



MISE EN PLACE D'UN SYSTÈME SÉCURISÉ POUR ÉCHANGER ET CONSOMMER LES SERVICES ET LES DONNÉES ENTRE DJANGO REST API ET ANGULAR 11

RÉALISÉ PAR
BROUK SAFAA

ENCADRÉ PAR
ENCADRANT :
PR.NOUH SAID
ENCADRANT PROFESSIONNEL :
(ENTREPRISE INDATABEE)
MR.TAMOUEH AMINE

EXAMINATRICE
PR. MARYAM MAHDAOUI

EXAMINATEUR
PR. IMRAN CHEMSEDDINE IDRISI

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

Dédicaces

C'est avec profonde gratitude et sincères mots, que je dédie ce modeste travail de fin d'année à mes chers parents ; qui ont sacrifié leur vie pour ma réussite et m'ont éclairé le chemin par leurs conseils judicieux.

J'espère qu'un jour, je pourrais leur rendre un peu de ce qu'ils ont fait pour moi, que Dieu leur prête bonheur et longue vie.

Je dédie ce travail à mes frères, ma famille, mes amis, tous mes professeurs et à tous ceux qui sont chers pour moi.

Remerciements

Après avoir rendu grâce à Dieu le Tout-Puissant et le Miséricordieux, je tiens à remercier vivement tous ceux qui ont contribué à la réalisation de cet humble travail

Tout d'abord, j'adresse mes sincères remerciements et mon extrême gratitude à mon encadrant Monsieur NOUH Saïd, pour son encadrement, ses précieux conseils et le soutien inconditionnel qu'il a bien voulu me prêter tout au long de ce stage.

Ensuite, je tiens à remercier Monsieur TAMOUH Amine pour son aide, ses conseils méthodologiques et sa présence tout au long de cette période.

Enfin, je me permets de dédier ce travail à mes parents pour leur soutien moral, et remercie toute personne ayant contribué de près ou de loin pour la réussite de ce projet de fin d'année.

Liste des abréviations

SPA : Single page application

IDE : Integrated Development Environment

API : Application Programming Interface

DB : Data Base

SGBDR : Système Gestion Base Données Relationnelles

HTTP : Hyper Text Transfer Protocol

UML : Unified Modeling Language

URL : Uniform Resource Locator

WWW : World Wide Web

DOM : Document Object Model

ORM : Object Relational Mapping

MVT : model vue Template

MVC : Model Vue Controller

JSON : JavaScript Object Notation

HTML : Hypertext Markup Language

CSS : Cascading Style Sheets

JS : JavaScript

Ng : Angular

NPM : Nodejs Paquet management

ES : ECMAScript

TS : TypeScript

JWT : JSON Web Token

CLI : Command-Line Interface

DI : Dependency injection

Rest : Representational State Transfer

NgRx : Angular Reactive Extensions

Liste des figures

Figure 1 : Organigramme des services administratifs de la FSBM	14
Figure 2 : Organigramme de service commun de la FSBM	15
Figure 3 : Diagramme de Cas d'utilisation de médecin de l'assistante	25
Figure 4 : Scénario de cas d'utilisation<Authentification médecin>	26
Figure 5 : Scénario de cas d'utilisation<Consulter assistante >.....	27
Figure 6 : Scénario de cas d'utilisation<Ajouter assistante >	27
Figure 7 : Scénario de cas d'utilisation<Supprimer assistante >	28
Figure 8 : Scénario de cas d'utilisation<Ajouter consultation >	28
Figure 9 : Scénario de cas d'utilisation<Chercher consultation >.....	29
Figure 10 : Scénario de cas d'utilisation<Consulter consultation >.....	29
Figure 11 : Scénario de cas d'utilisation<Supprimer consultation >	30
Figure 12 : Scénario de cas d'utilisation<Authentification assistante>.....	31
Figure 13 : Scénario de cas d'utilisation<Ajouter dossier >.....	32
Figure 14 : Scénario de cas d'utilisation<Chercher dossier >	33
Figure 15 : Scénario de cas d'utilisation<Consulter dossier >	33
Figure 16 : Scénario de cas d'utilisation<Modifier dossier >.....	34
Figure 17 : Scénario de cas d'utilisation<Supprimer dossier >	35
Figure 18 : Diagramme de classe de l'application	36
Figure 19 : Les classes qui concernent les examens.....	36
Figure 20 : Les classes qui concernent Dossier, Consultation, Traitement médical	37
Figure 21 : Les classes qui concernent les acteurs et leur profil	37
Figure 22 : Architecture physique de l'application.....	38
Figure 23 : Architecture REST	39
Figure 24 : Le modèle MVT de Django Rest Framework	40
Figure 25 : Architecture logique de django Rest Framework.....	40
Figure 26 : Object Relationship Mapping (ORM)	41
Figure 27 : Architecture du projet adopté en django.....	42
Figure 28 : Modèle MVC en Angular12	43
Figure 29 : Single Page Application in Angular	44
Figure 30 : Architecture du projet adopté en Angular	45
Figure 31 : Configuration du setting.py.....	52
Figure 32 : Configuration du fichier urls.py	52
Figure 33 : Problème d'authentification sans token	57
Figure 34 : Obtention du jeton à base d'username, email et password	58
Figure 35 : L'utilisation du jeton d'accès pour communiquer avec l'API.....	59
Figure 36 : Expiration du jeton d'accès	60
Figure 37 : Obtention d'un nouveau jeton d'accès	60
Figure 38 : Interface d'authentification en Angular	61
Figure 39 : Stockage du jeton obtenu par l'api dans SessionStorage in angular.....	61

Liste des Tables

Tableau 1 : Tableau des acteurs	18
Tableau 2 : Tableau des fonctionnalités de médecin.....	21
Tableau 3 : Tableau des fonctionnalités de l'assistante	21

Table des matières

Dédicaces.....	2
Remerciements	3
Liste des abréviations	4
Liste des figures.....	6
Liste des Tables.....	7
Table des matières	8
Introduction.....	10
Chapitre 1 : Présentation de l'organisme d'accueil	12
.I Introduction :.....	13
.II Présentation de la faculté Ben M'Sik :	13
.III Objet et missions :.....	14
.IV Organigramme des services administratifs de la FSBM :	14
.V Conclusion :	15
Chapitre II : Cahier de charge	16
.I Introduction.....	17
.II Problématique	17
.III Objectif de l'application	17
.IV Identification des acteurs.....	17
.V Besoins de l'application.....	18
.V.1 Besoins fonctionnels.....	18
.V.2 Besoins non fonctionnels	19
.VI Back Log.....	20
.VII Conclusion	22
Chapitre III : Analyse et conception	23
.I Introduction.....	24
.II Analyse	24
.II.1 Langage de modélisation UML	24
.II.2 Fonctionnalités de l'application	25
.II.3 Diagramme de séquence.....	26
.II.4 Diagramme de classe.....	35
.III Conception	38

.III.1	Architecture physique	38
.III.2	Architecture logique	38
.IV	Conclusion	45
Chapitre VI : Réalisation		46
.I	Introduction.....	47
.II	Outil et technologies utilisés	47
.II.1	Front end	47
.II.2	Back end	49
.II.3	Les bibliothèques.....	50
.II.4	Server.....	52
.II.5	Base de données.....	53
.II.6	Les logiciels	54
.II.7	IDE.....	56
.III	Interfaces de Site Web et d'application	57
Conclusion et perspectives.....		62
Bibliographie.....		63
Webographie.....		64
.I	Site	64
.II	Vidéos	64

Introduction

Ce projet intitulé « **Gestion d'un cabinet médical** », est réalisé dans le cadre de mon projet de fin d'année en master data science et big data (**DSBD**). C'est un système qui permettra la gestion des dossiers de patients. Cette gestion comprend aussi la manipulation des consultations, des examens que ce soit leur type, traitement médical

Il est présenté à travers l'étude de l'existant. A partir de laquelle nous avons dégagé des nouvelles conditions auxquelles doit répondre le futur système. Ce dernier va mettre en œuvre la gestion d'un cabinet médical. Cet outil devrait mettre l'**organisation** et l'**automatisation** de la gestion d'un cabinet médical, afin d'augmenter la fiabilité, l'efficacité de l'effort humain et faciliter les tâches pénibles au sein du cabinet.

Cette **application web** contient les fonctionnalités suivantes : Gestion des dossiers, gestion de rendez-vous pour l'assistante et pour le médecin, gestion des consultations : (Examen, Traitement médicale, diagnostique finale...).

Ensuite nous avons effectué un premier repérage des besoins fonctionnels et opérationnels afin de mieux définir les nouvelles fonctionnalités. Cela nous a permis par la suite d'identifier les différentes **entités** composant l'environnement d'interactions du système, à savoir : les acteurs humains et les messages échangés entre ces acteurs et le système. Nous avons aussi essayé d'adapter le plus possible notre système avec les besoins des utilisateurs à travers la **modélisation** des cas d'utilisations du système et la description textuelle de ces cas d'utilisations.

Par la suite nous avons présenté l'étude technique qui étale l'architecture logicielle mise en place, ainsi que les outils utilisés dans le développement de notre application telle que :

- a. Le **SGBDR MySQL** comme un serveur de base de données central pour le stockage des données.
- b. Le Framework **Angular12** qui va présenter la couche frontend.
- c. Django **Rest_Framework** qui va présenter la couche backend.
- d. IntelliJ IDEA et Atom comme des **IDE**.
- e. StrarUml et powerAMC pour la réalisation des diagrammes UML.

Puis nous avons présenté la démarche adoptée pour l'analyse et la conception détaillée de l'application où nous avons pu décrire le système cible dans sa globalité ainsi que les différentes fonctionnalités exigées par le médecin et l'assistante.

Chapitre 1 : Présentation de l'organisme d'accueil



.I Introduction :

Dans ce chapitre, nous allons présenter l'organisme d'accueil et nous allons décrire le cadre général du projet ainsi que les objectifs et les missions.

.II Présentation de la faculté Ben M'Sik :

La faculté des Sciences Ben M'Sik a ouvert ses portes en 1984. Elle est rattachée à l'Université Hassan II de Casablanca.

Dès son ouverture, la faculté des Sciences Ben M'Sik a accordé un intérêt particulier au développement de la recherche scientifique parallèlement à sa mission d'enseignement et de formation.

De 1984 à 2003, la FSBM a dispensé diverses formations du type premier cycle (DEUG) et deuxième cycle (Licences ès sciences : Bac+4) dans diverses spécialités.

À partir de 1989, les premières formations de 3ème cycle (CEA et DES) ont commencé grâce à la mise en place d'un certain nombre d'équipes et de laboratoires de recherche.

En 1997, avec la création des UFR, la nouvelle réorganisation de la formation à et par la recherche a donné lieu à de nouveaux regroupements de chercheurs autour de nouvelles thématiques.

Depuis 2003, la faculté des sciences Ben M'Sik dispense une formation modulaire et semestrielle dans le cadre de la réforme pédagogique de l'enseignement supérieur conformément au système LMD.

En 2008, suite à la réorganisation du cycle doctorat, la faculté des Sciences Ben M'Sik a mis en place le Centre d'études doctoral (CED) : « Sciences et applications ». Ce centre est adossé à l'ensemble des structures de recherches accréditées par l'université.

Dans le cadre de la structuration de la recherche, que l'Université Hassan II de Casablanca a entamée à partir de décembre 2015, la faculté des Sciences Ben M'Sik a procédé à une nouvelle organisation et restructuration de ses équipes et

Laboratoires de recherché. C'est ainsi que la recherche à la faculté des sciences s'est organisée en 19 laboratoires.

.III Objet et missions :

L'objectif principal consiste à concevoir et réaliser une application et un site Web permettant la gestion d'une bibliothèque. Une telle application devrait offrir l'intégrité, la sécurité et la confidentialité des données ainsi que de garantir l'accès à l'information au moment opportun. Pour atteindre ces objectifs, nous allons décomposer notre projet en deux parties La première partie concerne l'aspect administratif, et la deuxième partie concerne l'aspect utilisateur.

.IV Organigramme des services administratifs de la FSBM :

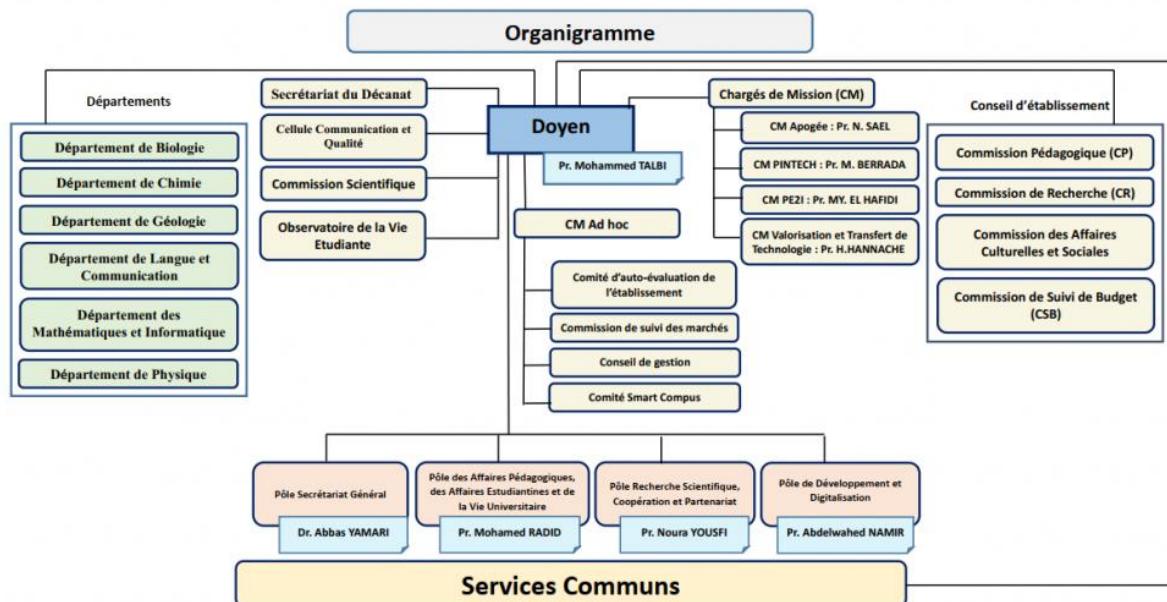


Figure 1 : Organigramme des services administratifs de la FSBM

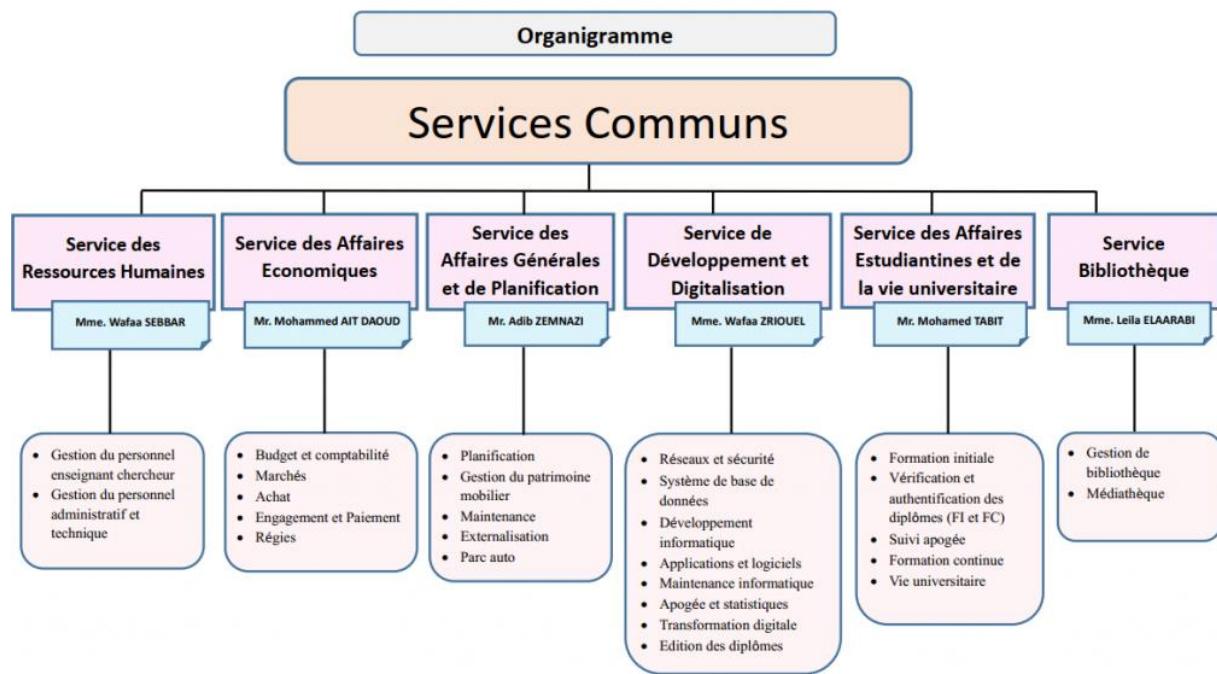


Figure 2 : Organigramme de service commun de la FSBM

.V Conclusion :

Au terme de ce chapitre nous avons présenté l'environnement où nous avons réalisé notre projet.

