



4IIR G5 2024/2025

Rapport de projet

Développement d'une application web intelligente pour l'assistance médicale

Ingénierie Informatique et Réseaux

Réalisé par :

BENHIDA Oussama

BENJIMA Yassir

JOUDAR Samia

LAACHER Mariam

Encadré par :

M. LOOUNOUSSE Jawad

Dédicace

“

Nous dédions ce travail à toutes les personnes qui nous ont soutenus de près ou de loin tout au long de la réalisation de ce projet.

Un remerciement particulier à nos familles respectives pour leur soutien inconditionnel, ainsi qu'à nos enseignants pour leurs conseils précieux.

Ce projet est le fruit d'un véritable travail d'équipe, basé sur la collaboration, le respect et la volonté commune de réussir. Nous tenons à exprimer notre reconnaissance envers chacun des membres du groupe pour leur engagement et leur esprit d'équipe.

Yassir Benjima, Mariam Laacher, Oussama Benhida et Samia Joudar

”

Abstract

This project aims to design and develop **MEDISENSE**, an innovative web application that offers personalized medical assistance and mental health support using artificial intelligence. The main objective is to improve access to healthcare by providing users with intelligent tools to analyze symptoms, receive tailored medical guidance, and interact in real time with a mental health chatbot powered by AI.

The platform enables users to describe their symptoms, obtain recommendations for appropriate medical specialists, and access mental health support through a conversational interface integrated with **Gemini**, a generative AI model. The system supports three user roles — **Doctors, Assistants, and Patients** — allowing secure interactions, follow-up management, and coordinated care. Doctors and assistants can access patient records, manage appointments, and provide support, while patients benefit from real-time, personalized recommendations.

Built using **Django** for the backend, the application offers a modular and secure architecture, with a RESTful API and a responsive web interface for a seamless experience across devices. The platform emphasizes data privacy, scalability, and usability for both patients and healthcare providers.

This solution responds to the growing need for accessible and intelligent healthcare services by reducing long wait times, improving communication between users and professionals, and providing reliable, AI-driven support.

Keywords : Django, Gemini, health assistant, mental health chatbot, patient management, AI healthcare, REST API, personalized medicine, responsive web platform

Résumé

Ce projet consiste à concevoir et développer **MEDISENSE**, une application web innovante d'assistance médicale reposant sur l'intelligence artificielle. L'objectif principal est d'améliorer l'accès aux soins de santé en fournissant aux utilisateurs des recommandations médicales personnalisées ainsi qu'un accompagnement en santé mentale, accessible à tout moment.

La plateforme permet aux utilisateurs de décrire leurs symptômes, d'obtenir des suggestions de spécialistes adaptés, et d'interagir avec un chatbot intelligent basé sur le modèle **Gemini**, dédié au soutien en santé mentale. Le système prend en charge plusieurs types d'utilisateurs : **Médecins**, **Assistants** et **Patients**. Chaque profil dispose d'un espace sécurisé lui permettant de gérer les informations médicales, les consultations, et les échanges.

L'application est développée avec le framework Django côté serveur, offrant une architecture sécurisée et évolutive via une API REST. L'interface utilisateur, intuitive et responsive, garantit une expérience fluide sur tous types de supports. Ce socle technique assure la performance, la confidentialité des données et une accessibilité optimale aux différents acteurs du système de santé.

Mots-clés : assistance médicale, santé mentale, chatbot médical, Django, API REST, intelligence artificielle, Gemini, gestion des patients, plateforme web responsive

Glossaire

API	<i>Application Programming Interface</i> : Interface logicielle permettant à différentes applications de communiquer via des points d'accès normalisés.
Django	Framework web open-source en Python facilitant le développement rapide d'applications web sécurisées et maintenables.
Gemini	Modèle d'intelligence artificielle générative développé par Google, utilisé ici pour analyser les symptômes et fournir un soutien conversationnel.
IA	<i>Intelligence Artificielle</i> : Domaine de l'informatique visant à simuler l'intelligence humaine, notamment dans l'analyse, la prédiction et la décision.
Chatbot	Agent conversationnel automatisé capable d'interagir avec les utilisateurs via un langage naturel, ici utilisé pour le soutien en santé mentale.
Base de données	Système structuré de stockage des informations liées aux patients, médecins, assistants, symptômes, et consultations.
Sécurité	Ensemble des mécanismes assurant la protection des données personnelles et le contrôle d'accès au système.
Frontend	Partie visible de l'application (interface utilisateur), développée pour être intuitive, responsive et accessible sur tout support.
Backend	Partie serveur de l'application, responsable du traitement des données, de la logique métier et de l'interfaçage avec la base de données.

Table des figures

1.1	une itération selon la méthode Scrum	6
2.1	UML logo	10
2.2	Diagramme de cas d'utilisation	11
2.3	Diagramme de Séquence du cas d'utilisation «Login»	14
2.4	Diagramme de classe	15
3.1	Architecture logique	19
4.1	Structure de la base de données	25
4.2	Structure du Backend	26
4.3	Structure du Frontend	27
4.4	Page de connexion	28
4.5	Page d'inscription	28
4.6	Tableau de bord médical	29
4.7	Liste des médecins	29
4.8	Formulaire d'ajout de médecin	30
4.9	Liste des médecins	30
4.10	Formulaire d'ajout de médecin	31
4.11	Liste des patients	31
4.12	Formulaire d'ajout de patient	32
4.13	Liste des rendez-vous	32
4.14	Formulaire d'ajout de rendez-vous	33
4.15	Paramètres du profil utilisateur	33
4.16	Recherche de médecins disponibles	34
4.17	Recherche d'assistant disponibles	34
4.18	Interface de messagerie intelligente	35

Liste des tableaux

1.1	Besoins fonctionnels	4
1.2	Besoins fonctionnels	5
2.1	Description textuelle du cas «S'inscrire»	12
2.2	Description textuelle du cas «Se connecter»	13
2.3	Dictionnaire de données basé sur le diagramme de classes médical	16
3.1	Tableau des exigences techniques	18

Table des matières

Introduction générale	1
1 Contexte général du projet	2
1.1 Introduction	3
1.2 Présentation du projet	3
1.2.1 Problématique	3
1.2.2 Objectifs	3
1.2.3 Solution proposée	4
1.3 Spécifications des besoins	4
1.3.1 Besoins fonctionnels	4
1.3.2 Besoins non fonctionnels	5
1.4 La Méthodologie de développement	6
1.4.1 Choix de la méthodologie de développement	6
1.4.2 Présentation de la méthode SCRUM	6
1.4.3 Principes essentiels de la méthode SCRUM	7
1.4.4 Organisation de la méthode SCRUM	7
1.4.5 Spécification des rôles de chaque membre de notre équipe dans SCRUM	8
1.5 Conclusion	8
2 Analyse et Conception	9
2.1 Introduction	10
2.2 Présentation du langage UML	10
2.3 Diagramme de cas d'utilisation	11
2.3.1 Description textuelle du cas « S'inscrire »	12
2.3.2 Description textuelle du cas « Se connecter »	13
2.4 Diagramme de séquence « Login »	14
2.5 Diagramme de Classes	15
2.6 Dictionnaire de données	16
2.7 Conclusion	16
3 Étude technique du projet	17
3.1 Introduction	18
3.2 Besoins techniques	18
3.3 Présentation de l'architecture du projet	18
3.3.1 Architecture logique	18
3.3.2 Schéma de flux recommandé	19
3.4 Technologies et Langages Utilisés	20

Table des matières

3.4.1	Langages de Programmation	20
3.4.2	Frameworks	21
3.4.3	Bases de Données	21
3.4.4	Technologies Frontend	21
3.4.5	Développement et Tests d'API	22
3.4.6	Gestion de Version et Déploiement	22
3.4.7	APIs et Services	22
3.4.8	Autres outils	23
3.5	Conclusion	23
4	Mise en œuvre	24
4.1	Introduction	25
4.2	Structure du projet	25
4.2.1	Structure de la base de données	25
4.2.2	Structure du projet Backend	26
4.2.3	Structure du projet Frontend	27
4.3	Interfaces de l'application	28
4.3.1	Interface de connexion	28
4.3.2	Interface d'inscription	28
4.3.3	Interface du tableau de bord	29
4.3.4	Interface de gestion des médecins	29
4.3.5	Interface d'ajout de médecin	30
4.3.6	Interface de gestion des assistants	30
4.3.7	Interface d'ajout d'assistant	31
4.3.8	Interface de gestion des patients	31
4.3.9	Interface d'ajout de patient	32
4.3.10	Interface des rendez-vous	32
4.3.11	Interface d'ajout de rendez-vous	33
4.3.12	Interface des paramètres utilisateur	33
4.4	Interfaces de l'application Patient	34
4.4.1	Interface de recherche de médecins	34
4.4.2	Interface de recherche d'assistants	34
4.4.3	Interface de chat médical	35
4.5	Conclusion	35
Références		37

Introduction générale

L'accès aux soins de santé représente aujourd'hui un défi majeur, accentué par la croissance démographique, le manque de professionnels disponibles, et les délais d'attente prolongés. À cela s'ajoutent les problématiques liées à la santé mentale, encore trop souvent négligées ou mal prises en charge, notamment en raison du manque de ressources spécialisées ou de la stigmatisation sociale. Dans ce contexte, les technologies numériques et l'intelligence artificielle apparaissent comme des leviers puissants pour améliorer l'accès aux soins et la qualité de l'accompagnement médical.

C'est dans cette optique qu'a été imaginée MEDISENSE, une application web intelligente dédiée à l'assistance médicale personnalisée et au soutien en santé mentale. Le projet vise à proposer une solution innovante permettant à tout utilisateur de décrire ses symptômes, d'obtenir des recommandations adaptées, et de bénéficier d'un accompagnement instantané via un chatbot intelligent basé sur le modèle IA Gemini. MEDISENSE intègre également une gestion complète des rôles : patients, médecins et assistants peuvent interagir au sein d'un écosystème sécurisé et cohérent, favorisant un suivi médical efficace.

Ce rapport présente en détail le processus de développement de cette application. Après une analyse du contexte et des besoins, nous exposerons les choix techniques et fonctionnels effectués, notamment l'utilisation du framework Django pour le développement du backend, ainsi que les principes d'architecture modulaire, sécurisée et évolutive adoptés. Nous détaillerons également les fonctionnalités clés de la plateforme, le rôle des différentes parties prenantes, et les bénéfices attendus en matière d'accessibilité, de fiabilité et d'intelligence décisionnelle dans le domaine de la santé.

