

Introduction Générale

Bonjour ! Bienvenue à ce projet de gestion d'un cabinet médicale avec une application réseau client/serveur. Ce projet consiste à développer une application informatique pour aider à gérer un cabinet médical en utilisant une architecture client/serveur. Le but est de faciliter la communication et la collaboration entre les membres de l'équipe médicale en automatisant certaines tâches et en fournissant des informations en temps réel pour une meilleure prise de décision.

Le projet implique la conception et le développement d'un système de gestion de dossiers de patients, de rendez-vous, de prescriptions, de facturation et de rapports médicaux. L'application doit être facile à utiliser pour les professionnels de la santé et doit être sécurisée pour protéger les informations confidentielles des patients.

Dans ce rapport de 30 pages, nous allons décrire les différentes étapes du développement de l'application, les choix technologiques et de conception qui ont été faits, ainsi que les résultats obtenus.

Enfin, nous allons conclure en évaluant la réussite du projet et en discutant des perspectives d'avenir pour l'application, notamment en termes d'amélioration des fonctionnalités, et de l'expérience utilisateur.

Ce projet est une occasion unique pour nous de développer nos compétences en matière de développement logiciel et de comprendre les défis et les opportunités liés à la gestion d'un cabinet médicale. Nous espérons que ce rapport sera informatif et utile pour tous les acteurs impliqués dans le projet.

Contexte :

La gestion d'un cabinet médicale peut être un défi de taille, car elle implique de nombreux processus tels que la gestion des dossiers des patients, la planification des rendez-vous, la facturation et la prescription des médicaments. L'utilisation d'une application informatique peut aider à automatiser certains processus et à améliorer la communication et la collaboration entre les membres de l'équipe médicale.

Objectifs :

L'objectif principal de ce projet est de concevoir et de développer une application de gestion d'un cabinet médicale en utilisant une architecture client/serveur pour permettre aux professionnels de la santé de travailler plus efficacement et d'améliorer la qualité des soins prodigués. Les objectifs spécifiques sont les suivants :

- Développer une application conviviale pour les professionnels de la santé qui permet de gérer les dossiers des patients, les rendez-vous, les prescriptions, la facturation et les rapports médicaux.
- Concevoir une architecture client/serveur pour permettre une communication rapide et efficace entre les membres de l'équipe médicale.
- Assurer la sécurité et la confidentialité des informations des patients.
- Évaluer la réussite du projet et discuter des perspectives d'avenir pour l'application.

Méthodologie :

Pour atteindre ces objectifs, nous avons adopté une méthodologie de développement itérative en utilisant le modèle en V. Nous avons commencé par l'analyse des besoins des utilisateurs pour identifier les fonctionnalités clés de l'application. Nous avons ensuite conçu l'architecture client/serveur et sélectionné les outils et les technologies appropriés pour la réalisation de l'application. Nous avons ensuite procédé à la phase de développement en implémentant les différentes fonctionnalités de l'application. Enfin, nous avons évalué la réussite du projet en utilisant des critères tels que la convivialité de l'application, la sécurité des données, l'efficacité et l'impact sur la gestion de la pratique médicale.

Résultats :

Le projet a abouti à la conception et au développement d'une application de gestion d'un cabinet médicale en utilisant une architecture client/serveur. L'application permet de gérer les dossiers des patients, les rendez-vous, les prescriptions, la facturation et les rapports médicaux. Elle est conviviale et offre une interface utilisateur intuitive pour faciliter son utilisation. Elle est également sécurisée, avec un accès contrôlé aux données des patients pour garantir leur confidentialité. L'utilisation de l'architecture client/serveur permet une communication rapide et efficace entre les membres de l'équipe médicale, ce qui améliore la coordination et la qualité des soins prodigués. L'évaluation du projet a montré que l'application répondait aux exigences spécifiées et était un succès. Les perspectives d'avenir pour l'application incluent la mise à disposition de fonctionnalités supplémentaires pour répondre aux besoins en constante évolution des professionnels de la santé.

1

L'analyse des besoins

Introduction :

L'analyse des besoins est une étape essentielle dans le processus de développement de toute application, y compris une application de gestion de cabinet médical. Cette étape permet de déterminer les fonctionnalités nécessaires pour répondre aux besoins des utilisateurs et de garantir que l'application répond aux exigences du marché.

Dans ce premier chapitre, nous allons examiner les besoins des professionnels de la santé en matière de gestion de cabinet médical. Nous allons d'abord décrire les tâches nécessaires pour gérer un cabinet médical, puis nous examinerons les problèmes qu'ils rencontrent dans la gestion de leur cabinet. Nous analyserons également les attentes des patients en matière de services de santé et leur impact sur la conception de l'application.

Enfin, nous allons répondre aux besoins des professionnels de la santé et des patients. Nous montrerons comment ces fonctionnalités permettront d'améliorer l'efficacité, la qualité et la satisfaction des utilisateurs de l'application.

1 Problématiques et objectifs :

1.1 Problématiques :

La gestion d'un cabinet médical comporte plusieurs défis, notamment la gestion des rendez-vous, la gestion des patients, la gestion des dossiers médicaux, la gestion des traitements et des ordonnances, et la gestion des paiements et des remboursements. Ces tâches peuvent être chronophages et sujettes à des erreurs humaines, ce qui peut entraîner des retards, des mécontentements des patients et des pertes financières.

1.2 Objectifs :

Les objectifs pour résoudre les problèmes liés à la gestion de cabinet médical sont nombreux et peuvent inclure :

- Améliorer la gestion des rendez-vous pour réduire les temps d'attente des patients et optimiser le temps de travail des professionnels de la santé.
- Faciliter la gestion des dossiers médicaux et des traitements pour une meilleure prise en charge des patients.
- Assurer une meilleure communication entre les professionnels de la santé pour une coordination optimale dans la gestion des patients.
- Faciliter la gestion des paiements et des remboursements pour une gestion financière efficace du cabinet médical.
- Améliorer la satisfaction des patients en offrant un service de qualité et une meilleure prise en charge de leurs besoins médicaux.
- Faciliter l'accès aux informations médicales des patients pour une meilleure prise de décision dans la gestion des soins de santé.

Ces objectifs peuvent être atteints grâce à une application de gestion de cabinet médical bien conçue, qui répond aux besoins des professionnels de la santé et des patients.

2 Analyse des besoins :

2.1 Identification des acteurs :

Un acteur représente un rôle joué par une personne externe ou par un processus qui interagit avec le système. Les acteurs de notre système sont :

2.1.1 • Secrétaire : il s'agit d'un acteur qui s'occupe de la gestion des dossiers médicaux, prise de rendez-vous et génération des bilans d'activités.

2.1.2 • Médecin : il s'agit d'un acteur qui consulte les dossiers médicaux des patients et accéder à leurs informations relatives pour une meilleure prise en charge des soins de santé.

2.2 Les besoins fonctionnels :

Ce sont les exigences du client spécifiant un comportement d'entrer et sortie du système.

Les besoins fonctionnels des différents acteurs peuvent être résumés comme suit :

2.2.1 Coté secrétaire :

- Gestion des dossiers des patients.
- Gestion des rendez-vous.
- Gestion de l'Agenda des Patients.
- Gestion de l'ordre d'arrivée des Patients.
- Gestion de Prescription complet (ordonnance, bilans, certificats ...)
- Gestion de liste des Médicaments (ajout, suppression, ...).
- Gestion de compatibilité

2.2.2 Coté Médecin :

- Consultation du planning des rendez-vous.
- Mise en disponibilité de plages horaires.
- Consultation des dossiers malades.

2.3 Les besoins non fonctionnels :

Ce sont des besoins en relation avec la performance du système, la facilité d'utilisation, l'ergonomie des interfaces, la sécurité etc...

Et parmi ses besoins nous citons :

- Accès à l'application via l'authentification.
- Simplicité et ergonomie de l'interface graphique.

3 Environnement de développement :

3.1 Eclipse IDE :



Eclipse IDE (Integrated Development Environment) est un environnement de développement open-source largement utilisé pour créer des applications logicielles dans différents langages de programmation tels que Java, C++, Python, etc. Il fournit des outils de développement avancés tels que l'édition de code, le débogage, la gestion de projets, le visionnage de code, la création d'interfaces graphiques utilisateur et bien plus encore. Eclipse IDE est personnalisable et extensible, ce qui signifie que les développeurs peuvent ajouter des plugins pour améliorer les fonctionnalités de l'environnement en fonction de leurs besoins spécifiques. [1]

3.2 php myAdmin :



PhpMyAdmin est une application web open-source conçue pour gérer des bases de données MySQL à l'aide d'une interface graphique conviviale. Il est souvent utilisé par les développeurs et les administrateurs de bases de données pour effectuer des tâches courantes telles que la création de tables, la modification de la structure de la base de données, la modification des données dans la base de données, l'importation et l'exportation de données, et bien plus encore. PhpMyAdmin facilite également la gestion de plusieurs bases de données et de plusieurs utilisateurs avec des autorisations spécifiques pour accéder aux bases de données. Il est écrit en PHP et est disponible en tant qu'application autonome ou comme un composant intégré dans un serveur Web tel que Apache. [2]

4 Langages de développements :

4.1 JAVA :



Java est un langage de programmation orienté objet, conçu pour être portable et indépendant de la plate-forme. Il est largement utilisé pour le développement d'applications, notamment pour les applications de bureau, les applications Web et les applications mobiles. Java est compilé en bytecode, qui peut être exécuté sur une machine virtuelle Java (JVM) installée sur n'importe quelle plate-forme compatible Java, ce qui en fait un langage de programmation multiplateforme. Java est également connu pour sa sécurité, car il est livré avec un système de sécurité intégré qui permet de contrôler les autorisations et les restrictions d'accès à certaines parties du code. Java est largement utilisé par les grandes entreprises et les organisations pour le développement d'applications robustes et évolutives. [3]

4.2 Java FX :



JavaFX est une plateforme logicielle permettant de créer et déployer des interfaces utilisateur riches en Java. Elle offre des bibliothèques et des outils pour construire des applications graphiques interactives pour diverses plateformes. JavaFX propose des fonctionnalités graphiques avancées, d'animation, de multimédia et de gestion des entrées utilisateur. [4]

4.3 CSS :



CSS (Cascading Style Sheets) est un langage de feuilles de style utilisé pour définir l'apparence et la mise en forme des documents HTML. Il permet de contrôler les couleurs, les polices, les marges, les espacements et d'autres aspects visuels d'un site web. [5]

4.4 MySQL :



MySQL est un système de gestion de base de données relationnelle open source qui permet de stocker, organiser et accéder à des données de manière efficace et sécurisée. Il est largement utilisé dans le développement web pour gérer des informations structurées telles que des articles, des commentaires, des profils d'utilisateurs et des transactions en ligne. MySQL utilise le langage SQL (Structured Query Language) pour interagir avec la base de données et permet une intégration facile avec différents langages de programmation tels que PHP, Java, Python, etc. [6]

Conclusion :

En conclusion, l'analyse des besoins est une étape essentielle dans le processus de développement d'une application de gestion de cabinet médical. Elle permet de comprendre les besoins des utilisateurs, les fonctionnalités nécessaires pour répondre à ces besoins, les contraintes techniques et les exigences de sécurité. Cette étape est cruciale pour s'assurer que l'application répondra aux attentes des utilisateurs et sera facilement utilisable et maintenable. En identifiant les besoins dès le début du projet, il est possible de réduire les risques d'erreurs et de coûts liés aux modifications ultérieures. Par conséquent, une analyse de besoins approfondie et rigoureuse est essentielle pour assurer le succès du projet et répondre aux attentes des utilisateurs.