Chapitre 11 : Cahier de charge



.I Introduction

Ce chapitre introduit le contexte général dans lequel le projet s'intègre, en Présentant à la fois son thème principal, la problématique générale et les objectifs du projet, ainsi que la conduite du projet. Identifier tous les acteurs et les fonctionnalités de notre futur système, et ceci en recensant les besoins fonctionnels et les besoins non fonctionnels.

.II Problématique

La gestion d'un cabinet médical est un emploi complexe. Cependant, les premières années de Construction des cabinets médicaux et les médecins s'orientaient à un objectif bien déterminé, celui-ci se résumait à comment travailler avec les patients à l'aide de moyens faciles et rapides. Mais malgré tous les efforts des médecins il reste des difficultés, parmi lesquelles : L'archivage des dossiers des patients sur des feuilles peut engendrer leur perte et perdre beaucoup de temps pour trouver un dossier ; la modification d'un élément dans un dossier peut obliger le médecin à refaire un nouveau dossier. A cause de ces difficultés, les médecins préfèrent s'approprier des bénéfices liés à l'informatisation de la gestion de leurs cabinets

.III Objectif de l'application



On a mis comme objectifs de ce projet la création d'une application web qui permet de gérer un cabinet médical pour concrétiser les connaissances informatiques dans le domaine de médecine.

.IV Identification des acteurs

Un acteur représente l'abstraction d'un rôle joué par des entités externes. Dans notre application on distingue principalement deux acteurs qui sont les suivants :

	<p>Médecin : Son activité principale est de débuter avec le motif de consultation, ensuite il passe à l'étape qui est les signes cliniques à travers le remplissage d'un formulaire. Après, il procède à l'examen clinique du patient et prescrit des examens paracliniques qui contiennent les différents examens. Enfin la diagnostique final. Dans toutes ces activités la partie la plus importante est le suivi du dossier médical pour chaque patient, le médecin tient à jour un dossier qui englobe les données médicales sous forme de consultations qui présentent l'historique des interventions, les résultats d'analyses médicales diverses subies par le patient, les médicaments qui lui ont été prescrits.</p>
	<p>Assistante : L'assistante a un rôle multiple dans le cabinet médical. Elle peut créer le dossier d'un patient (le nom, le prénom, la date de naissance ...) s'il s'agit d'une première consultation, ou de préparer son dossier antérieur pour l'entrevue avec le médecin. Ainsi que, elle est responsable sur toute importation et exportation d'un examen.</p>

Tableau 1 : Tableau des acteurs

.V Besoins de l'application

.V.1 Besoins fonctionnels

Le futur système doit permettre au médecin de :

- **Authentification**
- **Gestion assistante** : Ajouter, Modifier, Consulter, Détail, Chercher, Supprimer
- **Gestion profil** : Consulter profil, Modifier coordonnées
- **Gestion dossier** : Créer, Modifier, Consulter, Détail, Chercher, Supprimer
- **Gestion consultation** : Ajouter, Consulter, Détail, Chercher, Supprimer
- **Gestion Examens** : Ajouter, Modifier, Consulter, Détail, Supprimer, Importer, Exporter
- **Gestion Traitement** : Ajouter, Modifier, Consulter, Supprimer
- **Gestion Ordonnance** : Ajouter, Modifier, Détail, Supprimer, Exporter
- **Statistique**

Le futur système doit permettre à l'assistante de :

- ❖ **Authentification**
- ❖ **Gestion profil** : Consulter profil, Modifier coordonnées
- ❖ **Gestion dossier** : Créer, Modifier, Consulter, Détailier, Chercher, Supprimer
- ❖ **Gestion consultation** : Ajouter, Consulter, Détailier, Chercher, Supprimer
- ❖ **Gestion Examens** : Ajouter, Modifier, Consulter, Détailier, Supprimer, Importer, Exporter
- ❖ **Gestion Traitement** : Ajouter, Modifier, Consulter, Supprimer
- ❖ **Gestion Ordonnance** : Ajouter, Modifier, Détailier, Supprimer, Exporter
- ❖ **Statistique**

.V.2 Besoins non fonctionnels

Les besoins non fonctionnels sont des exigences qui ne concernent pas spécifiquement le comportement du système mais plutôt identifient des contraintes internes et externes du système. Les principaux besoins non fonctionnels de notre application se résument dans les points suivants :

❖ La maintenabilité

Le code doit être compréhensible par simple lecture, notamment en respectant les règles de gestion et les normes de développement.

❖ La rapidité

L'accès à la base de données doit être souple et rapide.

❖ Exploitation

Les impacts en termes de performances doivent être pris en compte lors de développement, ainsi que la consommation des ressources qui doit être minimisée.

❖ Capacité fonctionnelle et convivialité

Les composants développés doivent respecter les spécifications fournies. Le système doit être facilement utilisable et disposer d'interfaces conviviales, notamment par le respect des règles d'ergonomie.

❖ Sécurité

Chaque utilisateur, pour accéder à l'application, est obligé de s'authentifier par un nom d'utilisateur et un mot de passe. Il ne pourra accéder qu'aux pages qui lui sont permises par son profil ou les droits d'accès qui lui sont affectés par l'administrateur.

VI Back Log

Acteur	Fonctions	
		Authentification
 Médecin	Gestion assistante	Ajouter assistante
		Consulter assistante
		Détailler assistante
		Chercher assistante
		Supprimer assistante
	Gestion profil	Consulter profil
		Modifier coordonnées profil
	Gestion dossier	Créer dossier
		Modifier dossier
		Chercher dossier
		Détailler dossier
		Consulter dossier
		Supprimer dossier
Gestion Consultation	Gestion Consultation	Ajouter consultation
		Chercher consultation
		Consulter consultation
		Détailler consultation
		Supprimer consultation
	Gestion Examens	Ajouter examen
		Consulter examen
		Détailler examen
		Modifier examen
		Exporter examen
	Supprimer examen	
	Ajouter traitement	

	Gestion Traitement	Modifier traitement
		Consulter traitement
		Supprimer traitement
	Gestion Ordonnance	Ajoute ordonnance
		Modifier ordonnance
		Détailler ordonnance
		Exporter ordonnance
		Supprimer ordonnance
	<i>Statistique</i>	

Tableau 2 : Tableau des fonctionnalités de médecin

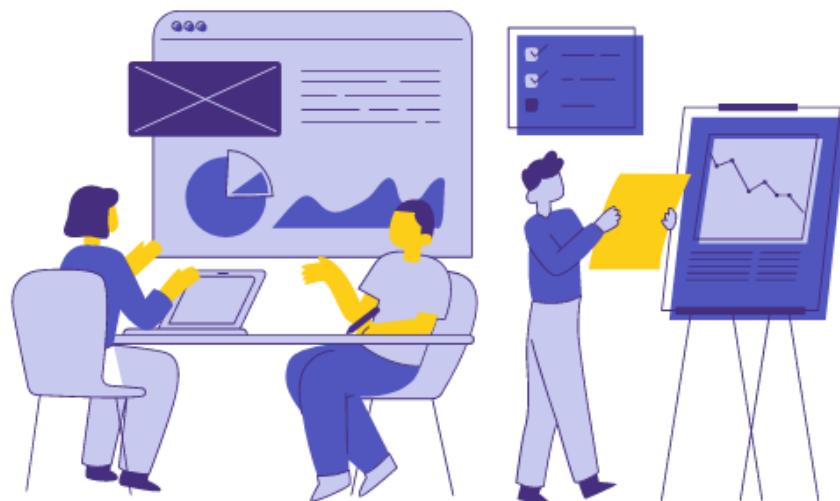
Acteur	Fonctions	
 Assistante	Authentification	
	Gestion profil	Consulter profil
		Modifier coordonnées profil
	Gestion dossier	Créer dossier
		Modifier dossier
		Chercher dossier
		Supprimer dossier
		Consulter dossier
		Détailler dossier
	Gestion Consultation	Chercher consultation
		Consulter consultation
		Détailler consultation
	Gestion Examens	Consulter examen
		Détailler examen
		Importer examen
		Exporter examen
	Gestion Traitement	Consulter traitement
	Gestion Ordonnance	Détailler ordonnance
		Exporter
		<i>Statistique</i>

Tableau 3 : Tableau des fonctionnalités de l'assistante

.VII Conclusion

Ce chapitre s'agissait d'une présentation du contexte générale du projet, ainsi que la conduite du projet. L'utilisation de cette méthode nous a beaucoup aidé pour bien cerner le projet, afin d'optimiser l'application et une présentation des acteurs du système, une vue globale sur les besoins fonctionnels, avec les autres besoins non fonctionnels qu'on doit respecter en cours de notre développement de l'application.

Chapitre III : Analyse et conception



.I Introduction

Cette partie met la lumière sur la phase d'étude et de conception en amont. Dans un premier temps, nous présenterons la méthodologie adoptée pour la conception ainsi que la démarche qui l'accompagne, puis nous présenterons les différents diagrammes modélisant à la fois les aspects fonctionnels, la structure statique et le fonctionnement dynamique de l'application.

.II Analyse

.II.1 Langage de modélisation UML

Langage de Modélisation (UML) est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet. UML propose de décrire un système à l'aide de 9 diagrammes :

- Diagrammes de cas d'utilisation
- Diagrammes d'objets
- Diagrammes de classes
- Diagrammes de composants
- Diagrammes de déploiement
- Diagrammes de collaboration
- Diagrammes de séquence
- Diagrammes d'états-transitions
- Diagrammes d'activités



Pour le cas de notre application, les diagrammes utilisés sont :

Le diagramme de cas d'utilisation : il permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système.

Le diagramme de séquence : il présente la vue dynamique du système. L'objectif du diagramme de séquence est de représenter les interactions entre les objets en indiquant la chronologie des échanges. Cette représentation se réalise par cas d'utilisation.

Le diagramme de classes : permet de donner une vue statique du système en matière de classes d'objets et des relations entre classes.

