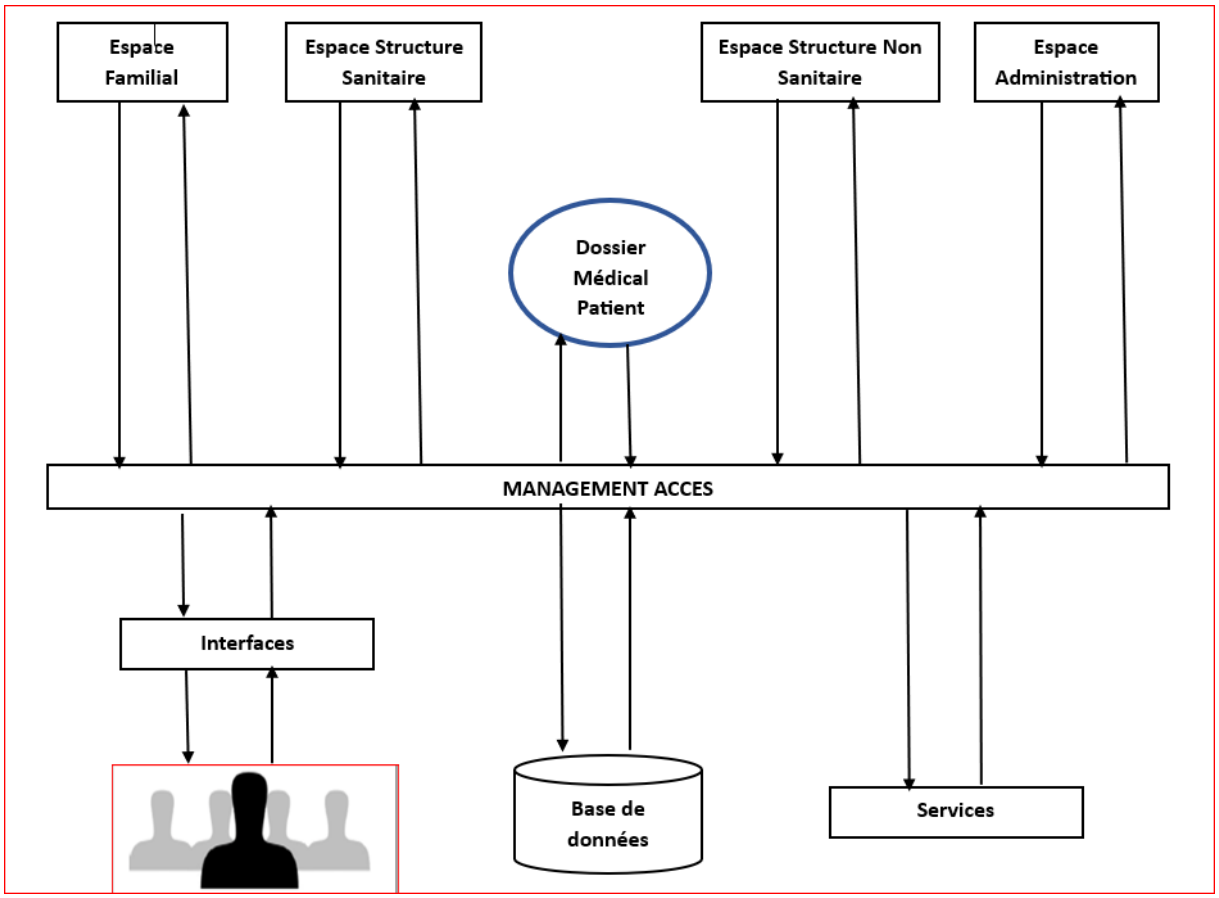

Figure 1. Architecture de l'application "GMU"

Le projet que nous présentons consiste à proposer une application de gestion de données médicales appelée « GMU (Guichet Médical Unique) » dont l'architecture est donnée dans la figure ci-dessus. Le choix de ce nom est motivé par l'orientation que nous donnons à notre plateforme. En effet, l'entité au cœur de cette application est le patient et son dossier médical. Il est l'*origine* et le *destinataire* de toutes les données du système. Chaque patient (et son dossier médical) existe de façon unique dans ce système malgré la multiplicité des autres entités pouvant y accéder.

Dans cette architecture, le module « Patient's Medical file » (ou dossier médical du patient) représente à la fois le dossier médical d'un patient et les dossiers médicaux de tous les patients. Il représente également toutes les données médicales, sanitaires et extra-sanitaires des patients, quelles que soient les sources, la nature et la complexité de ces données.

Toutes les entités du système interagissent avec le patient et son dossier médical par l'intermédiaire du module « Access manager » ou « gestionnaire des accès ». Ce module « Access manager » gère et contrôle les accès aux dossiers médicaux des patients. Il permet par exemple de définir et d'appliquer des règles restrictives telles que :

- Une donnée médicale ne peut être ajouté/modifié que par un médecin depuis l'espace d'une structure de santé.
- Un patient a accès total et en lecture seule à tout son dossier médical.
- Un patient peut accorder un accès en lecture/écriture à un médecin/structure de santé.
- Une donnée médicale cachée n'est visible que par le patient.
- Une structure de santé a accès en lecture/écriture à toutes les données médicales non cachées qu'elle a elle-même créées.

Dans cette architecture, nous avons un espace dédié pour chaque type d'entité impliquée dans la gestion, l'accès et/ou le traitement des dossiers médicaux :

- « Family area » ou « espace familiale ». C'est l'espace à partir duquel les patients accèdent à leurs dossiers médicaux. Un « Family area » est créée pour chaque famille souhaitant détenir ses propres données médicales. Il peut être administré par un gestionnaire qui gère (ajouter, modifier, supprimer) les membres. Chaque membre ajouté dans un espace familiale est un patient dans notre système. Le responsable d'un espace familial a accès à tous les dossiers médicaux des membres de la famille. De la même manière qu'il gère les membres de cet espace, il gère (ajout, modification, suppression) aussi les données ménagères et familiales (liens de parenté, mariages, fratrie, …) qui sont nécessaires pour déterminer un arbre généalogique médical et l'historique d'une maladie familiale, prédire l'apparition de certaines maladies "héréditaires". C'est aussi à partir de cet espace qu'un patient accorde un accès à son dossier médical à un médecin.
- « Health structure area » ou « espace des structures de santé ». Cet espace est réservé aux structures de santé et à leurs personnels (utilisateurs) pour leurs permettre la gestion des dossiers médicaux de leurs patients. Un espace est créé pour chaque structure de santé souhaitant utiliser notre application. Il est administré par un responsable qui gère (ajout, modification, suppression, suspension de comptes utilisateurs, …) les membres (médecins, aides soignant, secrétaires, etc.) et le paramétrage (définition et gestion des profils, accès et restrictions, services médicaux proposés, grille tarifaires des services médicaux, …). Une fois

qu'un espace est créé pour une structure, sa gestion est totalement assurée par le(s) responsable(s) désigné(s).

- « Non health structure area ». C'est l'espace à partir duquel les autres structures (assurances, banques, mutuelles de santé, …) peuvent accéder à certaines informations les concernant (dont les accès et les restrictions sont définis dans le module « access manager ») sur les dossiers médicaux des patients. C'est aussi à partir de leurs espaces respectives que ces structures fournissent à notre système des informations/données non médicales sur les patients, mais qui peuvent être particulièrement utiles dans certaines études et pratiques sanitaires.
- « Admin area » ou « espace d'administration » de l'application « GMU ». C'est dans cet espace que le système est globalement administré : créer un espace pour une famille et désigner le responsable, créer un espace pour une structure et désigner le responsable, définir et gérer les services que nous proposons.

L'application « GMU » propose également un ensemble de « Services » aux utilisateurs (patients, structures, médecins, directeurs des structures de santé, …) de la plateforme. Parmi ces services, il y'a entre autres :

- Les prises de rendez-vous médicaux à distance. Ce module, accessible depuis les espaces familiales, permet à un utilisateur de prendre rendez-vous avec un médecin dans une structure de santé. Il est aussi accessible aux médecins depuis les espaces de leurs structures pour leur permettre de prédéfinir des jours et des horaires de disponibilités qui vont constituer des rendez-vous pour les patients.
- Les paiements en ligne des frais médicaux. Ce module permet aux patients et à certaines structures telles que les mutuelles de santé de payer des frais médicaux avec les moyens de paiement classiques tels que la carte bancaire et le mobile money.
- La gestion des files d'attente. Ce module permet aux structures de santé de bien gérer les files d'attente des patients qui arrivent dans une structure sans avoir pris de rendez-vous.
- Arbre généalogique médical. Ce service permet aux familles de pouvoir disposer de leurs arbres généalogiques médical et de pouvoir, par conséquent, bénéficier des services y afférents : historique ou trajectoire d'une maladie, maladies héréditaires, incompatibilité sanguine dans les couples, etc.
- L'aide à la décision. Ce service permet entre autres :

- Aux directeurs des structures de santé de disposer de tableaux de bord sur les activités et pratiques médicales et administratives de leurs structures.
- Aux médecins de disposer de tableaux de bord sur leurs pratiques médicales.
- Aux patients de disposer de tableaux de bord sur leurs données médicales.
- D'utiliser des outils d'apprentissage automatique pour permettre d'améliorer et personnaliser les traitements proposés aux patients, améliorer le diagnostic des pathologies, etc.

Tous ces espaces et services sont accessibles aux utilisateurs via des « interfaces utilisateurs » spécifiques et toutes les données de l'application sont hébergées dans la base de données (« Database »).

4.2. Cadrage de nos travaux de mémoire

L'architecture présentée ci-dessus permet de voir que le projet de « Guichet Médical Unique » que nous présentons est composé de plusieurs modules indépendants. En effet, chaque espace et service présenté fera l'objet d'un module qui viendra se greffer sur ce que nous appelons le « noyau » du système. Le « noyau » est le dossier médical (« Patient's Medical File ») accompagné du module « Access Manager » qui gère son accès.

Nos travaux de mémoire consistent à mettre en place ce « noyau », le module « Access Manager » ainsi que la base de données « Databases ». Il s'agit de la première étape qui permettra de poser les fondements dans la construction du « Guichet Médical Unique ».

Nous allons donc proposer un *modèle* du dossier médical du patient et de toutes les entités, présentées plus haut, qui y accèdent. Ce *modèle* est une représentation des données du système, de leurs structures et leurs natures.

Dans la suite du document, nous employons le mot « système » pour désigner cette partie du projet que nous traitons dans ce mémoire.

5. Outils et techniques de développement

5.1.Le design pattern MVC

Le design pattern MVC qui définit un ensemble de règles architecturales et une nomenclature (conventions de nommage, casse, etc.) pour le développement d'un site web dynamique. MVC

signifie « Modèle - Vue - Contrôleur ». Il s'agit du design pattern le plus largement répandu sur le web, dont le rôle est de séparer la logique du code en trois parties que l'on retrouve dans des dossiers et fichiers spécifiques.

- Modèles (M)

Les modèles sont responsables de la structure et des traitements effectués sur les données d'une application. Ils regroupent à la fois :

Les entités, utilisées uniquement en POO (Programmation Orientée Objet) : les entités désignent la structure des données à manipuler (classes/objets mappées avec les tables de la base de données).

Les managers : leur rôle est d'interagir avec la base de données pour en écrire et lire des données (requêtes SQL d'insertion, modification, suppression, récupération). Ils agissent comme une couche d'abstraction qui va permettre au contrôleur d'interagir avec la base de données, sans se soucier du comment cela va être fait.

- Vues (V)

Le rôle des vues est de rassembler au sein de templates les éléments structurels des pages web. Autrement dit, l'ensemble du code HTML du site web.

Très souvent les vues intègrent des petites portions de code du langage back-end utilisé tels que l'affichage de variables, des boucles, des conditions, etc., venant dynamiser les templates.

Ces templates se divisent en 3 familles principales :

Le layout : il va contenir le squelette commun à l'ensemble des pages web. Il contient les balises de premier niveau (<html>, <head> et <body>).

Les pages : elles vont regrouper le contenu propre à chaque page web.

Les partials : ils vont contenir les éléments d'interface communs à plusieurs pages web (navbar, footer…).

- Contrôleurs (C)

Le rôle d'un contrôleur est de coordonner les actions à mettre en place pour retourner au client une réponse HTTP adaptée.

Pour cela, il travaille en étroite collaboration avec les modèles (données) et les vues (templates). Il joue en quelque sorte le rôle de chef d'orchestre du site [16].

5.2. Le scaffolding

Le terme "scaffolding" (échafaudage) est souvent utilisé dans le domaine de l'apprentissage et de la pédagogie, en particulier dans le contexte de la théorie de la ZPD (zone proximale de développement) développée par le psychologue russe Lev Vygotsky. Le scaffolding se réfère à un processus dans lequel un enseignant ou un pair fournit un soutien temporaire à un apprenant afin de l'aider à accomplir une tâche ou à acquérir une compétence qui serait difficile à réaliser seul. L'idée est de créer une structure de support qui peut être progressivement retirée à mesure que l'apprenant gagne en compétence et en confiance.