2

Conception et modélisation

Introduction :

Dans ce chapitre, nous allons explorer la phase de conception et de modélisation de notre application de gestion d'un cabinet médicale. Nous allons commencer par décrire les exigences fonctionnelles et non fonctionnelles de l'application et présenter les différents cas d'utilisation pour définir les interactions entre les différents acteurs. Ensuite, nous allons passer à la modélisation de l'architecture client/serveur en utilisant le langage de modélisation unifié (UML). Nous allons décrire en détail les différents diagrammes UML utilisés pour représenter l'architecture, tels que les diagrammes de cas d'utilisation, les diagrammes de classes et les diagrammes de séquence. Nous allons également discuter des décisions de conception clés prises pour garantir la convivialité, la sécurité et la performance de l'application. La phase de conception et de modélisation est une étape cruciale pour assurer le succès du projet, car elle permet de définir les spécifications et les détails techniques nécessaires pour la phase de développement ultérieure.

1 UML :



UML (Unified Modeling Language) est un langage de modélisation graphique utilisé pour concevoir, visualiser et spécifier des systèmes logiciels. Il fournit des diagrammes standardisés pour représenter les différents aspects d'un système. UML est largement utilisé dans l'industrie pour la conception de logiciels. [6]

2 Diagrammes de cas d'utilisation :

Un diagramme de cas d'utilisation UML est un diagramme de modélisation qui représente les interactions entre les utilisateurs et le système. Il permet de décrire les fonctionnalités du système du point de vue de l'utilisateur.

Voici le diagramme général de cas d'utilisation :

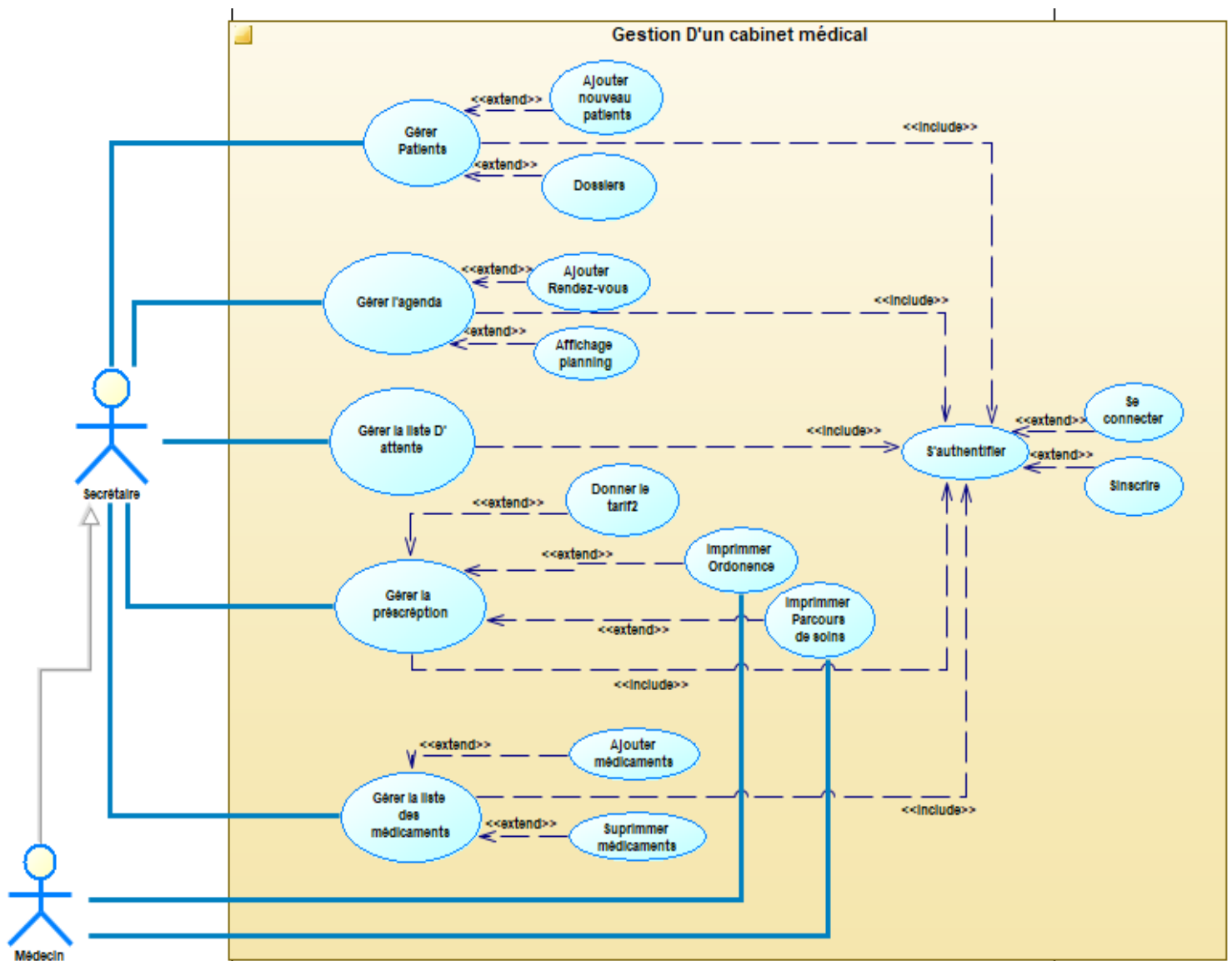


Figure2. 1 : Diagramme Général de cas d'utilisation

Ce diagramme résume tous les cas d'utilisation de notre application.

2.1 Cas d'utilisation « s'authentifier » :

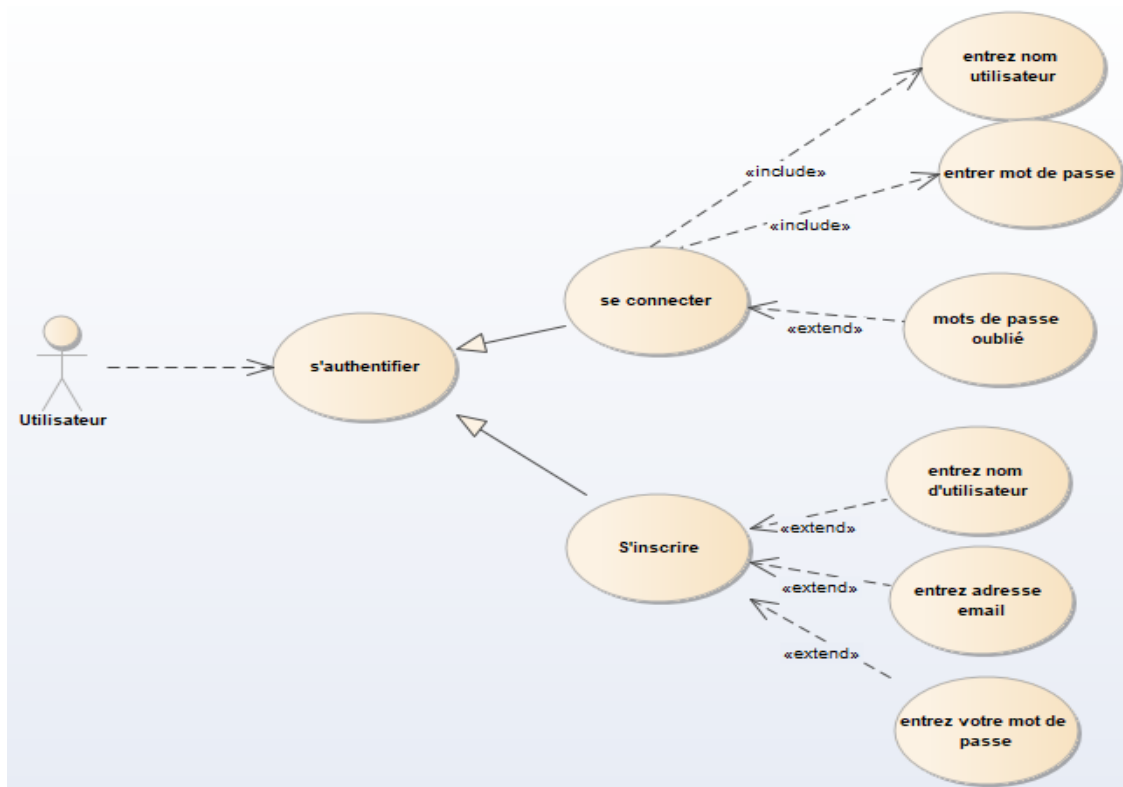


Figure2. 2 : Diagramme de cas d'utilisation s'authentifier

Ce diagramme illustre le cas d'utilisation "s'authentifier" qui permet à l'utilisateur de se connecter ou s'inscrire.

2.2 Cas d'utilisation « gérer patients » :

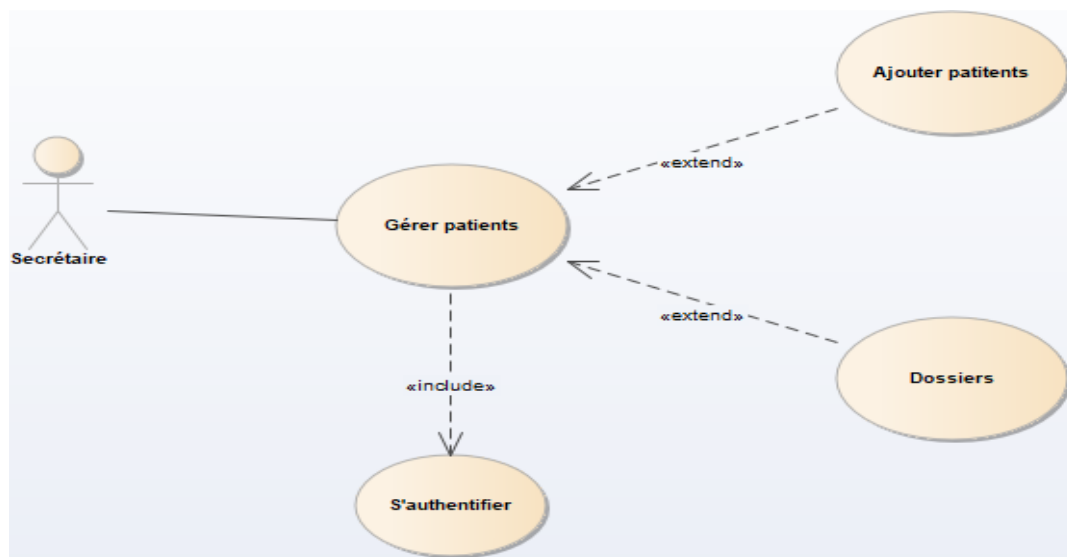


Figure2. 3 : Diagramme de cas d'utilisation gérer patients.