II.2 Fonctionnalités de l'application

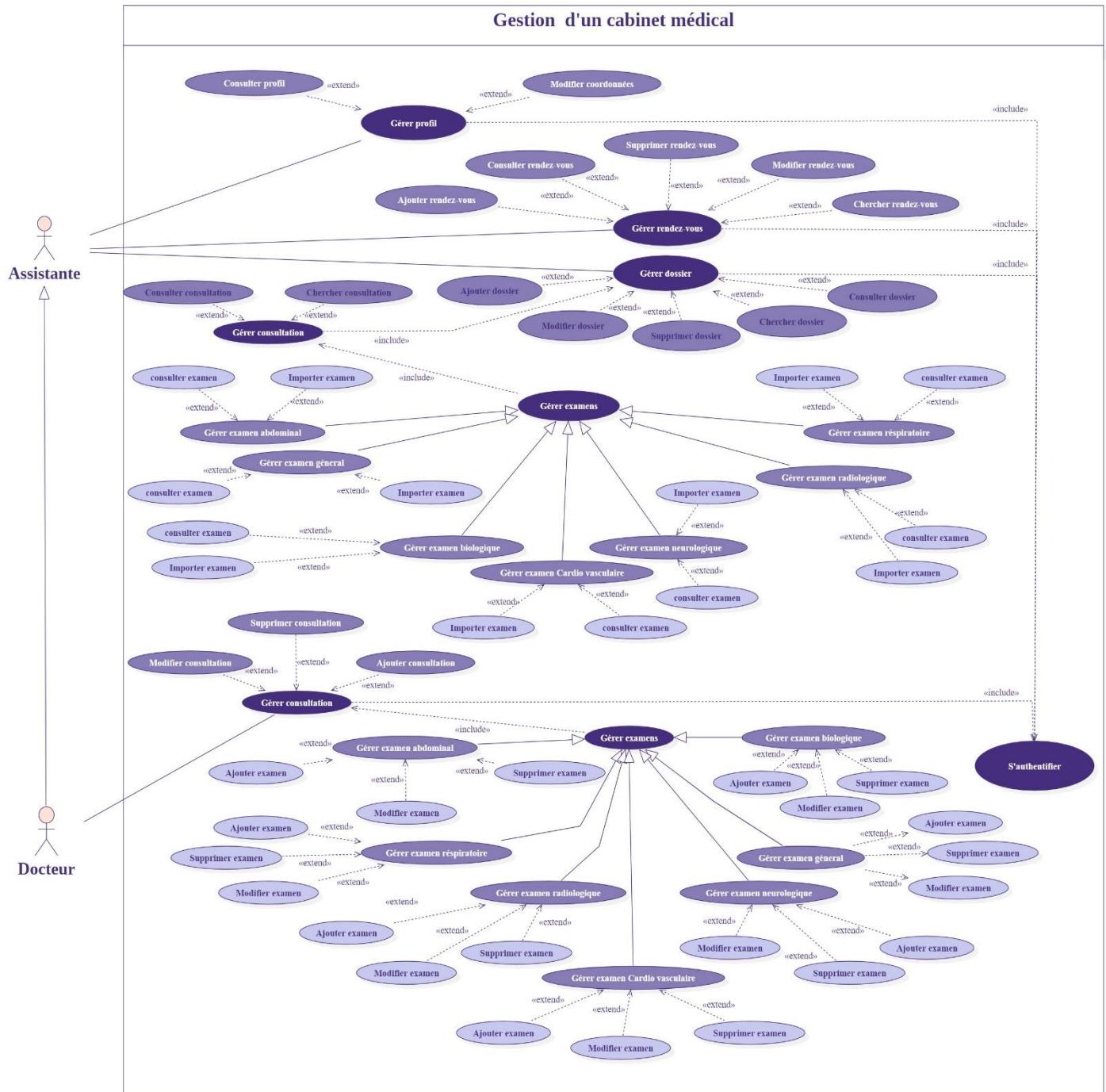


Figure 3 : Diagramme de Cas d'utilisation de médecin de l'assistante

.II.3 Diagramme de séquence

❖ Médecin

.II.3.1 Scénario de cas d'utilisation<Authentification>

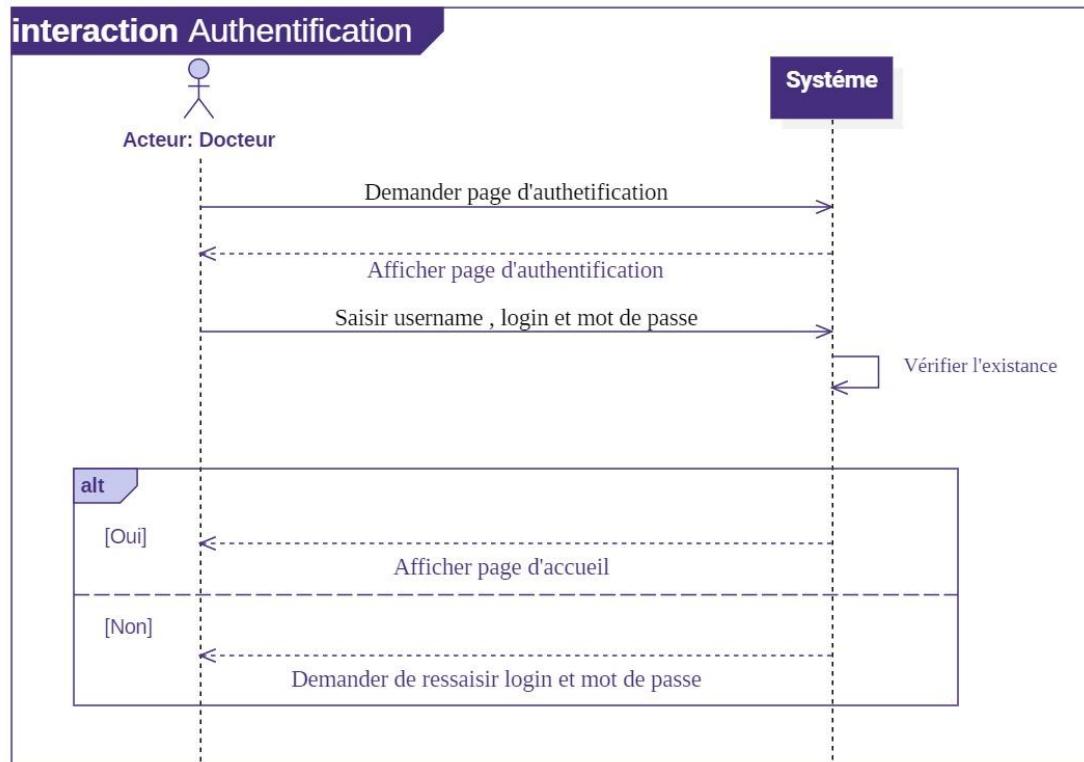


Figure 4 : Scénario de cas d'utilisation<Authentification médecin>

.II.3.2 Scénario de cas d'utilisation<Gérer assistante>

.II.3.2.1 Scénario de cas d'utilisation<Consulter assistante >

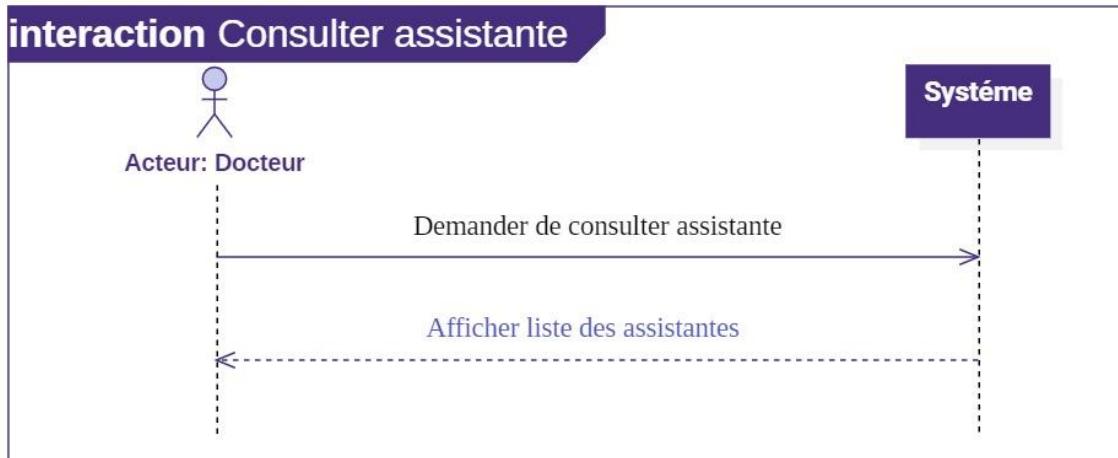


Figure 5 : Scénario de cas d'utilisation<Consulter assistante >

.II.3.2.2 Scénario de cas d'utilisation<Ajouter assistante >

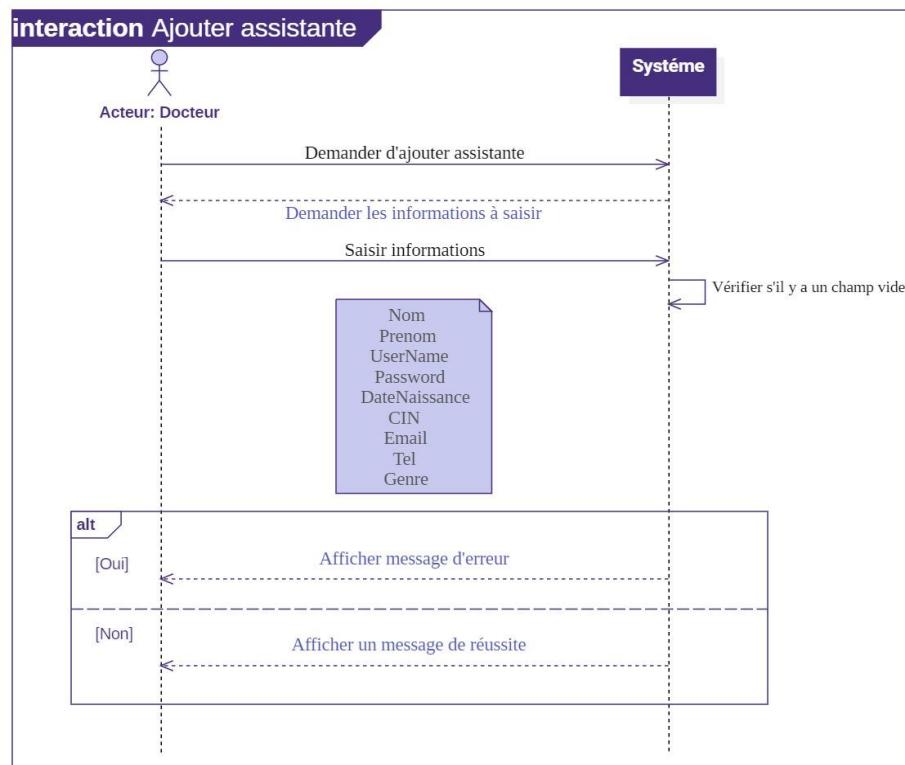


Figure 6 : Scénario de cas d'utilisation<Ajouter assistante >

.II.3.2.3

Scénario de cas d'utilisation<Supprimer assistante >

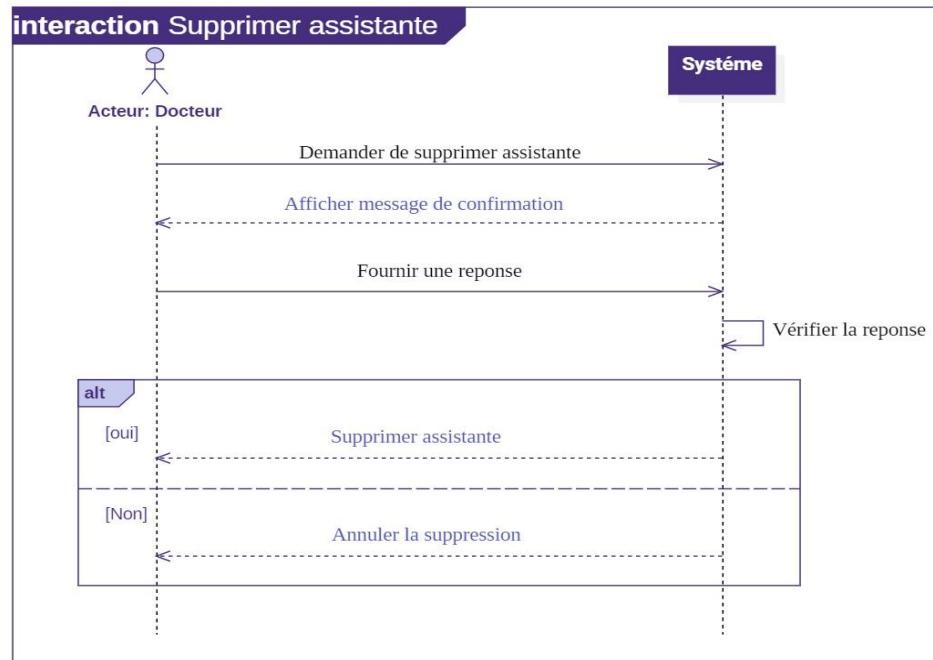


Figure 7 : Scénario de cas d'utilisation<Supprimer assistante >

.II.3.3

Scénario de cas d'utilisation<Gérer consultation>

.II.3.3.1

Scénario de cas d'utilisation<Ajouter consultation>

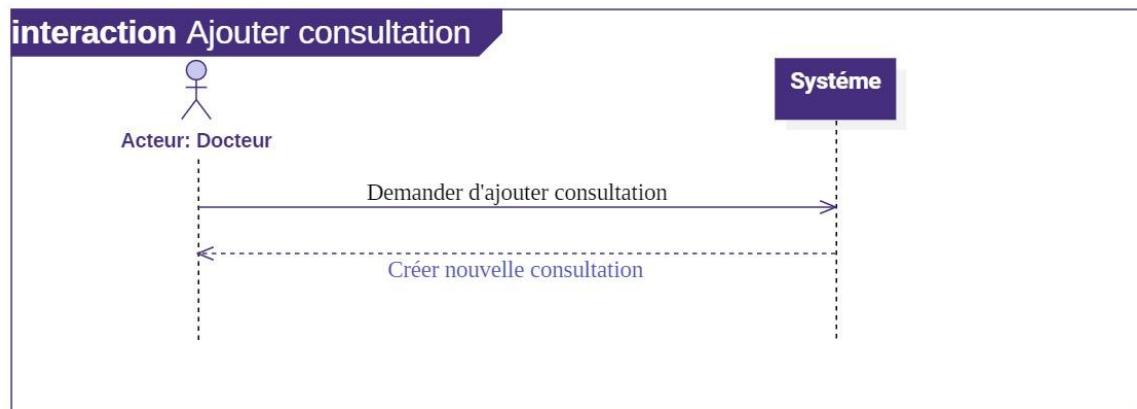


Figure 8 : Scénario de cas d'utilisation<Ajouter consultation >

.II.3.3.2

Scénario de cas d'utilisation<Chercher consultation >

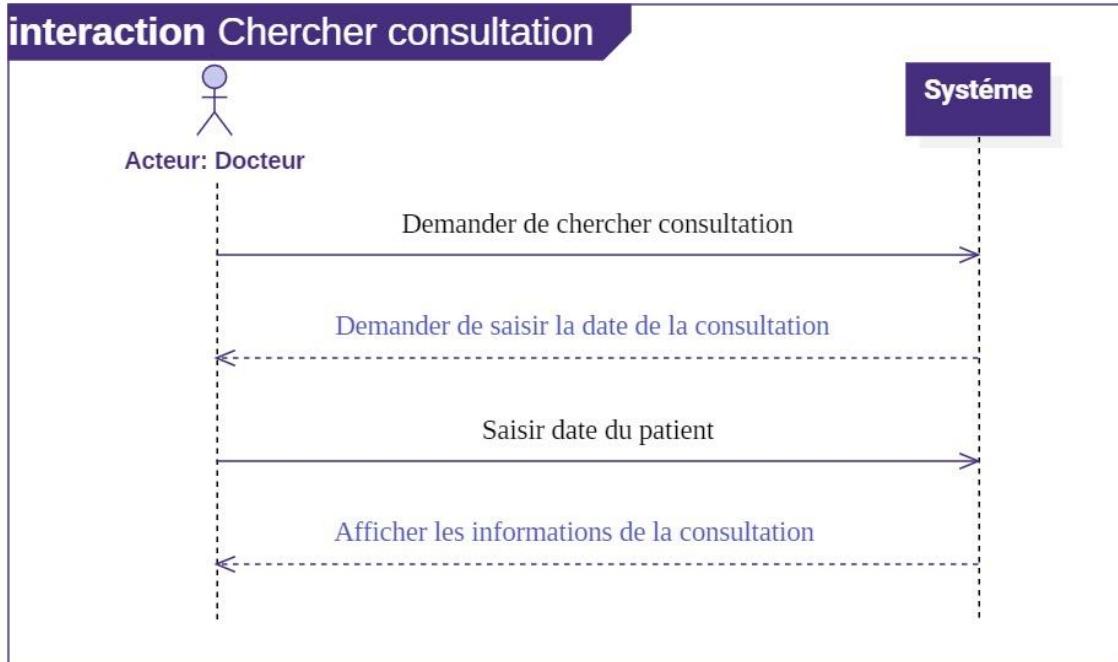


Figure 9 : Scénario de cas d'utilisation<Chercher consultation >

.II.3.3.3

Scénario de cas d'utilisation<Consulter consultation >

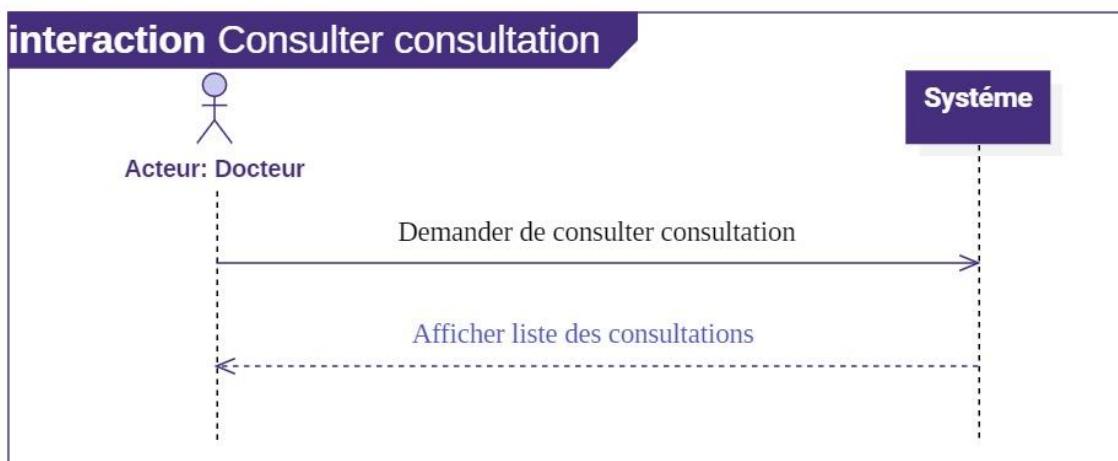