En informatique, le concept de scaffolding est également utilisé, mais il est souvent associé à des frameworks, des bibliothèques ou des outils qui fournissent une structure de base pour le développement logiciel. Dans ce contexte, le scaffolding agit comme un support initial qui permet aux développeurs de démarrer plus rapidement et de suivre les meilleures pratiques.

Voici quelques façons dont le scaffolding est utilisé en informatique :

- Génération de code initial : Certains outils de scaffolding peuvent générer automatiquement une structure de projet de base avec des fichiers et des dossiers préconfigurés, ce qui permet aux développeurs de se concentrer directement sur le code métier.
- Modèles de projet : Les frameworks et les outils de développement web, par exemple, peuvent fournir des modèles de projet qui incluent des configurations de base, des fichiers de configuration, et parfois même des morceaux de code pré-écrits. Cela aide les développeurs à éviter de réinventer la roue pour chaque nouveau projet.
- Génération de code CRUD : Certains frameworks proposent des générateurs de code pour les opérations CRUD (Create, Read, Update, Delete), permettant aux développeurs de créer rapidement des fonctionnalités de base de gestion des données sans avoir à écrire tout le code manuellement.
- Scaffolding de l'interface utilisateur (UI) : Certains outils fournissent des modèles et des composants d'interface utilisateur de base, permettant aux développeurs de commencer

rapidement le développement d'une interface utilisateur sans avoir à concevoir chaque élément à partir de zéro.

- Configuration automatique : Certains frameworks de configuration automatique simplifient le processus de configuration en ajustant automatiquement les paramètres en fonction des besoins du projet, ce qui réduit la charge de travail initiale des développeurs.

En résumé, le scaffolding en informatique vise à fournir une base solide et structurée pour le développement logiciel, accélérant ainsi le processus de démarrage et permettant aux développeurs de se concentrer sur des aspects plus spécifiques et complexes de leur application [18].

5.3. La base de données PostgreSQL

PostgreSQL est un système de gestion de base de données relationnelle orienté objet puissant et open source qui est capable de prendre en charge en toute sécurité les charges de travail de données les plus complexes. Alors que MySQL donne la priorité à l'évolutivité et aux performances, PostgreSQL donne la priorité à la conformité et à l'extensibilité SQL.

Les entreprises qui souhaitent maintenir un haut niveau d'intégrité et de personnalisation de leurs données choisissent généralement PostgreSQL en raison de sa fiabilité, l'intégrité de ses données, la robustesse de ses fonctionnalités, et parce qu'il fournit des solutions toujours performantes et innovantes. PostgreSQL fonctionne sur tous les principaux systèmes d'exploitation et est conforme à ACID depuis 2001.

PostgreSQL peut être téléchargé gratuitement et déployé sur du matériel standard, ou peut être exécuté dans le Cloud par le biais d'une variété de fournisseurs. Bien que PostgreSQL soit riche en fonctionnalités et adapté aux charges de travail OLAP, les performances de PostgreSQL ont tendance à atteindre une limite lorsque les volumes de données dépassent plusieurs téraoctets [19].

Chapitre 2. Analyse et conception de notre système

1. Aperçu sur UML

1.1. Définitions et historique

UML, ou Unified Modeling Language, est un langage de modélisation graphique standardisé utilisé principalement dans le domaine du génie logiciel pour visualiser, spécifier, construire et documenter les systèmes logiciels. Il a été créé pour unifier les approches de modélisation, en intégrant les meilleures pratiques et les concepts issus de différentes méthodologies de développement logiciel.

Voici quelques concepts clés de l'UML :

- Diagrammes UML : UML propose plusieurs types de diagrammes, chacun servant à représenter différents aspects d'un système logiciel. Les principaux diagrammes incluent les diagrammes de cas d'utilisation, les diagrammes de classes, les diagrammes d'objets, les diagrammes de séquence, les diagrammes d'activités, les diagrammes d'états, et les diagrammes de déploiement.
- Classes et objets : Les diagrammes de classes sont centraux à UML et sont utilisés pour représenter la structure statique d'un système. Ils comprennent des classes, des interfaces, des attributs, des opérations et les relations entre ces éléments.
- Cas d'utilisation : Les diagrammes de cas d'utilisation décrivent les fonctionnalités d'un système du point de vue de ses utilisateurs. Ils mettent en évidence les interactions entre les acteurs (utilisateurs) et le système.
- Séquence et collaboration : Les diagrammes de séquence et de collaboration décrivent le comportement dynamique d'un système en modélisant les interactions entre les objets au fil du temps. Ils sont particulièrement utiles pour représenter des scénarios d'utilisation et des flux d'exécution.
- Diagrammes d'activités : Ces diagrammes décrivent le flux de travail d'une opération ou d'un processus, mettant en évidence les étapes, les décisions et les activités parallèles.
- États : Les diagrammes d'états modélisent le comportement d'un objet en fonction de son état interne et des événements qui influent sur cet état.

- Déploiement : Les diagrammes de déploiement représentent la disposition physique des composants matériels et logiciels dans un environnement.

L'UML facilite la communication entre les membres de l'équipe de développement, les parties prenantes et d'autres intervenants en fournissant une notation graphique standardisée et compréhensible. Il favorise également la conception et la documentation claires des systèmes logiciels, contribuant ainsi à la qualité et à la maintenance du code. L'UML est largement utilisé dans l'industrie du logiciel et a été intégré dans de nombreuses méthodologies de développement, telles que le processus unifié (UP) et les méthodes agiles.

L'histoire de l'UML (Unified Modeling Language) remonte au début des années 1990. Voici une chronologie des événements clés dans le développement de l'UML :

1994 : Genèse de l'UML

L'UML trouve ses racines dans le travail de trois développeurs de méthodes de modélisation, Grady Booch, Ivar Jacobson et James Rumbaugh, qui ont développé séparément des méthodes de modélisation (Booch, OMT, et OOSE respectivement).

1995 : Rational Software Corporation

En 1995, Rational Software Corporation acquiert les droits sur les méthodologies de Booch, Rumbaugh et Jacobson. Ces trois approches distinctes sont alors fusionnées pour créer une méthodologie de modélisation unifiée.

1996 : Première version de l'UML

En novembre 1996, la première version de l'UML, appelée UML 1.0, est publiée. Cette version initiale combine les concepts les plus forts des trois méthodologies précédentes.

1997-1999 : Normalisation de l'UML

En 1997, l'Object Management Group (OMG), un consortium industriel, prend en charge la normalisation de l'UML. Une série de révisions de l'UML a lieu, aboutissant à la publication de l'UML 1.1 en 1997, puis de l'UML 1.2 en 1998, et enfin de l'UML 1.3 en 1999.

2001 : UML 2.0

L'UML 2.0, une version majeure, est publiée en 2001. Elle introduit des améliorations significatives, notamment une syntaxe plus précise, de nouveaux concepts et une meilleure prise en charge de la modélisation de systèmes complexes.

2005-2010 : Versions successives

L'UML subit plusieurs mises à jour mineures entre 2005 et 2010, avec la publication de l'UML 2.1, 2.2, et 2.3. Ces versions apportent des ajustements, des clarifications et des extensions de fonctionnalités.

2017 : UML 2.5

En juin 2017, l'UML 2.5 est publiée. Cette version intègre des améliorations notables, met l'accent sur la clarté et l'expressivité, et offre une meilleure prise en charge des modèles pour l'ingénierie des systèmes et l'ingénierie dirigée par les modèles.

L'UML est devenue une norme de facto dans l'industrie du génie logiciel, largement utilisée pour modéliser, concevoir et documenter des systèmes logiciels complexes. Elle a influencé de nombreuses méthodologies de développement logiciel et continue d'être une ressource précieuse pour les professionnels du domaine [12].

1.2. Le diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation (DCU) sont des diagrammes UML utilisés pour une représentation du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés. En effet, un cas d'utilisation (*use cases*) représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Ainsi, dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), et ils apparaissent dans les cas d'utilisation [13].

1.3. Le diagramme de classes

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que leurs relations. Ce diagramme fait partie de la partie statique d'UML, ne s'intéressant pas aux aspects temporels et dynamiques.

Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe.

Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique. Les classes sont utilisées dans la programmation orientée objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

Les classes peuvent être reliées grâce au mécanisme d'héritage qui permet de mettre en évidence des relations de parenté. D'autres relations sont possibles entre des classes, représentées par un arc spécifique dans le diagramme de classes.

Elles sont finalement instanciées pour créer des objets (une classe est un *moule à objet* : elle décrit les caractéristiques des objets, les objets contiennent leurs valeurs propres pour chacune de ces caractéristiques lorsqu'ils sont instanciés) [14].

1.4. Le diagramme d'activité

Un diagramme d'activité permet de modéliser un processus interactif, global ou partiel pour un système donné (logiciel, système d'information). Il est recommandable pour exprimer une dimension temporelle sur une partie du modèle, à partir de diagrammes de classes ou de cas d'utilisation, par exemple.

Le diagramme d'activité est une représentation proche de l'organigramme ; la description d'un cas d'utilisation par un diagramme d'activité correspond à sa traduction algorithmique. Une activité est l'exécution d'une partie du cas d'utilisation, elle est représentée par un rectangle aux bords arrondis.

Le diagramme d'activité est sémantiquement proche des diagrammes de communication (appelés *diagramme de collaboration* en UML 1), ou d'état-transitions, ces derniers offrant une vision microscopique des objets du système [15].

2. Analyse de notre système

2.1. Étude des besoins fonctionnels de notre système

Dans la section 3.2 du chapitre 1, nous avons présenté brièvement les contours de nos travaux de mémoire dans le projet de mise en place du guichet médical unique. Dans cette partie, nous présentons de façon détaillée ces travaux que nous appelons « notre système ». Il s'agit de proposer un modèle de dossier médical, de développer le module de gestion des accès aux

dossiers médicaux des patients et de mettre en place l'espace d'administration globale du guichet.

2.1.1. L'espace d'administration

L'espace d'administration est le « back-office » du guichet médical unique. C'est la partie « cachée aux utilisateurs » qui permet de gérer et d'administrer l'application. Il n'est donc accessible et visible que par le ou les « administrateurs » de l'application. L'administration du système consiste à :

- Gérer les clients. Ce que nous appelons « clients » ici, sont les utilisateurs responsables des espaces dédiés aux structures de santé, aux familles et aux autres structures non sanitaires (banques, assurances). En effet, chaque espace dédié est géré et administré par un client.
- Gérer les espaces dédiés. Dans l'architecture du guichet médical unique (Figure 1), nous avons un espace pour les structures de santé, un espace pour les familles et un espace pour les autres structures (banques, assurances, etc.). La gestion consiste à pouvoir ajouter, modifier, supprimer, bloquer un espace dédié. Un espace dédié à une structure de santé est donc un espace de type « structure de santé » propre à une structure. Par exemple, pour ajouter un espace dédié au « Cabinet médical du Docteur Simon Tendeng », un administrateur renseigne, via un formulaire, les informations sur le cabinet (Nom du cabinet, code, responsable, …) et lui associe un espace de type « Heath structure area ». Une fois cet espace dédié créé, il sera accessible via une adresse du type www.gmu-sn/simontendeng. Pour ajouter un espace dédié à la famille de Monsieur Papa Alioune CISSE par exemple, un administrateur renseigne, via un formulaire, les informations sur le responsable et lui associe un espace de type « Family area ». Dès lors, monsieur CISSE pourra se connecter sur son propre « Family area » en tant qu'administrateur et gérer les membres.
- Gérer les « données de paramétrage ». Les « données de paramétrage » sont les données administrées depuis l'espace d'administration pour le bon fonctionnement de l'application et qui sont accessibles aux utilisateurs finaux du guichet. Il s'agit, entre autres, des données relatives aux profils utilisateurs, aux groupes d'utilisateurs, aux accès et autorisations par profils et par groupes d'utilisateurs, aux types de services médicaux, aux pathologies, aux types de consultation médicale, aux types d'actes médicaux, etc.

2.1.2. La gestion des autorisations d'accès aux dossiers médicaux

Le module de gestion des accès aux dossiers médicaux permet de contrôler les accès des différents utilisateurs aux dossiers médicaux des patients. Comme nous l'avons vu dans la section 3.1 du chapitre 1, le dossier médical d'un patient est accessible depuis tous les espaces (Health structure, family, non health structure). Cependant, des restrictions sont appliquées sur les dossiers médicaux afin de garantir l'intégrité, la fiabilité et la confidentialité des données médicales. Ces restrictions permettent d'assurer, entre autres, que :

- Un acte médical ne peut être établi que par un médecin depuis un espace dédié à une structure de santé. De ce fait, cette donnée médicale est accessible en lecture/écriture au médecin. Ce qui veut dire que celui-ci a la possibilité de la modifier et la supprimer.
- Un patient a accès en lecture seule à tout son dossier médical.
- Etc.