Chapitre 1

Contexte général du projet

1.1 Introduction

Ce premier chapitre a pour vocation d'introduire le projet MEDISENSE en exposant les problématiques actuelles du secteur de la santé, les objectifs de l'application, ainsi que les spécifications fonctionnelles et non fonctionnelles qui ont guidé sa conception. Il établit ainsi le cadre général du développement de cette plateforme intelligente, en mettant en lumière les besoins concrets des utilisateurs : patients, médecins et assistants médicaux.

1.2 Présentation du projet

1.2.1 Problématique

L'accès aux soins médicaux et au soutien psychologique demeure un défi majeur dans de nombreuses régions du monde. Les délais d'attente prolongés, la pénurie de spécialistes, la surcharge des structures de santé, ainsi que le manque d'informations fiables sur les symptômes ou les troubles mentaux, limitent fortement la prise en charge rapide et personnalisée des patients. Par ailleurs, la santé mentale, souvent stigmatisée, reste sous-traitée malgré son importance croissante.

1.2.2 Objectifs

Le projet MEDISENSE a pour objectif de développer une plateforme web intelligente dédiée à l'assistance médicale et au soutien en santé mentale. Les objectifs principaux sont les suivants :

- **Automatiser l'analyse des symptômes** et fournir des recommandations médicales adaptées aux utilisateurs en fonction de leurs descriptions.
- **Offrir un accompagnement en temps réel en santé mentale** via un chatbot conversationnel, accessible 24h/24 et 7j/7.
- **Faciliter la mise en relation avec des spécialistes médicaux** : les utilisateurs peuvent obtenir des recommandations de professionnels de santé en fonction de leurs besoins spécifiques.
- **Garantir une sécurité et une confidentialité des données** à travers des mécanismes de protection avancés (authentification, cryptage des données).
- **Fournir une interface utilisateur intuitive et accessible**, adaptée aux différents profils : patients, médecins et assistants médicaux.

1.2.3 Solution proposée

La solution repose sur la création d'une application web full-stack, composée de deux parties principales :

Un backend développé avec Django chargée de la logique métier, de l'authentification des utilisateurs, de la gestion des patients, des médecins, des assistants et des interactions avec le chatbot IA. Cette architecture est renforcée par l'utilisation de JWT pour sécuriser les échanges et garantir la confidentialité des données des utilisateurs.

Une interface web côté frontend utilisant Bootstrap destinée à offrir une expérience utilisateur intuitive et responsive, adaptée à tous les profils : patients, professionnels de santé et administrateurs. L'interface permet aux utilisateurs de décrire leurs symptômes, d'obtenir des recommandations personnalisées et d'interagir avec le chatbot en temps réel.

1.3 Spécifications des besoins

1.3.1 Besoins fonctionnels

Notre système vise à développer une plateforme répondant aux besoins fonctionnels suivants, répartis selon les différents acteurs : Administrateur, Médecin, Assistant et Patient.

Acteur	Fonctionnalités principales
Administrateur	Gestion des utilisateurs : Créer, modifier et supprimer des comptes pour les patients, les médecins et les assistants. Gestion des rendez-vous : Planifier, modifier ou annuler des rendez-vous entre les patients et les médecins ou assistants. Suivi des interactions : Superviser les interactions entre utilisateurs et le chatbot IA, garantir la qualité des échanges. Gestion des droits d'accès : Attribuer des rôles et des priviléges spécifiques aux utilisateurs en fonction de leur profil.
Médecin	Gestion des patients : Consulter les profils des patients, gérer les rendez-vous et fournir des recommandations médicales. Consultation de l'historique médical : Accéder à l'historique des symptômes, consultations et résultats des patients. Gestion de son profil : Modifier ses informations personnelles, son emploi du temps et ses spécialités. Connexion sécurisée : Accéder à la plateforme via un système d'authentification sécurisé (login, mot de passe).

TAB. 1.1 : Besoins fonctionnels

Acteur	Fonctionnalités principales
Assistant	Gestion des patients : Assister les médecins dans la gestion des dossiers et des rendez-vous des patients. Consultation de son profil : Accéder et mettre à jour ses informations personnelles et ses horaires. Rendez-vous avec les patients : Aider à la planification et à la gestion des rendez-vous. Connexion sécurisée : Authentification sécurisée pour accéder à l'espace personnel.
Patient	Gestion de son profil : Créer, modifier et supprimer son profil personnel (informations de santé, contact, etc.). Consultation des médecins et assistants : Accéder aux profils des médecins et assistants disponibles pour des consultations. Utilisation du chatbot IA : Interagir avec le chatbot pour des conseils médicaux immédiats ou pour un soutien en santé mentale. Planification des rendez-vous : Demander des rendez-vous avec des médecins ou assistants, gérer son calendrier médical. Connexion sécurisée : Se connecter via une authentification sécurisée pour accéder à son espace personnel.

TAB. 1.2 : Besoins fonctionnels

1.3.2 Besoins non fonctionnels

Les besoins non fonctionnels sont des contraintes internes et externes désirées du système, indispensables pour l'amélioration de la qualité logicielle de notre système. Ce dernier doit répondre aux exigences suivantes :

- **Performance** : Le système doit répondre rapidement, avec un temps de réponse inférieur à 2 secondes pour une requête courante.
- **Sécurité** : L'authentification se fait via JWT avec gestion des rôles et des autorisations. Le système doit également se protéger contre les attaques courantes telles que XSS, CSRF, etc.
- **Fiabilité** : L'application doit être stable, fonctionnelle sans interruption et assurer une haute disponibilité.
- **Ergonomie** : L'interface doit être intuitive, responsive (adaptée aux mobiles), fluide et agréable à utiliser.
- **Scalabilité** : Le système doit pouvoir s'adapter à une augmentation du nombre d'utilisateurs sans dégradation des performances.

1.4 La Méthodologie de développement

1.4.1 Choix de la méthodologie de développement

Le choix entre une méthode et une autre dépend de la nature du projet et de sa taille. Pour des projets de petite taille et dont le domaine est maîtrisé, par exemple, un cycle de vie en cascade s'avère largement suffisant. Lorsqu'il s'agit d'un projet où les données ne sont pas réunies dès le départ, où les besoins sont incomplets voire flous, il faut s'orienter vers une méthode itérative ou orientée prototypes.

Parmi les méthodes itératives, nous pouvons distinguer les méthodes AGILE largement utilisées de nos jours à travers le monde. Une méthode AGILE est menée dans un esprit collaboratif et s'adapte aux approches incrémentales. Elle engendre des produits de haute qualité tout en tenant compte de l'évolution des besoins du client. Une méthode AGILE assure une meilleure communication avec le client et une meilleure visibilité du produit livrable. Elle permet aussi de gérer la qualité en continu et de détecter des problèmes le plus tôt au fur et à mesure, permettant ainsi d'entreprendre des actions correctrices sans trop de pénalités dans les coûts et les délais.

Il y a moult méthodes AGILE et il ne s'agit pas de choisir la meilleure méthode parmi celles existantes. Il s'agit plutôt de sélectionner la méthode la plus adaptée à notre projet.

1.4.2 Présentation de la méthode SCRUM

Le principe de la méthodologie SCRUM est de développer un logiciel de manière incrémentale en maintenant une liste totalement transparente des demandes d'évolutions ou de corrections à implémenter.

Avec des livraisons très fréquentes, toutes les 4 semaines en moyenne, le client reçoit un logiciel fonctionnel à chaque itération. Plus nous avançons dans le projet, plus le logiciel est complet et possède toujours de plus en plus de fonctionnalités. Pour cela, la méthode s'appuie sur des développements itératifs à un rythme constant d'une durée de 2 à 4 semaines.

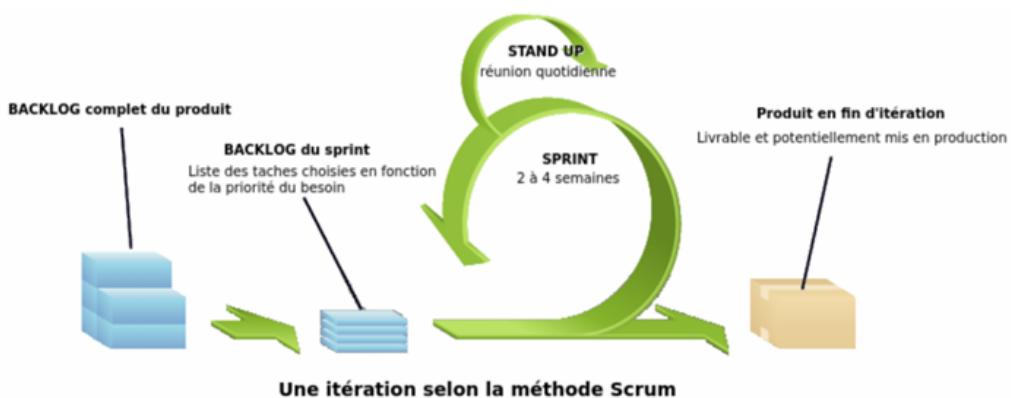


FIG. 1.1 : une itération selon la méthode Scrum

Comme nous pouvons le remarquer dans cette figure, pour mettre en place la méthode SCRUM, il faut tout d'abord définir les différentes fonctionnalités de notre application qui forment le backlog du produit. Ensuite, nous procémons à la planification du sprint pour définir le plan détaillé d'une itération. Les sprints durent généralement deux à quatre semaines. Durant un sprint, il y a toujours des réunions quotidiennes entre les différents collaborateurs du projet afin de présenter l'état d'avancement des différentes tâches en cours, les difficultés rencontrées ainsi que les tâches restantes à réaliser. Une fois le produit partiel est prêt, nous vérifions la conformité de ce qui a été fait durant le sprint et nous pouvons alors l'améliorer en procédant à l'étape de rétrospective.

1.4.3 Principes essentiels de la méthode SCRUM

Nous pouvons remarquer quatre valeurs principales dans les méthodes agiles :

- **L'équipe** : nous nous concentrons sur les personnes et leurs interactions plutôt que sur les processus et les outils.
- **L'application** : le plus important, c'est d'avoir une application fonctionnelle plutôt que d'avoir une documentation complète.
- **La collaboration** : cette méthode se base sur la collaboration avec le client.
- **L'acceptation du changement** : nous ne suivons pas un plan fixe, mais nous réagissons à chaque nouveau changement.