Figure 10 : Scénario de cas d'utilisation<Consulter consultation >

.II.3.3.4

Scénario de cas d'utilisation<Supprimer consultation >

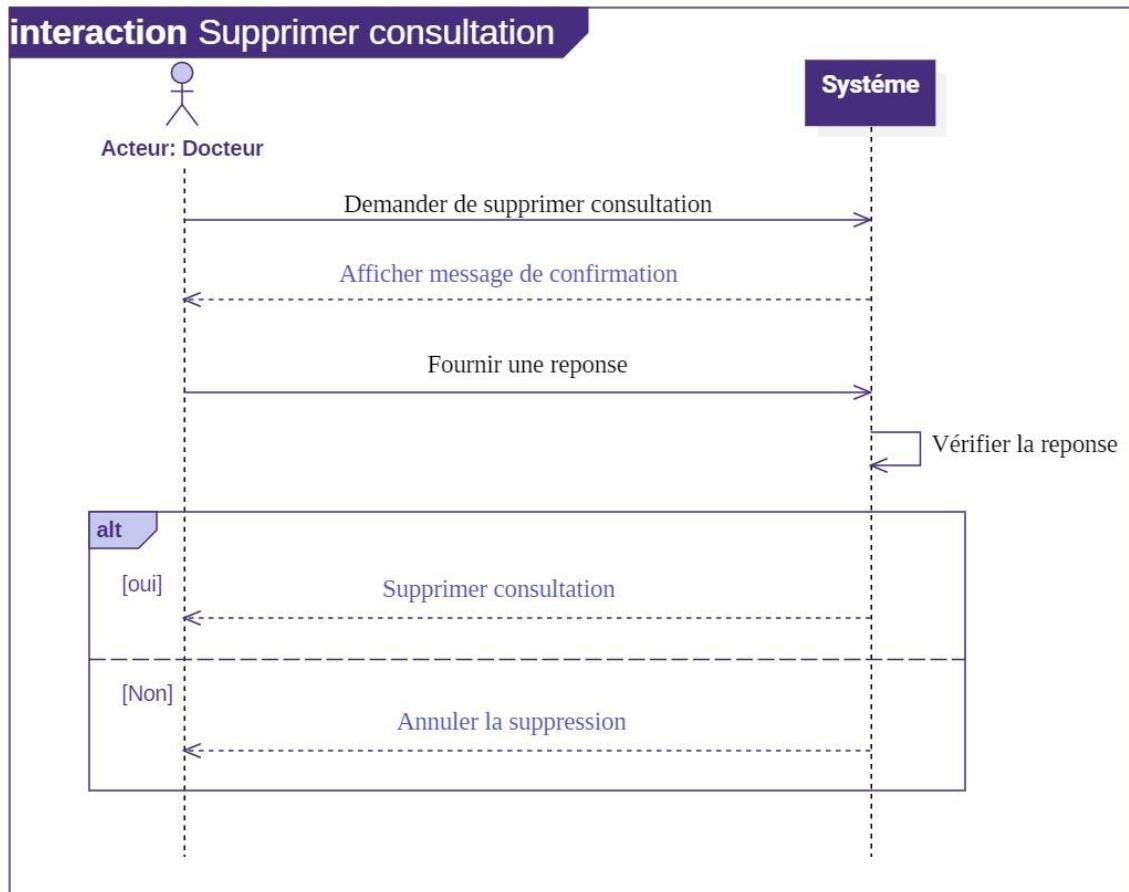


Figure 11 : Scénario de cas d'utilisation<Supprimer consultation >

❖ Assistante

.II.3.4 Scénario de cas d'utilisation<Authentification>

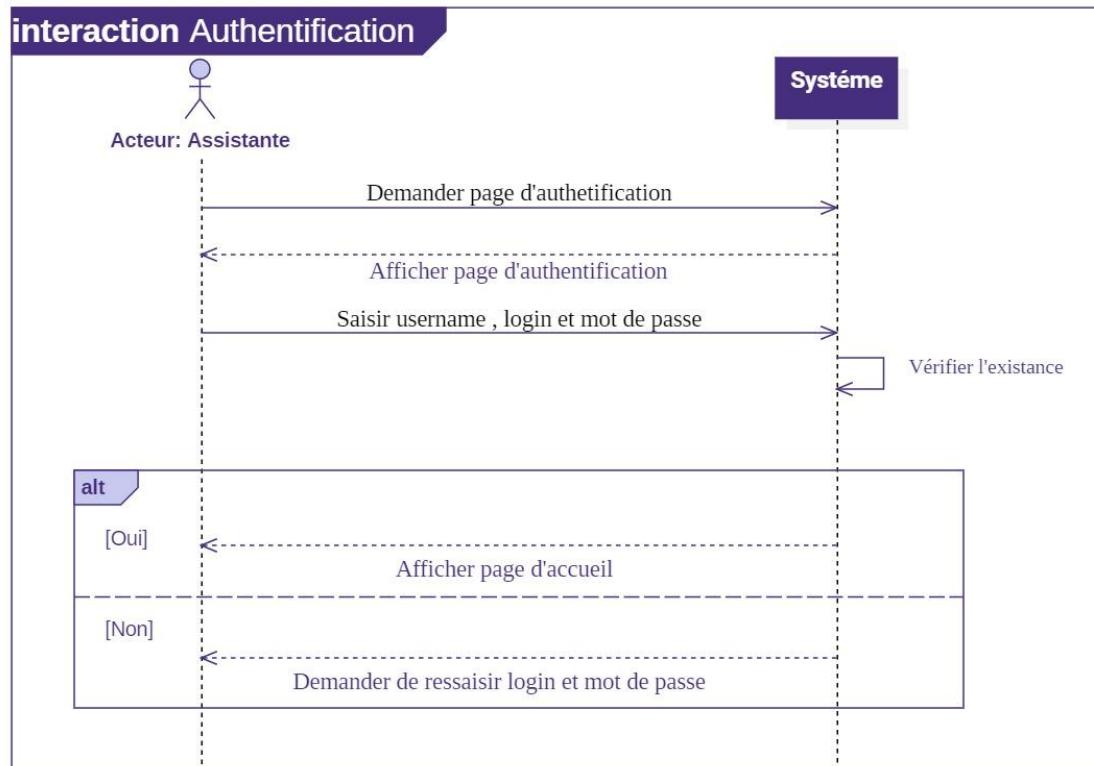


Figure 12 : Scénario de cas d'utilisation<Authentification assistante>

.II.3.5 Scénario de cas d'utilisation<Gérer dossier>

.II.3.5.1 Scénario de cas d'utilisation<Ajouter dossier >

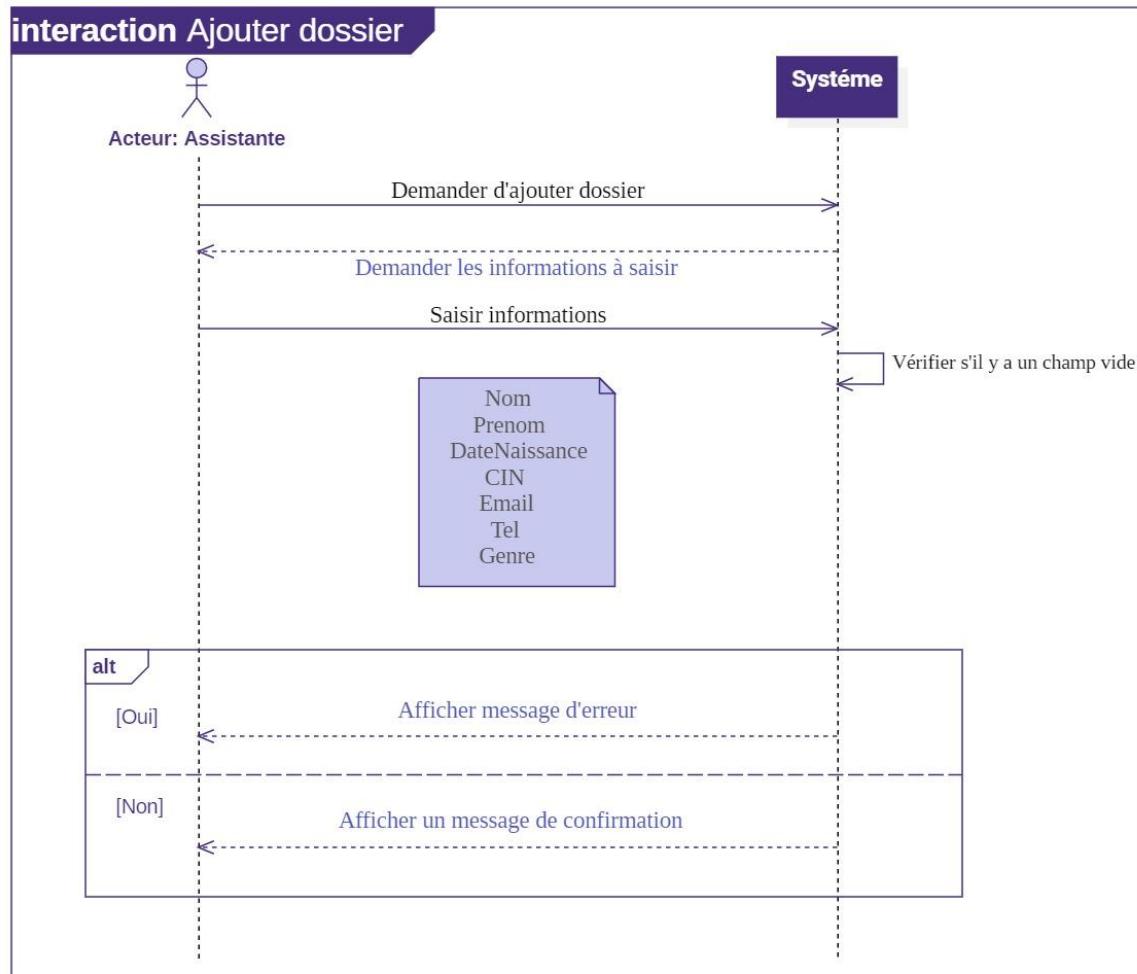


Figure 13 : Scénario de cas d'utilisation<Ajouter dossier >

.II.3.5.2

Scénario de cas d'utilisation<Chercher dossier >

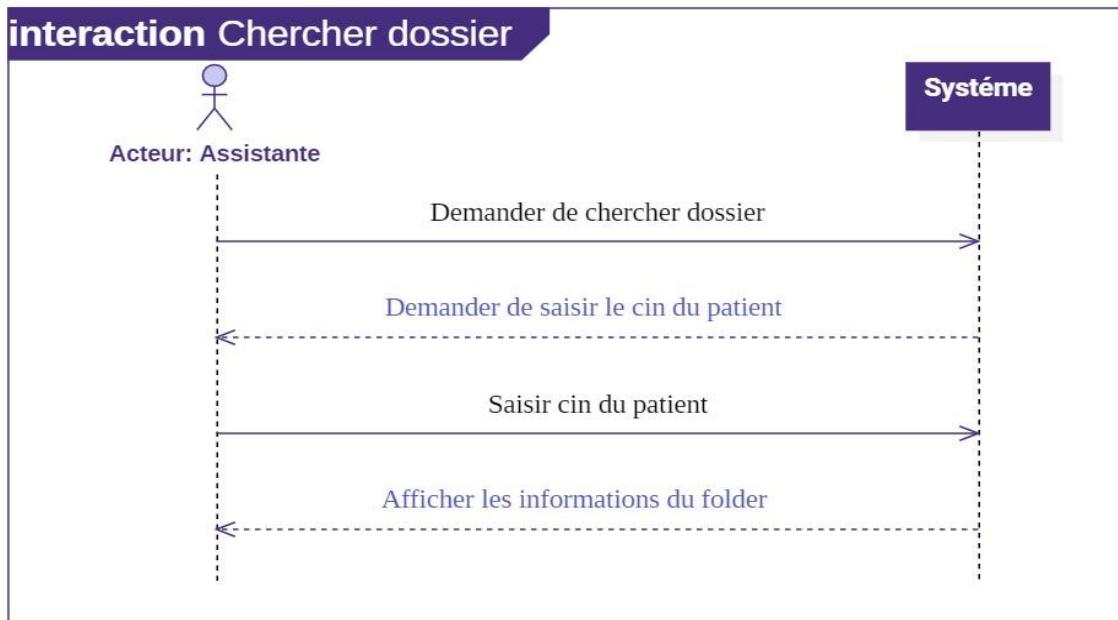


Figure 14 : Scénario de cas d'utilisation<Chercher dossier >

.II.3.5.3

Scénario de cas d'utilisation<Consulter dossier >

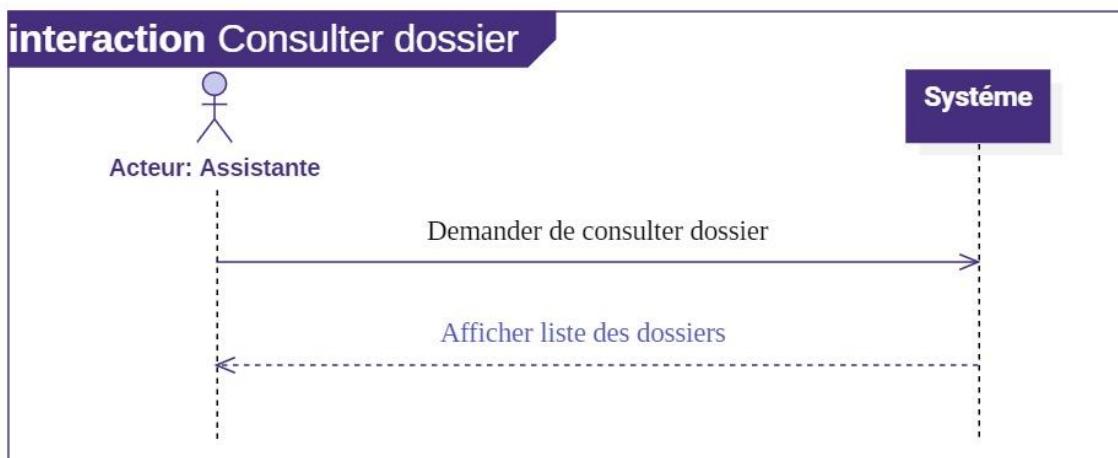


Figure 15 : Scénario de cas d'utilisation<Consulter dossier >

.II.3.5.4

Scénario de cas d'utilisation<Modifier dossier >

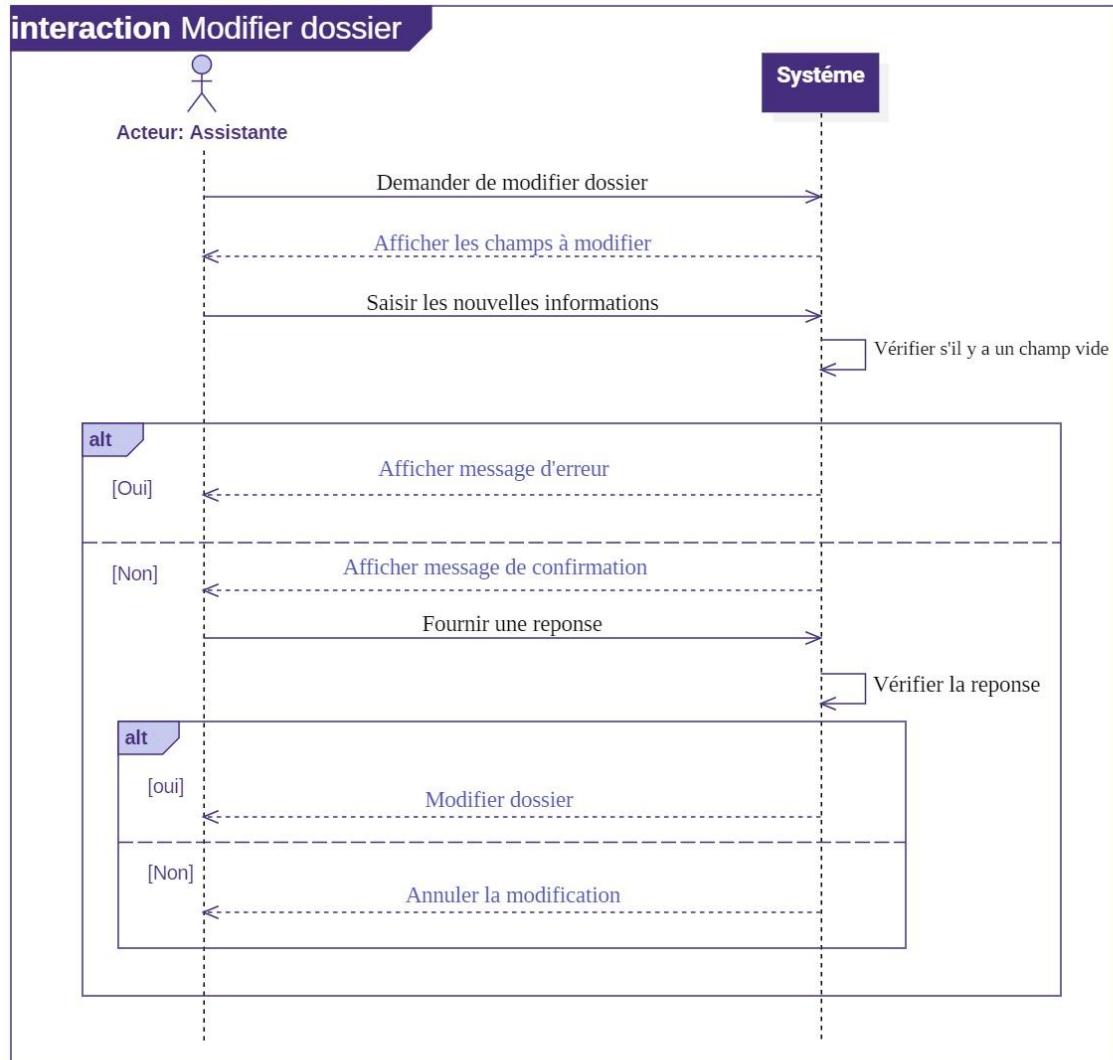


Figure 16 : Scénario de cas d'utilisation<Modifier dossier >

.II.3.5.5

Scénario de cas d'utilisation<Supprimer dossier >

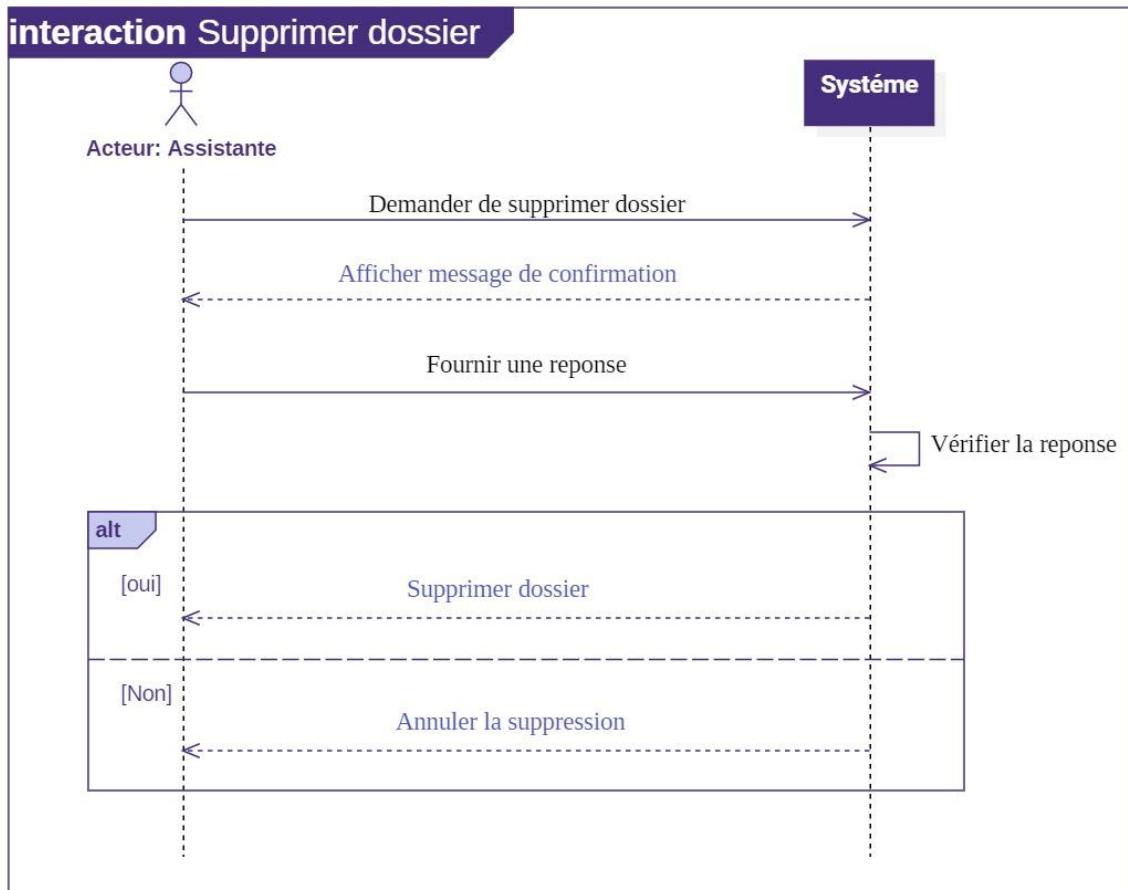


Figure 17 : Scénario de cas d'utilisation<Supprimer dossier >

.II.4 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 les différentes relations entre celles-ci. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques.

