جامعة محمد الأول بوجدة
UNIVERSITE MOHAMMED PREMIER OUJDA
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵎⵓⵃⴰⵎⵎⴷ ⴰⵏⴰⵎⵓⵔⵓ ⵏ ⵡⵓⵊⴷⴰ

---***---

École Supérieure de Technologie
OUJDA

Rapport de projet de Fin d'Études

Réalisation d'une application Web pour la gestion d'un hôpital

Réalisé par

Othmane BEKKAL

Naoufel GUENDOUZ

Mouad BOUBOUCH

Sous la direction de

M. TAHRICHI Mohamed

Année universitaire : 2022/2023

Remerciements

Avant tout développement sur notre projet de fin d'études, nous tenons à exprimer nos sincères remerciements à toutes les personnes qui ont contribué à sa réussite.

Nous souhaitons tout d'abord adresser nos remerciements les plus chaleureux à notre encadrant, M. TAHRICHI Mohamed, pour son accompagnement, son soutien et ses précieux conseils tout au long de ce projet. Sa disponibilité, son expertise et sa bienveillance ont été des atouts majeurs pour la réussite de ce travail.

Nous tenons également à remercier l'ensemble du corps professoral et administratif de l'ESTO pour leur enseignement de qualité, leur dévouement et leur motivation à nous accompagner dans ce projet. Leurs compétences, leurs encouragements et leur soutien ont été essentiels pour nous permettre de progresser et de réaliser ce projet dans les meilleures conditions.

Nous sommes conscients que ce travail n'aurait pas été possible sans l'implication, le soutien et la bienveillance de chacun d'entre vous. Merci encore pour cette expérience inoubliable et les enseignements précieux que nous en retirons.

Table des matières

Remerciements 2

Table des matières 3

Table des figures 5

INTRODUCTION GENERALE 6

CHAPITRE I : PRESENTATION DU PROJET 8

I. Introduction 9

II. État de l'art 9

A. Étude de l'existant 10

B. Limite de l'existant : 11

III. Les acteurs principaux 11

IV. Analyse du besoin 12

Besoin fonctionnel 12

Besoin non fonctionnel 13

V. Outil et langages utilisés 14

Langages de programmation 14

Environnement logiciel 15

CHAPITRE II : CONCEPTION TECHNIQUE 16

I. Introduction 17

II. ModélisationUML 17

a. Diagramme de cas d'utilisation 18

b. Diagramme de séquence 19

c. Diagramme de classe 24

Chapitre III : Réalisation et aperçus du projet : 26

I. Introduction 27

II. Interface graphique 27
III. Fonctionnalités de l'application 33
Bilan du projet 36
I. Les difficultés rencontrées 37
II. Compétences acquises 37
Conclusion 38
Bibliographie 40

Table des figures

FIGURE 1 DIAGRAMME DE CAS D'UTILISATION 18
FIGURE 2 DIAGRAMME DE SEQUENCE CREATION D'UN COMPTE PATIENT 20
FIGURE 3 DIAGRAMME DE SEQUENCE CREATION D'UN COMPTE MEDECIN 21
FIGURE 4 DIAGRAMME DE SEQUENCE CREATION D'UN COMPTE RECEPTIONNISTE 21
FIGURE 5 DIAGRAMME DE SEQUENCE CONNEXION A UN COMPTE 22
FIGURE 6 DIAGRAMME DE SEQUENCE DEMANDE DE RENDEZ-VOUS 23
FIGURE 7 DIAGRAMME DE CLASSE 24
FIGURE 8 PAGE D'ACCUEIL 27
FIGURE 9 PAGE DE CONNEXION 28
FIGURE 10 DASHBOARD RECEPTIONNISTE 29
FIGURE 11 DASHBOARD DOCTEUR 30
FIGURE 12 DASHBOARD CLIENT 30
FIGURE 13 PAGE DE CREATION D'UN COMPTE PATIENT 31
FIGURE 14 PAGE DE MISE A JOUR DU STATUT DU RENDEZ-VOUS 32
FIGURE 15 LA VUE APPOINTMENT_DETAIL 33
FIGURE 16 LA VUE RECHERCHE 34
FIGURE 17 LA VUE D'AUTHENTIFICATION POUR LES MEDECINS 35

INTRODUCTION GENERALE

Notre projet consiste à la réalisation d'une application web pour la gestion des rendez-vous et des rapports dans un établissement hospitalier. Ce projet a été proposé par notre encadrant, afin de répondre aux besoins d'un établissement de santé qui rencontre des difficultés dans la gestion des rendez-vous et des rapports.

Dans un hôpital, la gestion des rendez-vous est un enjeu majeur pour les patients, les médecins. En effet, la planification des rendez-vous peut avoir un impact important sur la qualité des soins, la satisfaction des patients et l'efficacité des ressources. Cependant, la gestion des rendez-vous peut être complexe et sujette à des erreurs humaines, notamment lorsqu'elle est effectuée manuellement.

Notre application web vise à résoudre ces problèmes en proposant une solution automatisée et centralisée pour la gestion des rendez-vous et des rapports médicaux. Elle permettra aux patients de prendre des rendez-vous en ligne, aux médecins de consulter leur agenda et de créer des rapports de consultation, et aux réceptionnistes de gérer les plannings et les informations des patients.

Nous avons choisi Django comme framework pour notre application web en raison de sa robustesse, de sa facilité de développement et de sa flexibilité. Nous avons également opté pour une approche basée sur l'UML pour la conception de notre application, afin de garantir une meilleure compréhension et une meilleure maintenabilité du code.

Notre projet vise donc à répondre aux besoins spécifiques d'un établissement de santé en proposant une solution innovante et efficace pour la gestion des rendez-vous et des rapports. Nous espérons ainsi contribuer à l'amélioration de la qualité des soins, à la satisfaction des patients et à l'optimisation des ressources.

CHAPITRE I :

PRESENTATION

DU PROJET

I. Introduction

Dans le cadre de notre projet de fin d'études, nous avons entrepris la conception et le développement d'une application web de gestion d'hôpital en utilisant le framework Django de Python. L'objectif principal de ce projet est de fournir une plateforme web complète permettant la gestion de toutes les activités d'un hôpital, tout en centralisant les informations dans une base de données unique.

Avant de commencer la phase d'analyse et de spécification des besoins, nous avons effectué une analyse de l'existant afin d'identifier les solutions existantes sur le marché et de déterminer les fonctionnalités et les services les plus pertinents à intégrer dans notre application. Cette analyse nous a permis de mieux comprendre les besoins des utilisateurs et les enjeux liés à la gestion d'un hôpital, ainsi que les éventuelles lacunes des solutions existantes.

Afin de répondre aux besoins des différents acteurs impliqués dans la gestion d'un hôpital, à savoir les médecins, les administrateurs et les réceptionnistes, il est primordial de réaliser une analyse détaillée des besoins fonctionnels et non fonctionnels de notre application. Cette étape de l'analyse et de la spécification des besoins est cruciale pour garantir la qualité et la performance de notre application.

II. État de l'art

L'étude de l'art, ou état de l'art, est une étape importante dans la réalisation d'un projet, car elle permet de faire un état des lieux des connaissances et des techniques existantes dans le domaine concerné. Dans le cadre de notre projet de fin d'études, qui consiste en la conception et le développement d'une application web de gestion d'hôpital, nous avons effectué une étude de l'art pour identifier les solutions existantes sur le marché, ainsi que les outils et les technologies les plus adaptés à notre projet.

Nous avons commencé notre étude de l'art en effectuant une recherche bibliographique, en consultant des articles scientifiques, des publications en ligne et des ouvrages de référence, afin de recueillir des informations sur les solutions de gestion d'hôpital existantes et les bonnes pratiques dans le domaine.

A. Étude de l'existant

Au cours de notre recherche, nous avons identifié plusieurs solutions de gestion d'hôpital existantes, telles que HospitalRun, OpenMRS et OpenEMR, qui proposent des fonctionnalités similaires à celles que nous souhaitons intégrer dans notre application. Nous avons étudié en détail chacune de ces solutions, en analysant les fonctionnalités proposées, l'interface utilisateur, l'architecture technique, la performance, la sécurité et la convivialité.

HospitalRun :

HospitalRun est une application de gestion hospitalière open-source qui permet aux professionnels de la santé de gérer les patients, les rendez-vous, les dossiers médicaux, les ordonnances, les factures et les rapports. L'application est disponible en plusieurs langues et peut être utilisée dans n'importe quel type d'hôpital ou de clinique, même avec une connectivité Internet limitée.

Fonctionnalités :

Gestion des patients : création de dossiers de patients, suivi des antécédents médicaux, enregistrement des allergies.
Gestion des rendez-vous : création, modification et annulation des rendez-vous, rappels de rendez-vous.
Gestion des ordonnances : création d'ordonnances, suivi des prescriptions, gestion des stocks de médicaments.
Gestion des factures : création de factures, suivi des paiements, etc.
Génération de rapports : rapports sur les patients, les rendez-vous, les factures.

OpenEMR :

OpenEMR est une application open-source de gestion de dossiers médicaux électroniques qui permet aux professionnels de la santé de gérer les patients, les rendez-vous, les dossiers médicaux, les ordonnances, les factures et les rapports. L'application est utilisée dans les hôpitaux, les cliniques et les cabinets médicaux du monde entier.

- **Fonctionnalités** :

Gestion des patients : création de dossiers de patients, suivi des antécédents médicaux, enregistrement des allergies.
Gestion des rendez-vous : création, modification et annulation des rendez-vous, rappels de rendez-vous.
Gestion des ordonnances : création d'ordonnances, suivi des prescriptions, gestion des stocks de médicaments.