1.4.4 Organisation de la méthode SCRUM

La méthodologie SCRUM fait intervenir 3 rôles principaux qui sont :

- **Product owner** : dans la majorité des projets, le responsable produit (Product owner) est le responsable de l'équipe projet client. C'est lui qui va définir et prioriser la liste des fonctionnalités du produit et choisir la date et le contenu de chaque sprint sur la base des valeurs (charges) qui lui sont communiquées par l'équipe.
- **Scrum Master** : véritable facilitateur sur le projet, il veille à ce que chacun puisse travailler au maximum de ses capacités en éliminant les obstacles et en protégeant l'équipe des perturbations extérieures. Il porte également une attention particulière au respect des différentes phases de SCRUM.
- **Equipe** : l'équipe s'organise elle-même et elle reste inchangée pendant toute la durée d'un sprint. Elle doit tout faire pour délivrer le produit.

1.4.5 Spécification des rôles de chaque membre de notre équipe dans SCRUM

- **Product owner** : Yassir Benjima.
- **Scrum Master** : M. LOUNOUSSE Jawad.
- **Equipe** : Yassir Benjima, Oussama Benhida, Mariam Laache, Samia Joudar.

1.5 Conclusion

Ce chapitre a posé les fondations du projet en exposant clairement le besoin de digitalisation des services d'assistance médicale, notamment en matière de recommandation de soins et de suivi en santé mentale. En identifiant les différents utilisateurs, leurs besoins spécifiques et les contraintes techniques, il a permis de définir les spécifications essentielles du système à concevoir. Cette vision structurée servira de base solide pour l'implémentation technique et le développement de l'application MEDISENSE, qui sera présentée dans les chapitres suivants. Grâce à cette approche, MEDISENSE pourra répondre de manière efficace aux défis actuels de l'accès aux soins, tout en offrant une expérience utilisateur optimale et sécurisée.

Chapitre 2

Analyse et Conception

2.1 Introduction

Dans ce chapitre, je vais présenter une conception et une analyse basées sur l'étude effectuée dans le chapitre précédent.

La plupart des langages de programmation sont orientés objets. Le passage de la programmation fonctionnelle à la programmation orientée objet n'est pas facile. L'un des défis est d'avoir une idée globale à l'avance de ce que nous devons programmer.

L'algorithmique utilisée dans la programmation fonctionnelle ne suffit pas à elle seule. Il est nécessaire d'avoir des méthodes ou des langages pour la modélisation des langages orientés objets. Ainsi, plusieurs méthodes ou langages ont vu le jour, notamment UML, qui nous a permis de concevoir notre système.

De nos jours, UML2 possède treize diagrammes classés en deux catégories : dynamique et statique. Dans ce chapitre, je vais commencer par les diagrammes de cas d'utilisation, qui permettent de donner une vue globale du système, non seulement pour un client non avisé qui aura une idée de sa future application, mais aussi pour le développeur qui s'en servira pour le développement des interfaces.

Ensuite, j'affinerai la conception en présentant la chronologie des opérations à l'aide de diagramme de cas d'utilisations et de diagramme de classe et de diagramme de séquence.

2.2 Présentation du langage UML



FIG. 2.1 : UML logo

FIG. 2.1 : Le Langage de Modélisation Unifié(UML), est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet.

UML est destiné à faciliter la conception des documents nécessaires au développement d'un logiciel orienté objet, comme standard de modélisation de l'architecture logicielle.

- Activité d'un objet/logiciel
- Acteurs
- Processus
- Schéma de base de données
- Composants logiciels
- Réutilisation de composants

2.3 Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation 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. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs, et ils apparaissent dans les cas d'utilisation.

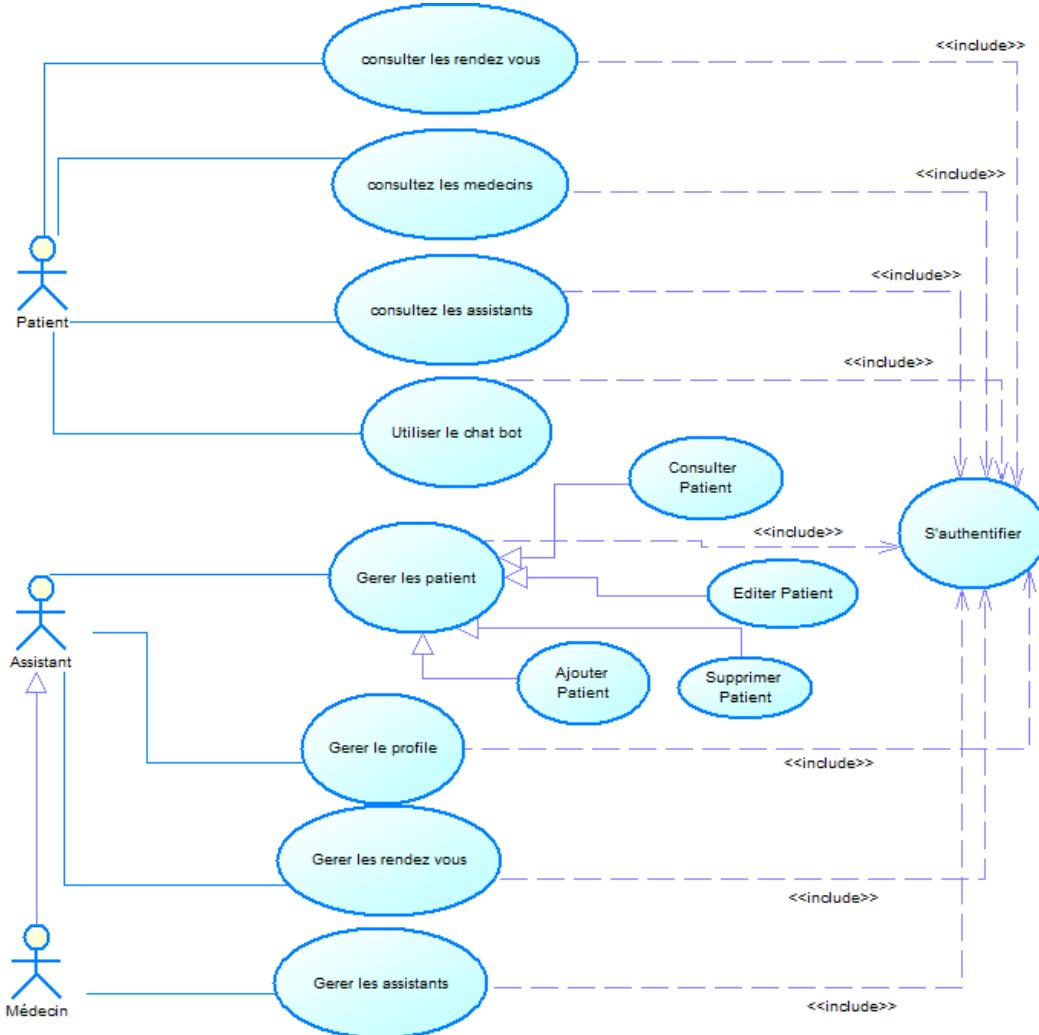


FIG. 2.2 : Diagramme de cas d'utilisation

FIG. 2.2 : Le diagramme de cas d'utilisation présenté ci-dessus illustre les interactions entre les différents acteurs et le système.

Ce diagramme est un outil essentiel pour comprendre les fonctionnalités principales du système et les interactions attendues avec les utilisateurs.

Les acteurs identifiés incluent les utilisateurs, et les administrateurs, chacun ayant des rôles et des permissions distinctes au sein du système.

Grâce à ce diagramme, il est possible de visualiser clairement les cas d'utilisation qui définissent les scénarios dans lesquels chaque acteur interagit avec le système.

Il permet également de repérer les points de connexion entre les acteurs et les différentes fonctionnalités, facilitant ainsi la détection de potentielles lacunes ou améliorations à apporter au système.

Enfin, le diagramme de cas d'utilisation sert de base pour la conception détaillée et le développement du système, assurant que toutes les exigences des utilisateurs sont bien prises en compte.

2.3.1 Description textuelle du cas « S'inscrire »

Cas d'utilisation	S'inscrire
Acteurs	Patient
Résumé	Ce cas d'utilisation permet à un utilisateur de créer un nouveau compte dans le système fournissant ainsi les informations nécessaires pour accéder aux fonctionnalités de l'application.
Précondition	1- Le système est accessible. 2- Aucune session n'est ouverte. 3- L'utilisateur n'a pas encore de compte dans le système.
Postcondition	L'utilisateur est enregistré dans le système avec un nouveau compte et peut accéder aux fonctionnalités disponibles après l'inscription.
Scénario principal	1- L'utilisateur accède à l'interface d'inscription du système. 2- L'utilisateur passe toutes les étapes d'inscription avec succès. 3- Le système vérifie la validité des informations fournies par l'utilisateur. 4- Si les informations sont valides, le système enregistre le nouveau compte utilisateur et envoie une notification de réussite d'inscription. 5- L'utilisateur peut maintenant utiliser ses identifiants pour se connecter au système et accéder à ses fonctionnalités.
Scénario alternatif	E1- L'utilisateur fournit des informations incorrectes ou manquantes. E2- Le système affiche un message d'erreur indiquant les champs nécessitant une correction ou un complément.

TAB. 2.1 : Description textuelle du cas « S'inscrire »

2.3.2 Description textuelle du cas « Se connecter »

Cas d'utilisation	Se connecter
Acteurs	Assistant
Résumé	Ce cas d'utilisation permet à un utilisateur existant de s'authentifier dans le système à l'aide de ses identifiants (email et mot de passe) afin d'accéder à son espace personnel et aux fonctionnalités autorisées.
Précondition	1- L'utilisateur possède déjà un compte valide. 2- Le système est disponible. 3- Aucune session active n'est en cours pour cet utilisateur.
Postcondition	L'utilisateur est authentifié et redirigé vers son tableau de bord personnel. Une session est établie.
Scénario principal	1- L'utilisateur accède à la page de connexion. 2- L'utilisateur saisit son email et son mot de passe. 3- Le système vérifie les informations d'identification. 4- Si les identifiants sont valides, le système authentifie l'utilisateur et crée une session sécurisée. 5- L'utilisateur accède à son espace personnel.
Scénario alternatif	E1- L'utilisateur entre un identifiant ou un mot de passe incorrect. E2- Le système affiche un message d'erreur précisant que les informations saisies sont invalides. E3- L'utilisateur peut réessayer ou demander une réinitialisation du mot de passe.

