# Contexte général du projet

## 1.1. Introduction

Le secteur de la santé est en pleine transformation numérique, avec une demande croissante d'outils permettant d'optimiser la gestion administrative et médicale des cabinets. L'organisation quotidienne d'un cabinet médical implique une multitude de tâches complexes, allant de la gestion des rendez-vous et des dossiers patients à la facturation et au suivi des stocks de médicaments. Traiter ces informations de manière manuelle ou avec des systèmes disparates est chronophage, source d'erreurs et nuit à la qualité du service patient. Ce projet s'inscrit dans cette volonté de modernisation en proposant le développement d'une application web complète et sur-mesure pour la gestion d'un cabinet médical.

## 1.2. Problématique

Actuellement, de nombreux cabinets médicaux fonctionnent avec des processus manuels (agendas papier, dossiers physiques) ou utilisent des logiciels obsolètes et peu intégrés. Cette situation entraîne plusieurs problèmes majeurs :

Inefficacité administrative : Difficulté à gérer les rendez-vous (annulations, reports), recherche fastidieuse des dossiers patients, facturation manuelle longue et sujette aux erreurs.

Manque de centralisation de l'information : Les données patients (fiches, historiques, documents, ordonnances) sont souvent éparpillées, rendant le suivi médical difficile et fragmenté.

Absence de pilotage de l'activité : Il est complexe d'obtenir une vision claire et instantanée de l'activité du cabinet (statistiques de fréquentation, revenus, performances) pour prendre des décisions éclairées.

Problèmes de communication : La coordination entre les différents acteurs (médecins, secrétaires) peut être entravée par l'absence d'un outil unique et partagé.

La problématique centrale est donc : Comment concevoir et mettre en œuvre une application web centralisée, intuitive et sécurisée pour automatiser et optimiser l'ensemble des processus administratifs et médicaux d'un cabinet, améliorant ainsi tant l'efficacité du personnel que la qualité de la prise en charge des patients ?

## 1.3. Objectif du projet

L'objectif principal de ce projet est de développer une application web de gestion de cabinet médical (CPS) répondant aux besoins spécifiques des professionnels de santé. Cette application aura pour finalités de :

* Centraliser l'information en offrant une base de données unique et sécurisée pour tous les acteurs du cabinet (médecins, secrétaires, administrateurs).
* Automatiser les processus de gestion des rendez-vous, de création des dossiers médicaux et de facturation pour gagner en efficacité et en fiabilité.
* Améliorer le suivi patient en permettant un accès rapide à l'historique médical complet, aux documents associés et en facilitant la génération d'ordonnances.
* Fournir des outils de pilotage via un tableau de bord statistique offrant une vision