B. Limite de l'existant :

HospitalRun :

HospitalRun est une application relativement nouvelle et ne dispose pas encore d'une communauté de développeurs très active.
L'interface utilisateur peut être améliorée pour une meilleure expérience utilisateur.

OpenEMR :

L'interface utilisateur est considérée comme très complexe pour les utilisateurs non techniques.

III. Les acteurs principaux

Dans cette partie, nous allons présenter les différents acteurs principaux du système de gestion d'hôpital en ligne que nous avons développé. Les acteurs sont des utilisateurs de l'application qui interagissent avec le système pour accomplir certaines tâches spécifiques. La compréhension de leurs rôles, de leurs besoins et de leurs attentes est cruciale pour concevoir un système qui répond à leurs attentes.
Notre application de gestion d'hôpital en ligne sera utilisée par trois types d'utilisateurs principaux :

- **Médecin :**

Le médecin est l'un des acteurs clés du système de gestion hospitalière. Il utilisera l'application pour consulter les dossiers médicaux des patients, prescrire des médicaments et des traitements, ainsi que pour accéder à toutes les informations médicales nécessaires à la prise en charge des patients. Le médecin doit être en mesure d'accéder rapidement et facilement aux informations des patients afin de pouvoir les traiter efficacement.

- **L'administrateur :**

Il s'agit d'un utilisateur qui est responsable de la gestion du système dans son ensemble. Il doit être en mesure de gérer tous les utilisateurs, patients, réceptionnistes et docteurs. Il est également en charge de la maintenance du système.

- **Le réceptionniste :**

Il s'agit d'un utilisateur qui est responsable de l'accueil des patients à leur arrivée à l'hôpital. Il doit être en mesure de prendre des rendez-vous, d'enregistrer les patients et de gérer les flux de patients dans l'hôpital.

Les patients sont également des acteurs importants de l'application. Ils peuvent interagir avec l'application pour prendre rendez-vous, accéder à leur dossier médical, vérifier les résultats des examens, recevoir des rappels de rendez-vous et communiquer avec leur médecin.

Cependant, il est important de noter que les patients sont des utilisateurs externes de l'application et n'ont pas les mêmes permissions et accès aux fonctionnalités que les acteurs principaux tels que le médecin, l'administrateur ou le réceptionniste.

IV. Analyse du besoin

Besoin fonctionnel

- Gestion des patients :
  - Enregistrement des informations de base des patients, telles que le nom, le prénom, la date de naissance, l'adresse, le numéro de téléphone et CIN.
  - Planification des rendez-vous avec les médecins.
  - Historique médical enregistré pour chaque patient.
  - Gestions des dossiers médicaux.

- Gestion des médecins :
  - Enregistrement des informations de base des médecins, telles que le nom, l'âge, l'adresse, le numéro de téléphone, CIN.
  - Suivi des rendez-vous des patients.
  - Emettre des rapports aux patients.
  - Historique des dernières visites.
  - Création rendez-vous pour patient.

- Réceptionniste :
  - Planification des rendez-vous pour les patients.
  - Gestion des rendez-vous (annuler, accepter, proposer).
  - Gestion des patients (ajout, visualiser)

- Administrateur :
  - Gestion des patients (ajout, modifications, suppression).
  - Gestion des docteurs (ajout, modifications, suppression).
  - Gestion des réceptionnistes (ajout, modifications, suppression).

Besoin non fonctionnel

- Performance : L'application doit être suffisamment rapide pour permettre une utilisation fluide et rapide par les utilisateurs.

- Sécurité : L'application doit être sécurisée pour garantir la confidentialité des informations des patients et éviter les accès non autorisés.

- Maintenabilité : L'application doit être facilement maintenable et évolutive pour pouvoir ajouter de nouvelles fonctionnalités ou corriger les bugs.

- Ergonomie : L'interface de l'application doit être ergonomique et facile à utiliser pour permettre une prise en main rapide par les utilisateurs.

V. Outil et langages utilisés

La réalisation d'un projet nécessite souvent l'utilisation de différents outils pour assurer une bonne gestion du développement et garantir la qualité du produit final. Dans le cadre de notre projet, nous avons utilisé plusieurs outils et technologies.

Langages de programmation

HTML :

L'HTML est un langage informatique utilisé sur l'internet. Ce langage est utilisé pour créer des pages web. (L'acronyme signifie HyperText Markup Langage, ce qui signifie en français "langage de balisage d'hypertexte). Cette signification porte bien son nom puisqu'effectivement ce langage permet de réaliser de l'hypertexte à base d'une structure de balisage.

CSS :

Le terme CSS (l'acronyme anglais de CascadingStyle Sheets qui peut se traduire par "feuilles de style en cascade"). Le CSS est un langage informatique utilisé sur l'internet pour mettre en forme les fichiers HTML ou XML. Ainsi, les feuilles de style, aussi appelé les fichiers CSS, comprennent du code qui permet de gérer le design d'une page en HTML.

JavaScript :

Un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web. Avec les langages HTML et CSS, JavaScript est au cœur des langages utilisés par les développeurs web. Une grande majorité des sites web l'utilisent, et la majorité des navigateurs web disposent d'un moteur javascript pour l'interpréter.

Python :

Python est le langage de programmation open source le plus employé par les informaticiens. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels. En effet, parmi ses qualités, Python permet notamment aux développeurs de se concentrer sur ce qu'ils font plutôt que sur la manière dont ils le font.

Environnement logiciel

Django :

Django est un framework web open source en Python. Il a pour but de rendre le développement d'applications web simple et basé sur la réutilisation de code. Développé en 2003 pour le journal local de Lawrence, Django a été publié sous licence BSD à partir de juillet 2005.

Visual Studio Code :

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégré.

StarUML:

StarUML est un outil de génie logiciel dédié à la modélisation UML et édité par la société coréenne MKLabs. Il est multiplateforme et fonctionne sous Windows, Linux et MacOS.

Figma :

Figma est un éditeur de graphiques vectoriels et un outil de prototypage. Il est principalement basé sur le web, avec des fonctionnalités hors ligne supplémentaires activées par des applications de bureau pour macOS et Windows.

Bootstrap :

Bootstrap est un puissant framework front-end utilisé pour créer des sites web et des applications web modernes. C'est un logiciel libre et gratuit, qui comporte de nombreux modèles HTML et CSS pour des éléments clés de l'interface utilisateur tels que les boutons et les formulaires.

CHAPITRE II:

CONCEPTION TECHNIQUE

I. Introduction

Le chapitre de conception technique est une étape cruciale dans le développement de tout projet informatique. Il permet de traduire les besoins fonctionnels et non fonctionnels définis précédemment en une architecture technique claire et précise. Cette architecture détermine la structure du système, les différentes composantes logicielles qui le composent, ainsi que les interactions entre celles-ci. Dans ce chapitre, nous allons présenter en détail la conception technique de notre application de gestion hospitalière en nous appuyant sur les besoins fonctionnels et non fonctionnels définis dans le chapitre précédent.

II. ModélisationUML

Nous avons choisi UML (Unified Modeling Language) pour la conception technique de notre application de gestion hospitalière.

Définition :

UML (Unified Modeling Language) est un langage de modélisation graphique standardisé utilisé pour la conception de systèmes logiciels. Il permet de représenter visuellement les différents aspects d'un système, comme sa structure, son comportement et ses interactions avec d'autres systèmes. UML fournit une notation standardisée pour représenter les différents éléments du système, tels que les classes, les objets, les interfaces, les composants, les relations entre les éléments, les activités, les cas d'utilisation, etc.

Pourquoi UML ?