Ce diagramme illustre le cas d'utilisation "gérer patient" qui permet à la secrétaire de gérer les patients (ajouter...)

2.3 Cas d'utilisation « Ajouter patients » :

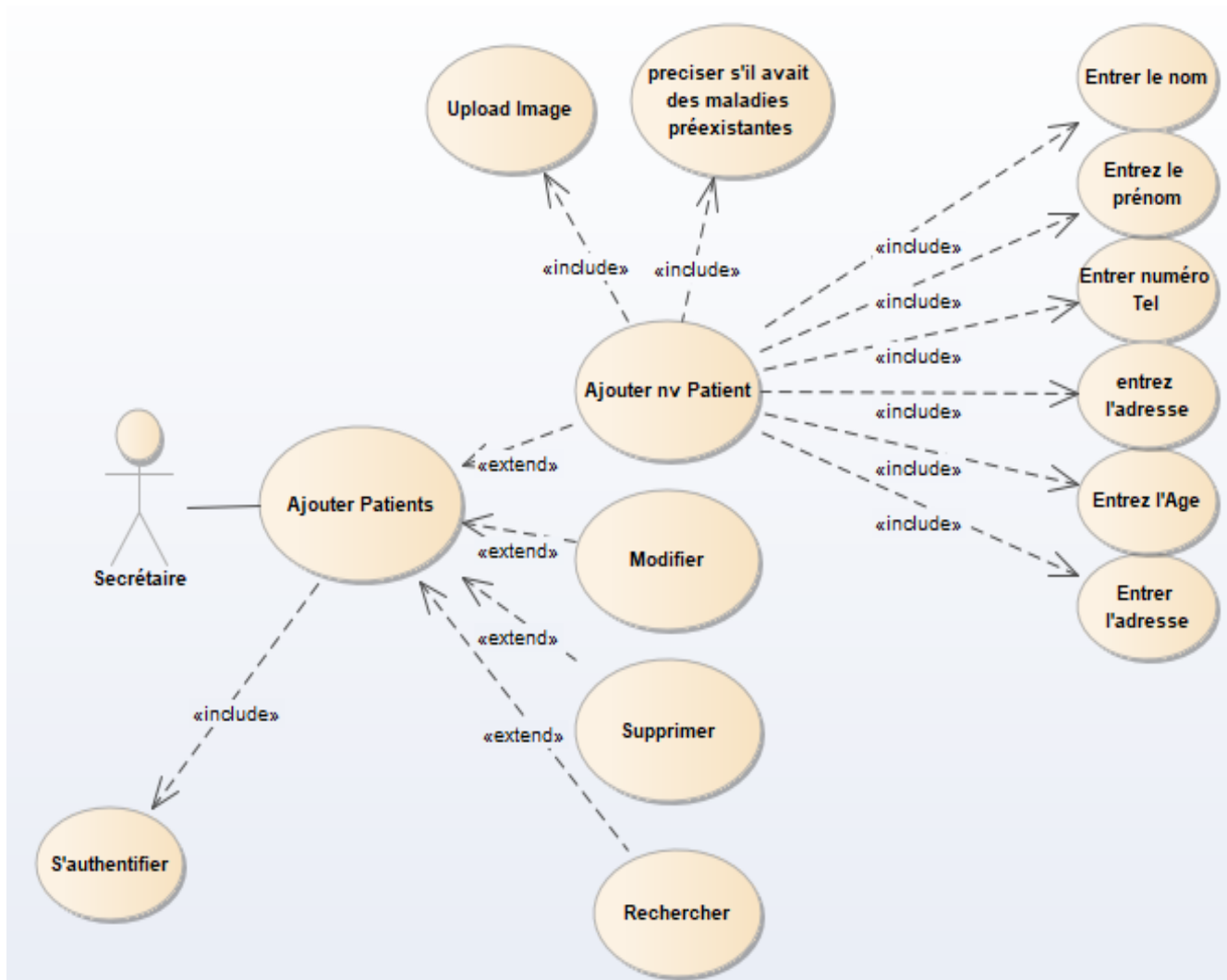


Figure2. 4 : Diagramme de cas d'utilisation ajouter patients

Ce diagramme illustre le cas d'utilisation "Ajouter patient" qui permet à la secrétaire d'ajouter des patients en rentrant tous leurs informations nécessaires et aussi de modifier, supprimer ou rechercher des patients.

2.3 Cas d'utilisation « gérer agenda » :

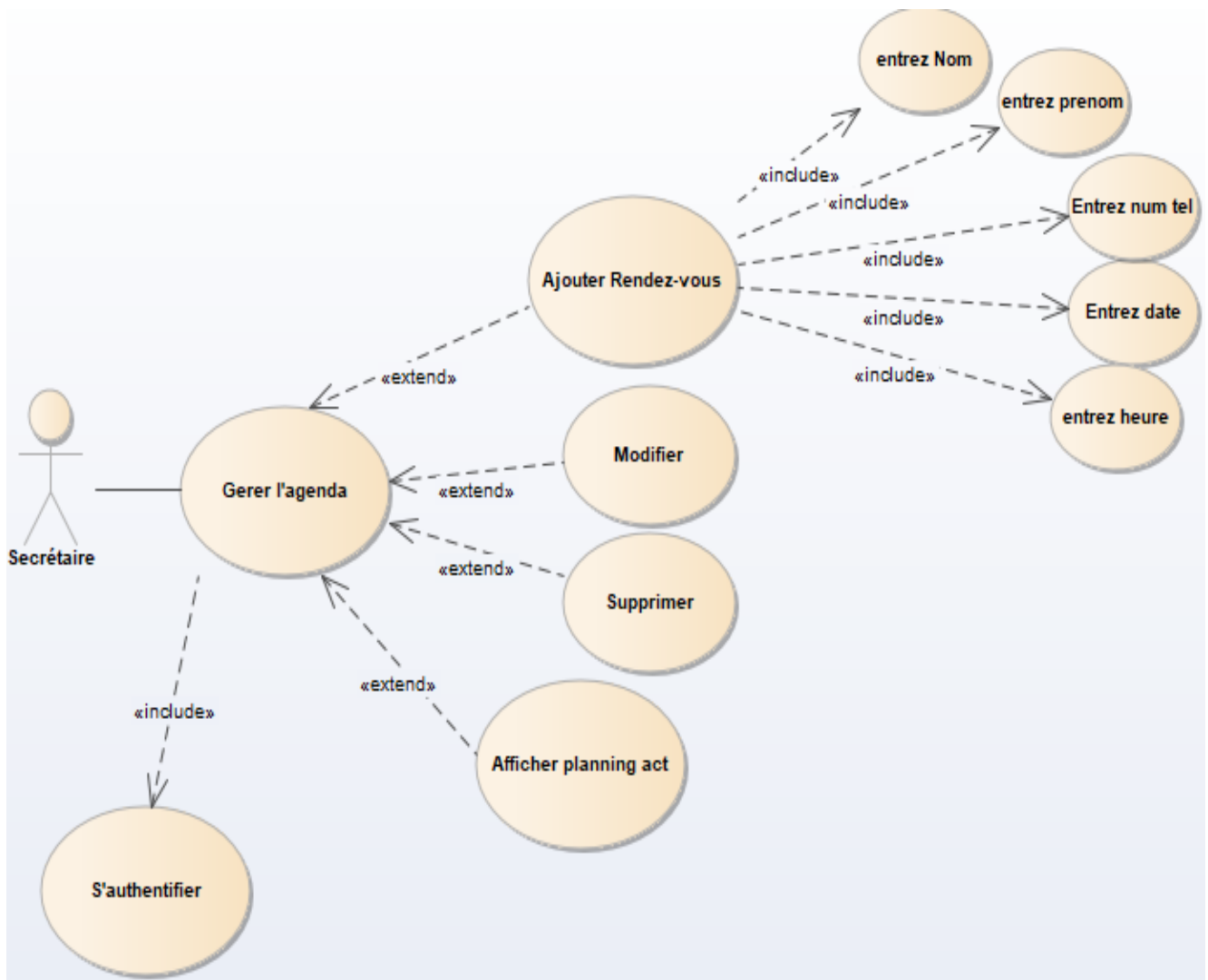


Figure2. 5 : Diagramme de cas d'utilisation gérer agenda.

Ce diagramme illustre le cas d'utilisation "gérer agenda" qui permet à la secrétaire de gérer les rendez-vous (ajouter, modifier, supprimer ...)