# Chapitre 2 : Analyse et conception

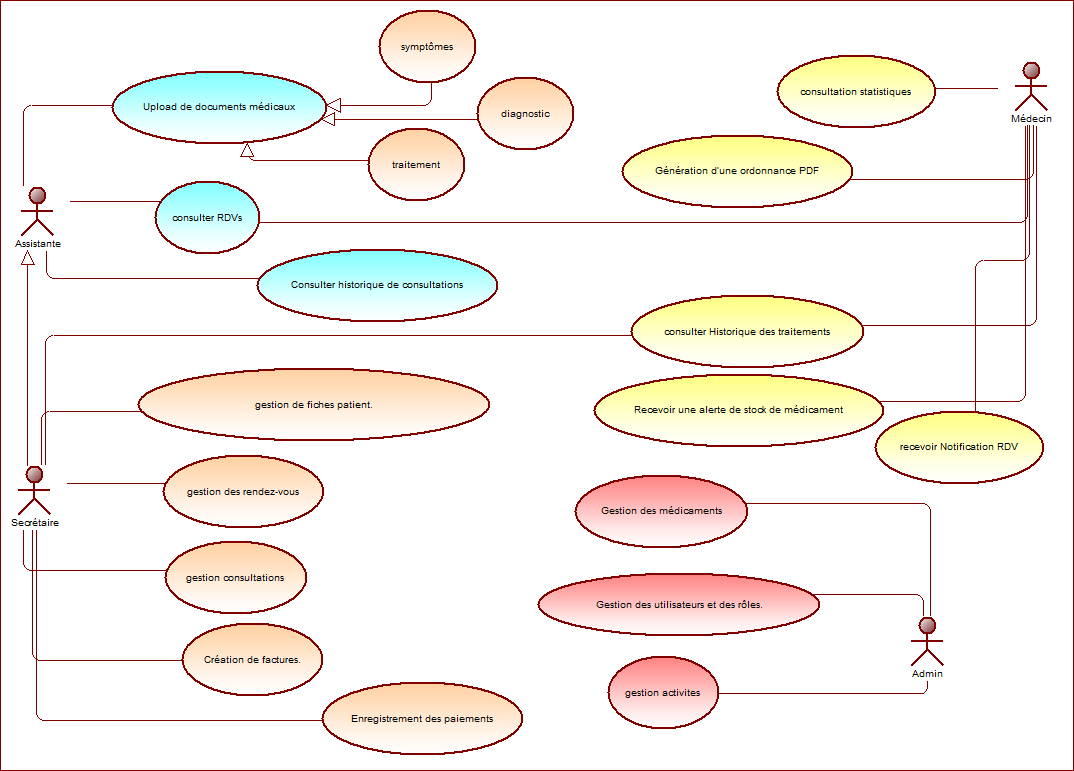
## 2.1. Introduction

Ce chapitre a pour objectif de traduire les besoins fonctionnels énoncés dans le cahier des charges en une modélisation précise et structurée, servant de base à la phase de réalisation. L'analyse permettra de définir le comportement attendu du système du point de vue des utilisateurs, tandis que la conception esquissera l'architecture et les modèles de données qui sous-tendent l'application. La modélisation UML (Unified Modeling Language) sera utilisée comme langage standard pour formaliser cette analyse et assurer une compréhension commune entre toutes les parties prenantes.

## 2.2. Le Langage UML

Le langage de modélisation unifie (Unified Modeling Language en anglais), est un langage standard utilise pour l’analyse, la conception et la modélisation des systèmes orientés objets. Il comprend 14 types qui sont répartis en deux grandes catégories : 1. Structurels comme le diagramme de class et diagramme d’objet. 2. Comportementaux comme le diagramme de cas d’utilisation, diagramme d’activités et diagramme de séquence. Ces diagrammes permettent de représenter les différents aspects des systèmes logiciels, ce qui facilite leur conception et leur compréhension.

## 2.3. Diagramme de cas d’utilisation global



## 2.4 Description détaillée des cas d’utilisations

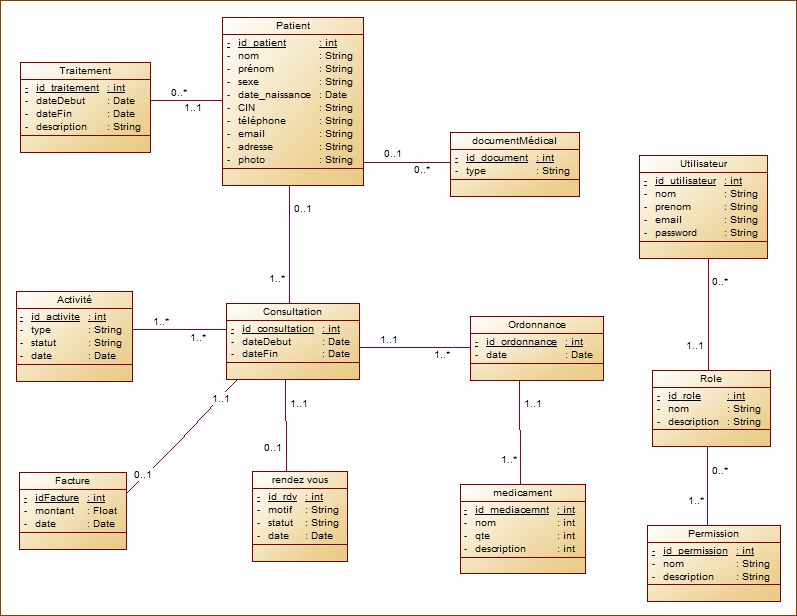
2.4.1 Cas d’utilisation "gestion des consultations"

|  |  |
| --- | --- |
| Nom du cas d’utilisation | Gestion des consultations |
| Acteur principal | Secrétaire ou assistante |
| Objectif | Gérer la consultation |
| Précondition | Utilisateur est authentifié et le patient exist |
| Scénario principal | 1. L’utilisateur clique sur le bouton 2. L’utilisateur sélectionne un patient 3. Le système affiche un formulaire dédié à la saisie de la consultation. 4. Utilisateur renseigne les champs 5. Le médecin valide la consultation en cliquant sur "créer". 6. Le système vérifie l'intégrité des données 7. Le système enregistre la nouvelle consultation dans la base de données |
| Scénario d’erreur | 1. Le système n'enregistre pas la consultation. 2. Il affiche un message d'erreur contextuel précis et surligne les champs en erreur. 3. Le formulaire reste affiché avec les données déjà saisies préservées. |