TAB. 2.2 : Description textuelle du cas « Se connecter »

2.4 Diagramme de séquence « Login »

La description textuelle présente des inconvénients puisqu'il est difficile de montrer comment les traitements se succèdent. Il est donc recommandé de compléter la description textuelle par un ou plusieurs diagrammes dynamiques UML. Dans ce qui suit on va présenter les principaux diagrammes de séquences élaborés.

FIG. 2.3 : Les diagrammes de séquence sont utiles pour visualiser le déroulement des interactions entre les différents éléments d'un système et pour comprendre le flux de contrôle lors de l'exécution d'un processus ou d'une fonctionnalité spécifique.

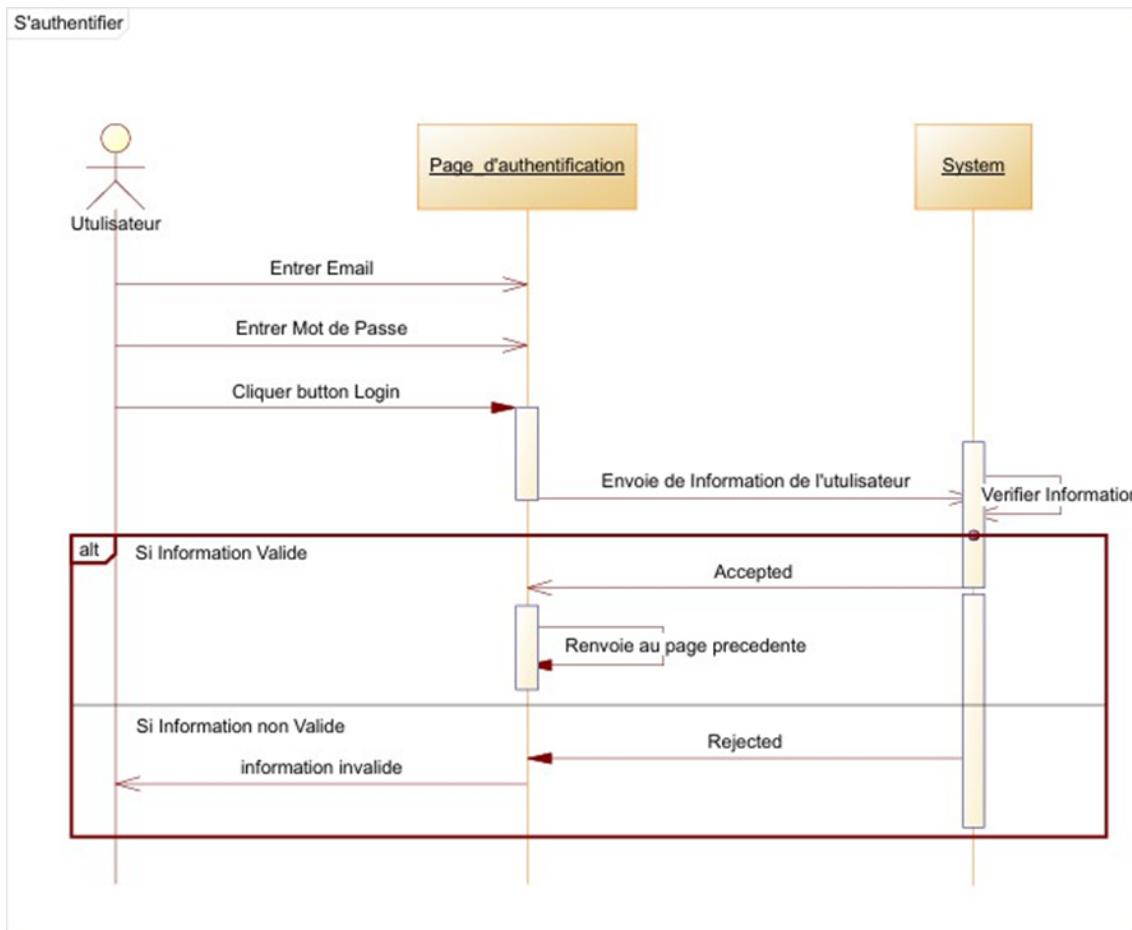


FIG. 2.3 : Diagramme de Séquence du cas d'utilisation « Login »

2.5 Diagramme de Classes

FIG. 2.4 : Les diagrammes de classes sont des outils de modélisation qui représentent la structure statique d'un système logiciel, montrant les classes, leurs attributs et leurs relations. Ils facilitent la compréhension et la communication des concepts de conception entre les membres de l'équipe.

Voici le diagramme de classe que nous avons élaboré :

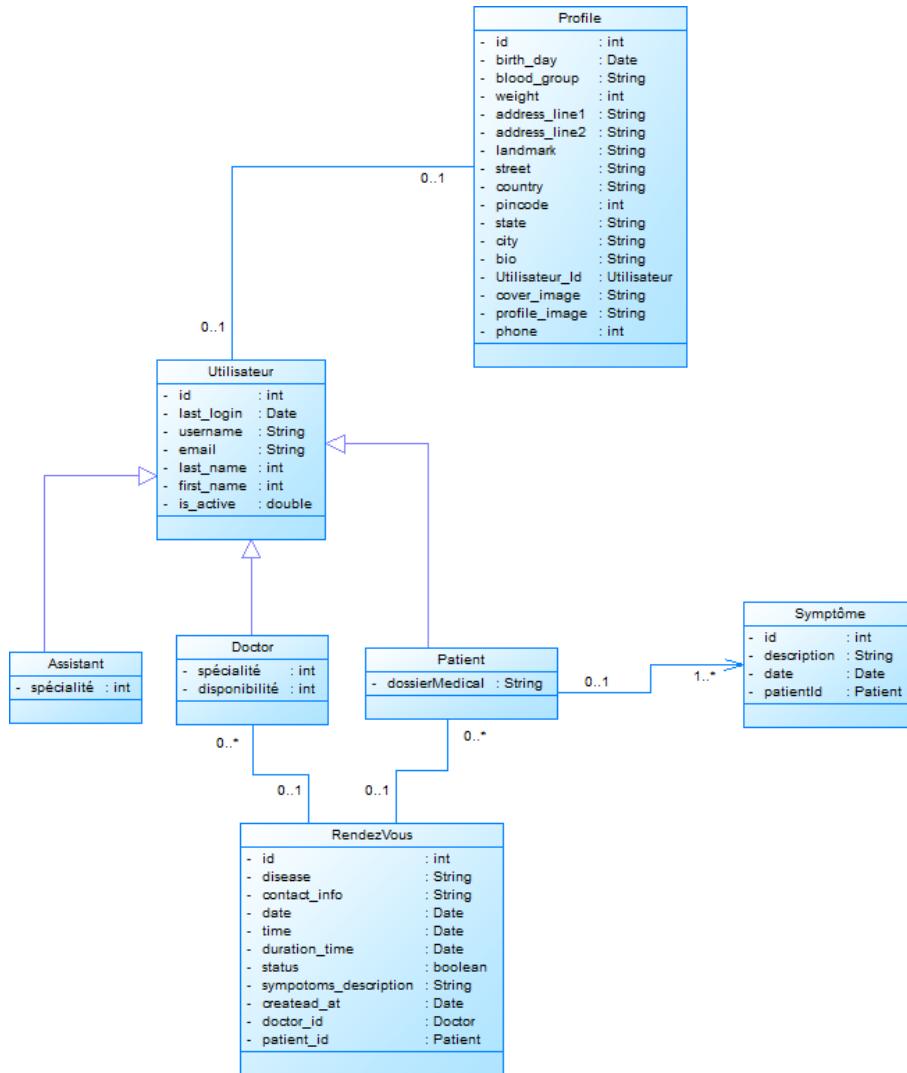


FIG. 2.4 : Diagramme de classe

Ce diagramme de classe montre clairement les relations entre les entités et les opérations que chaque classe peut effectuer, offrant ainsi une vue d'ensemble de l'architecture du système "MEDISENSE".

2.6 Dictionnaire de données

Classe	Description
Utilisateur	Informations de connexion : Contient les identifiants de base pour tous les types d'utilisateurs (patients, médecins, assistants), incluant <code>username</code> , <code>email</code> , <code>last_login</code> , et un indicateur d'activité. Héritage : Classe mère des entités Doctor, Assistant et Patient.
Doctor	Spécialité et disponibilité : Attributs propres aux médecins, incluant la spécialité et le niveau de disponibilité. Relations : Associé à plusieurs rendez-vous avec les patients.
Patient	Dossier médical : Contient un identifiant médical unique pour chaque patient. Relations : Peut avoir plusieurs symptômes et rendez-vous.
Assistant	Rôle de support : Possède un champ <code>spécialité</code> , indiquant le domaine de support médical. Relations : Hérite de Utilisateur, mais sans relations directes avec les patients ou rendez-vous.
Profile	Informations personnelles : Contient les détails personnels de l'utilisateur (date de naissance, poids, groupe sanguin, adresses, etc.). Association : Chaque profil est lié à un seul utilisateur.
Symptôme	Enregistrement des symptômes : Contient une description du symptôme, la date et le patient concerné. Relation : Un patient peut avoir plusieurs symptômes.
RendezVous	Gestion des consultations : Enregistre les informations sur les rendez-vous médicaux, incluant maladie, description des symptômes, date, durée, statut, etc. Relations : Lien direct avec un médecin et un patient.

TAB. 2.3 : Dictionnaire de données basé sur le diagramme de classes médical

2.7 Conclusion

Dans ce chapitre, j'ai procédé à l'identification et la présentation des diagrammes de cas d'utilisation et les diagrammes de séquence et de classes. Dans le prochain chapitre, je présenterai les contraintes techniques, et les différents outils que j'ai utilisés

Chapitre 3

Étude technique du projet

3.1 Introduction

Ce chapitre explore en détail les aspects techniques nécessaires à la mise en œuvre de la plateforme de gestion médicale intelligente **MEDISENSE**. Il aborde l'environnement de développement, l'architecture logicielle adoptée, ainsi que les outils, frameworks et technologies sélectionnés. Ces choix ont été guidés par les exigences du projet en matière de performance, de sécurité, de maintenabilité, d'accessibilité et de scalabilité.