2.4 Cas d'utilisation « gérer liste d'attente» :

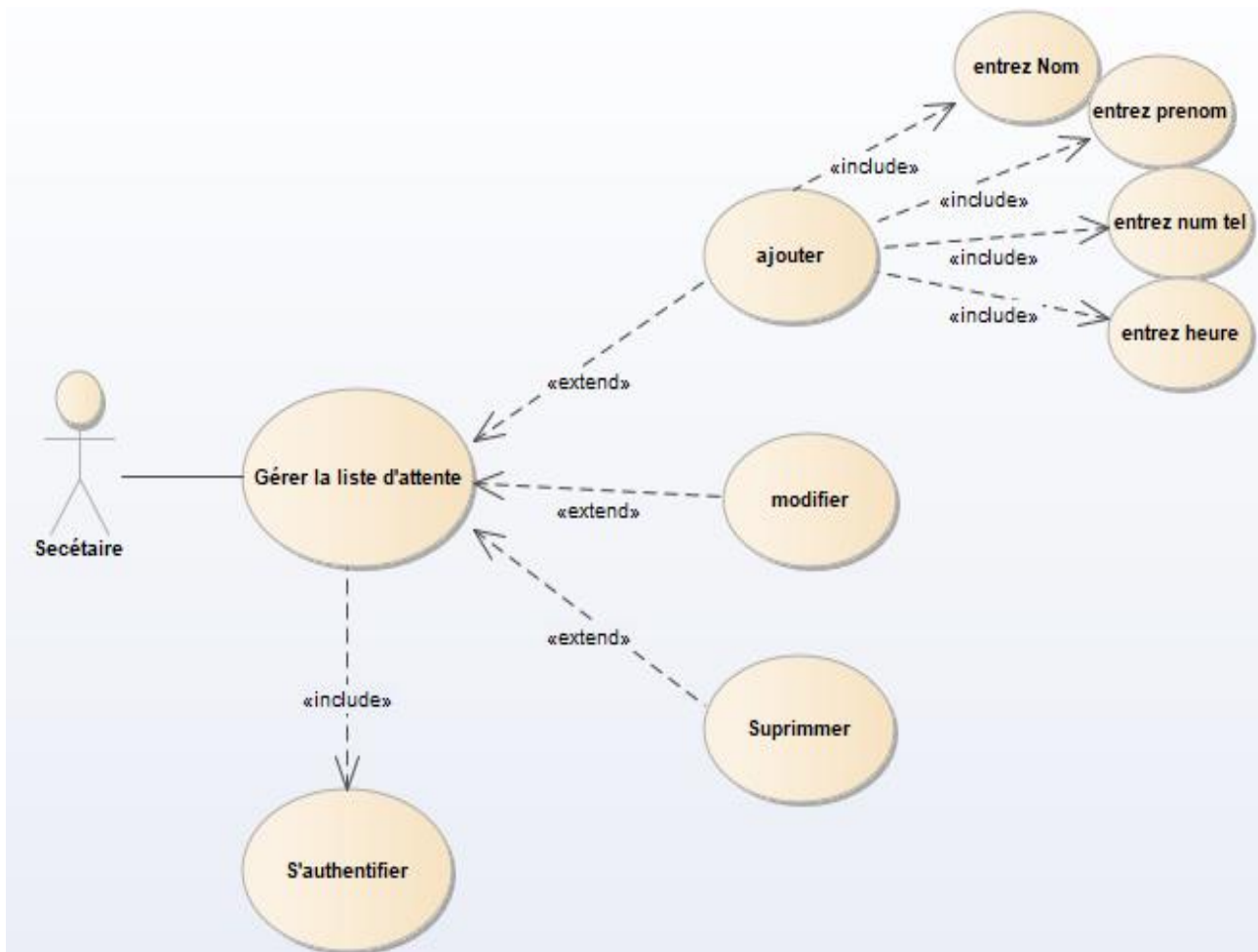


Figure2. 6 : Diagramme de cas d'utilisation gérer liste d'attente.

Ce diagramme montre le cas d'utilisation "gérer liste d'attente" qui permet à la secrétaire de gérer la liste d'attente et d'organiser les patients.

2.5 Cas d'utilisation « gérer prescription » :

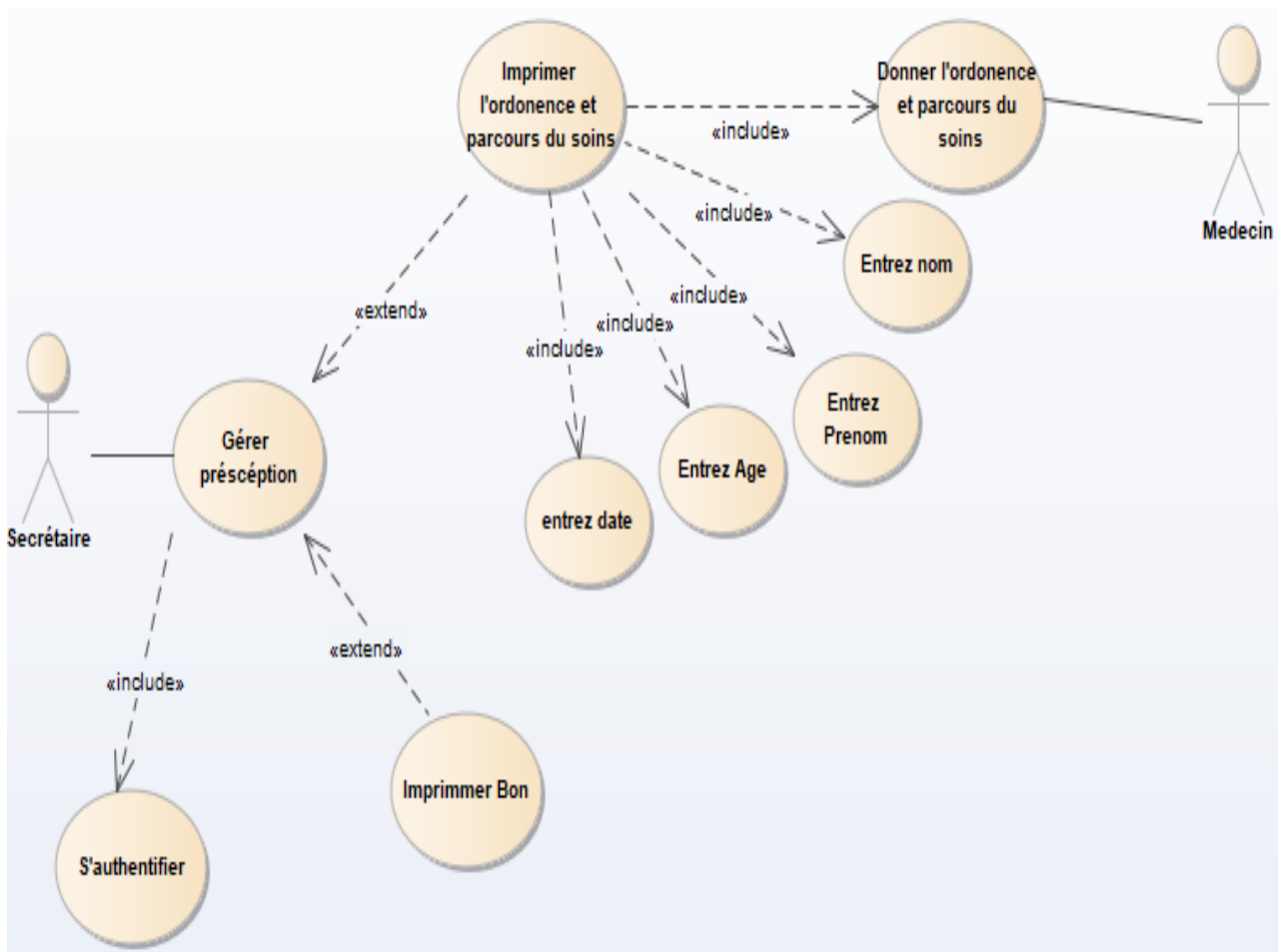


Figure2. 7 : Diagramme de cas d'utilisation gérer prescription.

Ce diagramme montre le cas d'utilisation "gérer prescriptions" qui permet à la secrétaire d'imprimer les ordonnances en rentrant toutes les informations nécessaires après que le médecin la donne aux patients

2.6 Cas d'utilisation « gérer médicaments » :

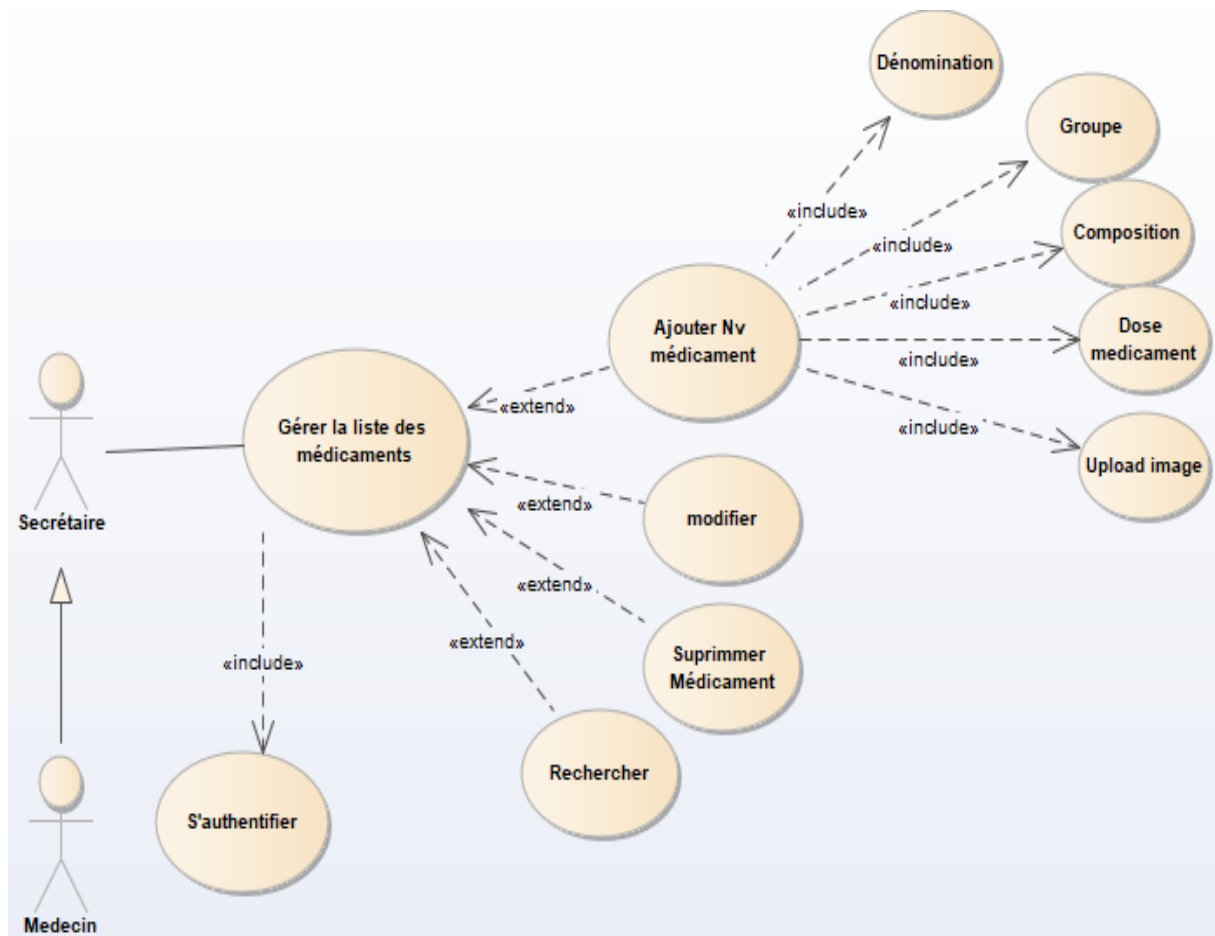


Figure2. 8 : Diagramme de cas d'utilisation gérer médicaments.

Ce diagramme illustre le cas d'utilisation «gérer médicaments» qui permet à la secrétaire ou au médecin d'ajouter, modifier, supprimer ou de rechercher des médicaments.