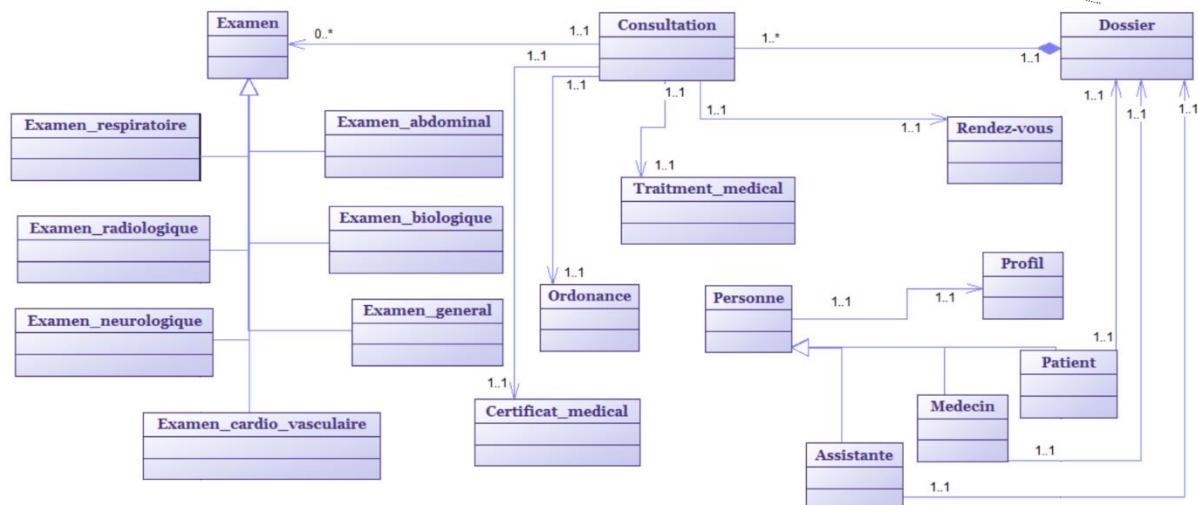


Figure 18 : Diagramme de classe de l'application

II.4.1 Les classes qui concernent les examens

Examen_biotique	Examen_abdominal	Examen_radiologique	Douleur_Masse
<ul style="list-style-type: none"> - id : int - gaj : String - hbac1 : String - uree : String - creat : String - nfshb : String - nfsgb : String - nfpq : String - ferritinemie : String - asat : String - alat : String - tsh : String - ct : String - tg : String - ldl : String - hdl : String - ecbu : String - vs : String - crp : String - autres : String 	<ul style="list-style-type: none"> - id : int - douleur : Douleur_Masse - masse : Douleur_Masse - ascite : boolean - tympanisme : boolean - distension_tympanique : boolean - hepatomegalie : boolean - splenomegalie : boolean - tr : int 	<ul style="list-style-type: none"> - id : int - thorax : String - rx_standard : String - tdm : String - irm : String 	<ul style="list-style-type: none"> - HD : String - HG : String - EPIG : String - FD : String - FG : String - O : String - HYPAG : String
Examen_respiratoire	Examen_general	Pleuresie	Pneumothorax
<ul style="list-style-type: none"> - id : int - rales_sibilants : boolean - ralents_ronflants : String - pleuresie : Pleuresie - pneumothorax : Pneumothorax - crepitauts : boolean 	<ul style="list-style-type: none"> - id : int - ta : String - fc : String - fr : String - poids : String - taille : String - conjonctive : Conjonctive 	<ul style="list-style-type: none"> - Dplus : String - G : String - B : String 	<ul style="list-style-type: none"> - D : String - G : String - B : String
Examen_cardio_vasculaire	Examen_neurologique	Conjonctive	Rythme
<ul style="list-style-type: none"> - id : int - rythme : Rythme - retricissement : Retricissement 	<ul style="list-style-type: none"> - id : int - normal : boolean - vertige : boolean - autre : String 	<ul style="list-style-type: none"> - normocolorees : String - decolores : String - icteriques : String 	<ul style="list-style-type: none"> - regulier : int - irregulier : int

Figure 19 : Les classes qui concernent les examens

.II.4.2 Les classes qui concernent Dossier, Consultation, Traitement médical,

Traitement_medical		Ordonnance	Dossier
- <u>id</u>	: int	- <u>id</u> : int	- <u>id</u> : int
- anti_biotique	: String	- date : Date	- date_creation : Date
- anti_inflammatoire	: String	- contenu : String	
- antalgique	: String		
- anti_spasmodique	: String		
- ipp	: String		
- anti_emetique	: String		
- anti_diarrheique	: String		
- laxatif	: String		
- Attribut_1charbon_active o	: String		
- anti_histaminique	: String		
- corticotode	: String		
- anti_hta	: String		
- b_bloquant	: String		
- anti_cholesterolemique	: String		
- ado	: String		
- insuline	: String		
- levothyrox	: String		
- dimazol	: String		
- magnesium	: String		
- auxiolytique	: String		
- anti_depresseur	: String		
- vitamine	: String		
- anti_mucosique	: String		
- ctc_local	: String		
- autres	: String		

Certificat_medical		Consultation
- <u>id</u>	: int	- <u>id</u> : int
- date	: Date	- date : Date
- contenu	: String	

Rendez-vous
- <u>id</u> : int
- date : Date
- heure : Date

Figure 20 : Les classes qui concernent Dossier, Consultation, Traitement médical

.II.4.3 Les classes qui concernent les acteurs et leur profil

Personne	Profil	<<enumeration>>
- <u>id</u> : int	- <u>id</u> : int	Role
- nom : String	- status : int	- assistante : String
- prenom : String	- url_image : String	- docteur : String
- cin : String		- patient : String
- login : String		
- tel : long		
- password : String		
- genre : Genre		
- role : Role		
- date_naissance : Date		
- date_creation : Date		

Assistante	Medecin	Patient	<<enumeration>>
- <u>id</u> : int	- <u>id</u> : int	- <u>id</u> : int	Genre
			- Homme : String
			- Femme : String

Figure 21 : Les classes qui concernent les acteurs et leur profil

.111 Conception

.111.1 Architecture physique

Voici l'architecture de l'application que nous allons construire :

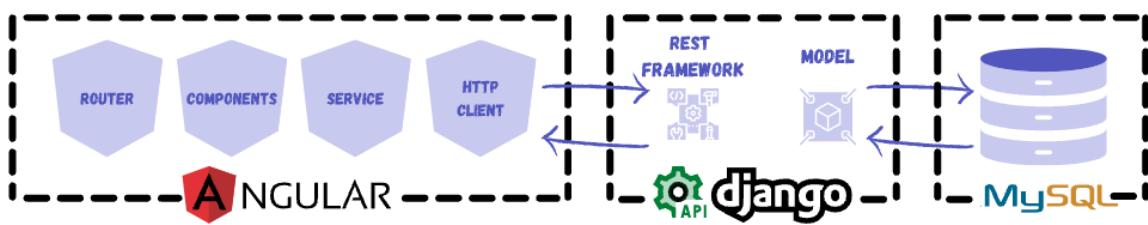


Figure 22 : Architecture physique de l'application

- **Django** exporte REST APIs à l'aide de Django Rest Framework et interagit avec la base de données MySQL à l'aide du modèle Django.
- **Angular Client** envoie des requêtes HTTP et récupère les réponses HTTP à l'aide du module HttpClient, affiche les données sur les composants et pour naviguer entre les pages, il y a Angular Router.
- **Serveur de base de données -MySQL** : MySQL est un serveur de bases de données relationnelles SQL, très rapide, multithread, robuste et multiutilisateurs. MySQL est un logiciel libre développé sous double licence GPL (Général public licence) et licence commerciale. Il est le serveur de base de données le plus utilisé dans le monde. Il fonctionne sur beaucoup de plates-formes différentes et il est accessible en utilisant plusieurs langages de programmation.

.111.2 Architecture logique

.111.2.1 Django Rest Api pour Back-end

.111.2.1.1 Architecture rest

Pour rendre les données accessibles via un site, il existe les méthodes dites REST (représentational state Transfer). Il s'agit d'un ensemble de conventions et de bonnes pratiques à respecter et non d'une technologie entière.

L'information de base, dans une architecture REST, est appelée ressource. Toute information qui peut être nommée est une ressource : la description d'un bâtiment, la liste des arrêts de bus ou n'importe

quel concept. Dans un système hypermédia, une ressource est tout ce qui peut être référencé par un lien.

L'interface entre les composants est simple et uniforme. En HTTP, cette interface est implantée par les verbes **GET, PUT, POST, DELETE**, . . . qui permettent aux composants de manipuler les ressources de manière simple. Par exemple quand un agent voudra récupérer la liste des arrêts de bus depuis l'application, il passera par la méthode GET qui lui retournera les ressources voulues.

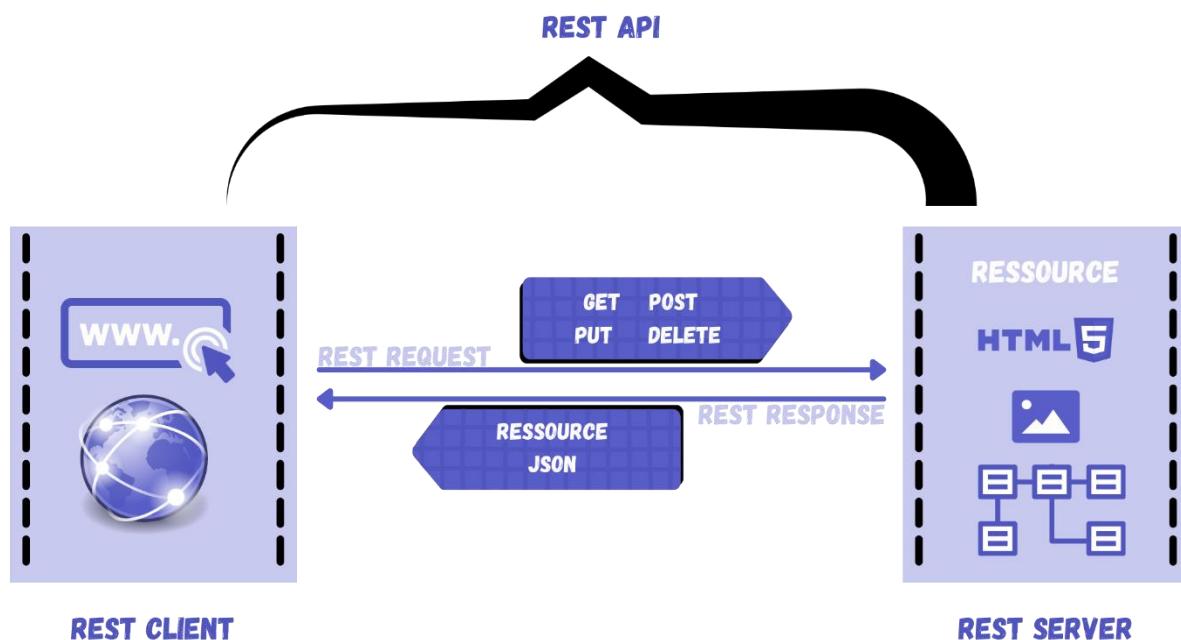


Figure 23 : Architecture REST

.III.2.1.2 Le modèle MVT

Django se base sur le modèle MVT, l'�égèrement diffrent du modle MVC, le Framework gre lui-mme le contrleur et laisse place au Template.

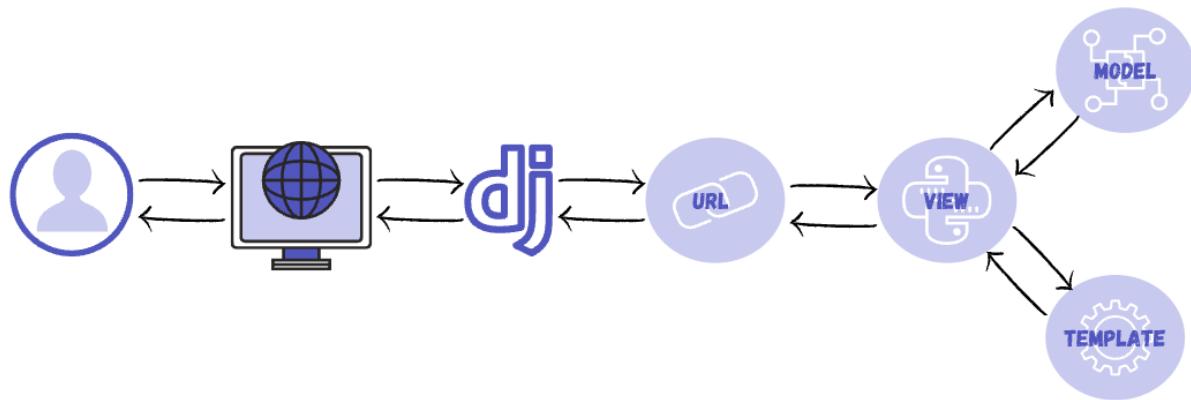


Figure 24 : Le modèle MVT de Django Rest Framework

Lors d'une requête venant de l'utilisateur, le Framework Django gère lui-même, via les règles de routage défini par le développeur (**urls.py**), de charger la bonne vue (**views.py**) correspondante au résultat voulu. Une Template est un fichier HTML qui sera récupéré par la vue pour être envoyé à l'utilisateur, mais entre cette étape, Django va exécuter la Template comme si c'était un fichier de code. Inclus dans les Templates, le Framework propose l'utilisation des structures conditionnelles, des boucles, des variables... afin d'avoir une grande liberté de développement.

Le schéma suivant montre l'architecture de notre application Django CRUD Rest Apis avec base de données MySQL :

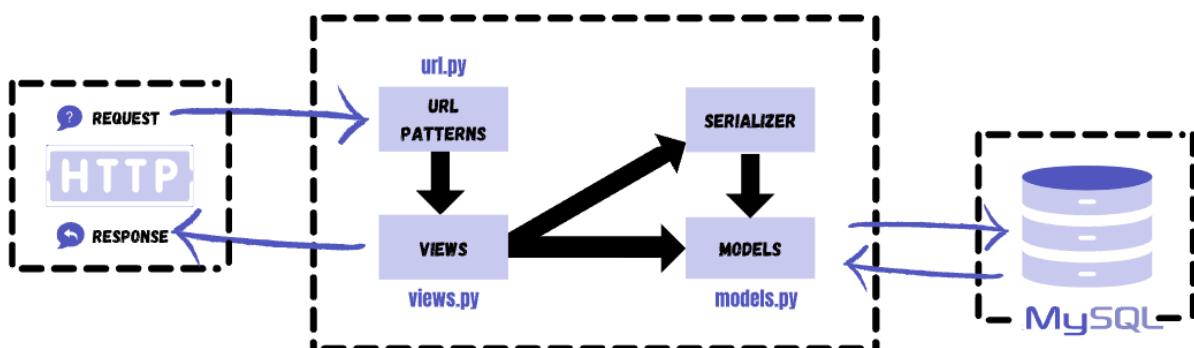


Figure 25 : Architecture logique de django Rest Framework

- Les requêtes HTTP seront mises en correspondance avec les modèles d'URL et transmises aux vues
- Views traite les requêtes HTTP et renvoie les réponses HTTP (à l'aide de Sérialiser)

- Le sérialiseur sérialise/désérialise les objets du modèle de données
- Les modèles contiennent des champs et des comportements essentiels pour les opérations CRUD avec la base de données MySQL

.III.2.1.3 **ORM :**

L'Object Relationship Mapping (ou ORM) est une technique de programmation qui permet de gérer une base de données orientée comme une base orientée objet, et cela de manière transparente pour le développeur. Dans notre cas, l'ORM permet également d'interagir plus rapidement avec la base de données, en passant par des requêtes standardisées et optimisées.