3.2 Besoins techniques

Pour garantir une expérience fluide de développement et d'exécution, les éléments suivants sont requis :

Exigence Technique	Description
Sécurité des Données	Le système doit garantir la protection des données utilisateurs
Performance Serveur	Le serveur doit être capable de gérer un grand nombre de requêtes simultanées
Scalabilité	Le système doit pouvoir évoluer pour supporter une augmentation du nombre d'utilisateurs sans dégradation significative des performances.
Gestion des Erreurs	Le système doit être capable de gérer les erreurs et exceptions avec des messages d'erreur clairs et des mécanismes de journalisation pour le suivi et la résolution des problèmes.
Déploiement Continu	Le système doit être intégré dans un pipeline de déploiement continu (CI/CD) pour automatiser les tests, le build, et le déploiement des nouvelles versions.

TAB. 3.1 : Tableau des exigences techniques

3.3 Présentation de l'architecture du projet

L'architecture du projet repose sur le principe de séparation des responsabilités, inspiré du modèle MVC (Model – View – Controller), bien que Django adopte une structure propre qui se rapproche du modèle MTV (Model – Template – View). Cette structure permet une organisation claire du code et facilite la maintenabilité et l'évolutivité du projet.

3.3.1 Architecture logique

- **Model** : Représente les entités du domaine (Patient, Médecin, Assistant, Rendez-vous, etc.) sous forme de classes Python héritant de `django.db.models.Model`.

Chaque modèle correspond à une table dans la base de données.

- **View (logique)** : Gère la logique de traitement des requêtes. Ce sont des fonctions ou des classes Python qui reçoivent la requête HTTP, effectuent les traitements nécessaires (appel aux modèles, vérification des droits, etc.) et retournent une réponse (généralement HTML ou JSON).
- **Template** : Fichiers HTML (souvent avec le moteur de templates de Django) permettant de générer dynamiquement l'interface utilisateur. Ils reçoivent des données depuis les vues et les affichent à l'écran.
- **URL dispatcher** : Système de routage propre à Django qui associe chaque URL à une vue spécifique.
- **Middleware et Authentification** : Gèrent les aspects de sécurité, comme l'authentification JWT, la gestion des rôles, et les vérifications globales des requêtes entrantes.

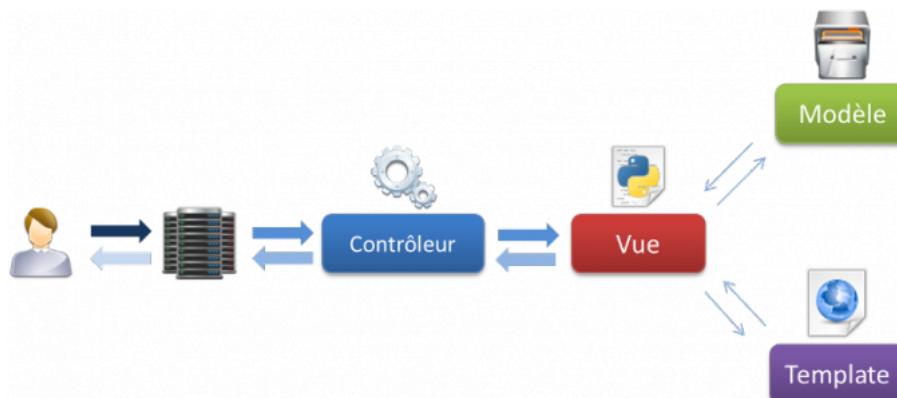


FIG. 3.1 : Architecture logique

3.3.2 Schéma de flux recommandé

Le schéma de flux recommandé illustre l'interaction entre les différentes couches de l'application Django et les processus métiers. Voici un exemple de flux pour l'inscription d'un utilisateur à un tournoi :

- **Étape 1 : Saisie des informations par l'utilisateur (Patient, Médecin, etc.)**
L'utilisateur remplit le formulaire d'inscription ou de prise de rendez-vous via l'interface web (Template). Les données sont envoyées au serveur via une requête HTTP POST.
- **Étape 2 : Traitement dans la vue (View)**
La vue **Django** correspondante (inscription, login, création de rendez-vous, etc.) reçoit les données, applique les règles de gestion métier (vérification de la disponibilité du médecin, format des données, etc.).

- **Étape 3 : Interaction avec les modèles (Model)**

Si les données sont valides, la vue interagit avec les **modèles Django** pour créer ou modifier les enregistrements liés : compte utilisateur, dossier médical, rendez-vous, etc.

- **Étape 4 : Réponse à l'utilisateur**

Après traitement, la vue redirige vers une page de confirmation ou renvoie un message contextuel (succès ou erreur) via le **template**.

- **Étape 5 : Accès personnalisé aux données**

Une fois connecté, l'utilisateur peut :

- Le **patient** accède à ses rendez-vous, dossiers et au chat IA.
- Le **médecin** visualise et gère ses patients, assistants, rendez-vous.
- L'**assistant** organise les rendez-vous et gère le suivi.
- L'**administrateur** supervise tous les comptes et configurations.

Ce flux repose sur l'architecture MTV propre à Django, garantissant une séparation des responsabilités claire entre présentation (Template), traitement (View), et persistance des données (Model).

3.4 Technologies et Langages Utilisés

3.4.1 Langages de Programmation



Python [1] est un langage de programmation polyvalent et très utilisé, apprécié pour sa simplicité et sa lisibilité. Il supporte plusieurs paradigmes de programmation, notamment la programmation impérative, orientée objet, et fonctionnelle. Python est un langage interprété, ce qui signifie qu'il est exécuté directement par l'interpréteur, facilitant ainsi le développement rapide et les tests en temps réel.

3.4.2 Frameworks



Django [2] est un framework web open-source conçu pour le langage Python, qui facilite le développement rapide d'applications web puissantes et sécurisées. Il suit le principe de conception "DRY" (Don't Repeat Yourself), ce qui permet d'éviter la duplication du code et d'améliorer l'efficacité du développement. Django fournit une structure cohérente pour la gestion des bases de données, l'authentification des utilisateurs, et la création de services RESTful, entre autres fonctionnalités.

3.4.3 Bases de Données



SQLite est un système de gestion de bases de données relationnelles léger, intégré et sans serveur, qui fonctionne directement avec des fichiers de base de données. Contrairement à d'autres systèmes de gestion de bases de données qui nécessitent un serveur dédié pour le traitement des données, SQLite s'exécute directement sur le fichier de la base de données, ce qui simplifie son déploiement et sa gestion.

3.4.4 Technologies Frontend



Bootstrap [3] est un système de gestion de bases de données distribué conçu pour être rapide, scalable et flexible. Il s'appuie sur une architecture décentralisée qui permet de gérer de grandes quantités de données réparties sur plusieurs noeuds, tout en maintenant une cohérence élevée. Grâce à sa capacité à gérer de manière efficace les transactions et les requêtes complexes, Gemini est particulièrement adapté aux environnements nécessitant une haute disponibilité, une faible latence et une gestion dynamique des données.

3.4.5 Développement et Tests d'API

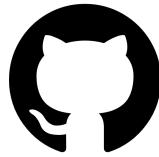


Postman est un outil de collaboration pour le développement d'API. Il offre une interface intuitive pour créer, tester, documenter et partager des APIs. Postman est largement utilisé pour tester les appels API de manière rapide et efficace.

3.4.6 Gestion de Version et Déploiement



Git [4] est un système de gestion de versions décentralisé, très populaire dans le développement logiciel. Il permet de suivre les modifications du code source, de collaborer efficacement entre développeurs et de gérer les différentes versions d'un projet de manière fiable et performante.



GitHub Classroom [5] est une plateforme proposée par GitHub, destinée à l'enseignement. Elle permet aux enseignants de créer, distribuer et gérer des devoirs de programmation, tout en intégrant directement les fonctionnalités de Git et GitHub pour suivre le travail des étudiants de manière centralisée.

3.4.7 APIs et Services



Gemini est une plateforme de gestion de projet et de collaboration dédiée aux équipes de développement. Elle fournit des outils pour le suivi des tâches, la gestion des versions, ainsi que la documentation technique et la gestion des exigences d'un projet. Gemini permet de structurer les projets en fonction de leurs besoins, offrant une interface intuitive et des outils de gestion de workflow qui facilitent la collaboration entre les membres d'une équipe. Grâce à son intégration avec d'autres outils comme Git, il permet un suivi fluide du code et des versions tout en restant flexible dans sa configuration pour répondre aux spécificités des projets.

3.4.8 Autres outils



XAMPP est un package de pile de solutions de serveur Web multiplateforme gratuit et open source développée par Apache Friends, composé principalement du serveur HTTP Apache, de la base de données Maria DB et d'interpréteurs pour les scripts écrits dans les langages de programmation PHP



Visual Studio Code est un éditeur de code source léger, open-source et multiplateforme, développé par Microsoft. Bien qu'il soit souvent comparé à des IDE plus lourds, VS Code se distingue par sa rapidité, sa flexibilité et sa capacité à être personnalisé grâce à une multitude d'extensions disponibles.

3.5 Conclusion

Ce chapitre a détaillé les choix techniques réalisés dans le cadre du développement de la plateforme Medisense, en particulier l'architecture MTV propre à Django, qui permet une séparation claire entre les différentes couches de l'application : Modèle, Template, et View.

L'architecture logique a facilité une gestion optimisée de la persistance des données avec des modèles Django bien définis. Le framework Django a permis d'assurer une gestion simplifiée de la logique métier, de l'authentification des utilisateurs et de la communication entre les différentes parties de l'application. La base de données SQLite a été choisie pour sa légèreté et sa facilité d'utilisation pendant les phases de développement.

Chapitre 4

Mise en œuvre

4.1 Introduction

Dans le cadre de la mise en œuvre de notre application Medisense, une attention particulière a été portée à la création et à la gestion des interfaces administratives. Ces interfaces sont essentielles pour garantir une gestion fluide et efficace des opérations de la plateforme.