3 Diagrammes de séquences :

3.1 Diagramme de séquence « s'authentifier » :

Un utilisateur doit s'authentifier en saisissant ses propres coordonnées (identifiant et mot de passe), puis le système procède à la vérification des informations introduites pour les comparer avec les données stockées dans la base de données, si l'une des coordonnées est incomplète, login ou mot de passe est incorrect le système affiche un message d'erreur sinon l'accès est autorisé.

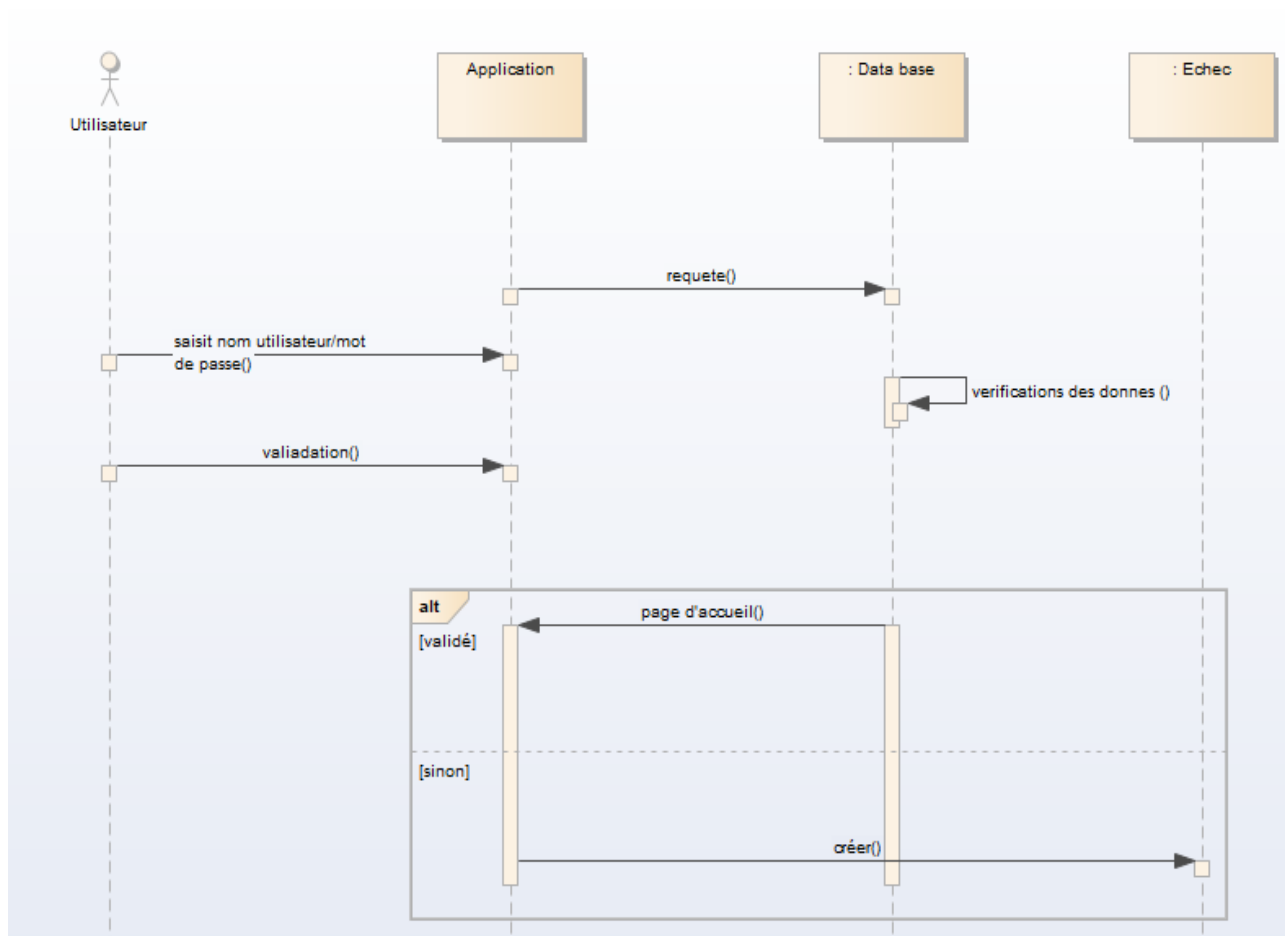


Figure2. 9 : Diagramme de séquence s'authentifier

3.2 Diagramme de séquence « récupérer mot de passe » :

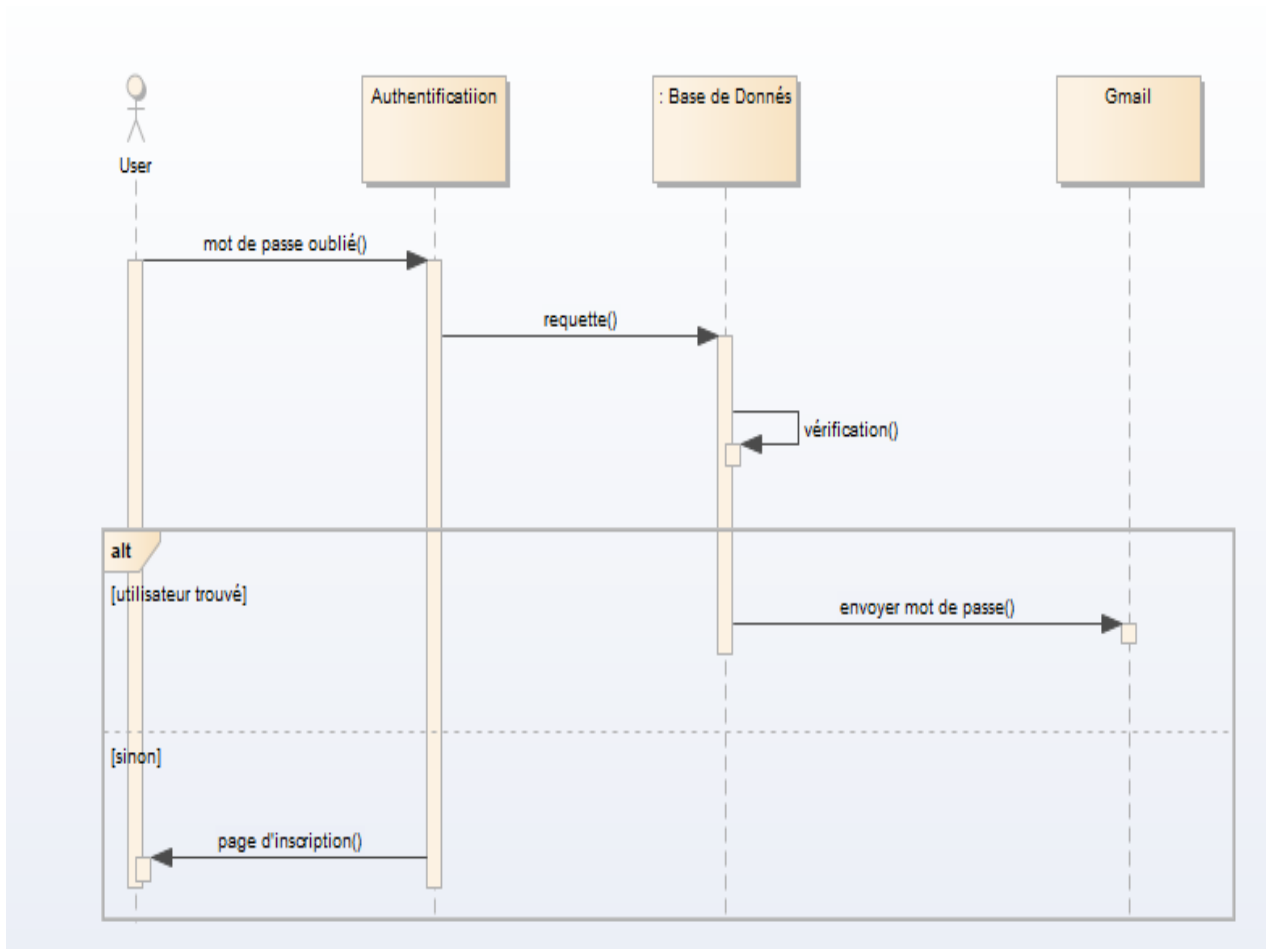


Figure2. 10 : Diagramme de séquence récupérer mot de passe.

Quand l'utilisateur oublie son mot de passe, le système vérifie les données puis envoie un mot de passe à l'utilisateur dans son email pour que le dernier puisse se connecter