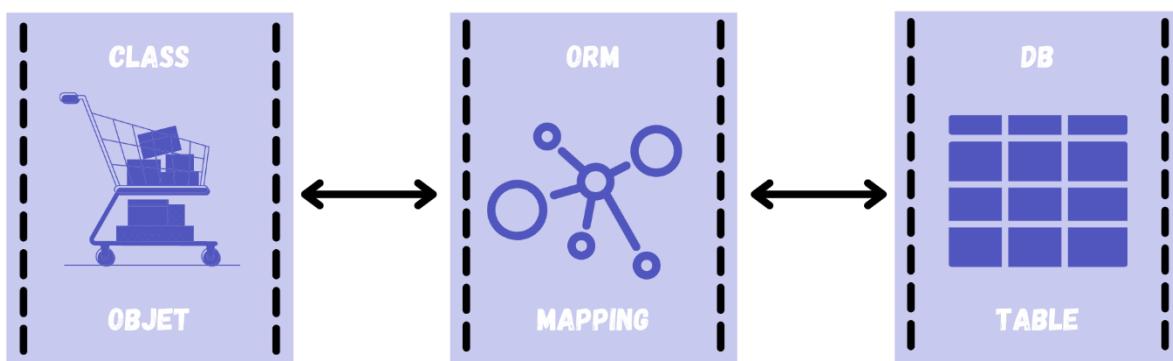


Figure 26 : Object Relationship Mapping (ORM)

.III.2.1.4 **Architecture du projet adopté**

Voici l'architecture de l'application au niveau d'API django Rest Framework :

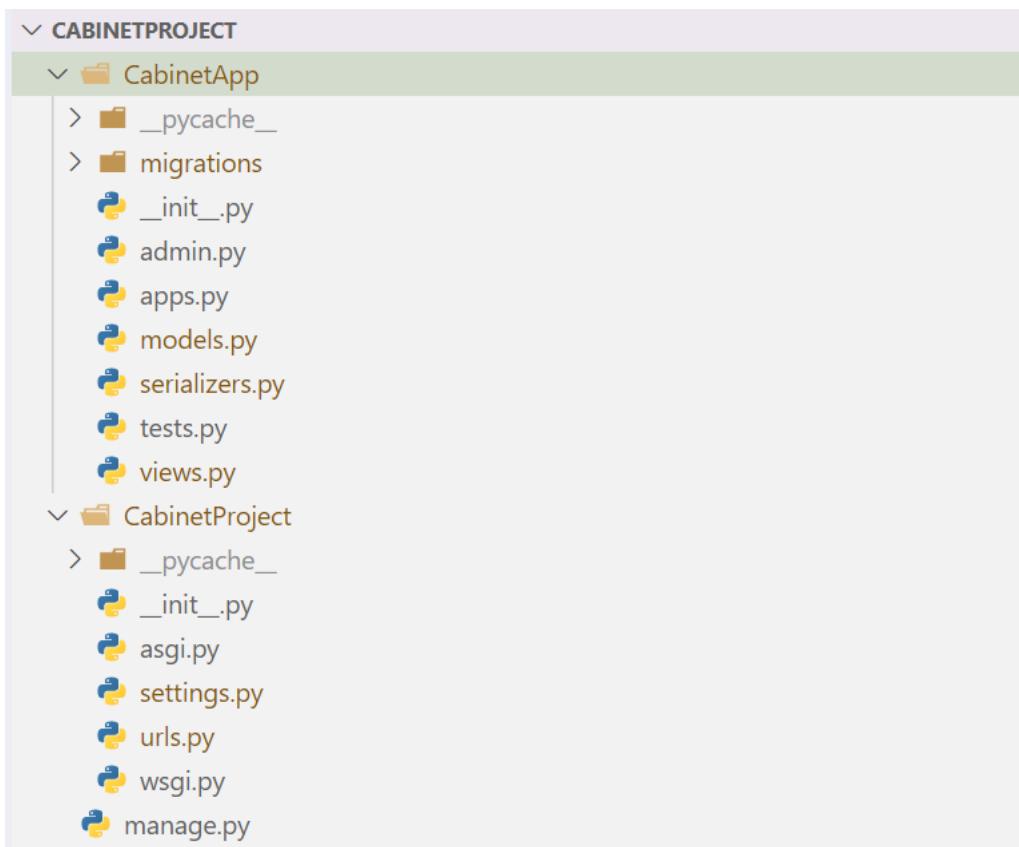


Figure 27 : Architecture du projet adopté en django

Un projet Django (**CabinetProject**) est composé d'un ou plusieurs applications (**CabinetApp**). Ces applications peuvent être d'ailleurs déplacées dans d'autres projets. C'est une des forces de Django : les développeurs peuvent s'échanger des applications compatibles très facilement.

Chaque projet Django doit contenir les fichiers suivants :

- ❖ **manage.py** : est le point d'entrée de commande de management du projet. Il permet de créer des applications via la commande « **python manage.py startapp CabinetApp** », démarrer le serveur via la commande « **python manage.py runserver** ».
- ❖ **urls.py** : est une sorte de table de routage. Pour chaque URL définie on peut l'envoyer vers une fonction choisie.
- ❖ **settings.py** : qui est comme son nom l'indique un fichier de configuration. Comme souvent en python le fichier de configuration n'est pas un simple fichier texte mais un script qui déclare des variables d'environnement. C'est ici que vous pourrez indiquer quelle base de

données utiliser ou quelles applications ajouter au projet, où se situent les fichiers statiques, etc.

.III.2.2 *Angular pour front-end*

.III.2.2.1 *Le modèle MVC*

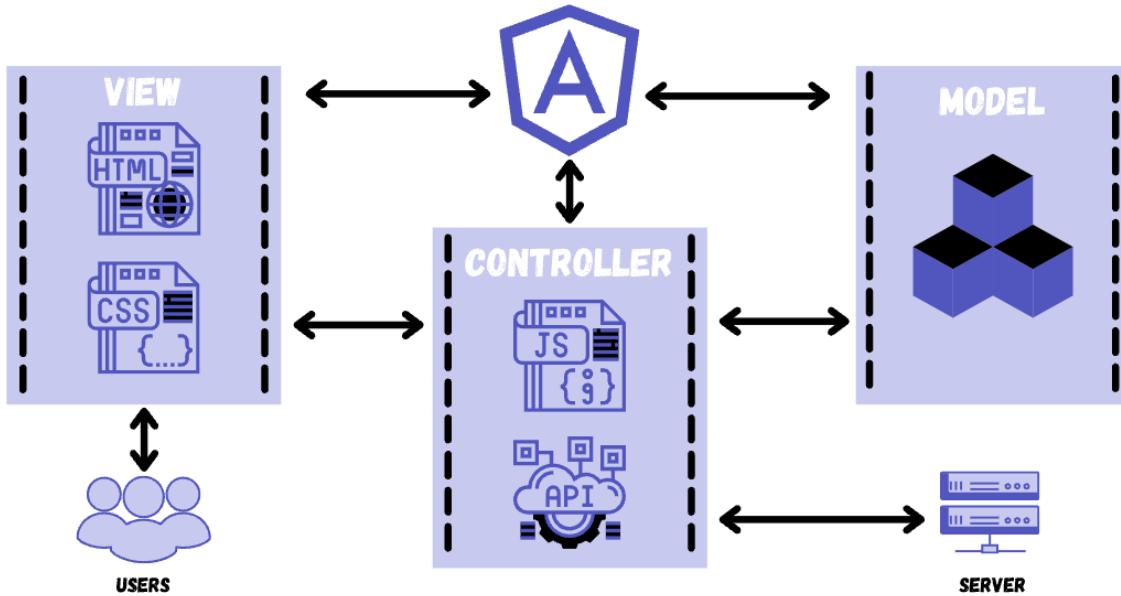


Figure 28 : Modèle MVC en Angular12

AngularJS est un Framework frontend basé sur l'architecture Modèle-Vue-Contrôleur (MVC) dans laquelle les couches sont séparées en trois types d'objets :

- ❖ **Modèle** : Présente le comportement de l'application : traitements des données, interactions avec la base de données, etc.
- ❖ **Vue** : Correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, bouton, . . .).
- ❖ **Contrôleur** : Prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer.

.III.2.2.2 *SPA*

Une SPA (Single Page Application) est une application monopage qui exécute principalement l'interface utilisateur dans le navigateur et qui communique avec le serveur via des services web. Les avantages d'une SPA par rapport à une application web standard sont :

- Fluidifier l'expérience utilisateur en évitant de charger une nouvelle page à chaque action de l'utilisateur ;
- Permettre de s'exécuter localement sur le navigateur de l'utilisateur.

Que vous pouvez le constater au niveau du fonctionnement d'Angular à travers le schéma suivant :

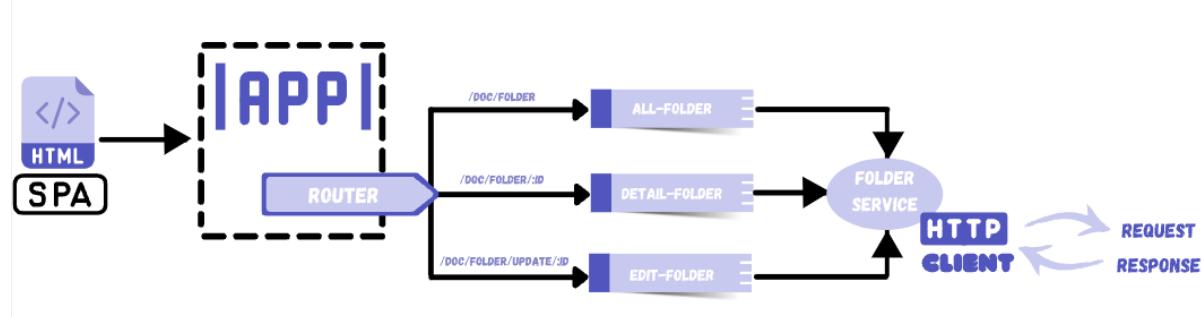


Figure 29 : Single Page Application in Angular

Chaque projet Angular contient **index.html**, qui est la page unique de notre application. Lorsque vous exécutez « **ng serve** », la CLI appelle le compilateur TypeScript. Le compilateur génère des fichiers JavaScript. La CLI les regroupe ensuite et ajoute les éléments de script nécessaires au fichier index.html (en utilisant Webpack dans les coulisses).

Le routeur dans Angular a un objectif simple : la création d'URL significatives reflétant l'état de notre application, et chaque URL sachant quel composant doit être initialisé et inséré dans la page. Ce sera exécuter tout cela sans rafraîchir la page et sans déclencher une nouvelle requête vers notre serveur du backend : c'est tout l'intérêt d'avoir une Single Page Application.

.III.2.2.3 Architecture du projet adopté

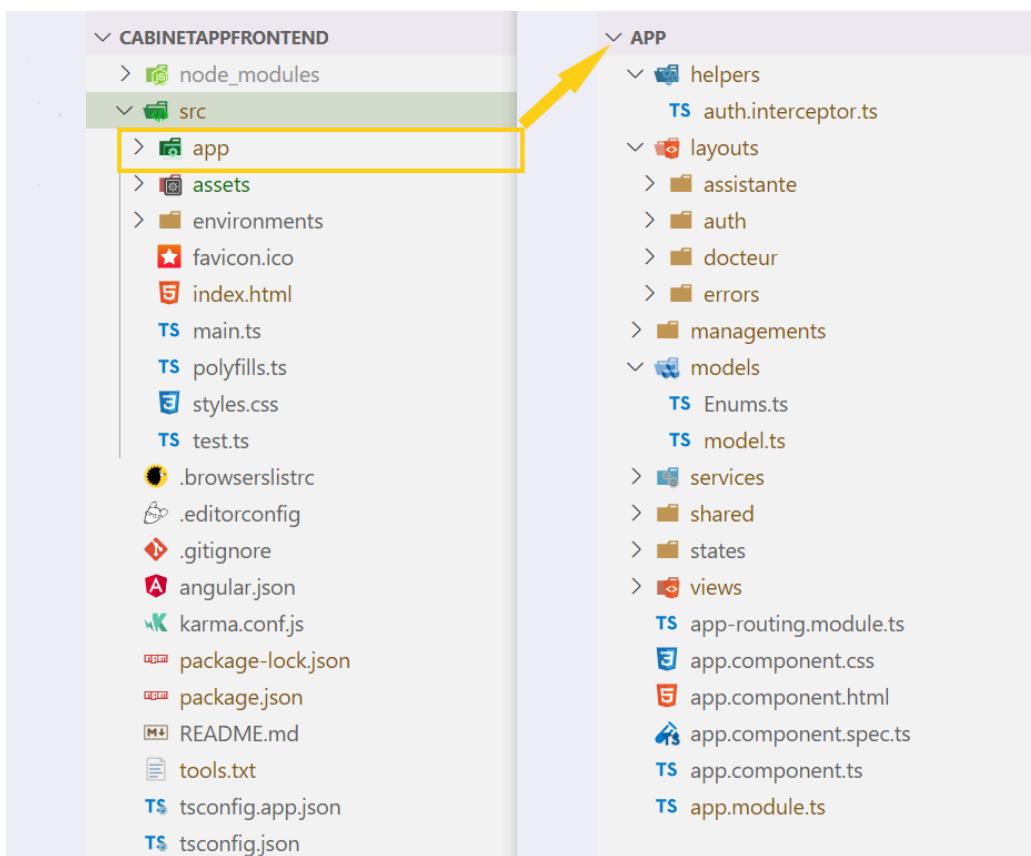


Figure 30 : Architecture du projet adopté en Angular

IV Conclusion

À travers ce chapitre, nous avons vu les différentes phases de pré-développement, qui sont l'analyse et la conception, ces deux phases qui définissent l'application, et permettent de parler de la spécification des besoins verbaux en une application, et nous avons vu aussi les différents diagrammes utilisés pour avoir une bonne idée sur le fonctionnement de l'application, avant même de démarrer la phase de développement.

Chapitre VI : Réalisation



.1 Introduction

Après la partie analyse et conception, nous abordons maintenant la phase de réalisation, où nous allons présenter l'architecture de l'application, ainsi que l'acheminement des interfaces réalisées.

.1.1 Outil et technologies utilisés

.1.1.1 Front end



Le langage HTML : (HyperText Markup Language): langage de balisage conçu pour représenter les pages web. C'est un langage permettant d'écrire de l'hypertexte, d'où son nom. HTML permet également de structurer sémantiquement et logiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie et des programmes informatiques. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web.



Le langage CSS : (Cascading Style Sheets) : est un langage informatique utilisé sur Internet pour la mise en forme de fichiers et de pages HTML ou XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.



Le langage JS : (JavaScript) : désigne un langage de développement informatique, et plus précisément un langage de script orienté objet. On le retrouve principalement dans les pages Internet. Il permet, entre autres, d'introduire sur une page web ou HTML des petites animations ou des effets. Mais il est aussi utilisé dans de nombreux environnements extérieurs aux navigateurs web tels que node.js ou Apache CouchDB. C'est un langage à objets utilisant le concept de prototype, disposant d'un typage faible et dynamique qui permet de programmer suivant plusieurs paradigmes de programmation : fonctionnelle, impérative et orientée objet.



Angular 12 et TypeScript : Angular est une plate-forme et un Framework permettant de créer des applications clientes d'une seule page à l'aide de HTML et de TypeScript. Angular est écrit en TypeScript. Il implémente les fonctionnalités principales et facultatives sous la forme d'un ensemble de bibliothèques TypeScript que vous importez dans vos applications.

.11.2 Back end



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. Il a libéré les développeurs des contraintes de formes qui occupaient leur temps avec les langages plus anciens. Ainsi, développer du code avec Python est plus rapide qu'avec d'autres langages.



Django : est un Framework Python de haut niveau, permettant un développement rapide de sites internet, sécurisés, et maintenables. Créé par des développeurs expérimentés, Django prend en charge la plupart des tracas du développement web, vous pouvez donc vous concentrer sur l'écriture de votre application sans avoir besoin de réinventer la roue. Il est gratuit, open source, a une communauté active, une bonne documentation, et plusieurs options pour du support gratuit ou non.



Django Rest Framework : est une extension à Django pour développer rapidement des API REST robustes au goût du jour. Reprenant la philosophie Django, la prise en main est rapide et efficace. Néanmoins, certaines notions nécessitent une attention plus particulière que nous allons mettre en avant dans cette suite d'articles.

.11.3 Les bibliothèques

.11.3.1 [@autho/angular-jwt](#)

.11.3.1.1 **Definition**

Cette bibliothèque fournit un **HttpInterceptor** qui attache automatiquement un **jeton Web JSON** aux **HttpClient** demandes.

Cette bibliothèque n'a aucune fonctionnalité pour (ou opinion sur) l'implémentation de l'authentification des utilisateurs et la récupération des JWT pour commencer. Ces détails varient en

fonction de votre configuration, mais dans la plupart des cas, vous utiliserez une requête HTTP standard pour authentifier vos utilisateurs, puis enregistrer leurs JWT dans un stockage local ou dans un cookie en cas de succès.

Remarque :

Cette bibliothèque ne peut être utilisée qu'avec Angular 4.3 et versions ultérieures car elle repose sur un **HttpInterceptor** fiche **HttpClient**, et ne peut être utilisée qu'avec Angular 4.3 et supérieur car elle s'appuie sur un Cette fonctionnalité n'est pas disponible sur les versions inférieures.

.II.3.1.2 Installation

- ❖ Installation avec npm : **npm install @auth0/angular-jwt**
- ❖ Installation avec yarn : **yarn add @auth0/angular-jwt**

.II.3.2 djangorestframework-simplejwt

.II.3.2.1 Définition

JSON Web Token (JWT) est une norme ouverte qui définit un moyen compact et autonome de transmettre en toute sécurité des informations entre les parties en tant qu'objet JSON. Ces informations peuvent être vérifiées et fiables car elles sont signées numériquement. JWT utilisé pour créer des jetons d'accès pour une application. JWT est bon pour l'authentification API et l'autorisation de serveur à serveur.