2.1.3. Le modèle de dossier médical

Le modèle de dossier médical est une représentation du dossier médical qui garantit d'une part les restrictions d'accès établis ci-dessus et qui comportent d'autre part toutes les données médicales, sanitaires et non sanitaires nécessaires pour prendre en compte les préoccupations établies dans l'introduction.

2.2. Les fonctionnalités et les acteurs

Le système que nous proposons de développer est constitué de 4 grandes fonctionnalités :

- Gérer les clients :
  - Acteur : Administrateur
  - Description : La gestion des clients est une grande fonctionnalité qui regroupe des sous fonctionnalités plus élémentaires : Ajouter un client et lui associer au moins un espace dédié ; modifier les informations sur un client ; supprimer un client ; visualiser sous forme de table la liste des clients ; visualiser les détails des informations sur un client ; rechercher un client.
- Gérer les espaces dédiés :
  - Acteur : Administrateur
  - Description : La gestion des espaces dédiés est une grande fonctionnalité qui regroupe des sous fonctionnalités plus élémentaires : Ajouter un espace dédié et lui associer au

moins un client ; modifier les informations sur un espace dédié ; supprimer un espace dédié ; bloquer l'accès à un espace dédié ; visualiser sous forme de table la liste des espaces dédiés ; visualiser les détails des informations sur un espace dédié ; rechercher un espace dédié.

- Gérer les données de paramétrage :
  - Acteur : Administrateur
  - Description : La gestion des données de paramétrage est une grande fonctionnalité qui regroupe des sous fonctionnalités plus élémentaires s'appliquant à chacune des données de paramétrage. Les données de paramétrage sont regroupées en deux catégories : les données d'accessibilité (Types de clients, Types d'utilisateurs, groupes d'utilisateurs, profils utilisateurs, permissions des groupes d'utilisateurs) et les données médicales (Types de services médicaux, Types de consultations médicales, Pathologies, etc.).
- Accéder au dossier médical d'un patient :
  - Acteur : utilisateur. L'acteur utilisateur représente tous les utilisateurs du système qui accèdent au guichet à travers les espaces dédiés.
  - Description : L'accès au dossier médical d'un patient est une grande fonctionnalité qui regroupe des sous fonctionnalités plus élémentaires : Ouvrir un dossier médical si on a les droits d'accès ; visualiser les données d'un dossier médical si on a les droits d'accès et si les données ne sont pas scellées ; modifier les données d'un dossier si on a les droits ; supprimer les données d'un dossier si on a les droits ; rechercher des données d'un dossier médical.

2.3. Les diagrammes de cas d'utilisation

2.3.1. Cas d'utilisation gestion des clients

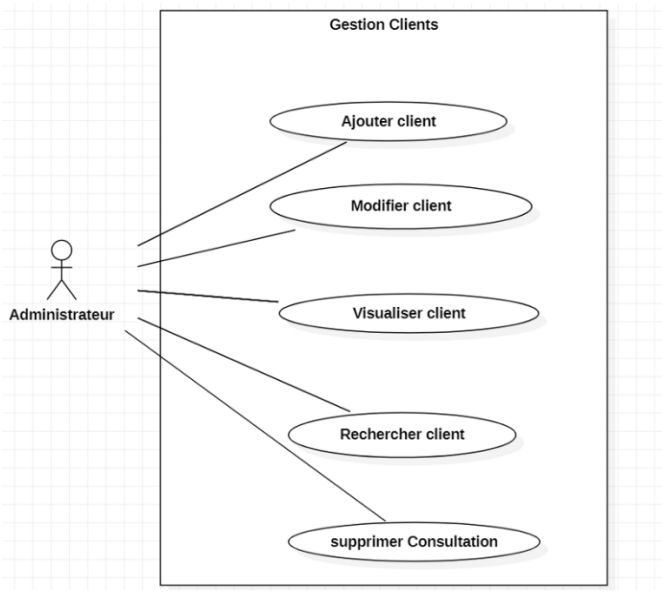

Figure 2:Cas d'utilisation gestion clients 'GMU'

L'administrateur ici est le responsable d'une structure de santé, ou la personne désignée pour gérer, qui à partir de son espace fera les opérations suivantes en fonction des droits d'accès qui lui ont été dédiés :

- Ajouter un nouvel patient
- Modifier client
- Visualiser un client
- Rechercher un client
- Supprimer un client

2.3.2. Cas d'utilisation gestion des espaces

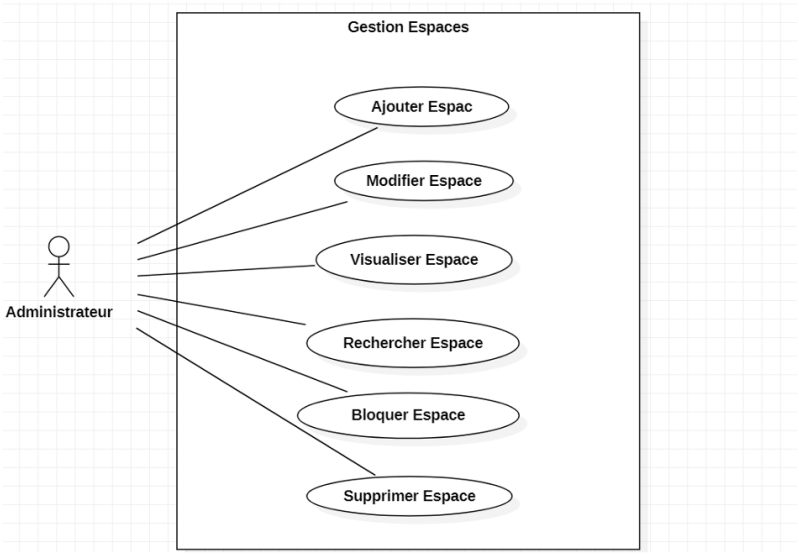

Figure 3:Cas d'utilisation gestion espaces 'GMU'

L'administrateur ici est celui qui gère la plateforme GMU. Il effectue les opérations suivantes :

- Ajouter un nouvel espace
- Modifier un espace
- Visualiser un espace
- Rechercher un espace
- Bloquer un espace
- Supprimer un espace

2.3.3. Cas d'utilisation gestion données de paramétrage

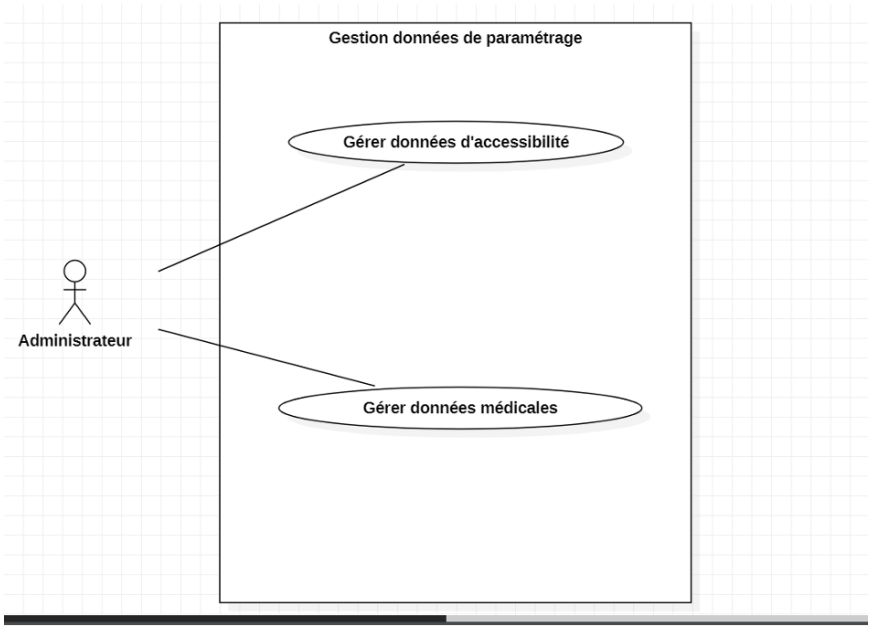

Figure 4:Cas d'utilisation gestion paramétrages 'GMU'

L'administrateur gère les données de paramétrage de la plateforme de façon générale. L'administrateur sera en mesure de faire les opérations suivantes :

- Gérer les données d'accessibilité
- Gérer les données médicales

2.4. Les diagrammes de séquences

2.4.1. Diagramme de séquence ajout client

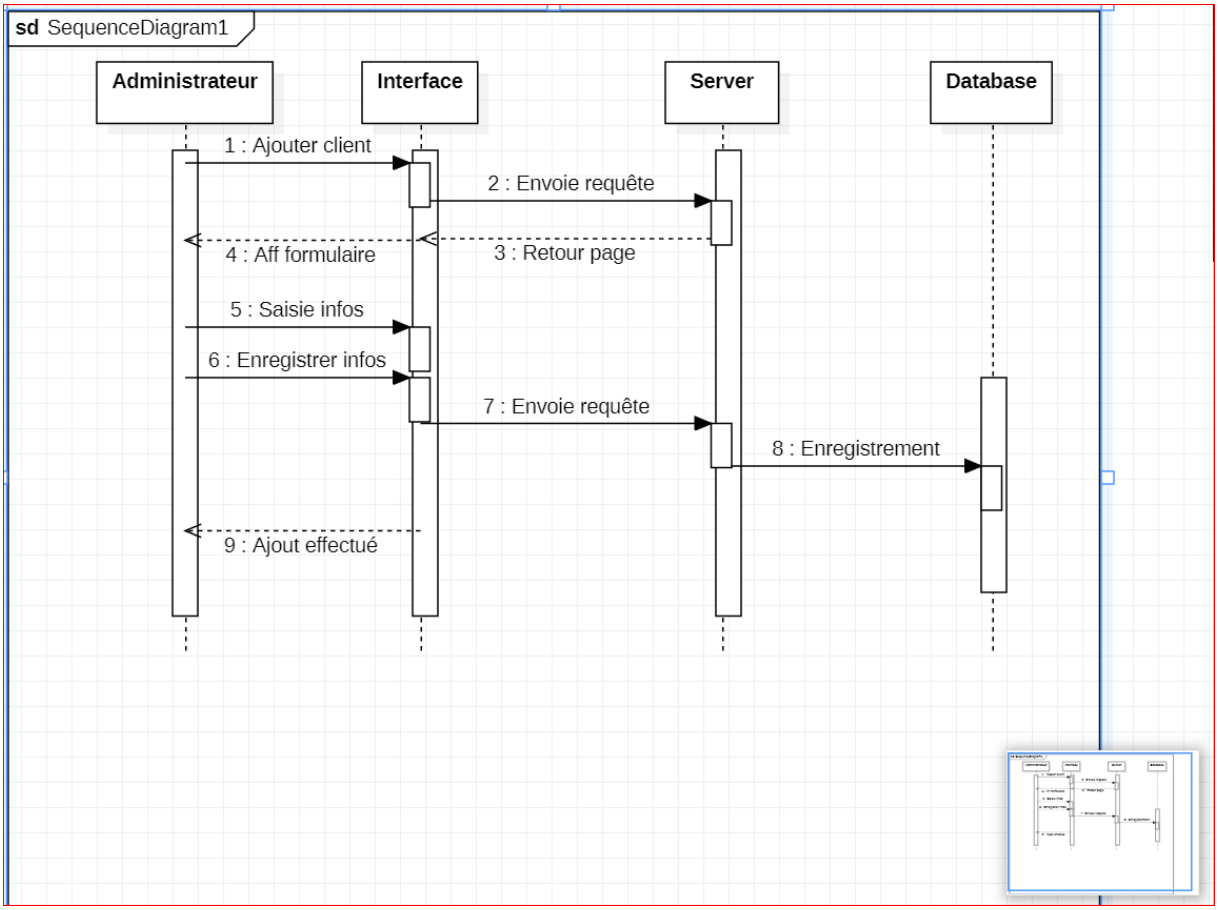

Figure 5:Diagramme de séquence ajout client

Dans cette figure nous expliquons la procédure d'ajout d'un client qui se fait comme suite :

D'abord l'administrateur clique sur le bouton ajouter un client à travers l'interface et par la suite l'interface transmet la requête au server qui envoie un retour à l'interface avec un formulaire, après remplissage du formulaire l'administrateur clique sur enregistrer et l'interface renvoie une requête au server et celui renseigne la base de données pour donner la réponse « Ajout effectué ».