4.2 Structure du projet

4.2.1 Structure de la base de données

FIG. 4.1 : Structure de la base de données

La FIG. 4.1 illustre la structure de la base de données du projet. Elle montre l'organisation des différentes tables et leurs relations, fournissant une vue d'ensemble de la manière dont les données sont stockées et interconnectées au sein du système. Cette structure est essentielle pour comprendre l'architecture de données et la gestion des informations dans l'application e-commerce.

4.2.2 Structure du projet Backend

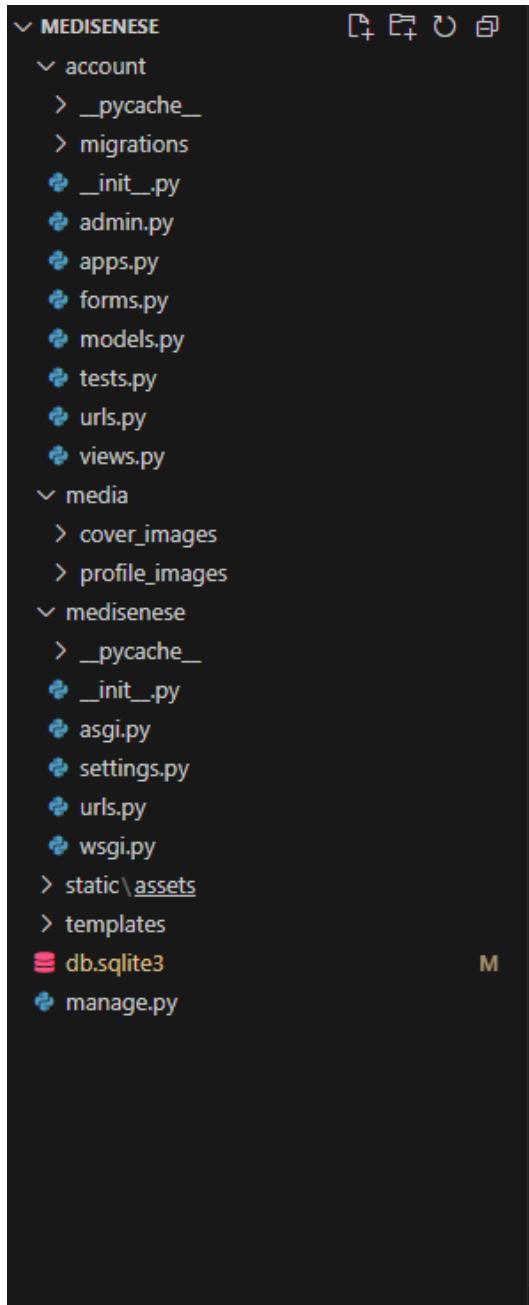


FIG. 4.2 : Structure du Backend

La FIG. 4.2 présente l’arborescence du projet backend Medisense. On y trouve une structure modulaire avec des dossiers pour différentes fonctionnalités telles que les views et les urls.

4.2.3 Structure du projet Frontend

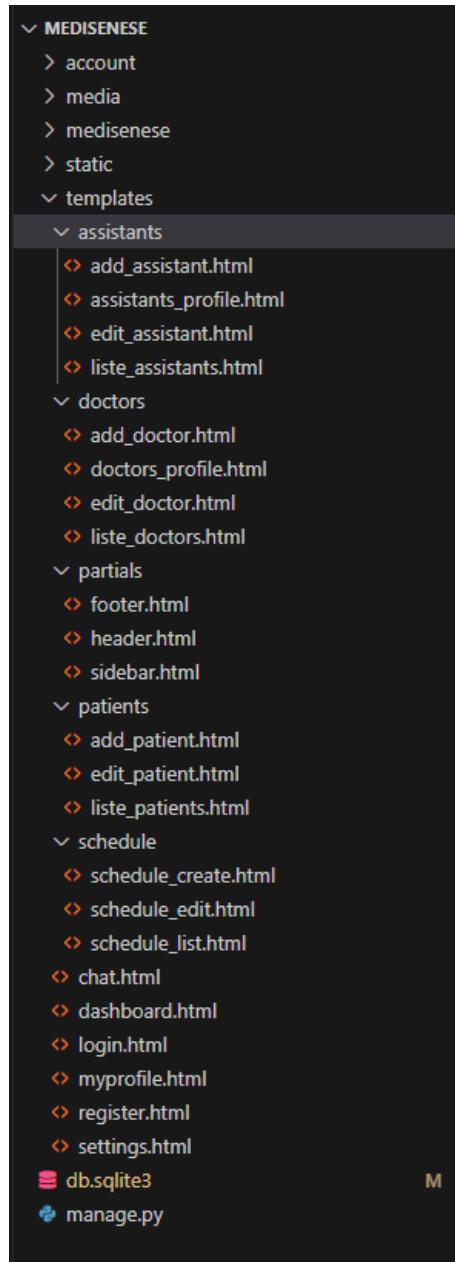


FIG. 4.3 : Structure du Frontend

La FIG. 4.3 illustre la structure des interfaces utilisateur de l’application eTournoiement Manager. Le dossier templates, situé sous resources, est organisé en deux modules principaux : admin et user. Le module admin contient des vues HTML permettant à l’administrateur de gérer les différentes entités du système, telles que les joueurs, les matchs, les équipes, les tournois et les utilisateurs. Chaque entité dispose de fichiers de formulaire et de consultation. Le module user, quant à lui, regroupe les interfaces destinées aux utilisateurs finaux, notamment la page de connexion, d’inscription et la page d’accueil utilisateur.

4.3 Interfaces de l'application

4.3.1 Interface de connexion

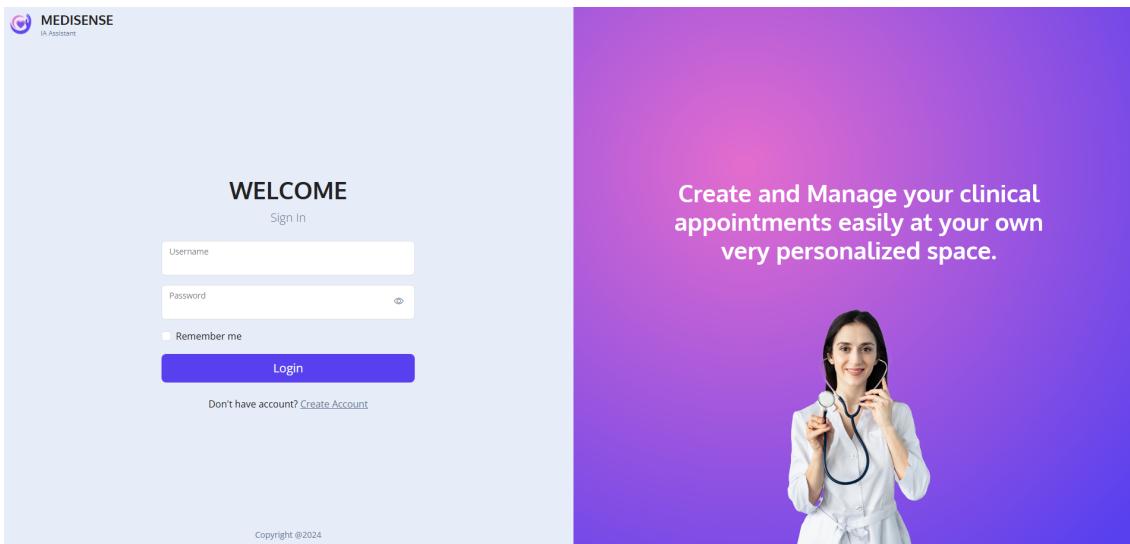


FIG. 4.4 : Page de connexion

La FIG. 4.4 présente l'interface de connexion de l'application Medisense. Cette page permet aux utilisateurs d'accéder à leur espace personnel en saisissant leurs identifiants.

4.3.2 Interface d'inscription

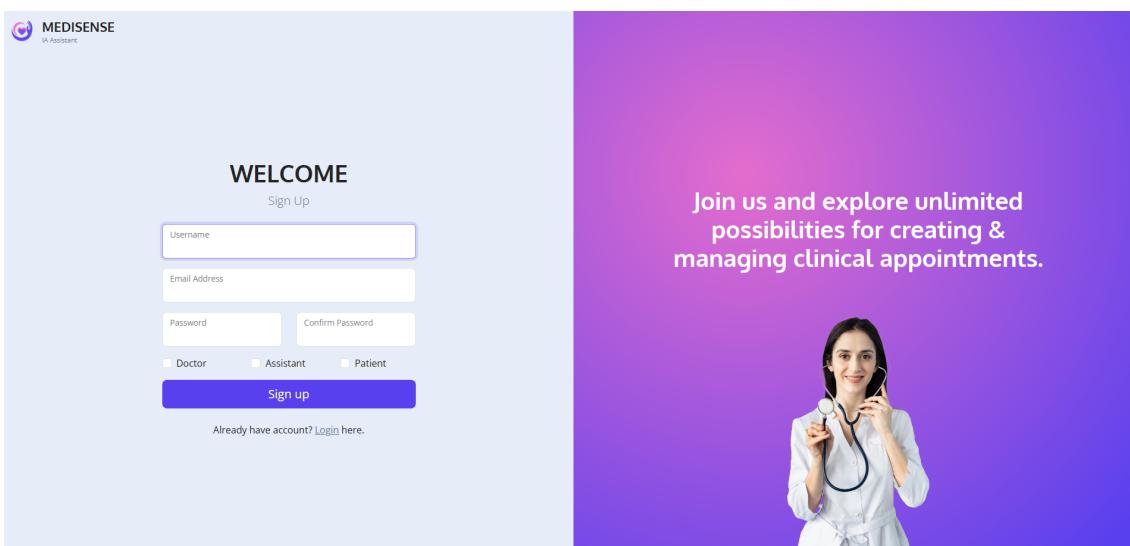


FIG. 4.5 : Page d'inscription

La FIG. 4.5 montre l'interface d'inscription de Medisense. Les utilisateurs peuvent s'inscrire en fournissant un nom d'utilisateur, une adresse email, un mot de passe, et en sélectionnant leur rôle (Docteur, Assistant ou Patient). Un lien permet également de revenir à la page de connexion.