Le serveur génère un jeton qui certifie l'identité de l'utilisateur et l'envoie au client. Le client renverra le jeton au serveur pour chaque demande ultérieure, afin que le serveur sache que la demande provient d'une identité particulière.

JSON jetons Web se composent de trois parties séparées par des points (.), qui sont :

- ❖ **En-tête** : Identifie quel algorithme est utilisé pour générer la signature.
- ❖ **Payload** : Contient un ensemble de revendications. Les revendications sont des déclarations sur une entité.
- ❖ **Signature** : Valide en toute sécurité le jeton.

.II.3.2.2 Installation

pip installer djangorestframework-simplejwt

Une fois l'installation terminée, nous devons explicitement indiquer à DRF quel backend d'authentification nous voulons utiliser. Ouvrez le fichier **settings.py** et faire les configurations suivants :

```

REST_FRAMEWORK = {
    'DEFAULT_SCHEMA_CLASS': 'rest_framework.schemas.coreapi.AutoSchema',
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    ],
}

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
  
```

Figure 31 : Configuration du setting.py

Puis allez au fichier **urls.py**, incluez les routes pour les vues **TokenObtainPairView** et **TokenRefreshView** de Simple JWT :

```

from django.contrib import admin
from django.urls import path
from django.urls import include
from rest_framework.authtoken.views import obtain_auth_token
from rest_framework.documentation import include_docs_urls
from rest_framework_simplejwt import views as jwt_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/docs/', include_docs_urls(title='Api Documentation')),
    path('api/register/', views.registerView.as_view()),
    path('api/jwt/token/', jwt_views.TokenObtainPairView.as_view(), name='token_obtain_pair'),
    path('api/jwt/token/refresh/', jwt_views.TokenRefreshView.as_view(), name='token_refresh'),
]
  
```

Figure 32 : Configuration du fichier urls.py

.11.4 Server



WampServer : est une plateforme de développement Web de type WAMP, permettant de faire fonctionner localement (sans avoir à se connecter à un serveur externe) des scripts PHP. WampServer n'est pas en soi un logiciel, mais un environnement comprenant trois serveurs

(Apache, MySQL et Maria DB), un interpréteur de script (PHP), ainsi que phpMyAdmin pour l'administration Web des bases MySQL . Il dispose d'une interface d'administration permettant de gérer et d'administrer ses serveurs au travers d'un trayions (icône près de l'horloge de Windows).



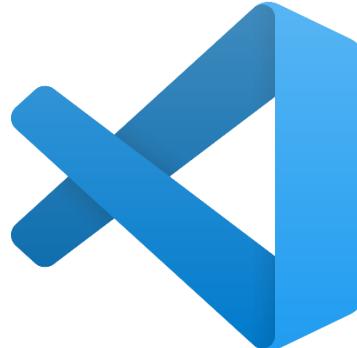
Node.js : est un environnement d'exécution JavaScript basé sur le moteur JavaScript V8 de Google qui exécute le code JavaScript en dehors du navigateur. Il aide les développeurs à créer des unités de code non bloquantes afin de tirer le meilleur parti des ressources système pour créer des applications plus réactives. L'idée était d'écrire le code Node.js en JavaScript, puis le moteur V8 le compilait en code machine prêt à être exécuté. Le système de modules de Node permet aux développeurs d'étendre la plate-forme à l'aide de modules tiers pour obtenir un maximum de fonctionnalités. Node.js permet simplement aux développeurs d'utiliser chaque projet JavaScript open-source sur le serveur de la même manière que sur le navigateur client, ce qui permet une intégration plus étroite et plus efficace entre le serveur Web et les scripts de l'application Web prise en charge.

.II.5 Base de données



MySQL : est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde³, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, PostgreSQL et Microsoft SQL Server.

.11.6 Les logiciels



Visual Studio Code (VSC par la suite) : est un éditeur de code open-source, gratuit et multi-plateforme (Windows, Mac et Linux), développé par Microsoft, à ne pas confondre avec Visual Studio, l'IDE propriétaire de Microsoft. VSC est développé avec Electron et exploite des fonctionnalités d'édition avancées du projet Monaco Editor. Principalement conçu pour le développement d'application avec JavaScript, Type Script et Node.js, l'éditeur peut s'adapter à d'autres types de langages grâce à un système d'extension bien fourni.



PowerDesigner (anciennement PowerAMC) : Est un logiciel de conception créée par la société SAP, qui permet de modéliser les traitements informatiques et leurs bases de données associées.

Il a été créé par SDP sous le nom AMC*Designor, racheté par Powersoft qui lui-même a été racheté par Sybase en 1995. Depuis 2010 Sybase appartient à l'éditeur allemand SAP1.

Avant mars 2016, la version française était commercialisée par SAP sous la marque PowerAMC2, jusqu'à la fusion avec la version internationale sous le nom PowerDesigner depuis la version 16.63.

PowerDesigner est disponible sous forme d'application native Microsoft Windows ou comme plugin eclipse. Par défaut, PowerDesigner stocke ses modèles sous forme de fichiers, dont l'extension dépend du type de modèle: bpm (pour business process model), cdm (pour conceptual data model)... La structure interne du fichier peut être du XML ou du binaire compressé. PowerDesigner peut aussi stocker ses modèles dans un Référentiel.



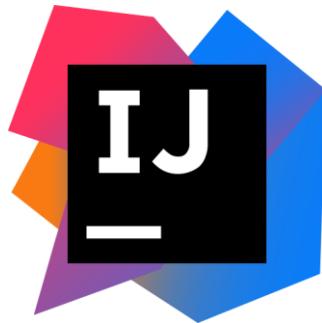
StarUML : est un logiciel de modélisation UML, qui a été "cédé comme open source" par son éditeur, à la fin de son exploitation commerciale (qui visiblement continue ...), sous une licence modifiée de GNU GPL. StarUML gère la plupart des diagrammes spécifiés dans la norme UML 2.0. StarUML est écrit en Delphi¹, et dépend de composants Delphi propriétaires (non open-source).



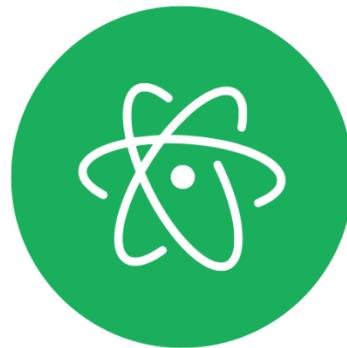
POSTMAN

Postman sert à exécuter des appels HTTP directement depuis une interface graphique. Vous pourrez simplement choisir l'URL, la méthode HTTP (le plus souvent GET, POST, PUT, PATCH et DELETE), les headers, les query params et dans certains cas le body de la requête.

.11.7 IDE



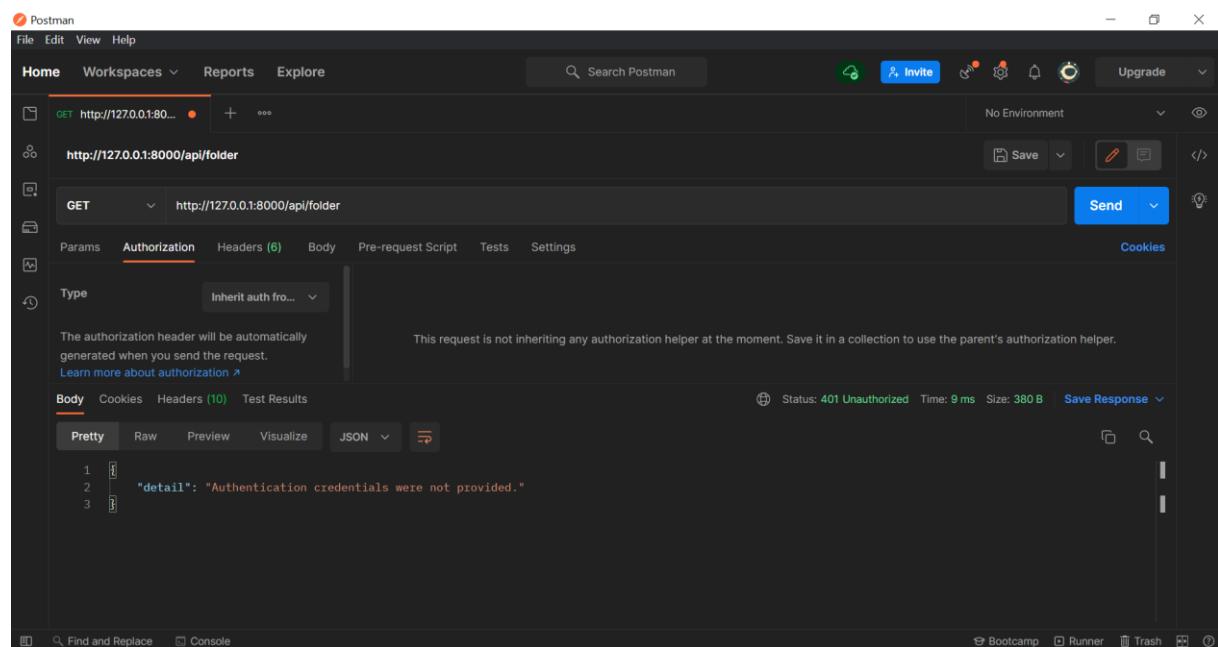
IntelliJ IDEA : est un IDE intelligent et tenant compte du contexte qui permet de travailler sur toutes sortes d'applications en Java et dans d'autres langages de la JVM tels que Kotlin, Scala et Groovy. De plus, IntelliJ IDEA Ultimate vous aide à développer des applications web full-stack grâce à ses puissants outils intégrés, à la prise en charge de JavaScript et de ses technologies connexes et à la prise en charge avancée de frameworks populaires tels que Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus et Helidon. IntelliJ IDEA peut être complété par des plugins gratuits développés par JetBrains afin de pouvoir travailler avec d'autres langages de programmation, parmi lesquels Go, Python, SQL, Ruby et PHP.



Atom : est un éditeur de texte libre pour macOS, GNU/Linux et Windows développé par GitHub. Il prend en charge des plug-ins écrits en Node.js et implémente Git Control. La plupart des extensions sont sous licence libre et sont maintenues par la communauté⁴. Atom est basé sur Chromium et Electron et est écrit en CoffeeScript⁵. Il est aussi utilisé en tant qu'environnement de développement (EDI).

III Interfaces de Site Web et d'application

L'authentification basée sur un jeton fonctionne en obtenant un jeton pour le nom d'utilisateur, email et le mot de passe corrects utilisés pour avoir l'accès qui a pour but de communiquer avec l'api et effectuer les requêtes directement au serveur sans aucun problème

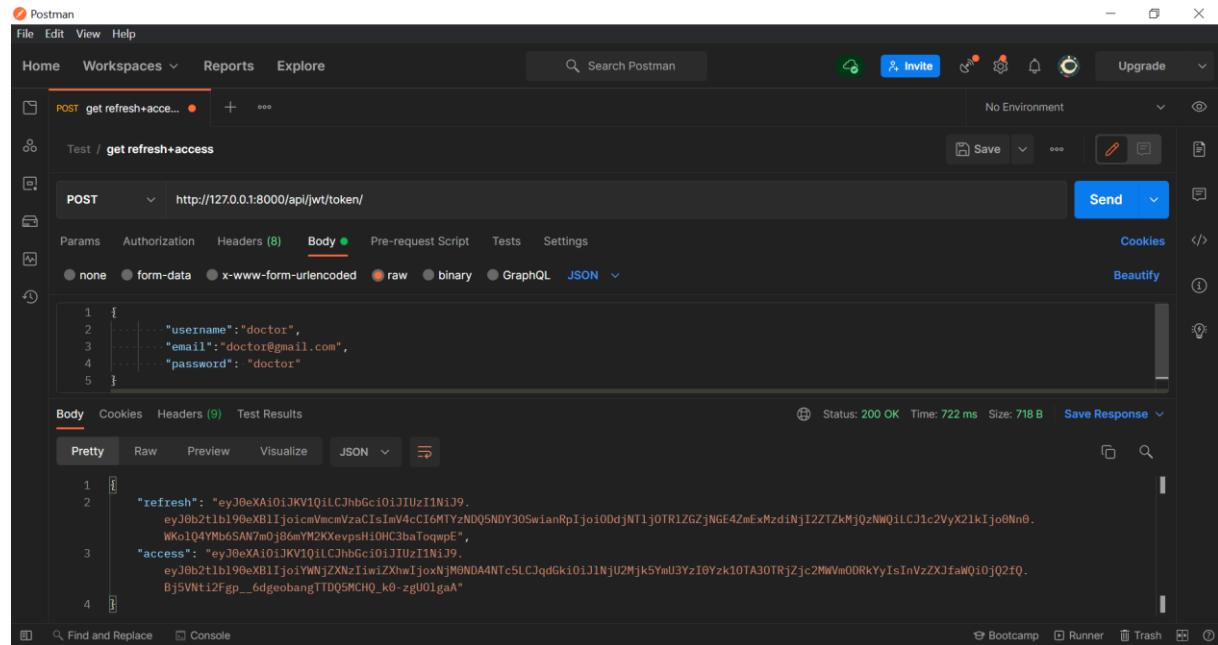


The screenshot shows the Postman application interface. A GET request is made to `http://127.0.0.1:8000/api/folder`. The 'Authorization' tab is selected under 'Params'. The response status is 401 Unauthorized, with the message: "Authentication credentials were not provided.".

Figure 33 : Problème d'authentification sans token

Obtention du jeton :

Pour obtenir les jetons d'accès et d'actualisation, on doit transmettre le nom d'utilisateur (username), l'email et le mot de passe en effectuant une requête POST vers <http://127.0.0.1:8000/api/jwt/token/> :



The screenshot shows a Postman interface with a POST request to `http://127.0.0.1:8000/api/jwt/token/`. The request body is set to `JSON` and contains the following data:

```

1 {
2   "username": "doctor",
3   "email": "doctor@gmail.com",
4   "password": "doctor"
5 }

```

The response status is `200 OK`, Time: 722 ms, Size: 718 B. The response body is displayed in Pretty format:

```

1 {
2   "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.  

    eyJ0b2tlbi90eXB1IjoicmVmcmVzaC1saW4cC16MTYzNDQ5NDY3OSwianRpIjo10djdNT1jOTR1ZGZjNGE4ZmExMzd1NjI2ZT2kMjQzNWQiLCJ1c2Vx21kIjo0Nn0.  

    Wk01Q4Ymb6SAN7m0j86mYM2KXevpsHi0HC3baToqwpE",
3   "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.  

    eyJ0b2tlbi90eXB1Ijo1YWNjZXNzIiwixhwiJoxNjMONDA4NTc5LCJqdGkiOiJ1NjU2Mjk5YmU3YzI0Yzk10TA30TRjZjc2Mm0DRkYyIsInVzZXJfaWQ1oJQ2fQ.  

    Bg5VNTi2Fgp_6dgeobangITDQ5MCHQ_k6-zgu01gaA"
4 }

```

Figure 34 : Obtention du jeton à base d'username, email et password

Pour accéder aux points de terminaison protégés dans notre backend, nous devons inclure le jeton d'accès dans l'en-tête de toutes nos demandes.