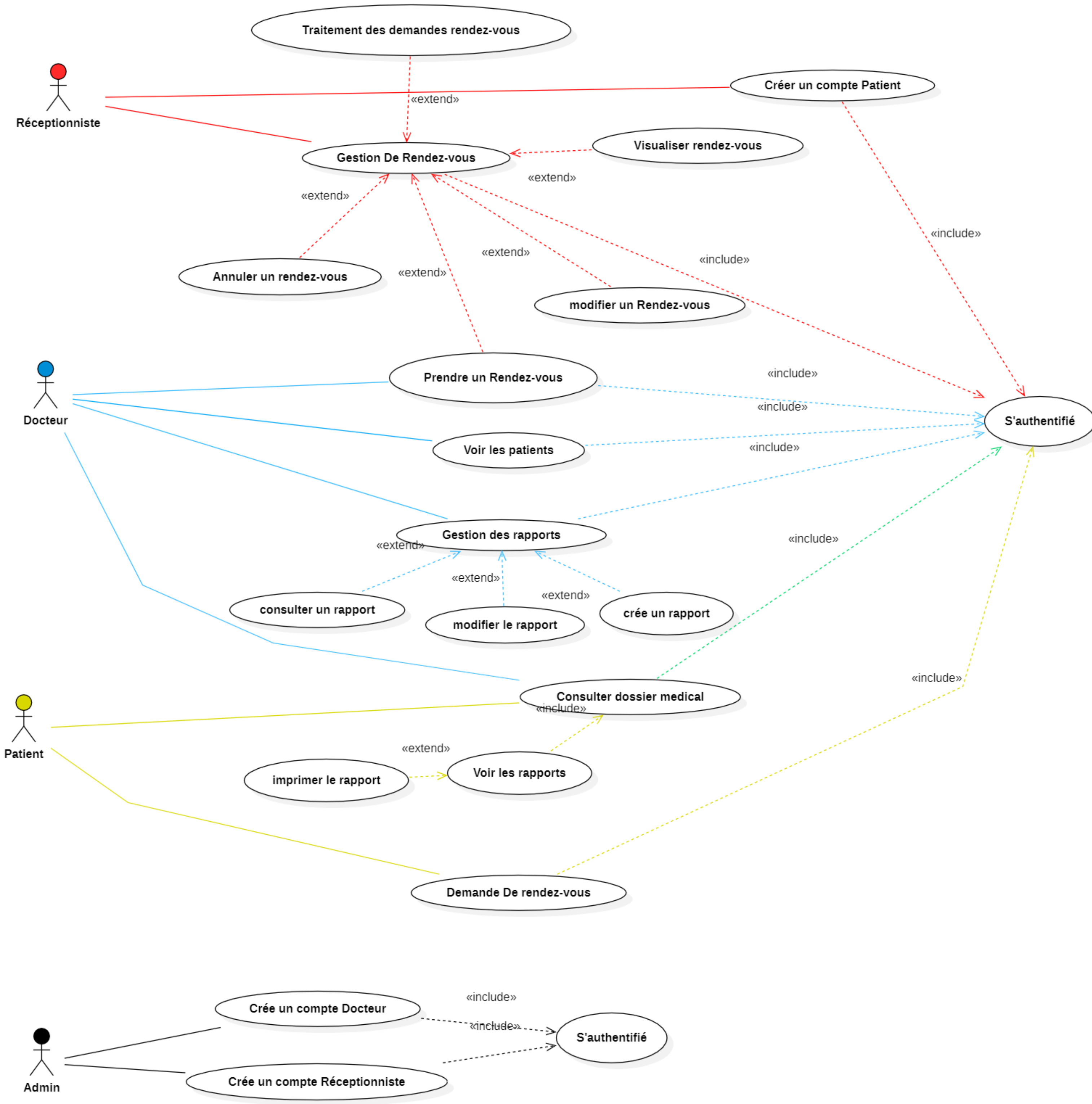
Ce choix s'explique en grande partie par le fait que notre projet est développé en POO (Programmation Orientée Objet). UML étant un langage de modélisation orienté objet, il nous a paru naturel de l'utiliser pour la conception de notre système. UML nous permet de modéliser les différents objets et leurs relations, les classes et leurs attributs, ainsi que les méthodes qui permettent aux objets de communiquer entre eux. Cela nous permet d'avoir une vue claire et précise de l'ensemble de notre système, et de mieux appréhender son fonctionnement.

De plus, UML offre une grande flexibilité et une adaptabilité à diverses méthodes de développement logiciel.

Dans notre projet étudié nous avons réalisé les diagrammes suivants :

a. 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.



Ce diagramme centre l'expression des exigences du système sur ses utilisateurs, et permet de structurer leurs besoins et les objectifs correspondants au système.

La description accompagnant ce diagramme est détaillée dans le tableau suivant :

1 Tableau Tableau de cas d'utilisation

Acteurs	**Cas d'utilisation**
Réceptionniste	· Créations de compte patient, gestion des rendez-vous le réceptionniste peut modifier, ajouter et annuler un rendez-vous.
Docteur	· Créations de rendez-vous, consulter et modifier les rapports, ajouter un rapport et accéder aux dossiers médicales des patients
Patient	· Peut consulter ses historiques de visites et prendre un rendez-vous
Admin	· Gère la création, modification, la suppression des comptes des réceptionnistes, des docteurs et des patients.

b. Diagramme de séquence

Un diagramme de séquence est un diagramme UML (Unified Modeling Language) qui représente la séquence de messages entre les objets au cours d'une interaction. Un diagramme de séquence comprend un groupe d'objets, représentés par des lignes de vie, et les messages que ces objets échangent lors de l'interaction.

