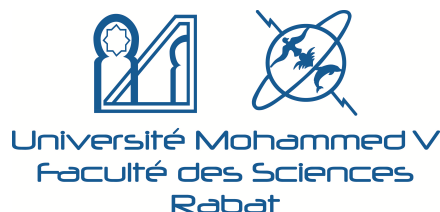


UNIVERSITÉ MOHAMMED V de Rabat  
**Faculté des Sciences**



**Département d'Informatique**  
**Filière Licence Fondamentale**  
**en Sciences Mathématiques et Informatique**

**PROJET DE FIN D'ÉTUDES**

intitulé :

**Application De Gestion Des Rendez-Vous D'un Cabinet  
Dentaire**

Présenté par :

**AJA OTHMANE**

**EL ABBADI YOUNESS**

soutenu le ?? Juin 2019 devant le Jury

Mme. Salma Mouline	Professeur à la Faculté des Sciences - Rabat	<i>Encadrant</i>
M. Prénom Nom	Professeur à la Faculté des Sciences - Rabat	<i>Président</i>
Mme Prénom Nom	Professeur à la Faculté des Sciences - Rabat	<i>Examineur</i>

Année universitaire 2018-2019

# Remerciements

Notre gratitude et nos remerciements les plus sincères vont à notre encadrante Professeur Mouline Salma, pour tout le temps qu'elle nous a consacré, son professionnalisme, ses conseils précieux et la diligence de son suivi durant toute la période de notre projet.

Nous ne saurions assez remercier les membres de jury pour le temps qu'ils nous accordent, l'intérêt qu'ils portent pour l'avenir universitaire des étudiants et la qualité de leur jugement.

Aussi, nous tenons à rendre un grand hommage à tout le corps professoral de la faculté des sciences de Rabat pour son enseignement de qualité.

Que nos familles qui ont notamment initiées nos pas voient dans ce travail le fruit de leur assistance sans faille à notre égard

# Table des matières

<b>1</b>	<b>Cahier des charges et environnement de travail</b>	<b>6</b>
1.1	Étude du cahier des charges et spécification des besoins . . . . .	6
1.1.1	Définition d'un cahier des charges . . . . .	6
1.1.2	Les Fonctionnalités d'un cahier des charges . . . . .	6
1.1.3	Problématique . . . . .	7
1.1.4	Solution Proposée . . . . .	7
1.2	Environnement de travail . . . . .	7
1.2.1	Les Bases Données No-SQL . . . . .	7
1.2.2	La différence entre les BD SQL et No-SQL . . . . .	7
1.2.3	Mongo DB . . . . .	8
1.2.4	Python . . . . .	9
<b>2</b>	<b>Conception</b>	<b>10</b>
2.1	Le langage UML . . . . .	10
2.2	Modélisation . . . . .	10
2.2.1	Diagramme de cas d'utilisation . . . . .	10
2.2.2	Diagramme de classe . . . . .	12
2.3	Passage du Model Conceptuel au Model Physique . . . . .	13
<b>3</b>	<b>Réalisation</b>	<b>14</b>
3.1	Partie Base de données . . . . .	14
3.1.1	Connexion à la Base de données . . . . .	14
3.1.2	Création des Collections . . . . .	15
3.1.3	Insertion dans la base de données . . . . .	15
3.2	Partie Traitements . . . . .	17
3.2.1	Création des Fenêtres . . . . .	17
3.2.2	Liaison entre les Fenêtres . . . . .	19
3.2.3	Les fonctions Utilisées . . . . .	20
3.2.4	Gestion des Comptes . . . . .	22
3.2.5	Gestion de Login . . . . .	24

---

3.2.6	Gestion des Médecins . . . . .	25
3.2.7	Gestion des patients . . . . .	28
3.2.8	Gestion des Rdv . . . . .	30
3.2.9	Gestion des Rapports Certificats . . . . .	32
<b>Bibliographie</b>		<b>36</b>
<b>Annexes</b>		<b>37</b>