2.4.2. Diagramme de séquence modification client

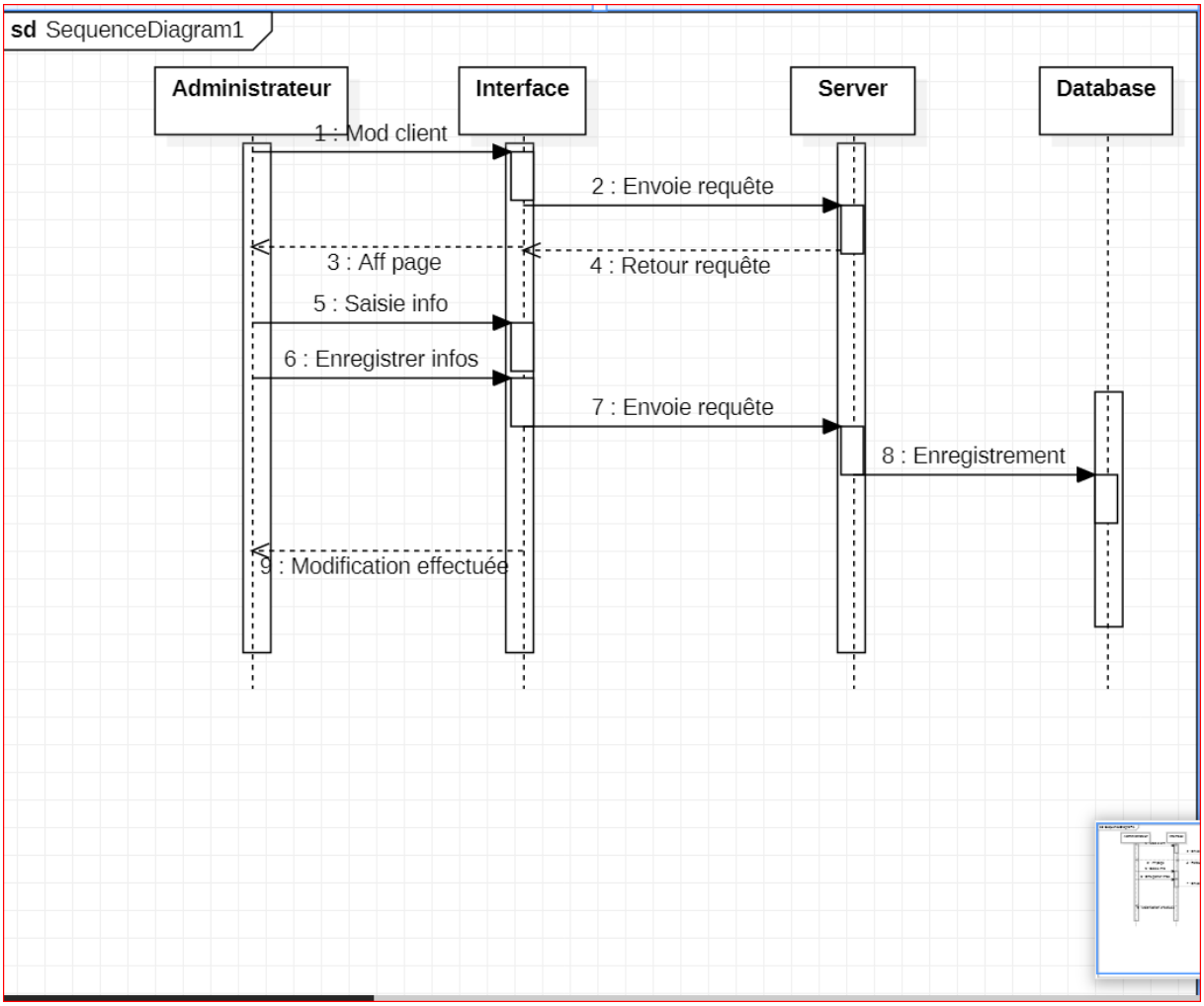

Figure 6:Diagramme de séquence modification client

Dans cette figure nous expliquons la procédure de modification d'un client qui se fait comme suite :

D'abord l'administrateur clique sur le bouton modifier un client à travers l'interface et par la suite l'interface transmet la requête au server qui envoie un retour à l'interface avec un formulaire préremplie, après modification du formulaire l'administrateur clique sur enregistrer et l'interface renvoie une requête au server et celui renseigne la base de données pour donner la réponse « Modification effectuée ».

2.4.3. Diagramme de séquence suppression client

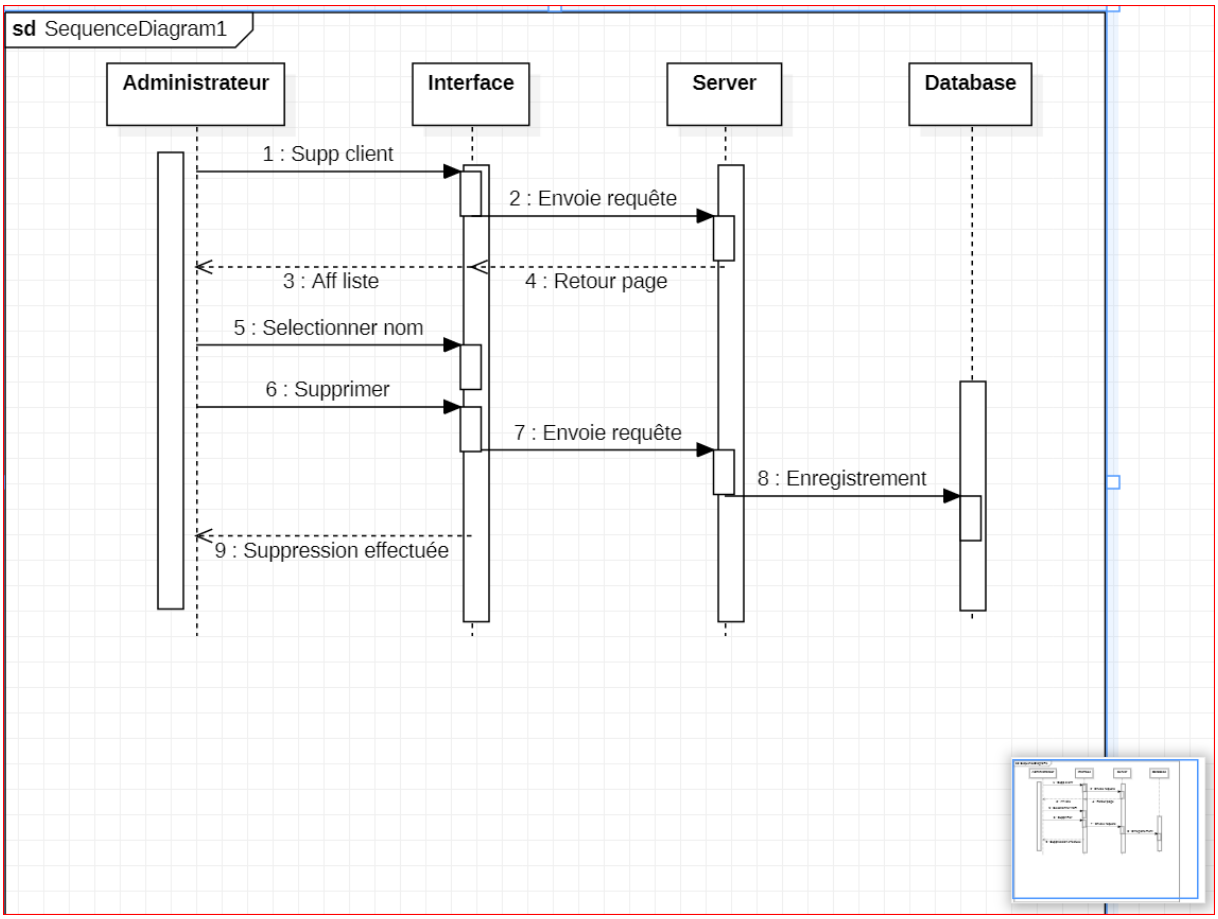

Figure 7:Diagramme de séquence supprimer client

Dans cette figure nous expliquons la procédure de suppression d'un client qui se fait comme suite :

D'abord l'administrateur clique sur le bouton supprimer un client à travers l'interface et par la suite l'interface transmet la requête au server qui envoie un retour à l'interface avec une liste, après affichage de la liste l'administrateur clique sur le client à supprimer et l'interface renvoie une requête au server et celui renseigne la base de données pour donner la réponse « Suppression effectuée ».

2.4.4. Diagramme de séquence ajout espace

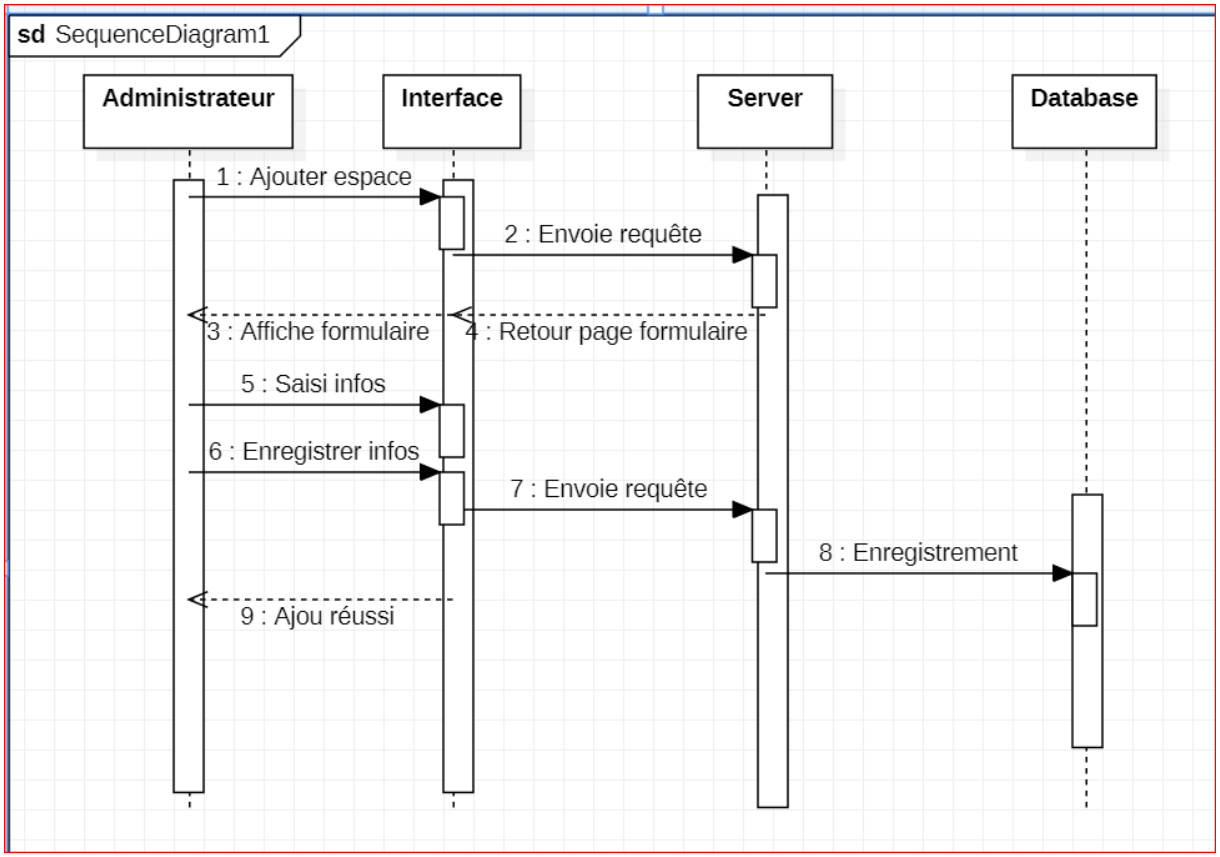

Figure 8:Diagramme de séquence d'ajout espace

Dans cette figure nous expliquons la procédure d'ajout d'un espace qui se fait comme suite :

D'abord l'administrateur clique sur le bouton ajouter un espace à travers l'interface et par la suite l'interface transmet la requête au server qui envoie un retour à l'interface avec un formulaire, après remplissage du formulaire l'administrateur clique sur enregistrer et l'interface renvoie une requête au server et celui renseigne la base de données pour donner la réponse « Ajout effectué ».