4 Diagrammes d'activité:

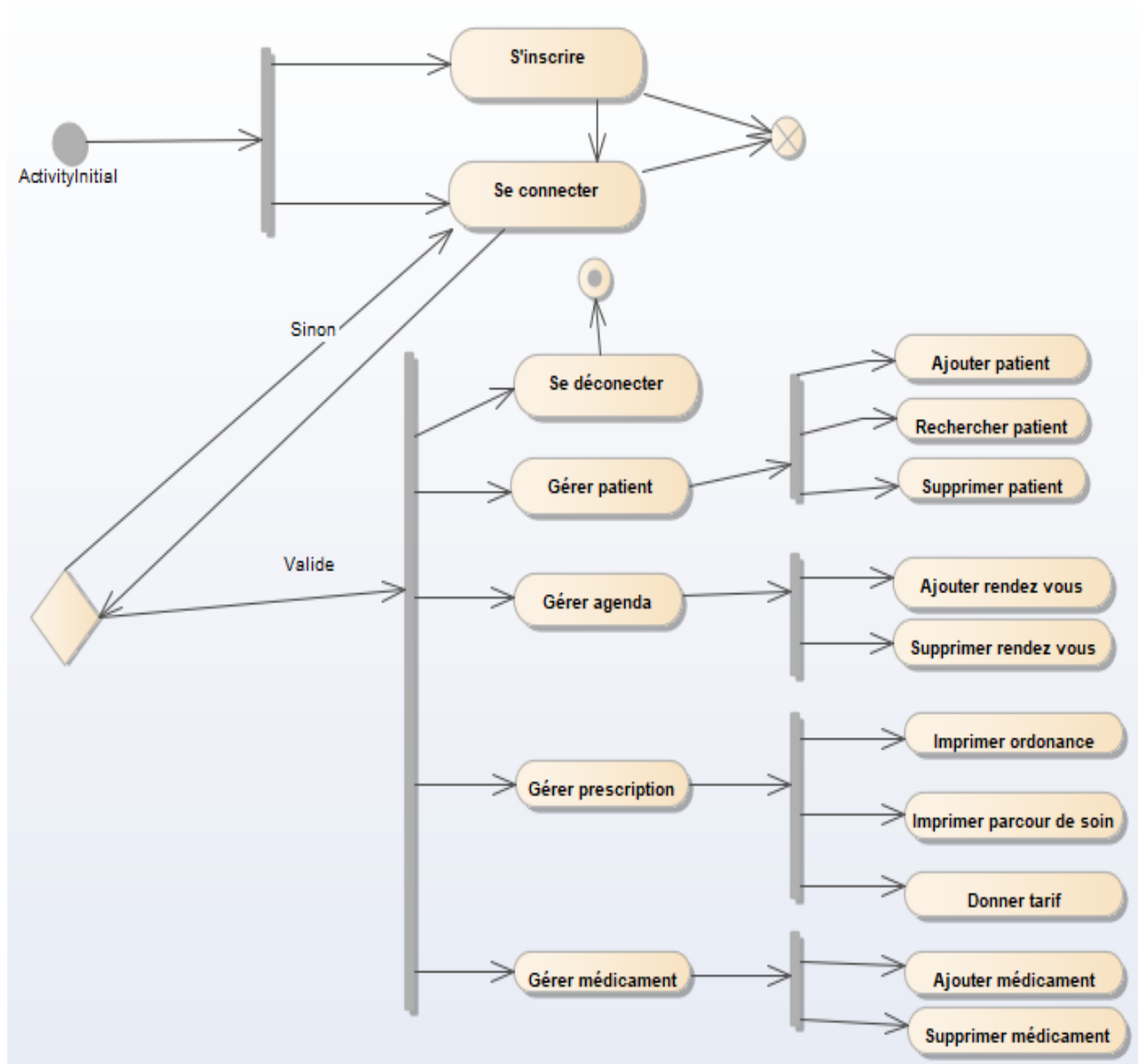


Figure2. 11 : Diagramme d'activité général

5 Diagrammes de classe :

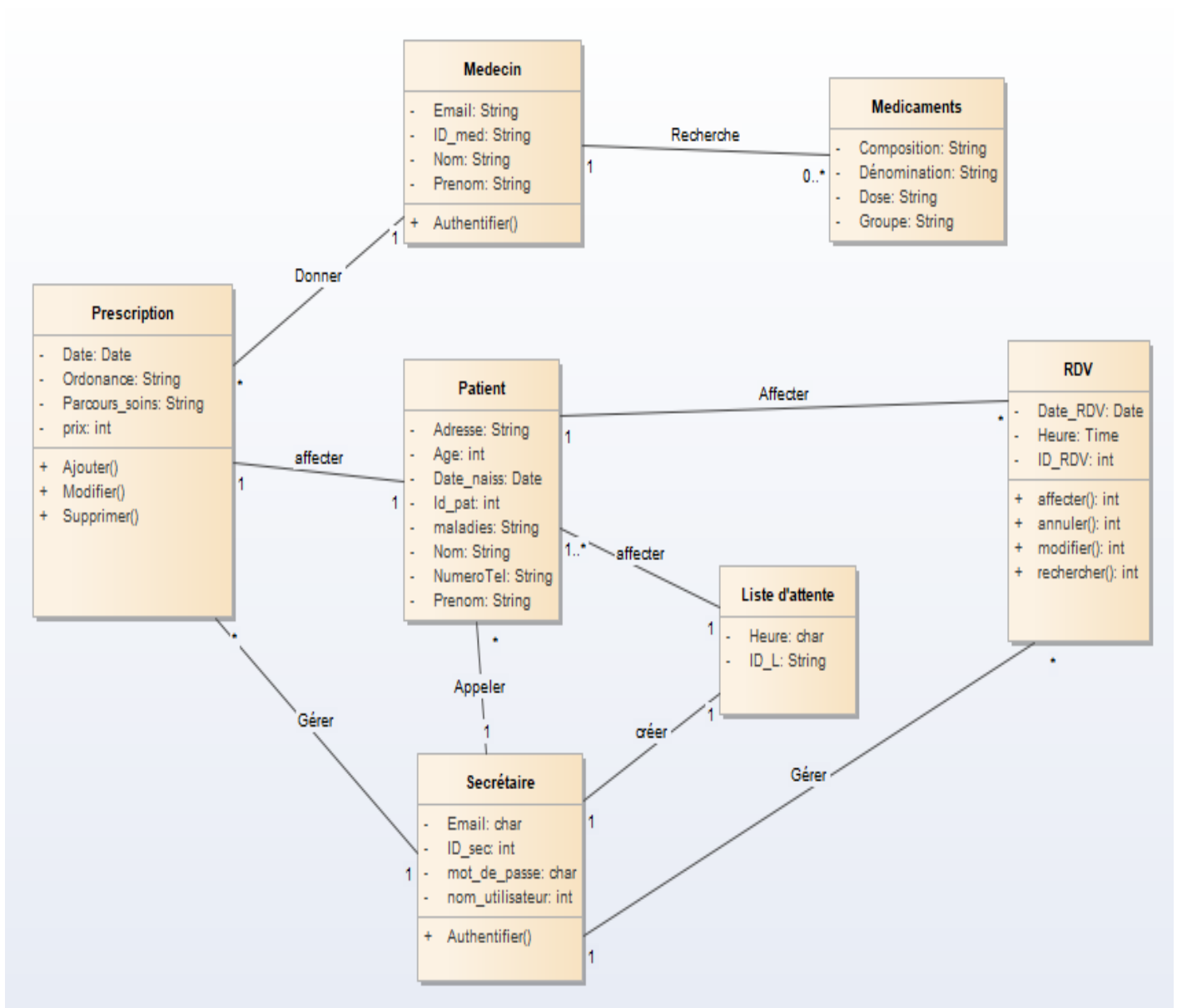


Figure2. 12 : Diagramme de classe.

6 Passage au modèle relationnelle :

Le modèle relationnel est le modèle logique de données qui correspond à l'organisation des données dans les bases de données relationnelles. Un modèle relationnel est composé de relations, appelée table. Ces tables sont décrites par des attributs ou champs. Pour décrire une relation, on indique tout simplement son nom, suivi du nom de ses attributs entre parenthèses. L'identifiant d'une relation est composé d'un ou plusieurs attributs qui forment la clé primaire. Une relation peut faire référence à une autre en utilisant une clé étrangère, qui correspond à la clé primaire de la relation référencée.

6.1 Règles du passage du diagramme de classe au modèle logique :

Les règles de passage sont :

1. Relation (1..*) : il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association. L'attribut aura le nom de la clé primaire de la relation père de l'association.
2. Relation (1..1) : il faut ajouter une relation qui prend les deux clé primaire des classes mère comme clé étrangère.
3. Relation d'héritage : la clé primaire de la classe mère est utilisée pour identifier chacune de ses classes filles : cette clé étant pour chaque classe fille, à la fois clé primaire et une clé étrangère vers la classe mère. [14]

6.2 Modèle logique des données :

Médecin (Id Med, Nom Med, Prénom Med, Email, mot de passe).

Secrétaire (Id Sec, Nom utilisateur, Email, mot de passe).

Patient (Id pat, Nom, Prénom, Date naiss, adresse, age, num tel).

RDV (Id RDV, date_RDV, heure).

Médicament (composition, nom, dose ,groupe).

Prescription (Id pres, prix, date, ordonnance).

Liste d'attente (Id L, heure).

Conclusion :

En conclusion de ce chapitre sur la conception de l'application, nous avons examiné les principaux aspects liés à la conception de l'application. Nous avons détaillé l'architecture et montré les différents diagrammes de cette application.

3

Implémentation

Introduction :

Ce chapitre représente le dernier volet de ce rapport. Nous entamons la réalisation après l'étape de conception défini au préalable. Nous présenterons les différentes interfaces de notre application réalisées à l'aide des outils cités et défini dans le premier chapitre.

Quelques Interfaces de notre application :

1.1 Interface d'authentification :

Voici L'interface d'authentification permettant aux différents utilisateurs d'avoir accès au contenu de l'application :