2.4.2 Cas d’utilisation  " Gestion des Patients"

|  |  |
| --- | --- |
| Nom du cas d’utilisation | Gestion des Patients |
| Acteur principal | Secrétaire |
| Objectif | Gestion des Patients |
| Précondition | Secrétaire est authentifié |
| Scénario principal | Saisie des informations personnelles  Le système enregistre le patient |
| Scénario d’erreur | 1. Le système n'enregistre pas le patient. 2. Il affiche un message d'erreur contextuel précis et surligne les champs en erreur. 3. Le formulaire reste affiché avec les données déjà saisies préservées. |

## 2.4 Diagramme de classe



# Chapitre 3 : Réalisation et Technologies utilisées

## 3.1 Introduction

Ce chapitre présente la phase de réalisation concrète de l'application de gestion de cabinet médical. Il détaille l'écosystème technologique complet qui a été mis en œuvre pour répondre aux exigences fonctionnelles et techniques du cahier des charges. Le choix de ces technologies a été guidé par plusieurs critères : la robustesse, la sécurité, la productivité du développement, la maintenabilité du code et la capacité à créer une interface utilisateur moderne et réactive. L'architecture retenue est de type client-serveur (frontend/backend découplés), permettant une grande scalabilité et une indépendance entre les couches métier et de présentation.

## 3.2. Technologies et Langages utilisées

### 3.2.1 Langages de programmation

**HTML** : Utilisé pour structurer et présenter le contenu de notre application web.

**CSS** : CSS est un langage de feuille de style utilisé pour décrire la présentation et la mise en page de documents écrits en HTML ou XML. CSS décrit la manière dont les éléments HTML ou XML apparaissent sur un écran, un papier ou un autre support. Il vous permet de contrôler la mise en form des éléments HTML ou XML en spécifiant des règles qui dé f inissent la couleur, la taille, la police, l’espacement, position et d’autres aspects visuels de l’élément

**TypeScript** : est un langage de programmation open-source développé par Microsoft en 2012. Il s’agit d’un sur-ensemble typé de JavaScript compilé en JavaScript, c’est-à-dire que TypeScript est considérée comme JavaScript avec quelques fonctionnalités supplémentaires,

Java : est un langage de programmation orienté objet, de haut niveau, fortement typé et multiplateforme. Créé par Sun Microsystems (aujourd'hui propriété d'Oracle)

### 3.2.2 Frameworks, bibliothèques et outils

**Angular** : C’est un cadre de conception d’applications et une plateforme de développement développé par Google et basé sur TypeScript [2]. Angular unous permet de créer des applications monopages (single-page applications) efficaces et sophistiquées.

**Tailwind** : est une bibliothèque CSS qui fonc tionne en ajoutant des classes prédéfinies par Tailwind direc tement aux éléments HTML, ce qui rend le stylage rapide et facile. De plus, nous pouvons également personnaliser ces classes pour répondre à nos besoins.

**PrimeNG** : C’est un ensemble riche de composants UI open source pour Angular, offrant une collection de composants tels que des dialogues de confirmation, des accordéons, des sé quenceurs, etc. PrimeNG fournit également une bibliothèque d’icônes [6].

**XAMPP** : [19] Nous aide à créer facilement un serveur web local sur nos machines, il est livré avec MySQL [12], le système de gestion de base de données que nous utilisons dans notre projet.

**Postman** : Postman est un outil qui nous fait gagner du temps en nous permettant de créer des requêtes HTTP vers le serveur local et de tester les réponses.

**Spring Boot** : Framework principal permettant de créer des applications stand-alone, productives et facilement déployables. Il a simplifié la configuration et le démarrage du projet.

**Spring Security** : Module indispensable pour gérer tout l’aspect sécurité et authentification. Il a été configuré pour gérer l'authentification par formulaire (ou JWT), le hachage sécurisé des mots de passe, l'autorisation basée sur les rôles et la protection contre les attaques CSRF.

**Spring Data JPA (Hibernate)** : Utilisé pour l'accès aux données et la persistance. JPA permet de mapper les objets Java (entités) sur les tables de la base de données relationnelle, simplifiant grandement les opérations CRUD et les requêtes complexes. Hibernate agit comme l'implémentation par défaut.

**Spring Web** : Pour la construction des contrôleurs RESTful qui exposent les API consommées par le frontend Angular.

## 3.3 Interfaces Graphiques