2.4.5. Diagramme de séquence modification espace

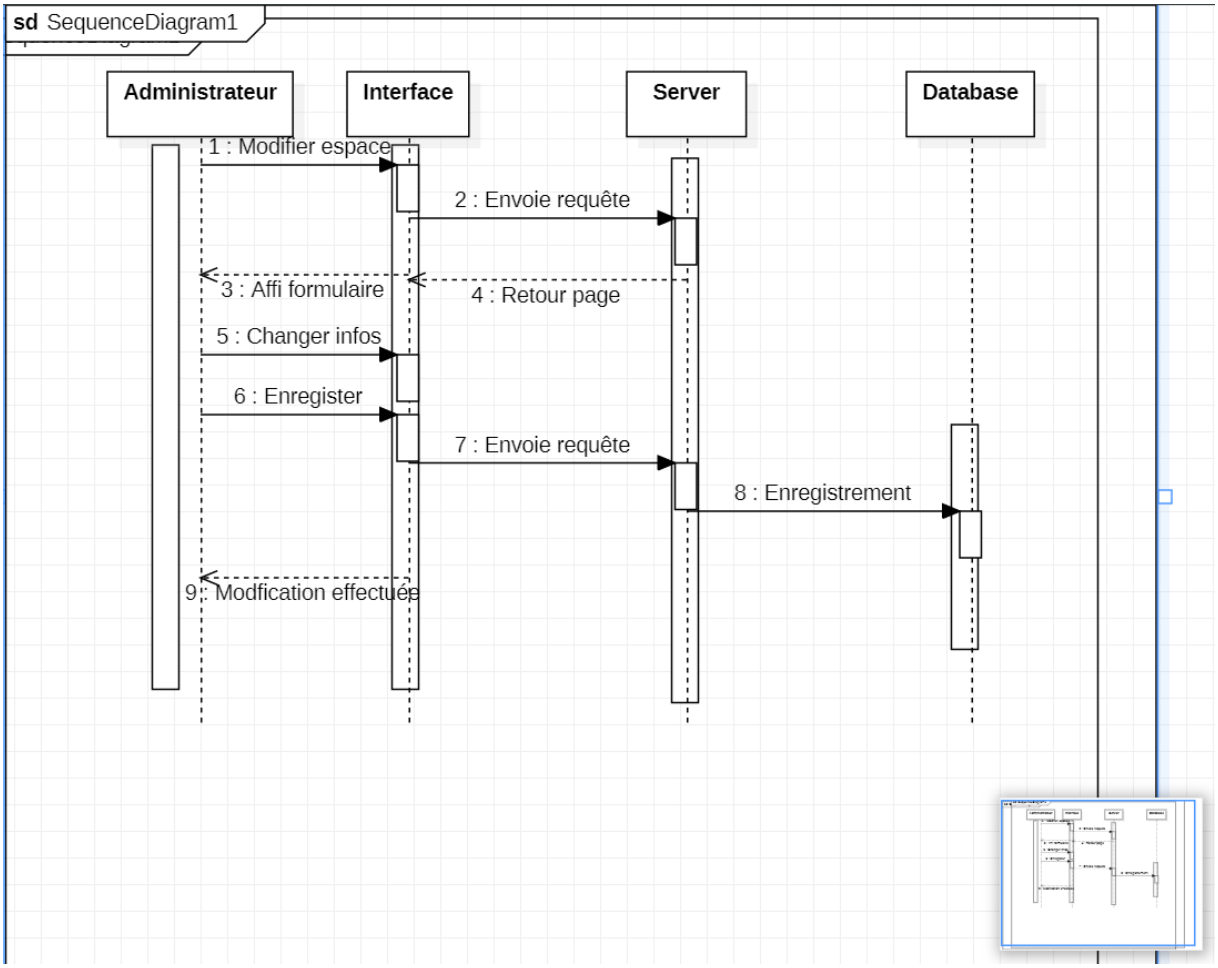

Figure 9:Diagramme de séquence modification espace

Dans cette figure nous expliquons la procédure de modification d'un espace qui se fait comme suite :

D'abord l'administrateur clique sur le bouton modifier un espace à travers l'interface et par la suite l'interface transmet la requête au server qui envoie un retour à l'interface avec un formulaire préremplie, après modification du formulaire l'administrateur clique sur enregistrer et l'interface renvoie une requête au server et celui renseigne la base de données pour donner la réponse « Modification effectuée ».

2.4.6. Diagramme de séquence blocage espace

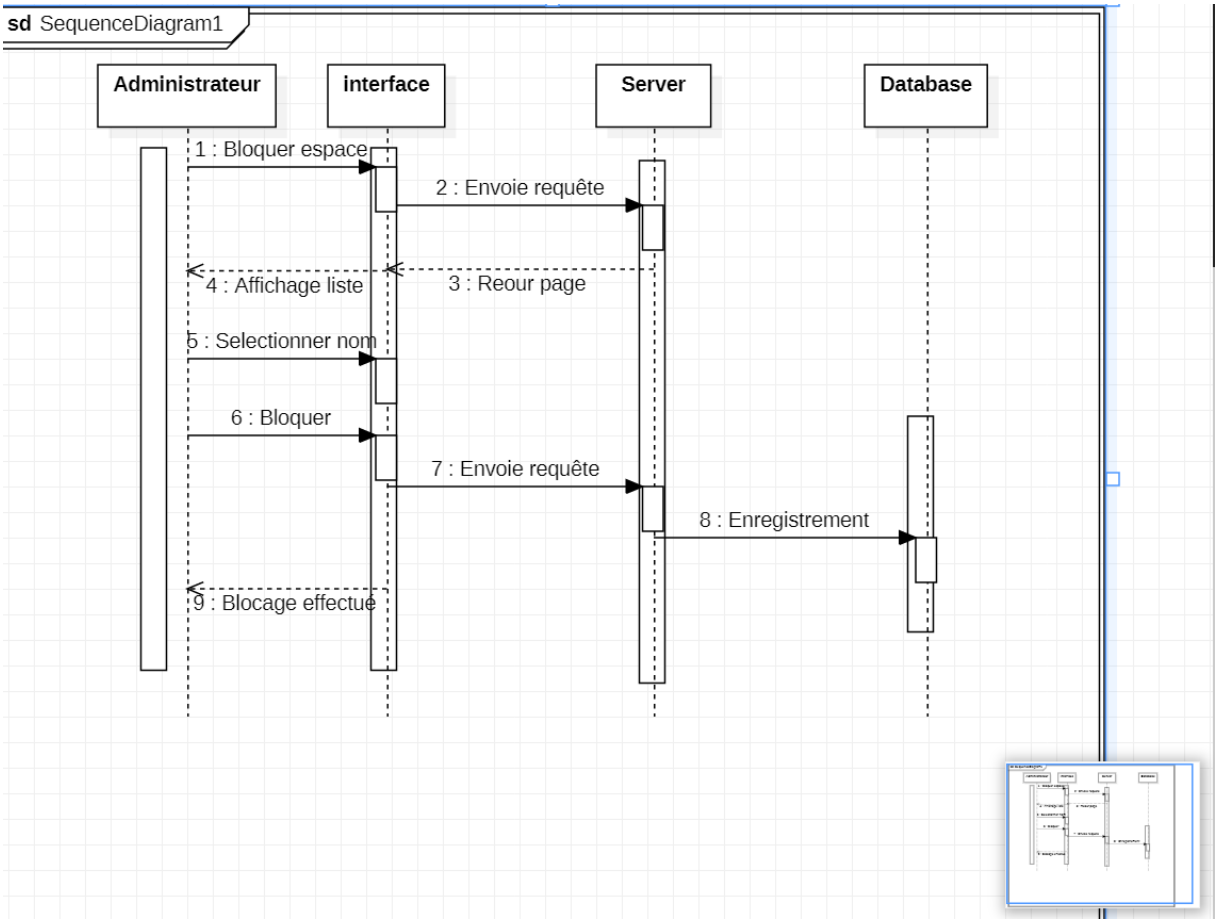

Figure 10:Diagramme de séquence blocage d'un espace

Dans cette figure nous expliquons la procédure de blocage d'un espace qui se fait comme suite :

D'abord l'administrateur clique sur le bouton bloquer un espace à travers l'interface et par la suite l'interface transmet la requête au server qui envoie un retour à l'interface avec une liste.

Après choix de l'espace à supprimer l'administrateur clique sur supprimer et l'interface renvoie une requête au server et celui renseigne la base de données pour donner la réponse « Suppression effectuée ».

2.4.7. Diagramme séquence supprimer espace

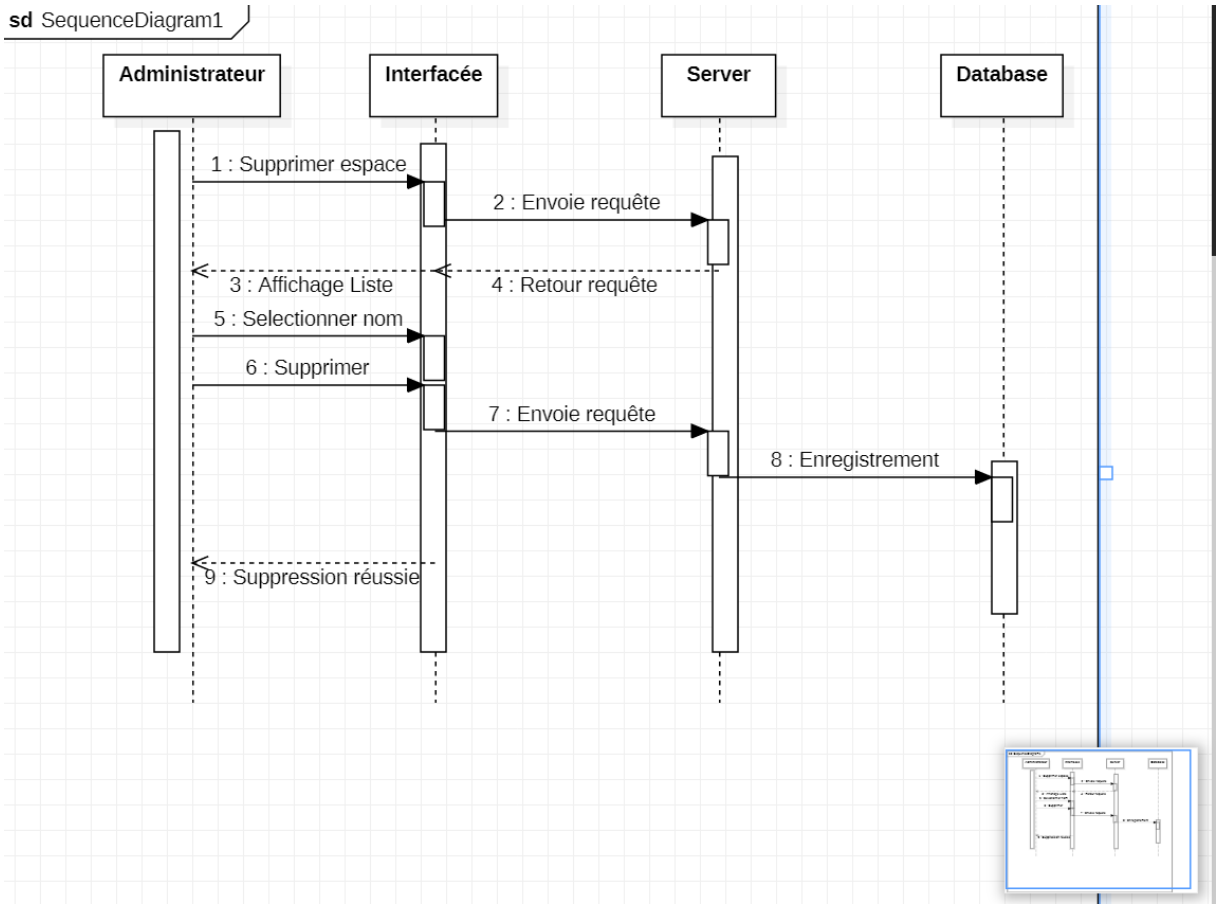

Figure 11:Diagramme de séquence suppression espace

Dans cette figure nous expliquons la procédure de suppression d'un espace qui se fait comme suite :

D'abord l'administrateur clique sur le bouton supprimer un client à travers l'interface et par la suite l'interface transmet la requête au server qui envoie un retour à l'interface avec une liste, après affichage de la liste l'administrateur clique sur l'espace à supprimer et l'interface renvoie une requête au server et celui renseigne la base de données pour donner la réponse « Suppression effectuée ».