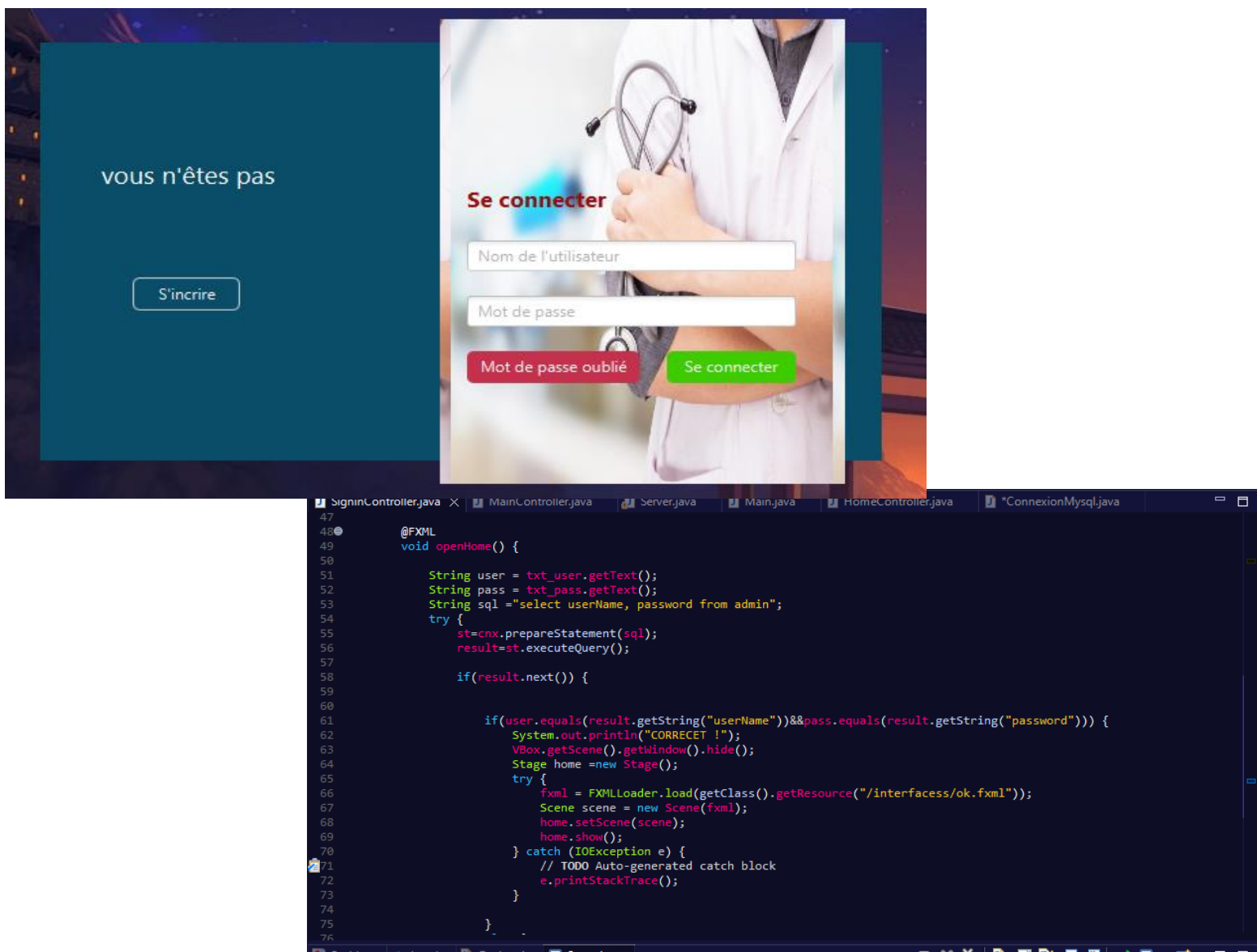
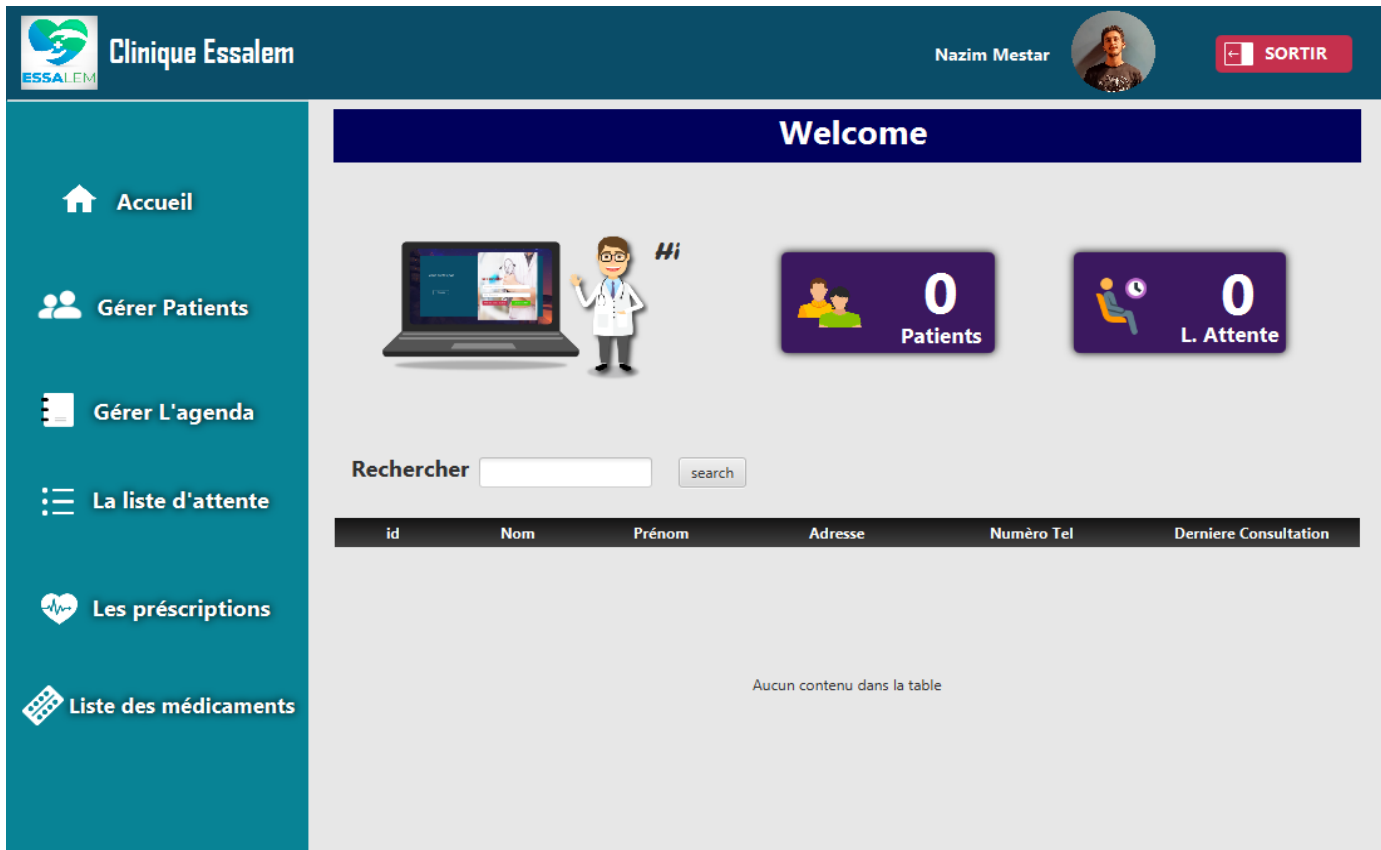


Figure 3.3 : Interface d'authentification.

1.2 Interfaces secrétaire :

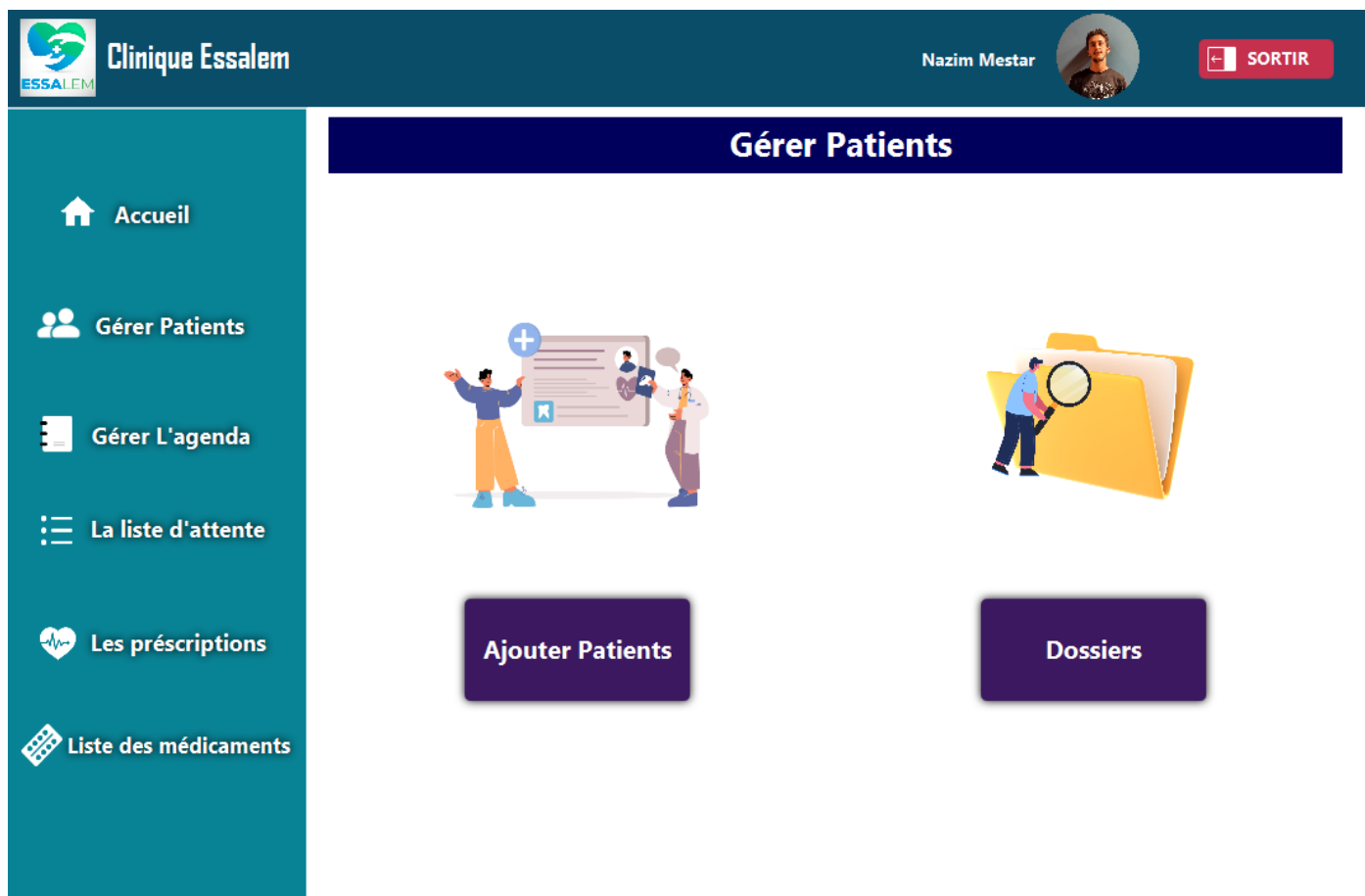
1.2.1 Accueil : Cette interface offre la possibilité à la secrétaire d'affecter des rendez-vous aux patients, quand ce dernier appelle pour prendre rendez-vous, la secrétaire choisit le service rendez-vous, elle peut aussi rechercher les patients déjà existant



```
1 package controller;
2
3 import java.io.IOException;
4
13 public class HomeController implements Initializable{
14
15     private Parent fxml;
16
17     @FXML
18     private AnchorPane root;
19
20     @FXML
21     void accueil(MouseEvent event) {
22         try {
23             fxml = FXMLLoader.load(getClass().getResource("/interfacess/Accueil.fxml"));
24             root.getChildren().removeAll();
25             root.getChildren().setAll(fxml);
26
27         } catch (IOException e) {
28             // TODO Auto-generated catch block
29             e.printStackTrace();
30         }
31     }
32
33
34
35
36     @FXML
37     void gererP(MouseEvent event) {
38
39         try {
40             fxml = FXMLLoader.load(getClass().getResource("/interfacess/GererPatients.fxml"));
41             root.getChildren().removeAll();
42             root.getChildren().setAll(fxml);
43         } catch (IOException e) {
44
45         }
46     }
47 }
```

Figure 3.3 : Interface d'accueil.