Création d'un compte patient

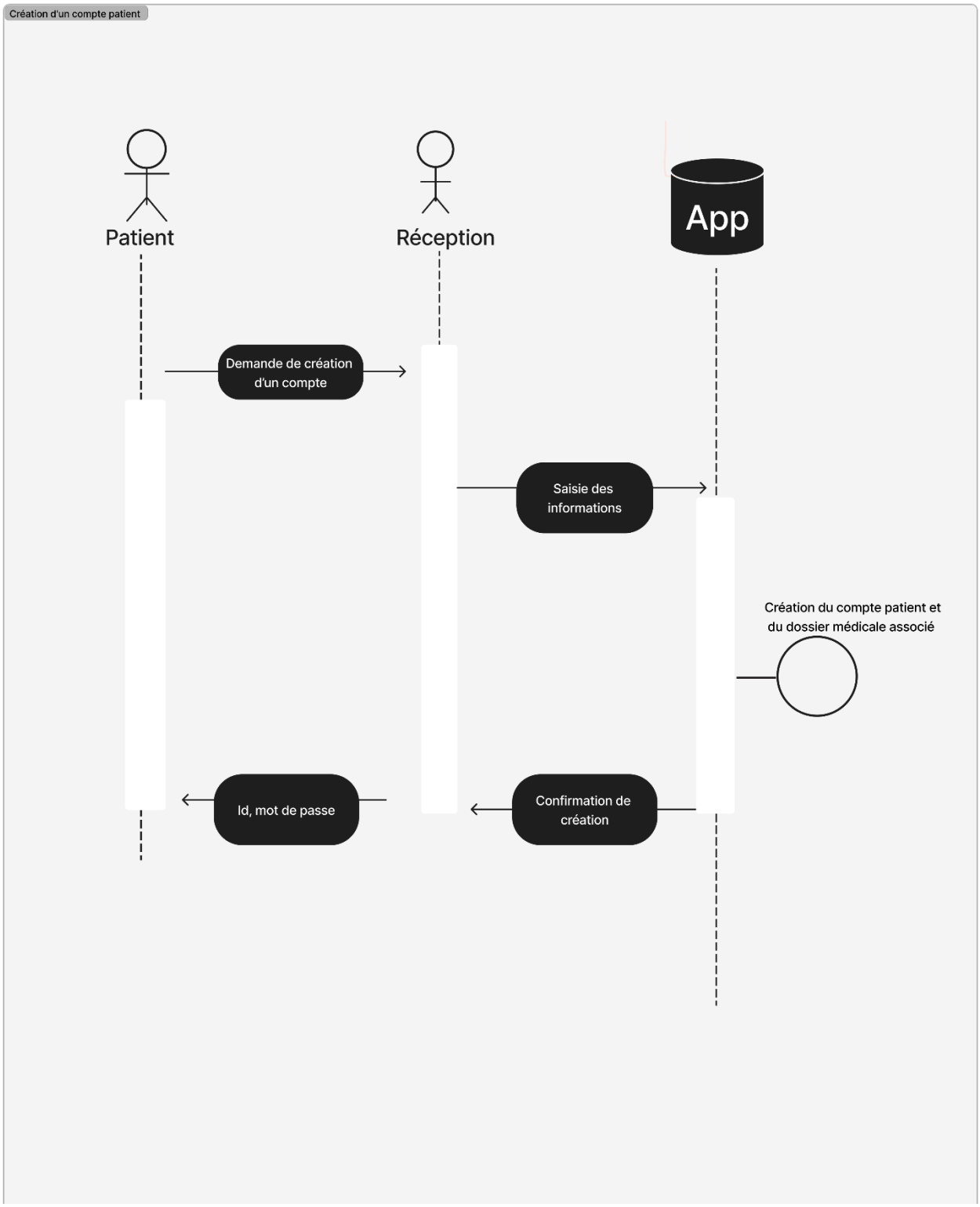


Figure 2 Diagramme de séquence Création d'un compte patient

Création d’un compte médecin

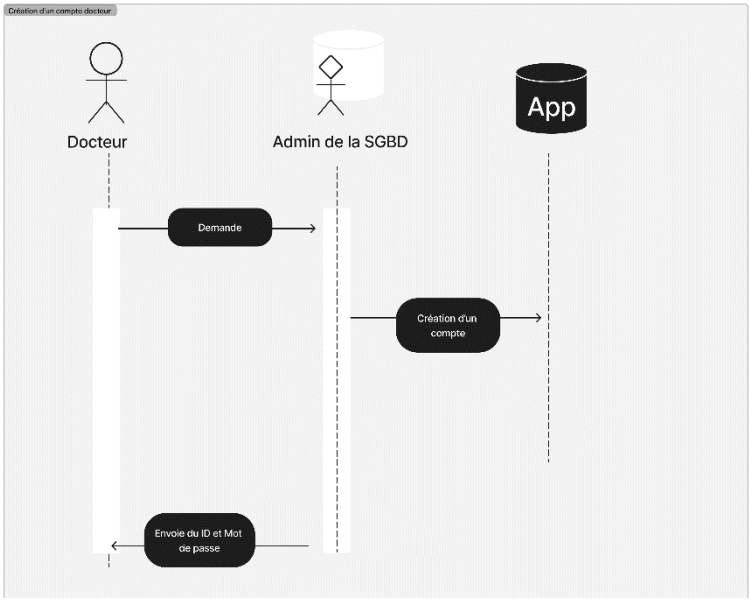


Figure 3 Diagramme de séquence Création d'un compte médecin

Création d’un compte réceptionniste

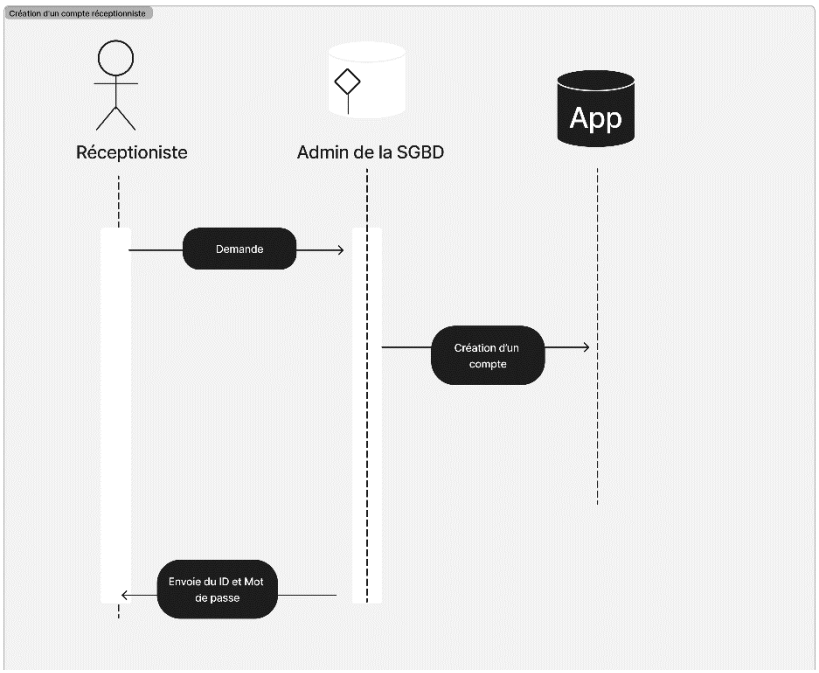


Figure 4 Diagramme de séquence Création d'un compte réceptionniste

Connexion à un compte

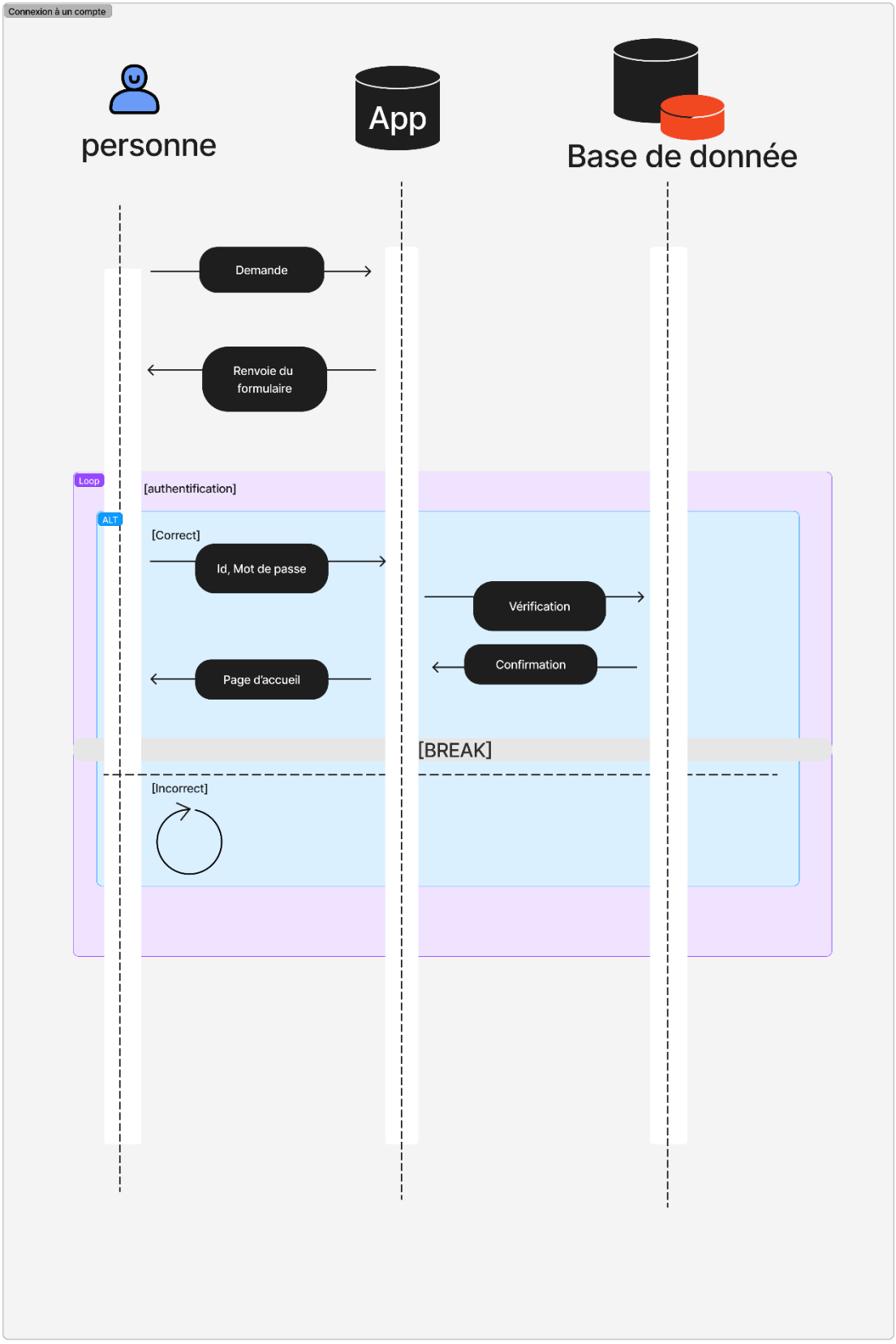


Figure 5 Diagramme de séquence Connexion à un compte

Demande de rendez-vous

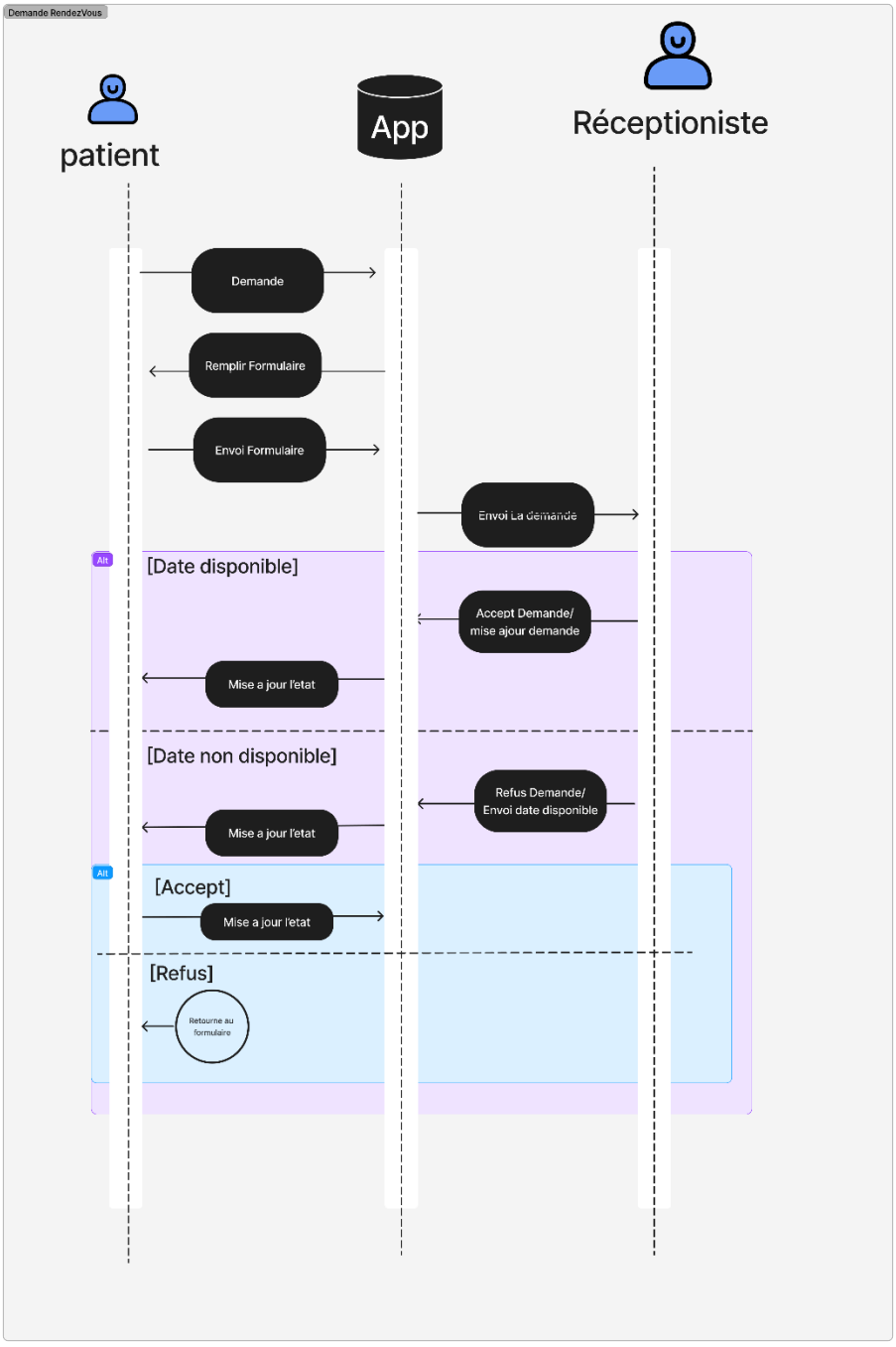


Figure 6 Diagramme de séquence demande de rendez-vous

c. Diagramme de classe

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.