2.5. Diagramme d'activité

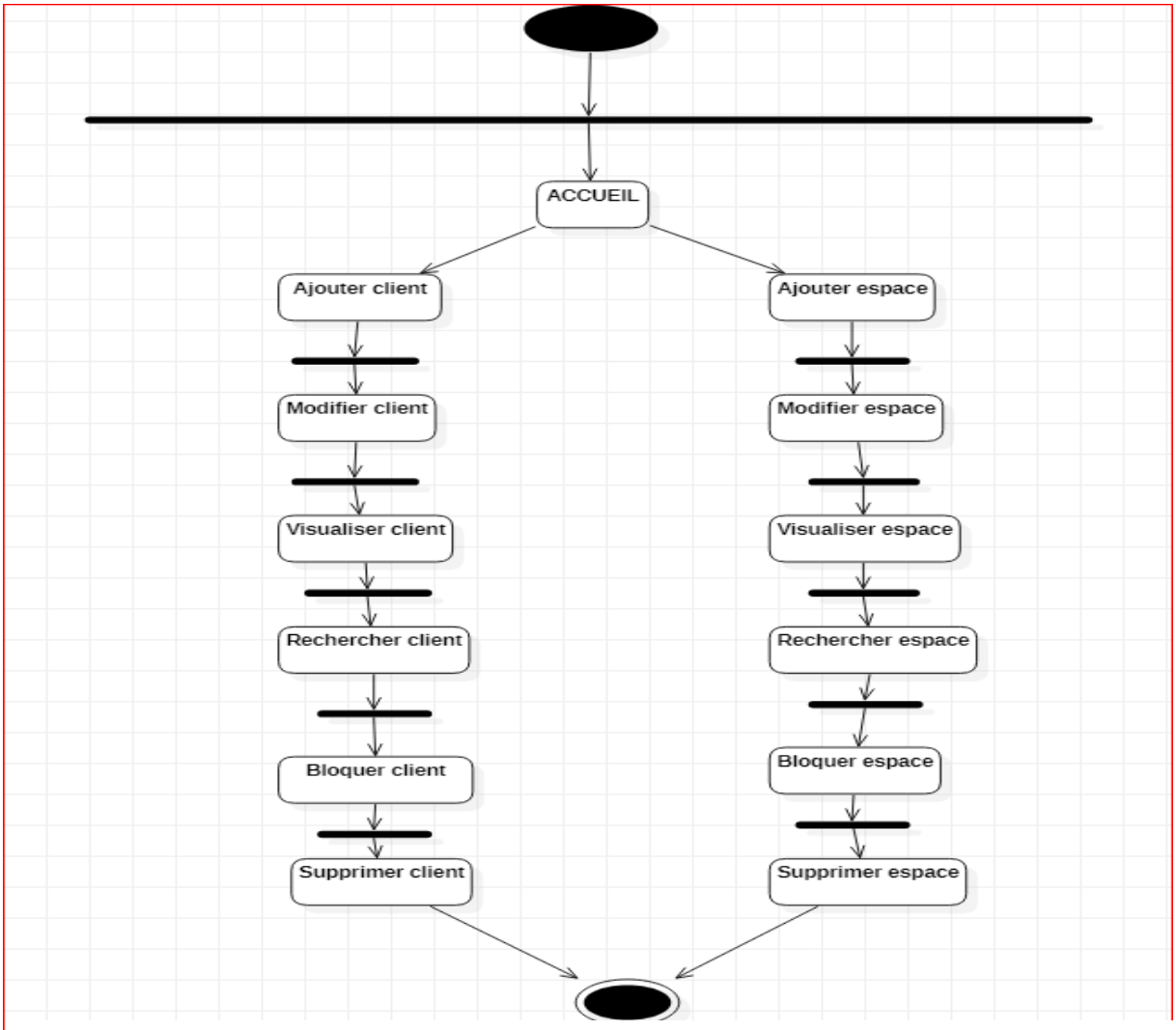

Figure 12:Diagramme d'activité

Dans cette figure nous expliquons le fonctionnement de l'application coté administrateur qui se fait comme suite :

L'administrateur agit sur la plateforme (côté administrateur) avec ces opérations.

2.6. Conception de notre système

2.6.1. Les données de notre système

Dans cette partie, nous proposons une liste des données (exprimées en termes de classes) de notre système en deux grandes catégories : les données de paramétrage et les données de production. Les données de paramétrage sont les données gérées depuis l'espace d'administration de l'application. Les données de production sont les données produites par les utilisateurs (ainsi que les données sur les utilisateurs eux-mêmes) depuis les différents espaces dédiés.

2.6.2. Les données de paramétrage

Classe	Description
Area	Une classe abstraite qui représente tout espace dédié dans notre application. Elle est héritée par les classes concrètes représentant les espaces familiaux (Family Area), des structures de santé (Health Structure Area) et des structures non sanitaires (Non Health Structure Area).
Family Area	Représente les espaces dédiés aux familles pour gérer les dossiers médicaux des membres.
Health Structure Area	Représente les espaces dédiés aux structures de santé.
Non Health Structure Area	Représente les espaces dédiés aux structures non sanitaires telles que les banques, les assurances, …
User Type	Représente les types ou profils utilisateurs qui accèdent à l'application via les espaces dédiés : responsable d'espace, médecin, secrétaire, membre de famille, …
Permission	Représente les permissions accordées aux profils utilisateurs. On dira par exemple que les responsables d'espace peuvent ajouter des utilisateurs dans leurs espaces, les médecins peuvent accéder aux dossiers médicaux, les médecins peuvent ajouter des actes médicaux, etc.
User	Représente les utilisateurs qui accèdent à l'application via les espaces dédiés.
Customer type	Représente les types de clients de notre application : responsable d'espace familiale ; responsable de structure de santé, responsable de structure non sanitaire.

Customer	Représente les clients (responsables des espaces dédiés) de notre application. Un client est aussi un utilisateur.
Fonctionnality	Représente les fonctionnalités accessibles aux utilisateurs d'un espace dédié à une structure de santé depuis le menu de l'application. Elle permet à un responsable de structure de santé de donner des accès aux utilisateurs de l'espace de sa structure.
Pathology	Représente l'ensemble des pathologies pouvant être diagnostiqués par les médecins lors des consultations médicales.
Consultation Type	Représente les types de consultations médicales.
Type of Medical Procedure	Représente les différents types d'actes médicaux.

Tableau 3:Liste des classes représentant les données de paramétrage de notre système

2.6.3. Les données de production

Classe	Description
Patient	Représente un patient. Elle comporte les données identitaires du patient et ses constantes médicales.
Consultation	Représente une consultation, c'est-à-dire une rencontre entre un médecin et un patient au cours de laquelle le médecin peut produire un acte médical.
Medical Procedure	Représente un acte médical, c'est-à-dire, « *tout acte dont la réalisation par des moyens verbaux, écrits, physiques ou instrumentaux, est effectué par un membre d'une profession médicale, dans le cadre de son exercice et les limites de sa compétence* ».

Tableau 4. Liste des classes représentant les données de production de notre système

2.7. Le diagramme de classes

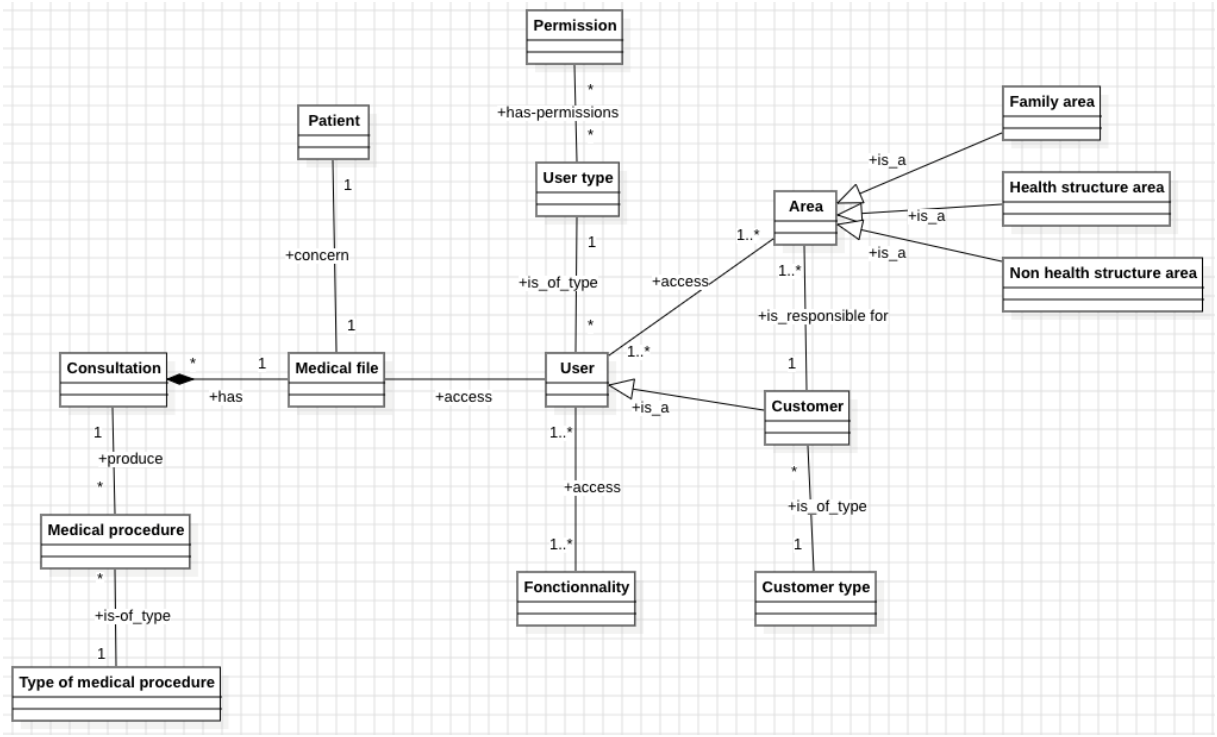

Figure 13. Diagramme de classes de notre système

Comme on peut le voir dans le diagramme de classes, un dossier médical concerne un et un seul patient (un patient a aussi un et un seul dossier médical) et comporte pour le moment que des consultations[1]. Les consultations produisent des actes médicaux qui sont les données médicales du patient. Les utilisateurs accèdent aux dossiers médicaux via les espaces dédiés, selon les permissions accordées et les fonctionnalités dont ils ont accès.

Chapitre 3. Implémentation et présentation de notre système

1. Implémentation de notre système

1.1. Vue d'ensemble sur l'implémentation de notre système

Il faut noter que nos travaux de mémoire constituent la première étape dans la construction du « Guichet Médical Unique ». Nous avons donc initié le développement de la plateforme en créant un « projet Django ». En effet, pour mettre en place une plateforme avec Django, il faut d'abord créer un « projet Django », puis y ajouter des « applications Django » qui constituent les modules applicatifs de la plateforme. Ensuite, développer/implémenter un « projet » dans Django, c'est développer/implémenter une-à-une chaque application ajoutée. Le développement « d'application Django » consiste à implémenter chacun des composants du MVT : le modèle, les Vues et les Templates.

Pour notre cas, le « projet Django » est créé avec le nom « gmu-sn » (pour guichet médical unique du Sénégal). Nous avons ensuite ajouté une « application Django » appelée « medicalfile » pour représenter ce que nous avons appelé le « noyau » de la plateforme dans la section 3.2 du chapitre 1. Cette application comporte à la fois le modèle de dossier médical que nous proposons et une personnalisation du « modèle utilisateur » de Django qui nous sert de base pour implémenter et utiliser son « système d'authentification et d'autorisations » afin de gérer les accès aux dossiers médicaux (le module « Access manager »).

Par ailleurs, le scaffolding Django nous a fourni le module d'administration générale de la plateforme (il s'agit de l'application « admin » intégrée dans Django) qui nous permet d'administrer les données de paramétrage présentées dans la section 4.2 du chapitre 2.

Dans la suite, nous présentons les détails d'implémentation de l'application « medicalfile » en se focalisant sur l'implémentation de notre modèle de dossier médical, du module d'administration et du module de gestion des accès aux dossiers médicaux.

2. Implémentation de l'application « medicalfile »

2.1.Implémentation de notre modèle de dossier médical

Dans Django, le développement du "Model" d'une application consiste à ajouter les différentes classes « métiers » de l'application dans le fichier "models.py". A priori, pour chaque classe ajoutée dans ce fichier, une table est créée par l'ORM Django dans la base de données. Ce qui permet d'enregistrer et de pérenniser un objet d'une classe donnée dans la table correspondante.