4.3.3 Interface du tableau de bord

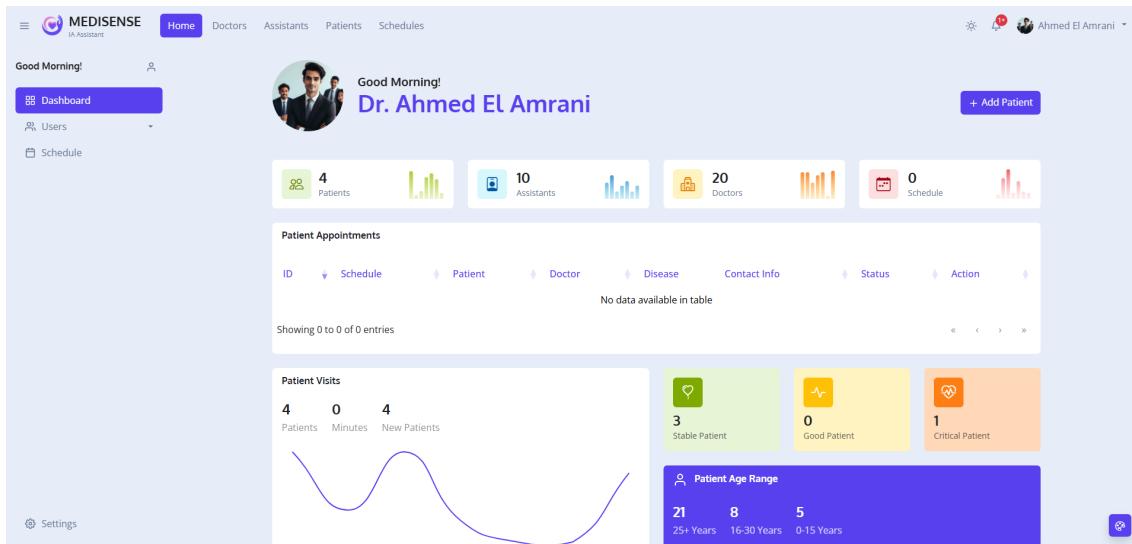


FIG. 4.6 : Tableau de bord médical

La FIG. 4.6 présente l’interface principale du tableau de bord de l’application médicale, laquelle est structurée en plusieurs sections. On y trouve une section de navigation rapide permettant d'accéder aux utilisateurs et à l'emploi du temps, ainsi qu'un tableau des rendez-vous. L'interface inclut également des statistiques sur les visites des patients ainsi qu'une répartition de ceux-ci par tranche d'âge.

4.3.4 Interface de gestion des médecins

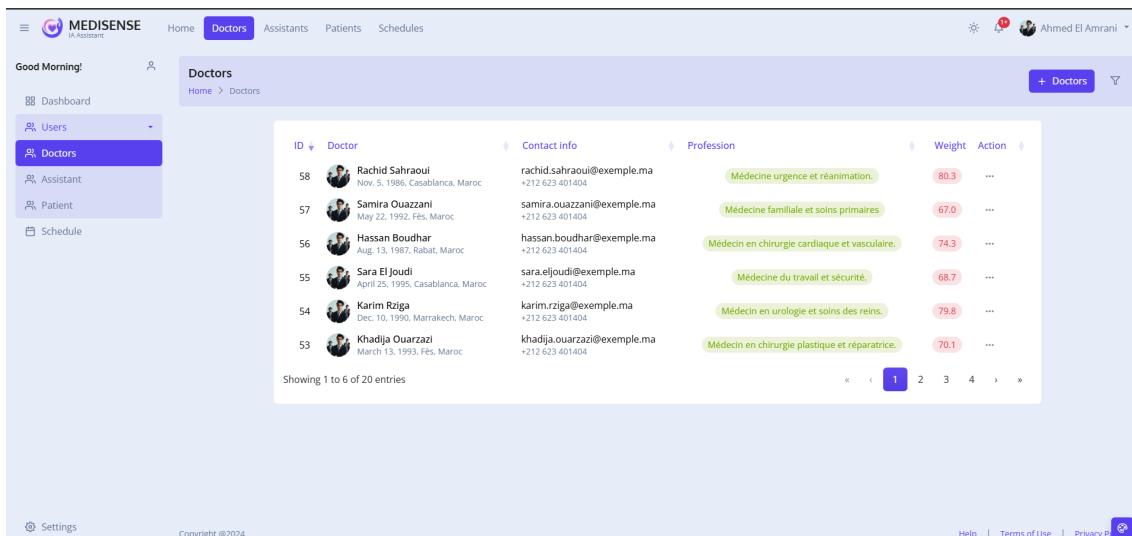


FIG. 4.7 : Liste des médecins

La FIG. 4.9 présente l’interface de gestion des médecins dans Medisense. Elle affiche une liste de médecins avec leurs informations personnelles, coordonnées et spécialités médicales. Un système de pagination permet de naviguer entre les entrées.

4.3.5 Interface d'ajout de médecin

The screenshot shows the 'Add Doctor' page of the Medisense application. The left sidebar shows navigation options like Dashboard, Users, Doctors, Assistants, Patients, and Schedules. The main area has a title 'Add Doctor' with a breadcrumb path: Home > Doctors > Add Doctor. It contains several input fields for doctor details, address information, and a professional bio section. On the right, there are sections for 'Doctor Photo' and 'Cover Photo' with upload buttons.

FIG. 4.8 : Formulaire d'ajout de médecin

La FIG. 4.10 présente le formulaire d'ajout d'un nouveau médecin dans Medisense. L'interface est divisée en trois sections principales : informations personnelles, coordonnées et adresse, ainsi qu'une zone pour la biographie professionnelle. Un système de champs organisés permet de saisir toutes les données nécessaires à l'enregistrement d'un nouveau praticien.

4.3.6 Interface de gestion des assistants

The screenshot shows the 'Assistants' list page of the Medisense application. The left sidebar shows navigation options like Dashboard, Users, Doctors, Assistant (which is selected and highlighted in purple), and Patient. The main area has a search bar and a dropdown for 'Blood Group' set to 'All'. Below is a table listing seven assistants with their details. The table columns are: ID, Assistant, Contact info, Profession, Weight, and Action. Each row shows the assistant's ID, name, email, birth date, contact info, profession, weight, and an 'Action' button. Pagination at the bottom indicates there are 6 entries.

FIG. 4.9 : Liste des médecins

La FIG. 4.9 présente l'interface de gestion des assistants dans Medisense. Elle affiche une liste assistants avec leurs informations personnelles, coordonnées et spécialités médicales. Un système de pagination permet de naviguer entre les entrées.

4.3.7 Interface d'ajout d'assistant

FIG. 4.10 : Formulaire d'ajout de médecin

La FIG. 4.10 présente le formulaire d'ajout d'un nouveau assistant dans Medisense. L'interface est divisée en trois sections principales : informations personnelles, coordonnées et adresse, ainsi qu'une zone pour la biographie professionnelle. Un système de champs organisés permet de saisir toutes les données nécessaires à l'enregistrement d'un nouveau praticien.

4.3.8 Interface de gestion des patients

ID	Patient	Contact Info	Blood Group	Weight	Action
18	Patient Emsi Feb. 12, 2002, Marrakech, Marrakech	patientems@gmail.com 0620020714	O	77.9	...
8	Samia Joudar Feb. 4, 1999, Morocco, Morocco	samiajoudar@gmail.com +216 623-422123	O	79.0	...
7	Oussama Benhida Aug. 9, 2001, Casablanca, Morocco	yassirbenjima29@gmail.com +212 624-423523	A	55.0	...
5	Mariam Laacher May 30, 2003, Marrakech, Morocco	Mariami@gmail.com +212 624-423522	O	45.0	...

Showing 1 to 4 of 4 entries

Track patient's statistic report easily
Take good care for your patient with data oriented approach to streamline medical facility

3 Stable Patient 0 Good Patient 1 Critical Patient

Patient Age Range

1 25+ Years 3 16-30 Years 0 0-15 Years

FIG. 4.11 : Liste des patients

La FIG. 4.11 présente l'interface de gestion des patients dans Medisense. Elle affiche une liste de 4 patients avec leurs informations médicales essentielles (groupe sanguin, poids) et coordonnées. La section inférieure propose des statistiques par tranche d'âge.

4.3.9 Interface d'ajout de patient

FIG. 4.12 : Formulaire d'ajout de patient

La FIG. 4.12 présente le formulaire d'enregistrement d'un nouveau patient dans Medisense. L'interface comprend deux sections principales : informations personnelles (nom, date de naissance, coordonnées) et données médicales (groupe sanguin, poids), ainsi qu'une partie dédiée aux informations d'adresse. Un bouton de soumission permet de valider l'enregistrement.

4.3.10 Interface des rendez-vous

ID	Schedule	Patient	Doctor	Disease	Contact Info	Status	Action
5	11:21 AM 11 May 2025	Mariam Laacher Marrakech, Morocco	Ahmed El Amrani Rabat, Maroc		+212 623-421421	pending	...

FIG. 4.13 : Liste des rendez-vous

La FIG. 4.13 présente l'interface de gestion des rendez-vous dans Medisense. Elle affiche un tableau contenant un rendez-vous programmé avec les détails complets : patient, médecin, maladie, informations de contact et statut. La pagination indique qu'il s'agit du premier rendez-vous sur un total d'un enregistrement.

4.3.11 Interface d'ajout de rendez-vous

The screenshot shows the 'Add Schedule' page of the Medisense application. At the top, there's a navigation bar with links for Home, Doctors, Assistants, Patients, and Schedules. A user profile for 'Ahmed El Amrani' is visible on the right. The main area is titled 'Schedule Details' and contains several input fields: 'Doctor' (dropdown), 'Patient' (dropdown), 'Statut' (dropdown), 'Contact Information' (text input), 'Disease' (text input), 'Date' (date picker), 'Heure' (time picker), and 'Durée (minutes)' (text input). Below these is a 'Description des symptômes' section with a WYSIWYG editor. At the bottom are 'Submit' and 'Cancel' buttons.

FIG. 4.14 : Formulaire d'ajout de rendez-vous

La FIG. 4.14 présente le formulaire de programmation d'un nouveau rendez-vous dans Medisense. L'interface permet de sélectionner un médecin et un patient, de spécifier la date, le statut, les symptômes et autres détails médicaux nécessaires. Un bouton de soumission finalise la prise de rendez-vous.