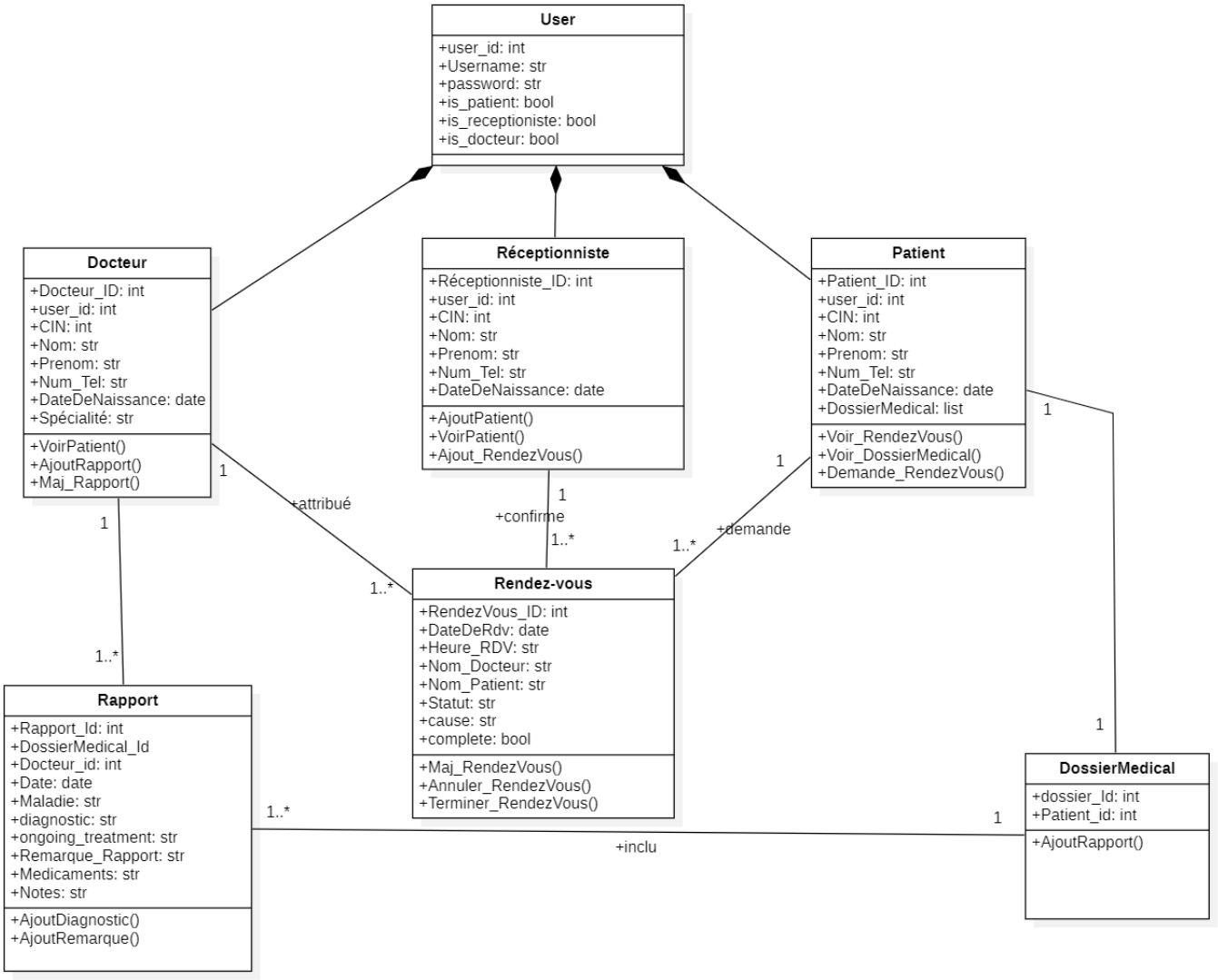


Figure 7 Diagramme de classe

Le diagramme de classe présenté dans le cadre de notre projet représente une relation de composition forte entre la classe User et les classes Patient, Docteur et Réceptionniste. Cette relation de dépendance forte signifie que la classe User est responsable de la création et de la destruction des instances de Patient, Docteur et Réceptionniste.

La classe User est étendue à partir de la classe AbstractUser et possède des attributs booléens pour indiquer si un utilisateur est un patient, un médecin ou un réceptionniste. Les classes Patient, Doctor et Receptionist sont toutes des sous-types de la classe User, avec des attributs spécifiques à chaque sous-type. Ces attributs spécifiques permettent de stocker les informations propres à chaque type d'utilisateur, telles que le nom, le prénom, la date de naissance, le numéro de téléphone, le numéro de carte d'identité nationale (CIN) et la spécialité pour les médecins.

Le diagramme de classe inclut également les classes Appointment, MedicalRecord et Report. La classe Appointment représente un rendez-vous entre un patient et un médecin, avec un statut qui indique si le rendez-vous est demandé, confirmé, annulé ou proposé. Cette classe contient également des informations sur la date et l'heure du rendez-vous, ainsi que sur la cause de l'annulation éventuelle. La classe MedicalRecord représente le dossier médical d'un patient et est associée à une instance de Patient à travers une relation OneToOneField. Cette classe contient des informations sur les antécédents médicaux du patient, les allergies, les médicaments prescrits et les traitements en cours. La classe Report représente un rapport médical créé par un médecin pour un patient, avec des champs pour les diagnostics, l'historique médical, les traitements en cours et les médicaments prescrits.

En outre, le diagramme de classe indique que les classes Patient, Doctor et Receptionist sont associées à une instance de MedicalRecord, ce qui permet aux médecins de consulter facilement les dossiers médicaux de leurs patients. De même, la classe Report est également lié à une instance de MedicalRecord, ce qui permet aux médecins de créer des rapports médicaux pour leurs patients.

Le diagramme de classe présenté est donc un outil essentiel pour comprendre la structure de l'application que nous avons développée. Il permet de visualiser les différentes classes et leurs relations, ainsi que les attributs et méthodes spécifiques à chaque classe. En outre, le diagramme de classe permet de comprendre comment les différentes classes interagissent pour fournir des fonctionnalités essentielles telles que la gestion des rendez-vous, des dossiers médicaux et des rapports médicaux.

Chapitre III :

Réalisation et aperçus du projet :

I. Introduction

Le chapitre de réalisation et aperçu du projet est l'un des chapitres les plus importants de ce rapport. En effet, ce chapitre va permettre de concrétiser tous les efforts de conception et de planification effectués dans les chapitres précédents en présentant une application web fonctionnelle pour la gestion d'un hôpital.

Ce chapitre va donc commencer par une présentation générale de l'application web développée, en mettant en évidence ses principales fonctionnalités et en expliquant comment elle répond aux besoins identifiés lors de la phase d'analyse. Ensuite, une vue d'ensemble de l'architecture technique sera présentée, en insistant sur les technologies utilisées pour le développement de l'application.

II. Interface graphique

Page d'accueil

Les différents acteurs et visiteurs externes de notre application Web accède à cette page d'accueil avant de se connecter à leurs comptes respectifs. Cette page présente des informations générales de l'hôpital.

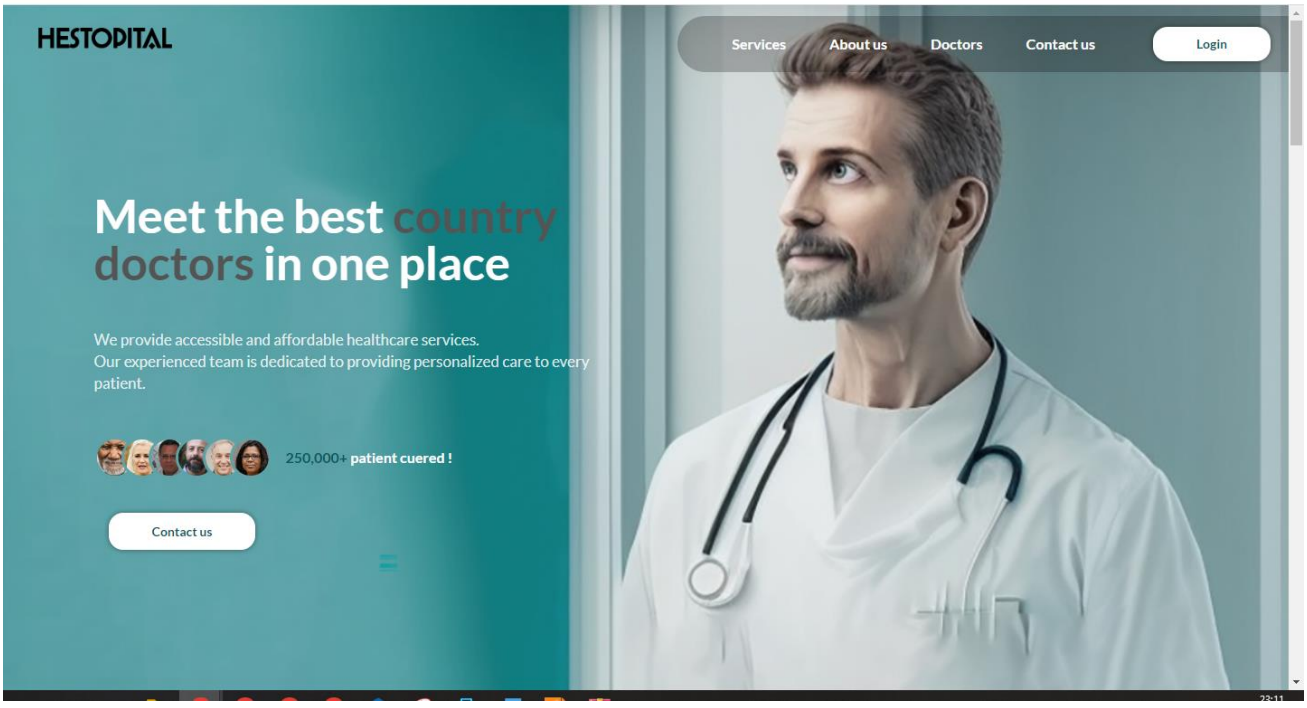


Figure 8 page d'accueil

Page de connexion :

L'utilisateur (réceptionniste, docteur, patient) se connecte avec son user Name et son mot de passe, si les données sont correctes il aura accès directement à son espace selon le type de compte.