1.2.2 Gestion des patients : Cette interface offre la possibilité à la secrétaire de choisir entre ajouter un patient ou d'aller à l'interface des dossiers médicaux :



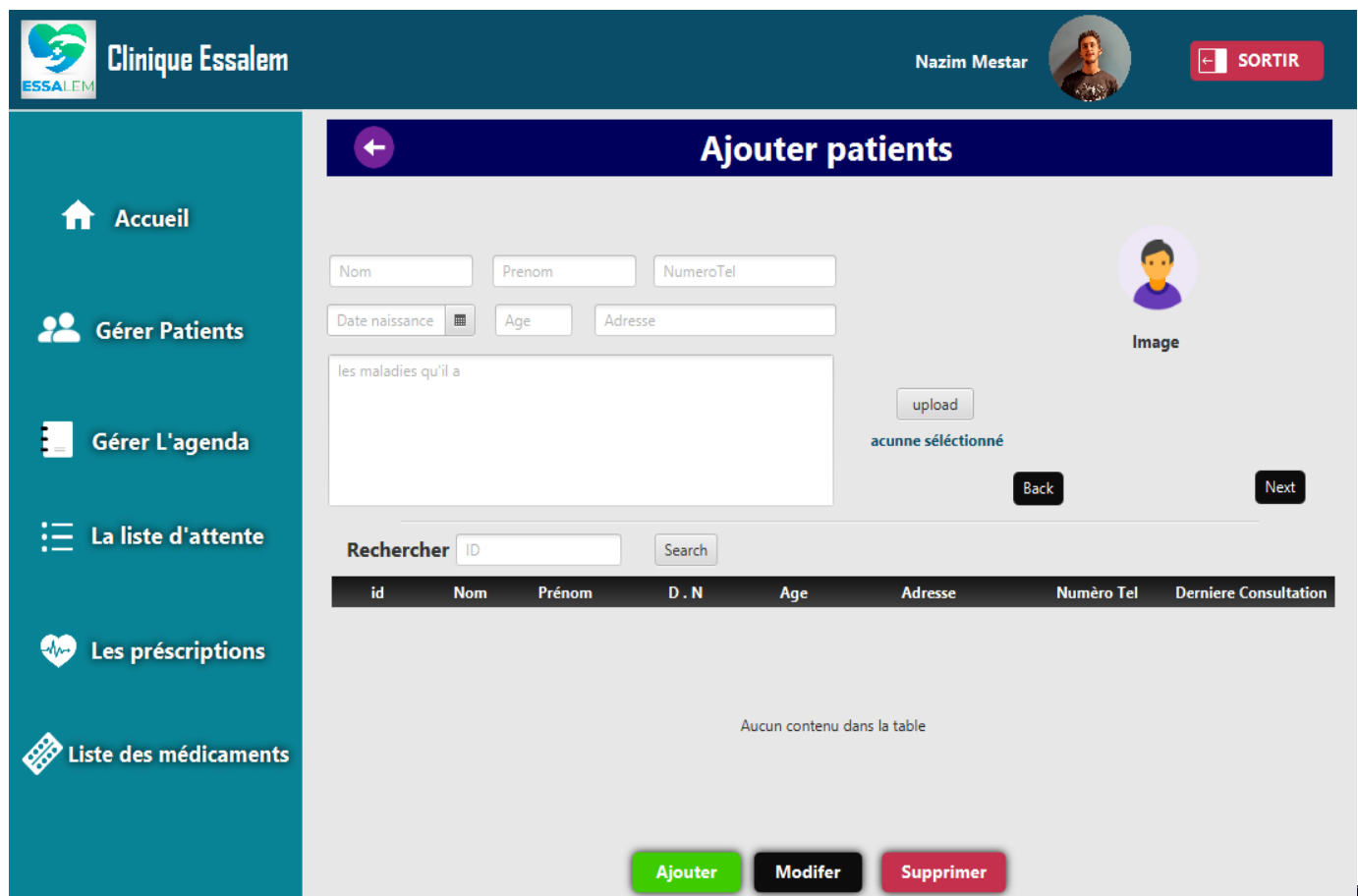
```

1 package controller;
2
3 import java.io.IOException;
4
13 public class GererPatientsController implements Initializable {
14
15
16 @FXML
17 private AnchorPane root;
18
19 private Parent fxml;
20
21 @FXML
22 void ajouter(MouseEvent event) {
23
24     try {
25         fxml = FXMLLoader.load(getClass().getResource("/interfacess/AjouterPatients.fxml"));
26         root.getChildren().removeAll();
27         root.getChildren().setAll(fxml);
28     } catch (IOException e) {
29         // TODO Auto-generated catch block
30         e.printStackTrace();
31     }
32
33 }
34
35 @FXML
36 void dossier(MouseEvent event) {
37
38     try {
39         fxml = FXMLLoader.load(getClass().getResource("/interfacess/DossiersM.fxml"));
40         root.getChildren().removeAll();
41         root.getChildren().setAll(fxml);
42     } catch (IOException e) {
43         // TODO Auto-generated catch block

```

Figure 3.4 : Interface gérer patients.

1.2.3 Ajouter patient : Cette interface permet d'ajouter, modifier ou supprimer les patient d'un rendez-vous et aussi rechercher si un patient existe déjà dans la base de données :



The screenshot shows the 'Ajouter patients' interface. The sidebar on the left contains the following menu items: Accueil, Gérer Patients, Gérer L'agenda, La liste d'attente, Les prescriptions, and Liste des médicaments. The main content area has a title 'Ajouter patients' with a back arrow. Below the title is a form with the following fields: Nom, Prenom, NumeroTel, Date naissance (with a calendar icon), Age, Adresse, and a text area for 'les maladies qu'il a'. To the right of the form is an 'Image' upload section with an 'upload' button and the text 'aucune sélectionné'. Below the form is a search bar with the label 'Rechercher' and a 'Search' button. Below the search bar is a table with the following columns: id, Nom, Prénom, D . N, Age, Adresse, Numéro Tel, and Dernière Consultation. The table is currently empty, with the text 'Aucun contenu dans la table' displayed below it. At the bottom of the main area are three buttons: 'Ajouter' (green), 'Modifier' (black), and 'Supprimer' (red).

```

16 <?import javafx.scene.text.Font?>
17
18 Bind to grammar/schema...
19 <AnchorPane fx:id="root" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="623.0" prefWidth="800.0">
20 <children>
21 <Pane layoutX="20.0" layoutY="14.0" prefHeight="44.0" prefWidth="619.0" style="-fx-background-color: #00005C;" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
22 <children>
23 <Label layoutX="304.0" layoutY="3.0" text="Ajouter patients" textFill="WHITE">
24 <font>
25 <Font name="System Bold" size="27.0" />
26 </font>
27 </Label>
28 <ImageView fitHeight="39.0" fitWidth="41.0" layoutX="19.0" layoutY="2.0" onMouseClicked="#retour" pickOnBounds="true" preserveRatio="true">
29 <image>
30 <Image url="@../images/retour.png" />
31 </image>
32 </ImageView>
33 </children>
34 </Pane>
35 <TableView layoutX="23.0" layoutY="361.0" prefHeight="209.0" prefWidth="819.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
36 <columns>
37 <TableColumn prefWidth="75.0" text="id" />
38 <TableColumn prefWidth="75.0" text="Nom" />
39 <TableColumn prefWidth="75.0" text="Prénom" />
40 <TableColumn prefWidth="117.0" text="D . N" />
41 <TableColumn prefWidth="58.0" text="Age" />
42 <TableColumn prefWidth="162.0" text="Adresse" />
43 <TableColumn prefWidth="113.0" text="Numéro Tel" />
44 <TableColumn prefWidth="138.0" text="Dernière Consultation" />
45 </columns>
46 </TableView>
47 <ImageView fitHeight="79.0" fitWidth="84.0" layoutX="657.0" layoutY="83.0" pickOnBounds="true" preserveRatio="true" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
48 <image>
49 <Image url="@../images/user.png" />
50 </image>
51 </ImageView>
52 </children>
53 </AnchorPane>

```

Figure 3.9 : Interface ajouter patient

Conclusion :

Au cours de ce chapitre nous avons présenté les différents aspects et fonctionnalités de notre application ainsi que quelques interfaces des cas d'utilisation cités au préalable.

Conclusion Générale

En conclusion, ce rapport a présenté une application de gestion de cabinet médical efficace et personnalisable qui répond aux besoins des professionnels de la santé et des patients. Nous avons développé une application qui permet une gestion facile et pratique des dossiers médicaux, des rendez-vous, des prescriptions et des factures. Nous avons surmonté les défis techniques et organisationnels grâce à une planification minutieuse et une collaboration étroite avec les parties prenantes.

Nous sommes convaincus que cette application améliorera la qualité des soins de santé en fournissant une solution efficace pour la gestion des dossiers médicaux et la coordination des soins. Nous espérons que cette étude sera utile pour d'autres développeurs souhaitant développer une application de gestion de cabinet médical ou des applications similaires.

Table de matières

Introduction générale	3
1 L'analyse des besoins	5
1 Problématiques et objectif	6
1.1 Problématiques	6
1.2 Objectifs	7
2 L'analyse des besoins	7
2.1 Identification des acteurs	7
2.1.1 Secrétaire	7
2.1.2 Médecin	7
2.2 Les besoins fonctionnelles	8
2.2.1 coté secrétaire	8
2.2.2 coté médecin	8
2.3 Les besoins non fonctionnelles	8
3 Environnement de développement	9
3.1 Eclipse IDE	9
3.2 PHP myadmin	9
4 Langages de développements	10
4.1 Java	10
4.2 Java fx	10
4.3 CSS	10
4.4 My SQL	11
Conclusion	11

2 Conception et modélisation	12
1 UML	13
2 Diagrammes de cas d'utilisation	14
2.1 cas d'utilisation s'authentifier	15
2.2 cas d'utilisation gérer patient	15
2.3 cas d'utilisation ajouter patient	16
2.4 cas d'utilisation gérer agenda	17
2.5 cas d'utilisation gérer liste d'attente	18
2.6 cas d'utilisation gérer prescription	19
2.7 cas d'utilisation gérer médicaments	20
3 Diagramme de séquences	21
3.1 Diagramme de séquence "s'authentifier"	21
3.2 Diagramme de séquence "récupérer mot de passe"	22
4 Diagramme d'activité	23
5 Diagramme de classe	24
6 Passage au modèle relationnel	25
6.1 Règles du passage du diagramme de classe au modèle logique	25
6.2 Modèle logique des données	25
7 Conclusion	26
3 Implémentation	27
1 Interfaces	28
1.1 Interface d'authentification	28
1.2 Interface secrétaire	29
1.2.1 Accueil	29
1.2.2 Gérer patient	30
1.2.3 Ajouter patient	31
Conclusion	32
Conclusion générale	33
Bibliographie	36

Bibliographie

- [1] <https://www.eclipse.org/> site officiel d'éclipse.
- [2] [Http: //www.siteduzero.com/informatique/tutoriels/concevez-votre-site-webavecphpet-mysql/presentation-des-bases-de-donnees-2](http://www.siteduzero.com/informatique/tutoriels/concevez-votre-site-webavecphpet-mysql/presentation-des-bases-de-donnees-2). O. Classrooms, "présentation des bases de données."
<http://www.siteduzero.com/informatique/tutoriels/concevez-votresiteweb-avecphp-et-mysql/presentation-des-bases-de-donnees-2>.
- [3] A. H. S. T. A. BOUBEKRI Abdelmalek, CHIKOU Youghorta, Conception et réalisation d'une application de gestion de stock des matières premières. Université de Bejaia, 2014.
- [4] <https://docs.oracle.com/javase/8/javafx/get-started/tutorial/jfxoverview.htm>
- [5] Site du W3C : <https://www.w3.org/Style/CSS/>
- [6] Pascal Roques, les cahiers du programmeur UML2 modélisé une application web, Eyrolles, 2007, 4eme édition.