4.3.12 Interface des paramètres utilisateur

The screenshot shows the 'My Profile' settings page. The top navigation bar includes 'Home', 'Doctors', 'Assistants', 'Patients', and 'Schedules'. A user profile for 'Ahmed El Amrani' is at the top right. The main area is titled 'Basic Details' and contains fields for First Name ('Ahmed'), Last Name ('El Amrani'), Email Address ('ahmed.elamrani@example.ma'), Phone ('+212 635-683433'), Birth Date ('1985-04-12'), Weight ('75.5'), and Blood Group ('A'). Below this is an 'Address Details' section with fields for Address Line 1 ('123 Avenue Mohammed V') and Address Line 2 ('Quartier Hassan'). A 'Landmark' field ('près du jardin public') is also present. On the left, a 'Settings' button is visible. A large circular placeholder for a profile picture is in the center, with a 'Change Cover' button above it.

FIG. 4.15 : Paramètres du profil utilisateur

La FIG. 4.15 présente l'interface de gestion des paramètres personnels dans Medisense. Elle permet à l'utilisateur de modifier ses informations de base (nom, prénom, coordonnées), ses données médicales (groupe sanguin, poids) et ses informations d'adresse complètes. L'interface est organisée en sections claires pour une mise à jour aisée des informations personnelles.

4.4 Interfaces de l'application Patient

4.4.1 Interface de recherche de médecins

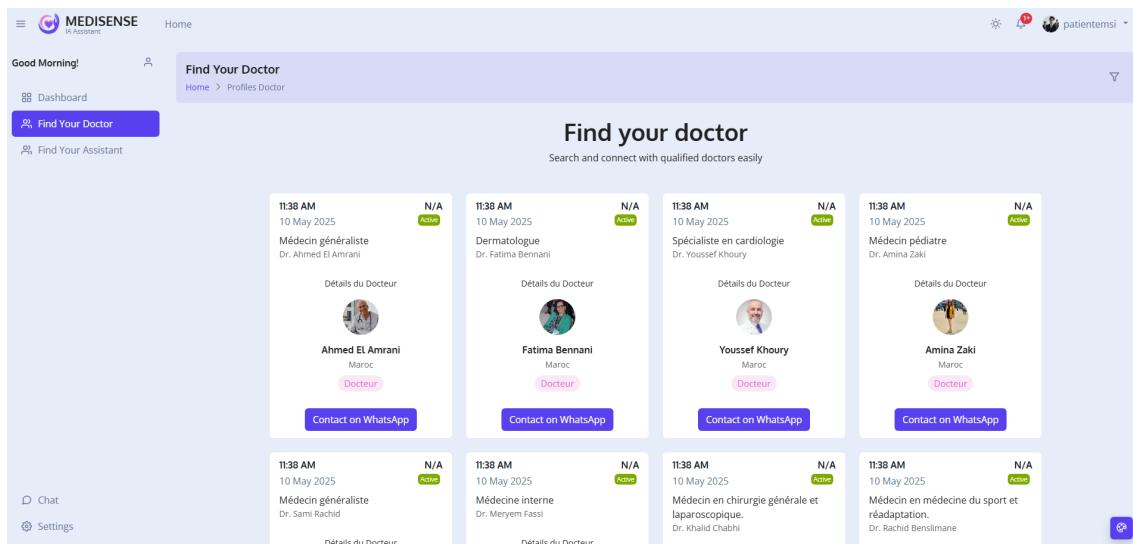


FIG. 4.16 : Recherche de médecins disponibles

La FIG. 4.16 présente l'interface de recherche de médecins dans Medisense. Elle affiche plusieurs profils médicaux avec leurs spécialités, noms, pays, date et heure, statut de disponibilité, ainsi qu'un bouton permettant de les contacter directement via WhatsApp.

4.4.2 Interface de recherche d'assistants

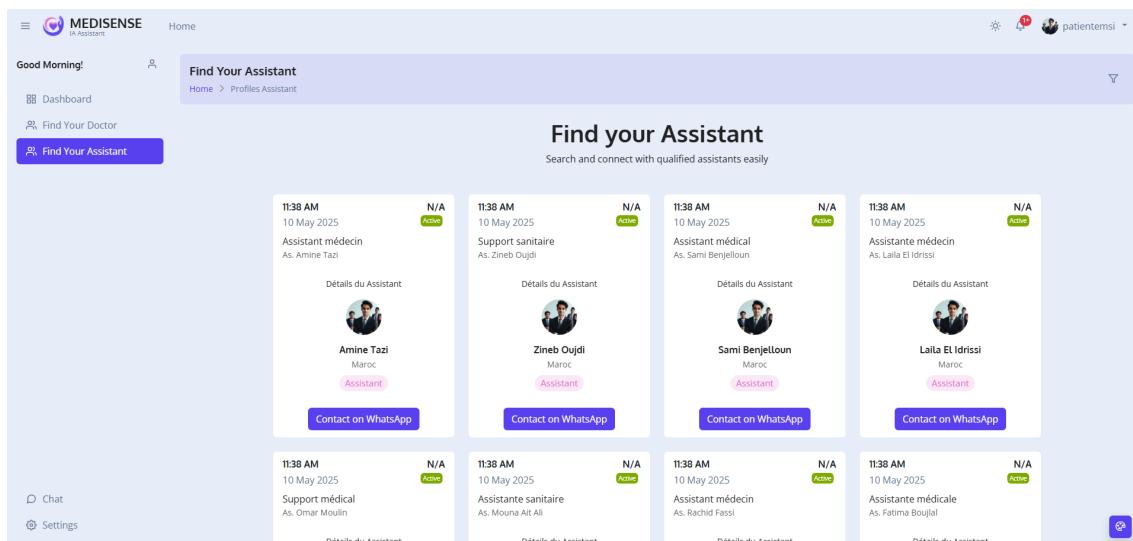


FIG. 4.17 : Recherche d'assistant disponibles

La FIG. 4.17 présente l'interface de recherche d'assistant dans Medisense. Elle affiche plusieurs profils avec leurs spécialités, noms, pays, date et heure, statut de disponibilité, ainsi qu'un bouton permettant de les contacter directement via WhatsApp.

4.4.3 Interface de chat médical

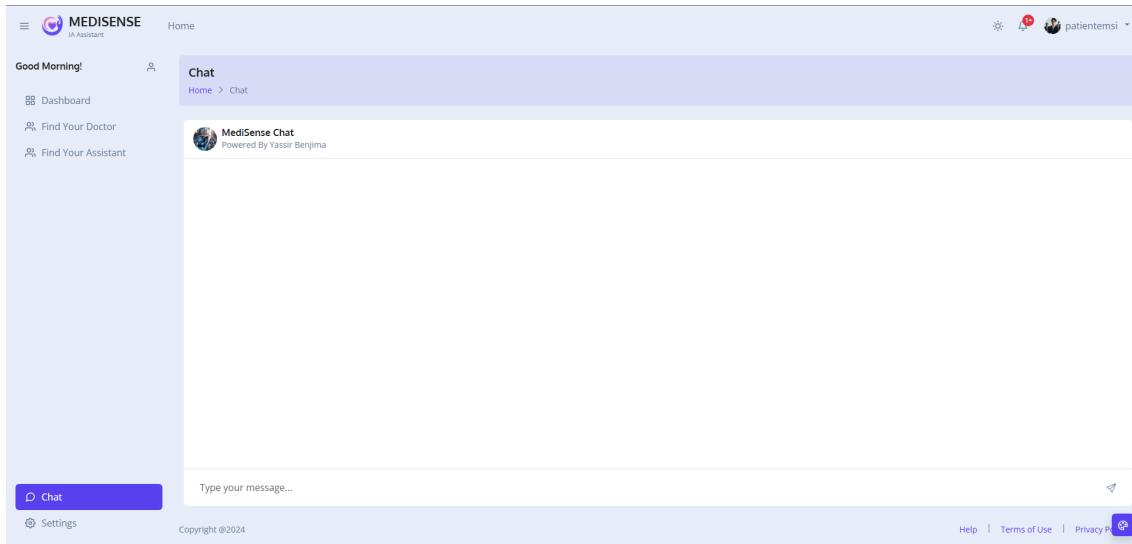


FIG. 4.18 : Interface de messagerie intelligente

La FIG. 4.18 présente l'interface de chat intelligent de Medisense.

4.5 Conclusion

Dans ce chapitre, nous avons présenté les différentes interfaces de l'application Medi-sense, en détaillant les fonctionnalités et les objectifs de chacune. Ces interfaces ont été conçues pour répondre aux besoins spécifiques des utilisateurs finaux, et des administrateurs.

Conclusion et perspectives

La plateforme Medisense constitue une solution numérique moderne et efficace pour la gestion des activités médicales, incluant la gestion des patients, des rendez-vous, des professionnels de santé (médecins et assistants), ainsi que l'intégration d'une assistance intelligente via un module de chat IA. Grâce à l'utilisation du framework Django et de l'architecture MTV, le projet bénéficie d'une structure claire, sécurisée, évolutive et adaptée à un usage web.

Cette première version de la plateforme a permis d'automatiser plusieurs processus auparavant manuels, tout en garantissant une interface utilisateur fluide, des fonctionnalités sécurisées (authentification par rôles) et un accès centralisé aux informations médicales essentielles.

Perspectives d'évolution :

- Intégration d'un système de téléconsultation en visioconférence.
- Mise en place de rappels automatiques de rendez-vous par SMS ou email.
- Extension du module de chat IA pour fournir des recommandations de santé préventives.
- Déploiement d'une application mobile patient/assistant/médecin (via Flutter ou React Native).
- Tableau de bord statistique pour l'analyse des données de santé et l'optimisation des ressources.
- Interconnexion avec des systèmes de dossiers médicaux électroniques (DME) existants.

Cette base solide ouvre ainsi la voie à un système de santé intelligent et connecté, au service d'une meilleure coordination des soins et d'un suivi médical amélioré.

Références

- [1] *Python 3 Documentation*. URL : <https://docs.python.org/3/> (visité le 01/05/2025).
- [2] *Django 4.x Documentation*. URL : <https://www.djangoproject.com/> (visité le 01/05/2025).
- [3] *Bootstrap : Documentation*. URL : <https://getbootstrap.com/> (visité le 04/05/2025).
- [4] *Git - Distributed version control system*. URL : <https://git-scm.com/> (visité le 02/05/2025).
- [5] *GithHub Classroom : Documentation*. URL : <https://classroom.github.com/> (visité le 02/05/2025).