L'ensemble des classes du diagramme de la Figure 9 sont donc ajoutées dans le fichier « models.py » du « noyau », comme on peut le voir dans la figure suivante :

```
class UserType(models.Model): ...
class CustomerType(models.Model): ...
class MedicalConsultationType(models.Model): ...
class MedicalProcedureType(models.Model): ...
class MedicalConsultation(models.Model): ...
class MedicalProcedure(models.Model): ...
class Patient(models.Model): ...
class CustomUser(AbstractBaseUser, PermissionsMixin): ...
class Customer(AbstractBaseUser, PermissionsMixin): ...
class HelathStructure(models.Model): ...
class NonHelathStructure(models.Model): ...
class Family(models.Model): ...
```

Figure 14. Les différentes classes "métier" du "noyau"

2.2.Implémentation du module d'administration

Django fournit une interface d'administration dont on peut paramétrer et surcharger les comportements de base pour en faire un véritable back-office pour notre plateforme. Pour l'utiliser, il faut simplement activer l'application Django nommée "**django.contrib.admin**" en l'ajoutant dans la variable "**INSTALLED_APPS**" du fichier de configuration "**settings.py**" :

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'medicalfile',
]
```

Figure 15. Ajout de l'application "admin" de Django comme back-office du "gmu"

Pour administrer notre application « medicalfile » depuis le back-office, il suffit d'ajouter les classes de son fichier "**models.py**" qui comportent les données de paramétrage (il s'agit des classes présentées dans la section 4.2 du chapitre 2), sur lesquelles nous souhaitons effectuer des opérations CRUD, dans un fichier « **admin.py** » (fichier à créer). De cette façon, pour chaque classe ajoutée, nous disposons d'une interface d'administration qui permet les opérations CRUD sur les données de cette classe.

2.3.Implémentation du « système d'authentification et d'autorisations »

Django fournit un système d'authentification et d'autorisation, sur la base d'utilisation des « sessions », qui permet de vérifier les informations d'identification des utilisateurs et de définir les actions que chaque utilisateur est autorisé à effectuer. Ce système comporte des modèles intégrés pour représenter des utilisateurs (classe Django « User »), des groupes d'utilisateurs (class Django « Group », une manière générique d'appliquer des autorisations à plusieurs utilisateurs à la fois) et des autorisations qui indiquent si un utilisateur peut effectuer certaines tâches ; accéder à des formulaires et des vues (c'est-à-dire des fonctionnalités) ; etc.

Le « système d'authentification et d'autorisations » est une application intégrée dans Django qui est activé automatiquement à la création d'un projet. Sa configuration est faite dans les variables « INSTALLED_APP » et « MIDDLEWARE » du fichier de configuration du projet.

Pour notre cas, nous utilisons ce « système d'authentification et d'autorisations » de Django en personnalisant son « modèle utilisateur » avec la classe « CustomUser » de la figure 7. Cette classe est ensuite « administré » dans le back-office pour gérer les utilisateurs/groupe d'utilisateurs de la plateforme et leurs autorisations.

Nous avons enfin mis en place les différents « templates » (fenêtres) d'authentification : connexion, déconnexion, changement de mot de passe ; récupération de mot de passe ; etc.

3. Présentation des interfaces de notre système

3.1.Le module d'administration

Le module d'administration est le back-office de la plateforme qui permet de gérer les données de paramétrage de la plateforme, les utilisateurs, groupes d'utilisateurs et les autorisations. Dans cette partie, nous montrons l'interface d'administration des données de paramétrage (avec l'exemple de gestion des espaces dédiés aux structures de santé) ; l'interface de gestion des utilisateurs et l'interface de gestion des autorisations.

3.1.1. Interfaces d'administration des données de paramétrage

Ces interfaces permettent d'implémenter les opérations CRUD sur les données de paramétrage avec une fenêtre de visualisation (Read) offrant la possibilité de choisir un élément et le supprimer (Delete), des formulaires d'ajout (Create) et de modification (Update).

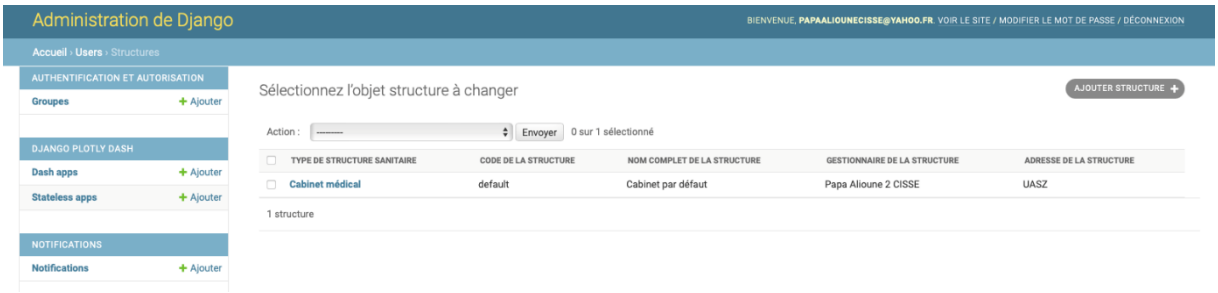

Figure 16. Fenêtre de visualisation de la liste des structures de santé

Cette fenêtre montre la liste des structures de santé ajoutées et dont des espaces de type « Health Structure area » leurs sont dédiés. Il est possible de sélectionner une structure (en cochant la case devant) et d'appliquer l'action « supprimer » pour la supprimer, comme sur la fenêtre suivante :

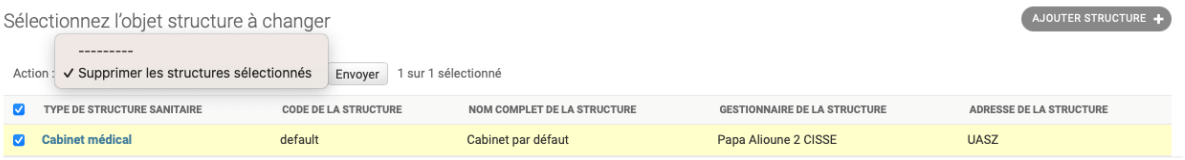

Figure 17. Fenêtre de suppression d'une structure de santé

En cliquant sur le boutton « Ajouter Structure », le formulaire suivant s'ouvre pour permettre d'ajouter une structure de santé :

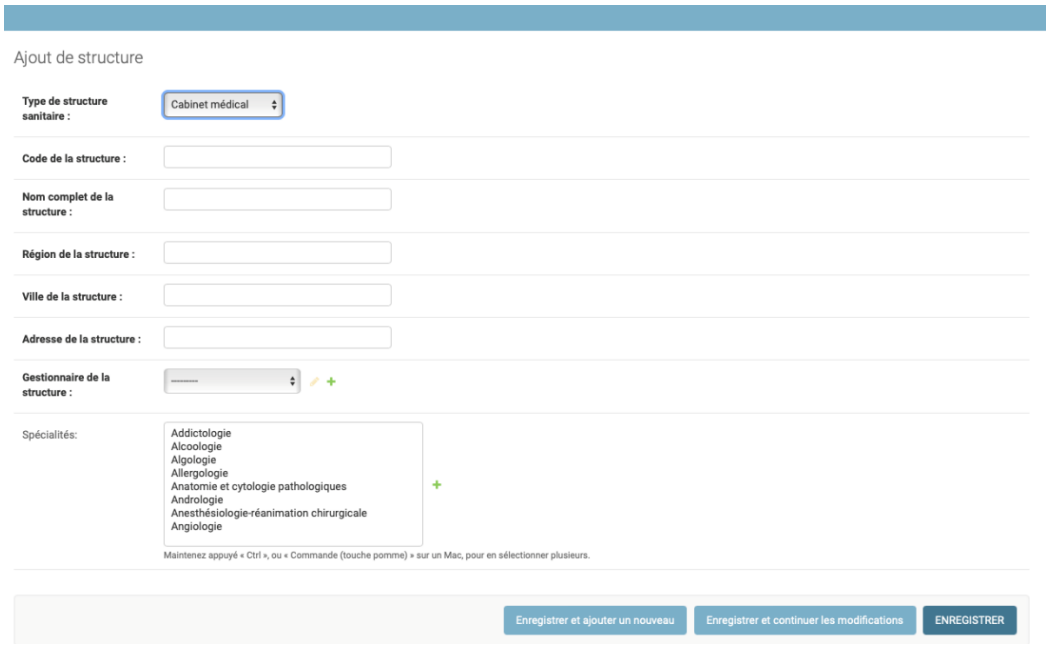

Figure 18. Formulaire d'ajout d'une structure de santé

En cliquant sur une structure de santé (l'expression en « bleue ») de la figure 9, un formulaire prérempli avec les données de la structure s'ouvre pour permettre de la mettre à jour :

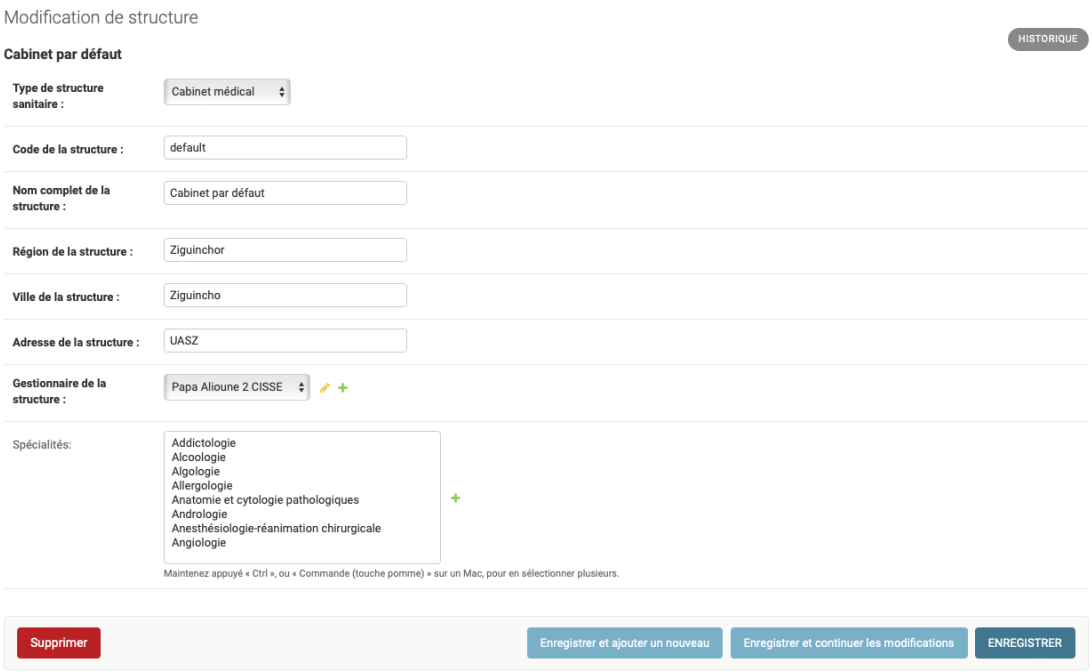

Figure 19. Formulaire de mise à jour d'une structure de santé

3.1.2. Interfaces de gestion des utilisateurs

La gestion des utilisateurs suit la même logique que celle présentée précédemment : il y a des interfaces pour les opérations CRUD. Nous montrons ici seulement la fenêtre de visualisation de la liste des utilisateurs ajoutés :

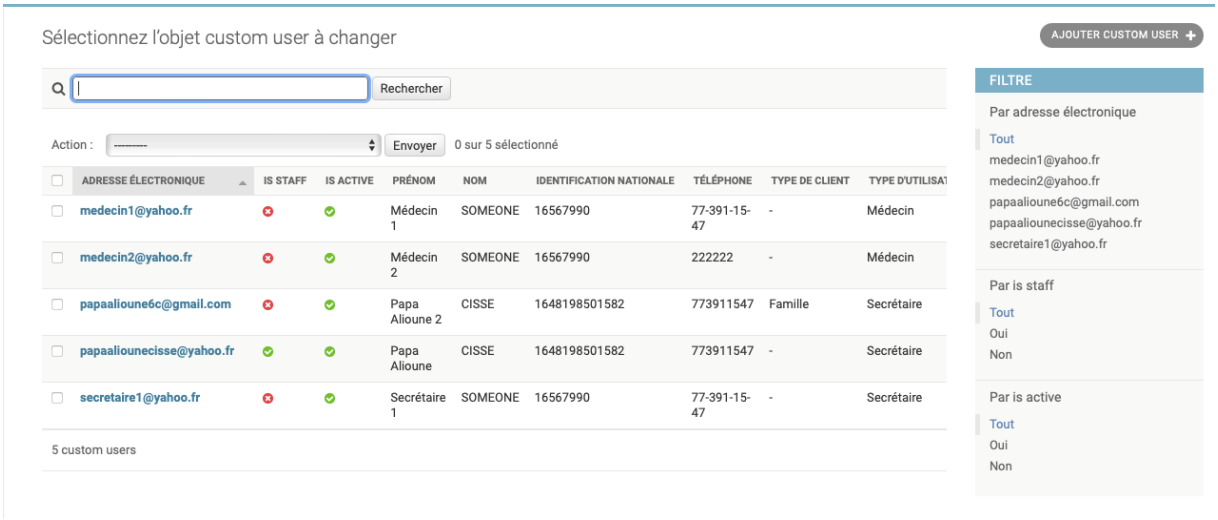

Figure 20. Fenêtre de visualisation de la liste des utilisateurs

3.1.3. Interface de gestion des autorisations

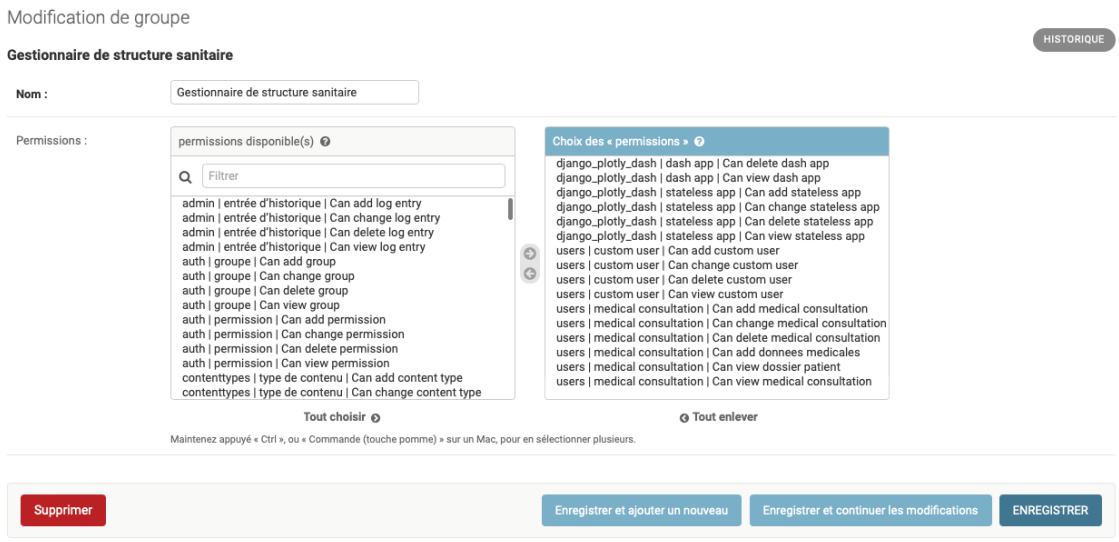

Figure 21. Fenêtre de gestion des permissions (autorisations) dans les structures de santé

Dans cette fenêtre, nous définissons les autorisations accordées aux gestionnaires de structure de santé. On peut voir qu'ils peuvent, entre autres, gérer des utilisateurs (il s'agit ici des utilisateurs de l'espace dédié à une structure de santé), accéder aux dossiers médicaux enregistrés dans leurs structures, … Quand un utilisateur de notre plateforme est ajouté comme

« responsable d'une structure de santé », il appartient au "groupe" « gestionnaire de structure de santé » et possède toutes les autorisations dans « Choix des permissions » de la figure précédente.

Ensuite, plus tard, il suffira de mettre par exemple le "décorateur" « *@permission_required(medicalfile.add_patient')* » devant une fonctionnalité (une « view ») pour restreindre son accès aux utilisateurs possédant l'autorisation d'ajouter un patient.

Dans l'exemple suivant :