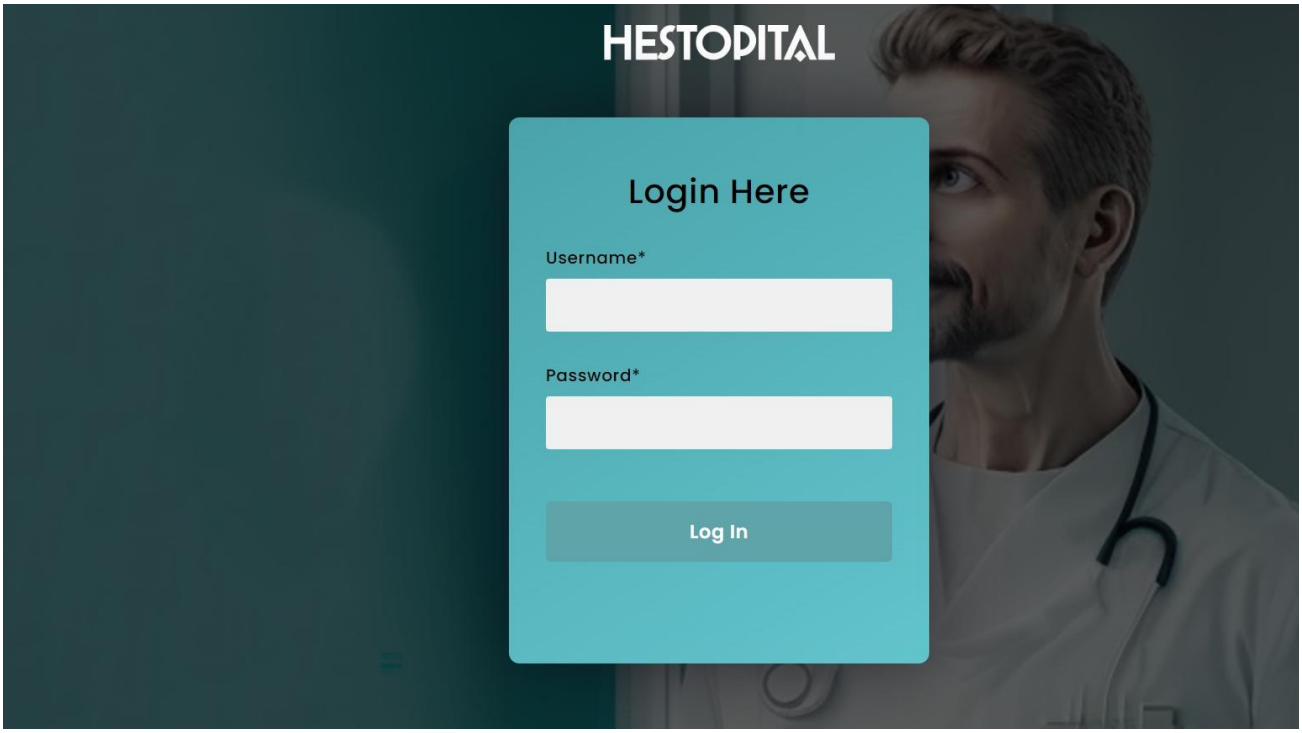


Figure 9 Page de connexion

Page d'accueil réceptionniste (Dashboard) :

Cette page représente la page d'accueil de la réceptionniste après sa connexion et en même temps permet de visualiser tous les patients, leur créer un rendez-vous et visualiser leur compte.

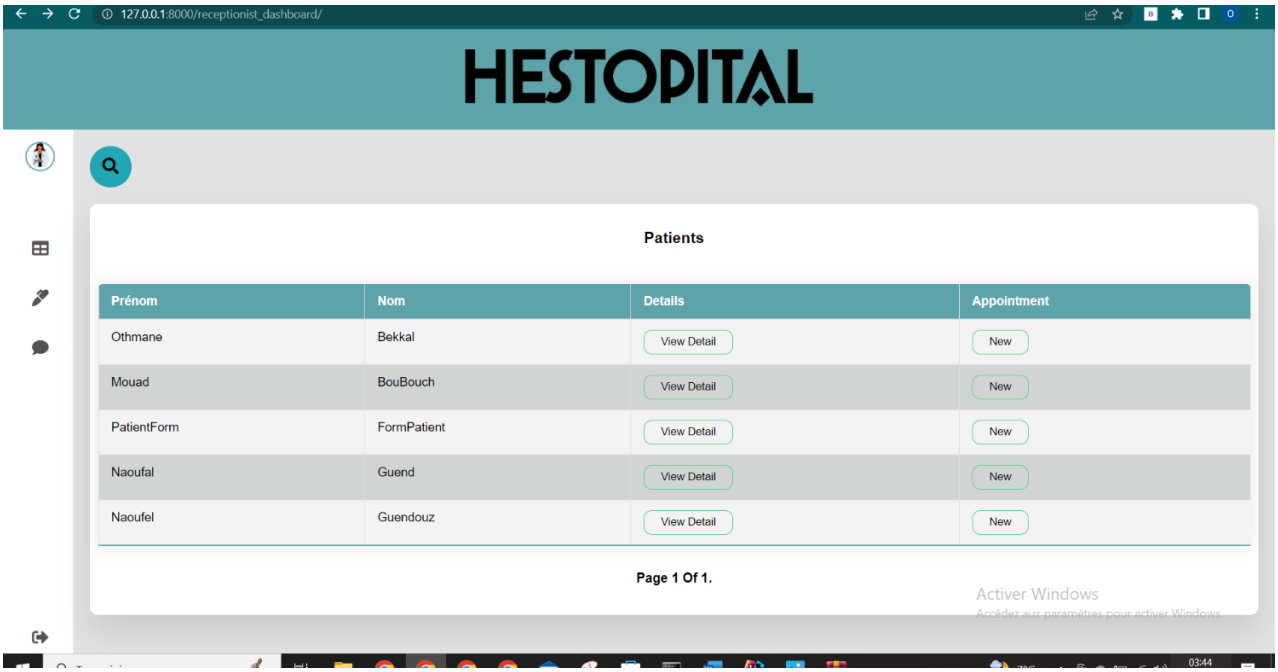


Figure 10 Dashboard Réceptionniste

Page d'accueil docteur (Dashboard) :

Cette page représente la page d'accueil du docteur après sa connexion et en même temps permet de visualiser tous les patients, leur créer un rapport et visualiser leur compte et leur dossier médicale.

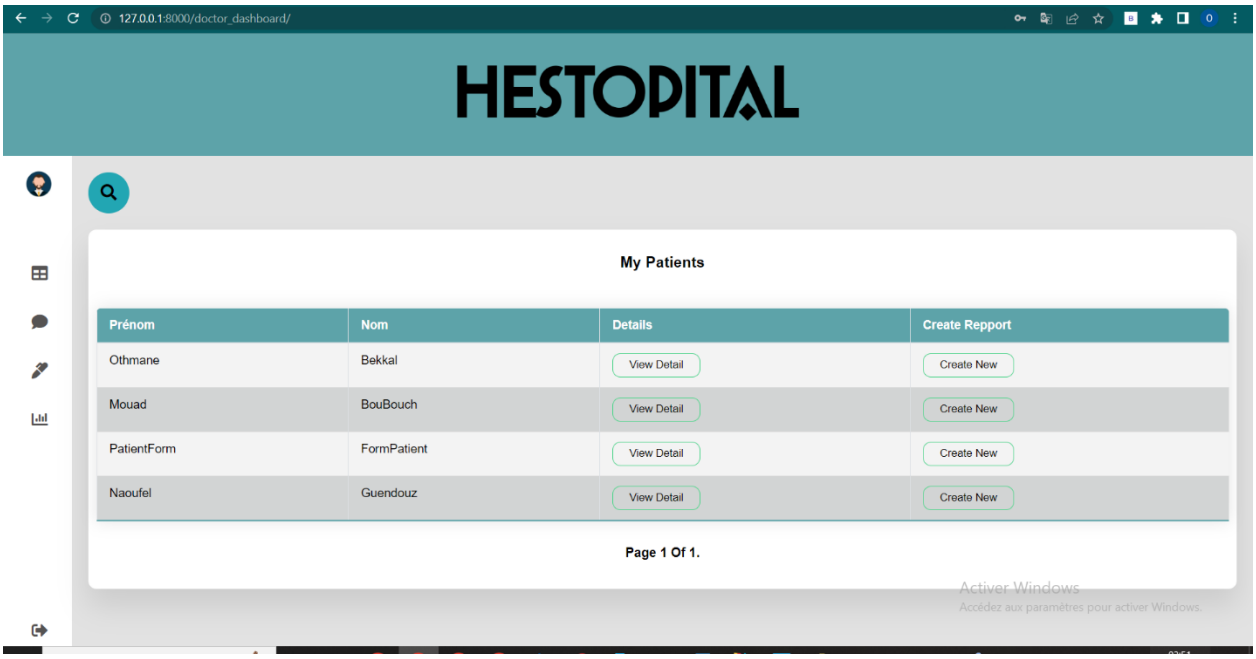


Figure 11 Dashboard docteur

Page d'accueil patient (Dashboard) :

Cette page représente la page d'accueil du patient après sa connexion et en même temps permet de visualiser son historique médicale.