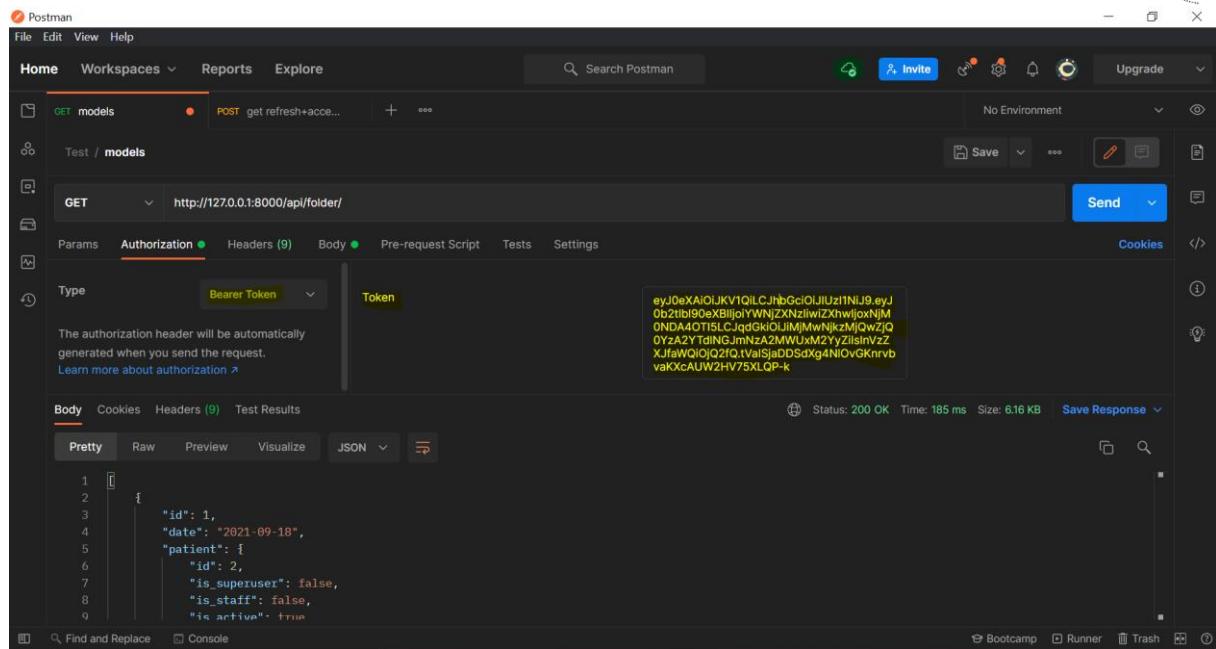
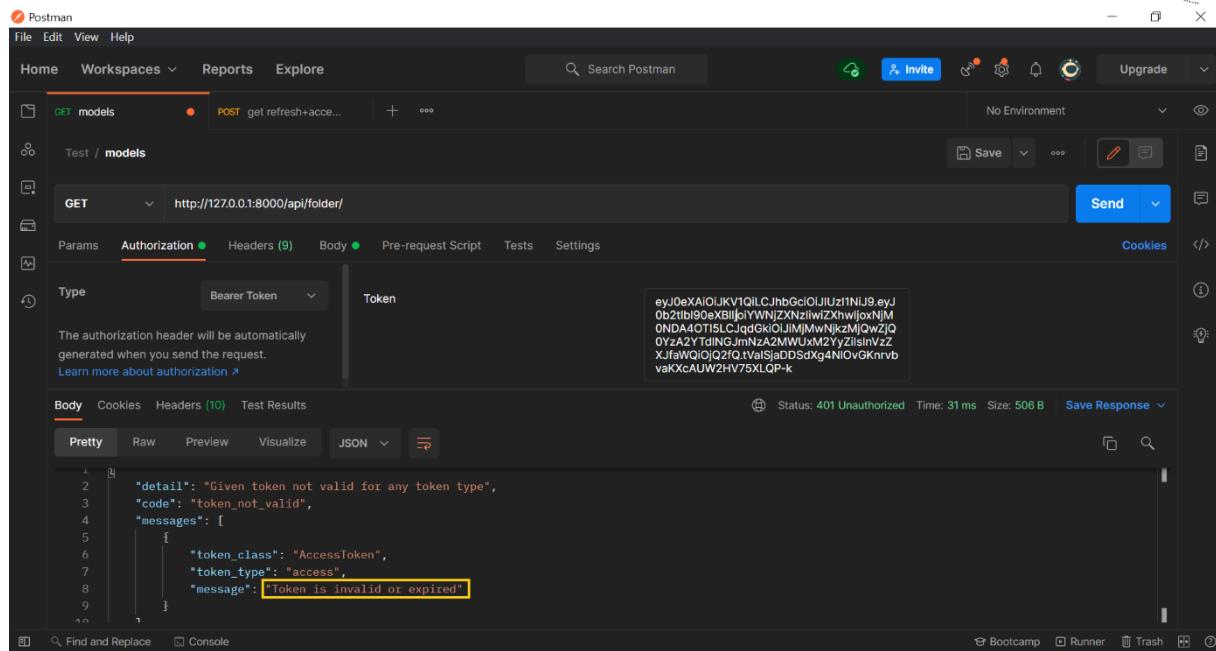


Figure 35 : L'utilisation du jeton d'accès pour communiquer avec l'API

Nous pouvons utiliser le jeton d'accès dans les 5 minutes avant son expiration. Après cela, nous devrons obtenir un autre jeton d'accès à l'aide du jeton d'actualisation que nous avons obtenu de la demande d'API précédente.

Lorsque nous essayons de faire des demandes aux points de terminaison protégés, nous obtenons l'erreur ci-dessous.



The screenshot shows the Postman application interface. A GET request to `http://127.0.0.1:8000/api/folder/` has failed with a 401 Unauthorized status. The response body is a JSON object:

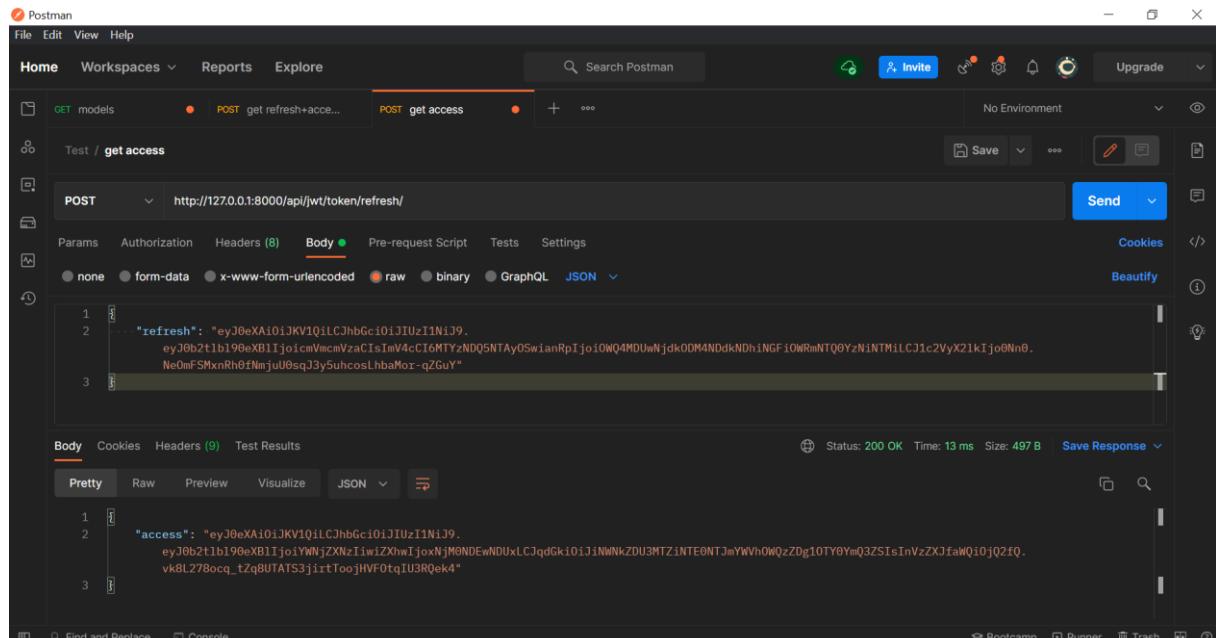
```

{
  "detail": "Given token not valid for any token type",
  "code": "token_not_valid",
  "messages": [
    {
      "token_class": "AccessToken",
      "token_type": "access",
      "message": "Token is invalid or expired"
    }
  ]
}

```

Figure 36 : Expiration du jeton d'accès

Pour obtenir un nouveau jeton d'accès, nous ferons une demande post vers `http://127.0.0.1:8000/api/jwt/token/refresh/` pour publier le nouveau jeton d'actualisation.



The screenshot shows the Postman application interface. A POST request to `http://127.0.0.1:8000/api/jwt/token/refresh/` has succeeded with a 200 OK status. The response body is a JSON object:

```

{
  "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbi90eXBIIjoicmVmcmVzaTIsImV4cCt6MTYzNDQ5NTAyOSwanRpIjoiOWQ4MDUwNjdkODM4NDdkNDhiNGFiOWRmNTQ0YzNINTMiLCJ1c2VyX21kIjoi0Nn0NeOmSMXnRh0fNmju0sqJ3y5uhcosLhbaMor-qGuY",
  "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbi90eXBIIjoicmVmcmVzaTIsImV4cCt6MTYzNDQ5NTAyOSwanRpIjoiOWQ4MDUwNjdkODM4NDdkNDhiNGFiOWRmNTQ0YzNINTMiLCJ1c2VyX21kIjoi0Nn0vk8L278ocq_tZq8UTATS3jirtTooJHVF0tqiU3RQek4"
}

```

Figure 37 : Obtention d'un nouveau jeton d'accès

Implémentation de l'authentification par jeton au niveau d'Angular :

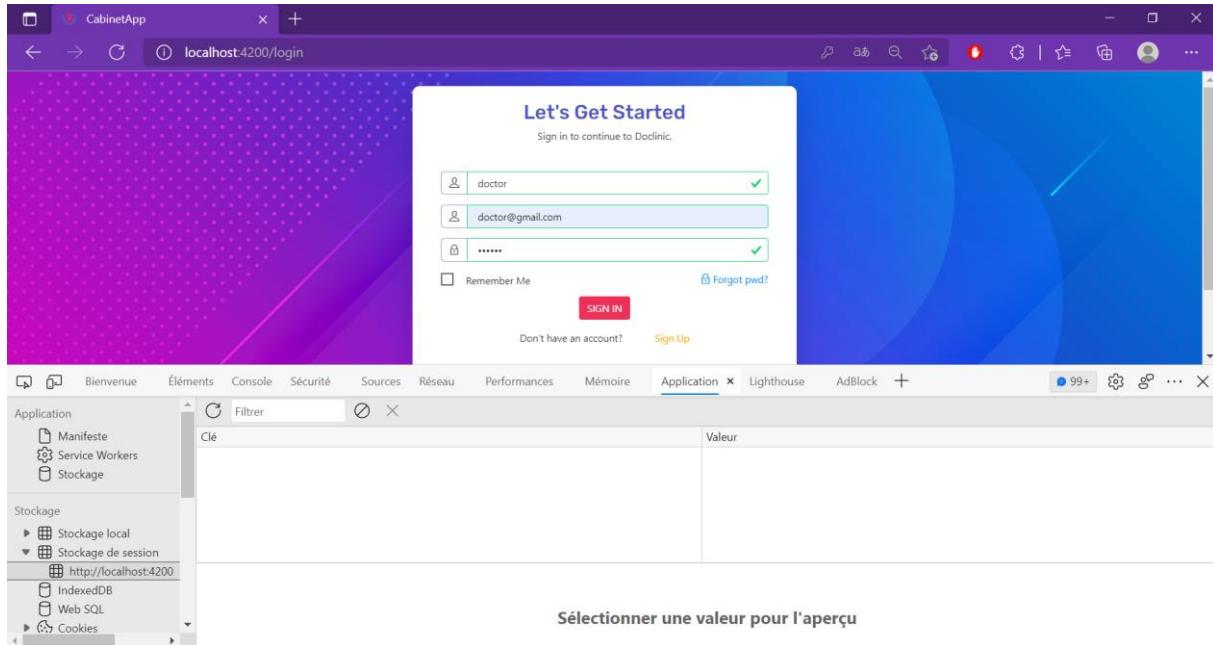


Figure 38 : Interface d'authentification en Angular

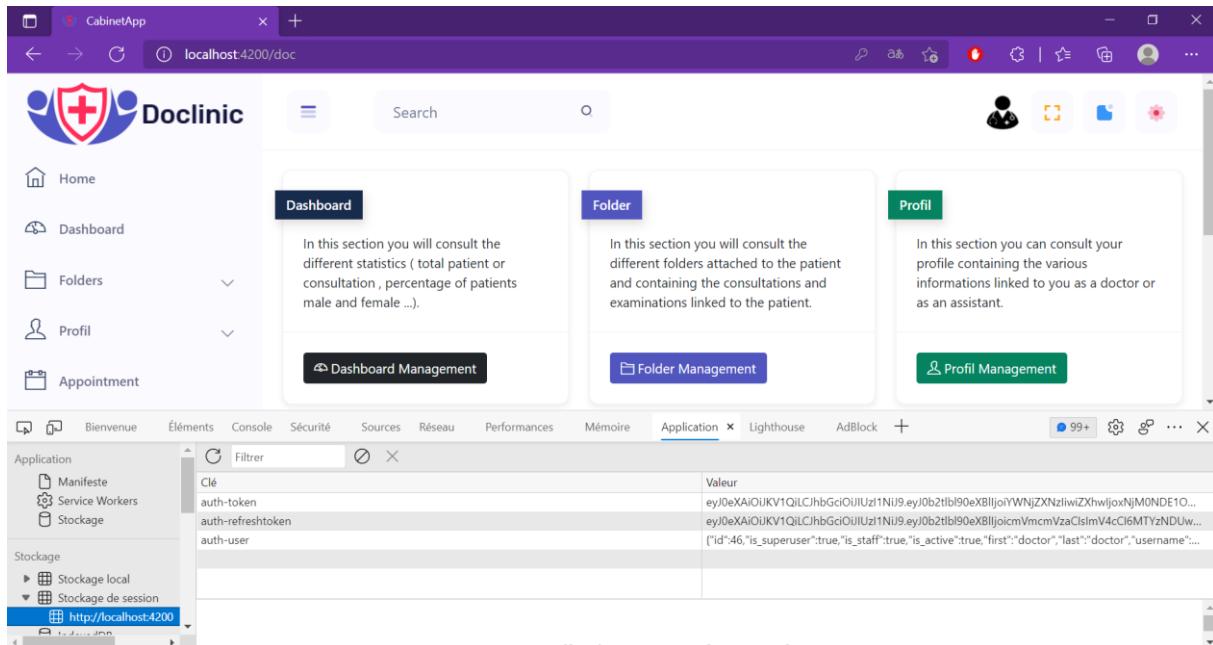


Figure 39 : Stockage du jeton obtenu par l'api dans SessionStorage in angular

Conclusion et perspectives

Dans ce rapport de projet de fin d'année, nous avons pu regrouper et appliquer toutes les étapes de la conception et la réalisation dans notre application sur la gestion de cabinet médical.

Pour mettre en œuvre ce projet nous avons suivi plusieurs étapes. Dans un premier lieu, nous avons commencé par définir les besoins fonctionnels et non fonctionnels de notre application et aussi rédiger le cahier des charges. Dans un second lieu, nous avons fait une analyse et conception du projet en se basant sur le formalisme UML. Enfin, nous avons mis en œuvre notre solution à l'aide des frameworks angular12 et Django Rest API. Ce projet a été pour nous une expérience très enrichissante vue que c'était notre premier contact avec les frameworks.

Bibliographie

DJANGO RESTFUL WEB SERVICES

Angular Up and Running Learning-Angular-Step-by-Step

Become a ninja with Angular by Ninja Squad

Webographie

.1 Site

<https://docs.djangoproject.com/fr/3.2/>

<https://python.doctor/page-django-introduction-python>

<https://www.section.io/engineering-education/securing-django-api/>

<https://www.section.io/engineering-education/django-api-documentation/>

<https://www.djangoproject.org/>

<https://www.hacksoft.io/blog/introducing-django-enum-choices-2>

<https://angular.io/docs>

<https://www.w3schools.com/angular/>

<https://www.ambient-it.net/comment-fonctionne-angular/>

<https://angular-enterprise.com/fr/ngcategorie/cours/>

<https://www.npmjs.com/package/ng2-charts>

<https://www.npmjs.com/package/ngx-perfect-scrollbar>

<https://www.npmjs.com/package/ngx-dropzone?fbclid=IwAR3yyHnQ9k6Mef6YV1g3VpxNzYJM8qmJ1-ulbLJV0CyIKjOxAuzLDGGOrfA>

https://www.bezkoder.com/angular-12-jwt-auth/?fbclid=IwAR0BCjzxkG88t0k74SydDxWNeSUM5ppi7Fak3ucAXDcgm_w5MefSIgQuNCU

<https://www.npmjs.com/package/@auth0/angular-jwt>

<https://sweetalert2.github.io/>

<https://stackoverflow.com/>

<https://scalablescripts.com/p/angular-django-admin>

<https://github.com/gund/ng-http-interceptor>

<https://github.com/scalablescripts/django-auth>

.11 Vidéos

<https://www.youtube.com/watch?v=ilqMEfIK-HM&list=PLmDLs7JbXWNjr5vyJhfGu69sowglUl8z5>

<https://www.youtube.com/playlist?list=PLXqhO5lRtxJV6oWcW2vIPHRzRFF6gVvc3>

<https://www.youtube.com/playlist?list=PLlxnHhzx9q8B0USj0wSpPfwHEogCj3ZxV>

https://www.youtube.com/playlist?list=PLdIu69UmPK4o_7xqsKBdMyN7C5aVGN4I

https://www.youtube.com/playlist?list=PLMya0JgwP7RyqmUa3pg_LknK-12NQIGqB

https://www.youtube.com/watch?v=hIvq-NzkRXM&ab_channel=MpembalInc

https://www.youtube.com/watch?v=-36YauTh4Ts&ab_channel=KrystianCzekalski

https://www.youtube.com/watch?v=PUzgZrS_piQ&ab_channel=ScalableScripts

https://www.youtube.com/watch?v=QdXHkybzrUU&ab_channel=BezKoder