```
@login_required
@permission_required('medicalfile.view_patient')
@permission_required('medicalfile.add_patient')
def addPatient(request):
```

Figure 22. Utilisation des permissions/autorisations pour restreindre l'accès à une fonctionnalité

Il est établi que la fonctionnalité d'ajout d'un patient n'est accessible qu'aux utilisateurs authentifiés et qui ont les autorisations de visualisation et d'ajout d'un patient.

3.2.Les espaces dédiés

Nous montrons ici quelques interfaces des espaces dédiés aux structures de santé (Health Structure area) et aux familles (Family area).

3.2.1. Espace dédié aux structures de santé

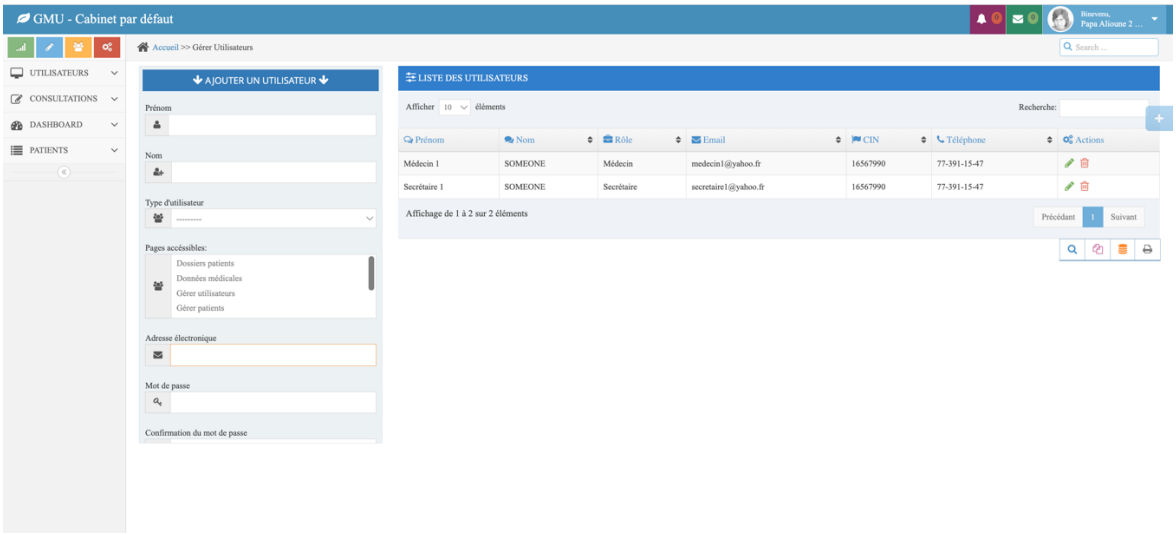

Figure 23. Fenêtre d'administration d'un espace dédié à une structure de santé

Dans la figure 10, nous voyons qu'une structure de santé nommé « Cabinet par défaut » (dont le code est « default » et le gestionnaire/responsable « Papa Alioune 2 CISSE ») est créée. La fenêtre ci-dessus (Figure 16) permet au responsable (connecté et visible au coin supérieur droit de la fenêtre) d'administrer les utilisateurs de son espace dédié.

La partie droite de cette fenêtre permet d'ajouter un utilisateur en précisant les fonctionnalités (« Pages accessibles ») qui lui seront accessibles. La partie à droite donne une visualisation, sous forme de tableau, de la liste des utilisateurs ajoutés. A côté de chacun des membres de cette liste, il y'a deux boutons :

- Le bouton de modification (crayon) sur lequel cliquer pour modifier les informations sur un utilisateur. En cliquant sur ce bouton, le formulaire de gauche est prérempli avec les informations de l'utilisateur pour permettre au responsable de les modifier.
- Le bouton de suppression (poubelle) qui permet de supprimer un utilisateur de l'espace dédié à cette structure.

3.2.2. Espace dédié aux familles

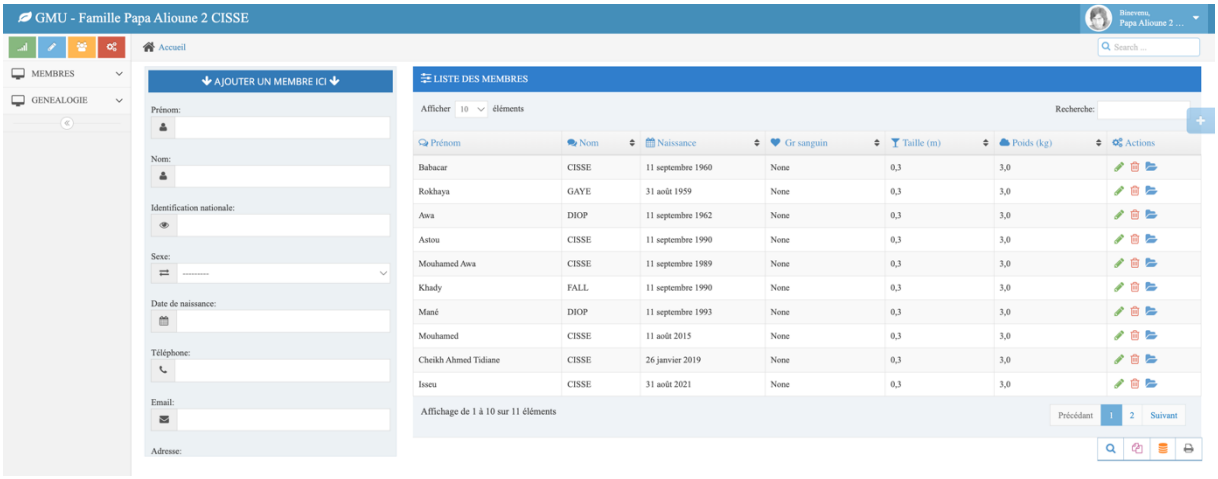

Figure 24. Fenêtre d'administration d'un espace dédié à une famille

Cette fenêtre a la même configuration que la précédente et permet de gérer les membres d'une famille. Il y a un bouton (le dossier) en plus à côté de chaque membre qui permet d'ouvrir et accéder à son dossier médical.

CONCLUSION

En conclusion, notre application de gestion des données médicales représente une pierre angulaire dans la modernisation des soins de santé, offrant des bénéfices immédiats et des perspectives prometteuses pour l'avenir. En rationalisant la collecte, le stockage et l'accès aux informations médicales, ces applications améliorent l'efficacité des professionnels de la santé, conduisant à des diagnostics plus rapides et à des traitements plus ciblés. Les avancées en matière d'intelligence artificielle et d'analyse de données ouvrent la voie à une médecine plus personnalisée, capable d'anticiper les besoins individuels des patients.

Cependant, pour concrétiser pleinement ces perspectives, il est impératif de surmonter des défis tels que la sécurité des données, la conformité réglementaire et l'interopérabilité des systèmes. L'évolution vers des normes communes et des protocoles de partage sécurisé des données médicales est essentielle pour favoriser une collaboration transparente entre les établissements de santé, améliorer la coordination des soins et stimuler la recherche médicale.

À mesure que la technologie continue de progresser, les applications de gestion des données médicales sont susceptibles de jouer un rôle croissant dans la prévention des maladies, la promotion de la santé et la création d'un écosystème médical plus connecté. En investissant dans la recherche et le développement, tout en restant attentif aux préoccupations éthiques et à la protection de la vie privée, nous pouvons anticiper une transformation positive du paysage médical, offrant des soins de santé plus accessibles, personnalisés et axés sur les résultats.

Références

[1] « Conception ». Consulté le: 30 novembre 2023. [En ligne]. Disponible sur: http://gpp.oiq.qc.ca/conception.htm

[2] « Application (informatique) », *Wikipédia*. 21 septembre 2023. Consulté le: 30 novembre 2023. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Application_(informatique)&oldid=208071460

[3] admin, « Qu'est ce qu'une application mobile ? », Numidev. Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://www.numidev.fr/une-application-mobile-cest-quoi/

[4] « Définitions et usages de : Application informatique - Dictionnaire de français sensagent ». Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://dictionnaire.sensagent.com/Application%20informatique/fr-fr/

[5] « Définitions et usages de : Application informatique - Dictionnaire de français sensagent ». Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://dictionnaire.sensagent.com/Application%20informatique/fr-fr/

[6] e.marciel@beta.one, « L'établissement de santé : définition et missions », Médical RH. Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://medical-rh.com/etablissement-de-sante/

[7] samuel, « Optimiser la gestion de rendez-vous », Petite Entreprise. Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://www.petite-entreprise.net/P-3986-83-G1-bien-gerer-la-prise-de-rendez-vous.html

[8] « Gestion du personnel : les bonne pratiques à mettre en place ». Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://payfit.com/fr/fiches-pratiques/gestion-du-personnel/

[9] Amandine, « Gestion des dossiers patients : le guide | CEF », Centre Européen de Formation. Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://www.centre-europeen-formation.fr/blog/sante-et-social/gestion-dossiers-patients/

[10] « À propos de DHIS2 - DHIS2 », https://dhis2.org/fr/. Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://dhis2.org/fr/about/

[11] « Projets – SIMENS ». Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://www.simens.sn/projets/

[12] « StarUML », *Wikipédia*. 10 octobre 2023. Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=StarUML&oldid=208598526

[13] « Diagramme de cas d'utilisation », *Wikipédia*. 21 novembre 2021. Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Diagramme_de_cas_d%27utilisation&oldid=188203003

[14] « Diagramme de classes », *Wikipédia*. 4 mars 2023. Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Diagramme_de_classes&oldid=201972343

[15] « Diagramme d'activité », *Wikipédia*. 17 janvier 2020. Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Diagramme_d%27activit%C3%A9&oldid=166458857

[16] « Apprendre PHP & MySQL : Design pattern : MVC ». Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://laconsole.dev/formations/php/design-pattern-mvc/

[17] « ChatGPT ». Consulté le: 1 décembre 2023. [En ligne]. Disponible sur: https://chat.openai.com

[18] « ChatGPT ». Consulté le: 1 décembre 2023. [En ligne]. Disponible sur: https://chat.openai.com

[19] « Qu'est-ce PostgreSQL », Oracle France. Consulté le: 24 novembre 2023. [En ligne]. Disponible sur: https://www.oracle.com/fr/database/definition-postgresql.html

TABLES DES MATIÈRES