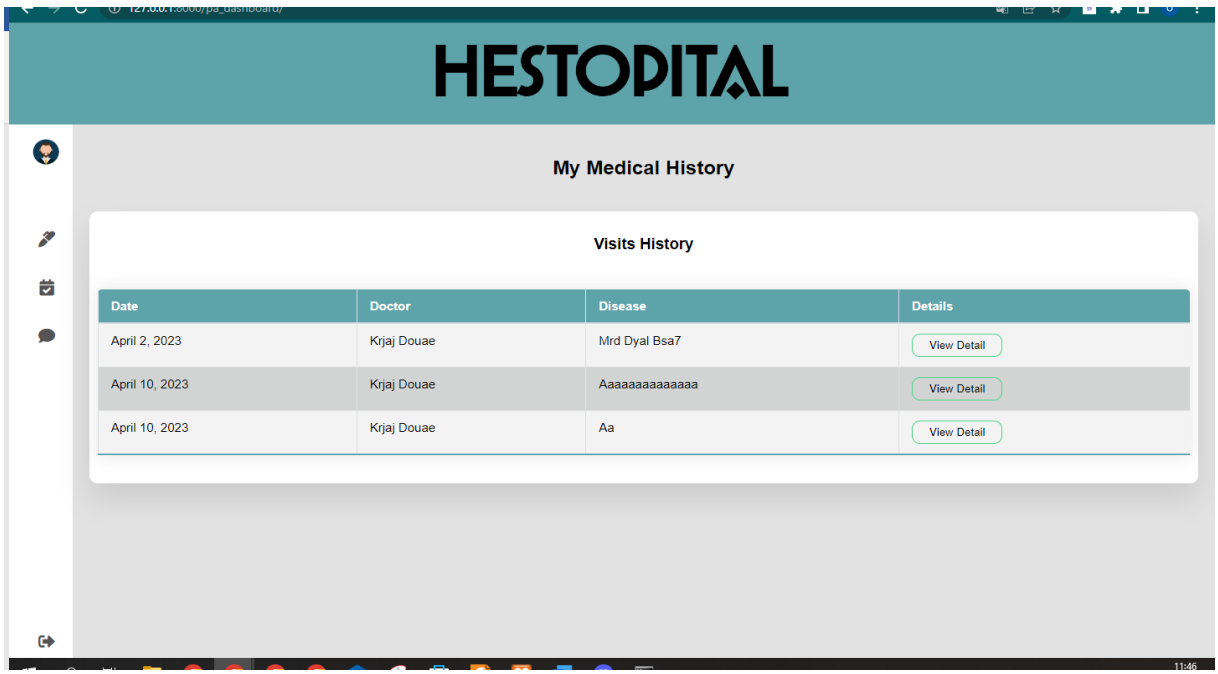


Figure 12 Dashboard client

Page de création d'un compte patient par réceptionniste :

Cette page représente la page de création d'un compte patient par un réceptionniste à travers la saisie de tous les champs du formulaire dans la page. Cette page contient des vérifications telles que la vérification des qui vérifie que le champs Password et Password confirmation doivent être similaire.

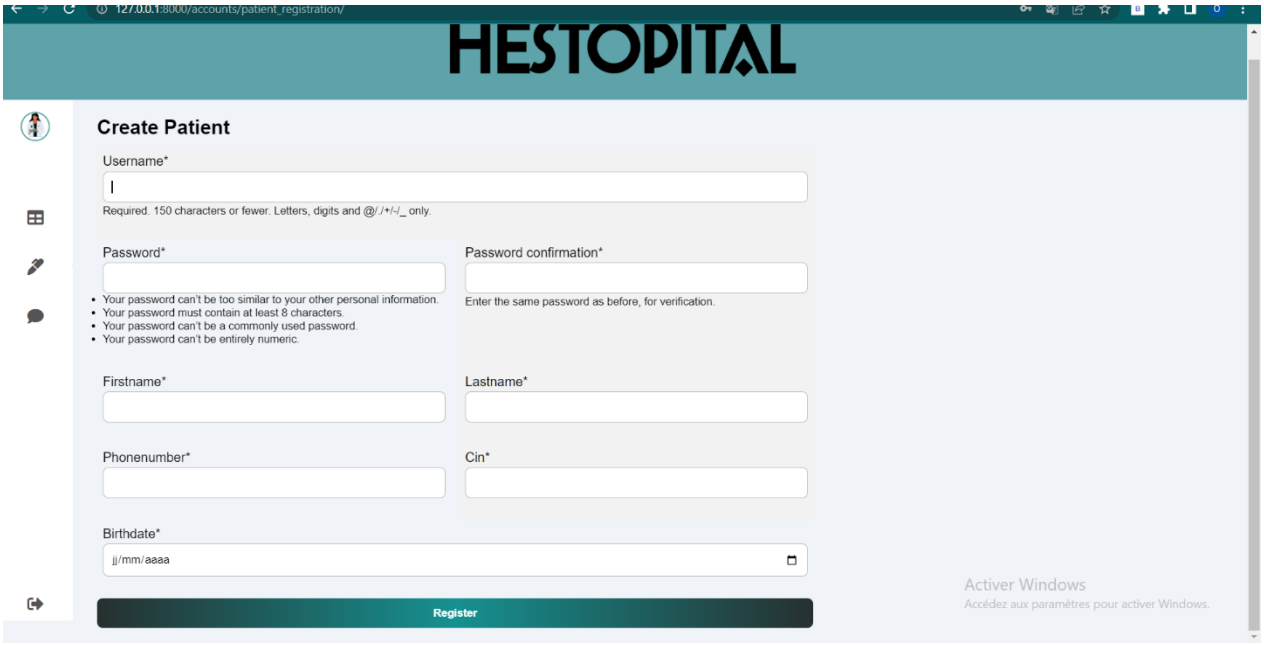


Figure 13 Page de création d'un compte patient

Page de mise à jour du statut des rendez-vous :

Cette page représente la page de mise à jour du statut des rendez-vous créer précédemment par un patient. La réceptionniste a aussi la possibilité de visualiser les rendez-vous accepter et demander ce jour-là avec le même docteur ou de visualiser ceux d'une autre date.

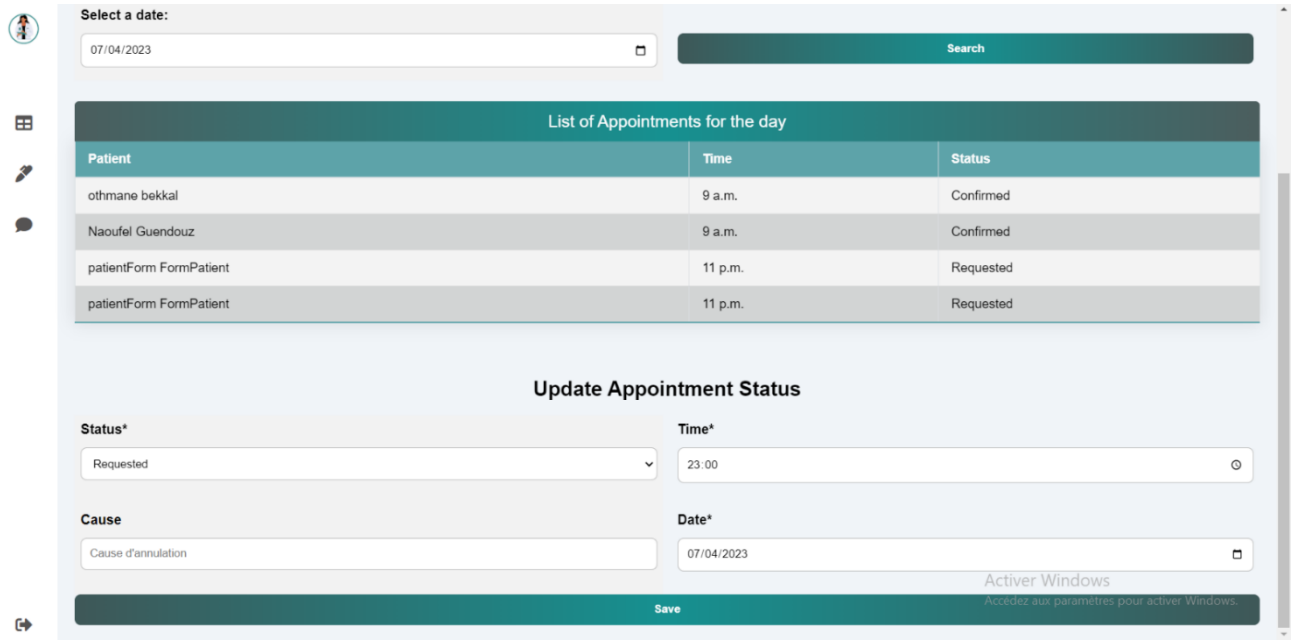


Figure 14 Page de mise à jour du statut du rendez-vous

III. Fonctionnalités de l'application

La vue appointment_detail :

La vue appointment_detail est utilisée pour afficher les détails d'un rendez-vous en particulier et pour permettre au réceptionniste de modifier l'état du rendez-vous. La vue prend en paramètre l'ID du rendez-vous (appointment_id).

La première étape de la vue consiste à récupérer le rendez-vous à partir de l'ID en utilisant la fonction get_object_or_404 de Django. Ensuite, la date sélectionnée est déterminée à partir de la requête GET, ou si elle n'est pas fournie, à partir de la date de rendez-vous existante. Les rendez-vous du médecin pour la date sélectionnée sont également récupérés.

Si la requête est de type POST, cela signifie que l'utilisateur a soumis un formulaire pour mettre à jour l'état du rendez-vous. Le formulaire est vérifié à l'aide de la méthode is_valid() de Django. Si le formulaire est valide, les données sont enregistrées dans la base de données et un message de succès est renvoyé à l'utilisateur.

La vue vérifie également certaines conditions avant d'enregistrer les modifications. Par exemple, si l'utilisateur essaie de confirmer un rendez-vous mais n'a pas fourni d'heure, un message d'erreur est renvoyé. De même, si l'utilisateur essaie de proposer ou d'accepter un rendez-vous sans fournir de date ou d'heure, un message d'erreur est renvoyé.

```python
def appointment_detail(request, appointment_id):
    appointment = get_object_or_404(Appointment, pk=appointment_id)
    # Utiliser la date de l'appointment comme date sélectionnée par défaut si aucune date n'est fournie
    selected_date = request.GET.get('selected_date') or appointment.date.strftime('%Y-%m-%d')
    # Rechercher les rendez-vous du médecin pour la date sélectionnée
    doctor_appointments = Appointment.objects.filter(doctor=appointment.doctor, date=selected_date, status__in=['proposed', 'accepted', 'requested']).exclude(status='cancelled')
    if request.method == 'POST':
        form = AppointmentStatusForm(request.POST, instance=appointment)
        if form.is_valid():
            updated_appointment = form.save(commit=False)
            status = updated_appointment.status

            if status == 'confirmed' and not updated_appointment.time:
                messages.error(request, "Veuillez entrer une heure pour le rendez-vous accepté.")
                return redirect('main:appointment_detail', appointment_id=appointment_id)

            if status == 'cancelled' and not updated_appointment.cause:
                messages.error(request, "Veuillez entrer une cause pour l'annulation du rendez-vous.")
                return redirect('main:appointment_detail', appointment_id=appointment_id)

            if status == 'proposed' and (not updated_appointment.time or not updated_appointment.date):
                messages.error(request, "Veuillez entrer une date et une heure pour la nouvelle proposition de rendez-vous.")
                return redirect('main:appointment_detail', appointment_id=appointment_id)
            # Vérifier si l'heure de rendez-vous est comprise entre 8h30 et 18h00
            if updated_appointment.time is not None:
                min_time = time(hour=8, minute=30)
                max_time = time(hour=18, minute=0)
                if updated_appointment.time < min_time or updated_appointment.time > max_time:
                    messages.error(request, "Les heures de rendez-vous doivent être comprises entre 8h30 et 18h00.")
                    return redirect('main:appointment_detail', appointment_id=appointment_id)
```

Figure 15 La vue appointment_detail

La vue recherche :

Cette vue est destinée aux utilisateurs authentifiés qui sont soit des médecins soit des réceptionnistes. La vue recherche les patients en fonction d'une chaîne de recherche donnée dans l'URL de la requête GET.

Si l'utilisateur est un médecin, la vue recherche les patients qui ont un rendez-vous confirmé avec ce médecin et qui correspondent à la chaîne de recherche. Les résultats de la recherche sont renvoyés dans le modèle de page 'main/search.html' avec l'objet médecin.

Si l'utilisateur est un réceptionniste, la vue recherche tous les patients qui correspondent à la chaîne de recherche. Les résultats de la recherche sont renvoyés dans le modèle de page 'main/searchrecep.html' avec l'objet réceptionniste.

La fonction lambda utilisée ici est pour autoriser seulement les utilisateurs qui ont l'attribut is_doctor ou is_receptionist. Si l'utilisateur n'a pas l'un de ces attributs, il sera redirigé vers la page de connexion.

La vue utilise également la décoration @login_required pour s'assurer que l'utilisateur est connecté avant d'accéder à la vue.

```python
@login_required
@user_passes_test(lambda user: user.is_doctor or user.is_receptionist)
def search(request):
    query = request.GET.get('q')
    doctor=None
    if request.user.is_doctor:
        doctor = Doctor.objects.get(user=request.user)
        results = Patient.objects.all().filter(appointment__doctor=doctor,appointment__status='confirmed').filter(
            Q(cin__icontains=query)
            | Q(firstName__icontains=query)
            | Q(lastName__icontains=query)
            | Q(phoneNumber__icontains=query)
            )
        return render(request, 'main/search.html', {'results': results,'doctor':doctor})
    else:
        receptionist = Receptionist.objects.get(user=request.user)
        results = Patient.objects.all().filter(
            Q(cin__icontains=query)
            | Q(firstName__icontains=query)
            | Q(lastName__icontains=query)
            | Q(phoneNumber__icontains=query)
            )
        return render(request, 'main/searchrecep.html', {'results': results,'doctor':receptionist})
```

Figure 16 La vue recherche

La vue d'authentification pour les médecins

La vue doctor_login permet à un médecin de se connecter. Si la méthode HTTP est POST, cela signifie que le formulaire de connexion a été soumis et le code vérifie si les données du formulaire sont valides. S'ils sont valides, la fonction form.get_user() est appelée pour récupérer l'utilisateur associé au formulaire et il est vérifié si l'utilisateur est un médecin en utilisant user.is_doctor. Si l'utilisateur est bien un médecin, alors la fonction login est appelée pour connecter l'utilisateur et le rediriger vers le tableau de bord du médecin. Si l'utilisateur n'est pas un médecin, un message d'erreur est affiché. Si les données du formulaire ne sont pas valides, un message d'erreur est également affiché.

```python
def doctor_login(request):
    if request.method == 'POST':
        form = DoctorAuthenticationForm(request, data=request.POST)
        if form.is_valid():
            user = form.get_user()
            if user.is_doctor:
                login(request, user)
                return redirect('doctor_dashboard')
            else:
                messages.error(request, 'You are not a doctor.')
        else:
            messages.error(request, 'Invalid login.')
    else:
        form = DoctorAuthenticationForm()
    return render(request, 'accounts/doctor_login.html', {'form': form})
```

Figure 17 La vue d'authentification pour les médecins

Bilan du projet

I. Les difficultés rencontrées

Lors de la réalisation de ce projet, notre équipe a été confrontée à plusieurs difficultés. Tout d'abord, la prise en main de l'outil Django a été un challenge pour nous, car il est rigide dans certains aspects. Nous avons dû passer beaucoup de temps à comprendre le fonctionnement de cet outil.

Ensuite, la configuration de l'environnement de développement a également posé des problèmes, ce qui nous a pris du temps pour mettre en place un environnement stable et fonctionnel. Malgré ces obstacles, nous avons persévéré et continué à travailler ensemble pour achever le projet.

Nous avons également compris l'importance de la communication entre les membres de l'équipe pour assurer le succès du projet. Nous avons donc encouragé les échanges d'idées et de propositions pour trouver les meilleures solutions aux problèmes rencontrés.

En somme, bien que nous ayons rencontré des difficultés, notre équipe a su les surmonter grâce à une organisation rigoureuse, une communication ouverte et une persévérance continue.

II. Compétences acquises

Au cours de notre projet de fin d'études, nous avons travaillé en équipe pour concevoir et développer une application web complète. Ce projet nous a permis d'acquérir de nombreuses compétences en développement web, en gestion de projet et en collaboration avec les membres de l'équipe.

Tout d'abord, nous avons dû maîtriser l'outil de développement Django pour créer l'application. Cela a nécessité une connaissance approfondie de Python, HTML, CSS et JavaScript, que nous avons acquise au fil de notre formation. Nous avons également utilisé des librairies et des frameworks tels que Bootstrap pour faciliter la conception et le développement de l'interface utilisateur.

En plus de nos compétences techniques, nous avons également développé notre capacité à travailler en équipe. Nous avons dû communiquer et collaborer efficacement pour répartir les tâches, faire des propositions et résoudre les problèmes ensemble.

Enfin, ce projet nous a permis de comprendre l'importance de l'analyse des besoins du client. Dans l'ensemble, ce projet nous a permis d'acquérir de nombreuses compétences qui nous seront utiles dans notre future carrière.

Conclusion

En conclusion, ce projet de fin d'études a été une expérience professionnelle enrichissante pour notre équipe. Nous avons été confrontés à des défis techniques, organisationnels et de communication, mais nous avons su les surmonter grâce à notre détermination, notre collaboration et notre acquisition de compétences. Nous avons pu approfondir notre maîtrise de l'outil Django, des langages de programmation tels que Python, HTML, CSS et JavaScript, ainsi que des librairies et des frameworks tels que Bootstrap. Nous avons également appris l'importance de la communication et de la collaboration au sein d'une équipe pour réussir la réalisation d'un projet. Nous sommes fiers du résultat final de notre projet et nous espérons que cette expérience nous sera bénéfique dans notre future carrière professionnelle.

Bibliographie

https://docs.djangoproject.com/en/

https://www.memoireonline.com/03/20/11628/

https://www.w3schools.com/

https://medium.com/

https://poe.com/

https://stackoverflow.com/

https://openclassrooms.com/

https://developer.mozilla.org/