# Table des figures

2.1	Diagramme de Cas d'utilisation . . . . .	11
2.2	Diagramme de class . . . . .	12
3.1	Code Creation des collections. . . . .	14
3.2	Code Création des collections. . . . .	15
3.3	Insertion d'un Médecin . . . . .	15
3.4	Insertion d'une Secrétaire . . . . .	16
3.5	Insertion d'un Rapport . . . . .	16
3.6	Insertion d'un Rendez-vous . . . . .	16
3.7	Insertion d'un Patient . . . . .	17
3.8	Initialisation de la fenêtre . . . . .	17
3.9	Fenêtre Tk . . . . .	18
3.10	Code Fenêtre login . . . . .	18
3.11	Menu Secrétaire . . . . .	19
3.12	Menu Secrétaire Après Clique Sur Gérer RDV . . . . .	19
3.13	Code Liaison entre fenêtres . . . . .	19
3.14	Fonction Affichage Rendez-vous Médecin . . . . .	20
3.15	Fonction Affichage Rendez-vous Patient . . . . .	20
3.16	Fonction Affichage Rendez-vous Journée . . . . .	21
3.17	Fonction Générer Id . . . . .	21
3.18	Fenêtre Ajout d'un compte . . . . .	22
3.19	Fenêtre Recherche un Utilisateur . . . . .	22
3.20	Fenêtre Modifier un Utilisateur . . . . .	23
3.21	Fenêtre Supprimer un Utilisateur . . . . .	23
3.22	Fenêtre Login . . . . .	24
3.23	Code Connexion . . . . .	24
3.24	Fenêtre d'erreur login . . . . .	25
3.25	Fenêtre Clôturer Soins . . . . .	25
3.26	Fenêtre Choix Médecin . . . . .	26
3.27	Fenêtre Emploi des rendez-vous . . . . .	26

---

3.28 Fenêtre Rédiger un rapport . . . . .	27
3.29 Fenêtre Rédiger un Certificat . . . . .	27
3.30 Fenêtre Ajout d'un Patient . . . . .	28
3.31 Fenêtre Ajout d'un Patient . . . . .	28
3.32 Fenêtre Modification d'un Patient . . . . .	29
3.33 Fenêtre Consultation d'un Patient . . . . .	29
3.34 Fenêtre Ajout d'un Rendez-vous . . . . .	30
3.35 Fenêtre Recherche d'un Rendez-vous . . . . .	30
3.36 Fenêtre Modification d'un Rendez-vous . . . . .	31
3.37 Fenêtre Suppression d'un Rendez-vous . . . . .	31
3.38 Dossier Des Patients . . . . .	32
3.39 Dossier D'un Patient . . . . .	32
3.40 Fenêtre Rédaction d'un Rapport . . . . .	32
3.41 Bases données du rapport . . . . .	33
3.42 Fenêtre Consulter un Rapport . . . . .	33
3.43 Affichage d'un Rapport . . . . .	34
3.44 Fenêtre Rédaction d'un Certificat . . . . .	34
3.45 Affichage d'un Certificat . . . . .	35

# Chapitre 1

## Cahier des charges et environnement de travail

### 1.1 Étude du cahier des charges et spécification des besoins

#### 1.1.1 Définition d'un cahier des charges

Notre objectif durant ce projet est de traiter un cahier des charges complet et faire la conception et évidemment la réalisation du programme demandé. Un cahier des charges est un modèle littéraire qui sert à poser les contraintes, formaliser les besoins et expliquer les différentes tâches pour chaque acteur qui va interagir avec le programme afin de produire une solution technique qui résout le problème posé.

#### 1.1.2 Les Fonctionnalités d'un cahier des charges

Le cahier des charges permet de simplifier l'étude de cas et parmi ces fonctionnalités :

- préciser et définir les objectifs et la finalité d'un projet
- détailler le contexte du projet (contraintes techniques, parties prenantes, exigences particulières, charte graphique, livrable attendu, ...)
- répertorier l'ensemble des besoins et des caractéristiques du projet
- identifier les contraintes, les intervenants et les interactions internes et externes au projet
- rassembler l'ensemble des éléments dans un même document afin que chaque intervenant dispose des mêmes informations
- répartir les charges et missions de chaque intervenant
- faciliter les consultations pour une mise en concurrence des différents prestataires.

---

### 1.1.3 Problématique

Le problème posé est la gestion des différentes tâches d'un cabinet dentaire, et automatiser ces derniers afin de faciliter la tâche pour l'admin, secrétaire et le médecin. Parmi les fonctions exécutées par ces derniers :

- La Gestion des RDVs(ajouter,supprimer,modifier,consulter).
- La Gestion des Patients(ajouter,supprimer,modifier).
- la Gestion des rapports et certificats ( Créer et consulter).
- La Gestion des Comptes .

### 1.1.4 Solution Proposée

Pour Résoudre Le problème posé précédemment, on propose de construire un système qui permet d'automatiser toutes les tâches et les fonctions de ce cabinet, en langage python et une base des données No-SQL Mongo Db.

## 1.2 Environnement de travail

### 1.2.1 Les Bases Données No-SQL

Les bases de données No-SQL (Not Only SQL) est une famille de bases de données qui s'écarte du model relationnel, ce qui lui permet de relâcher certaines contraintes lourdes du relationnelle pour créer une structure de données flexible et utile pour de très grands ensembles de données distribuée, et cette famille englobe une gamme étendue de technologies afin de résoudre les problèmes de performances en matière d'évolutivité et de BIG data que les bases de données relationnelles ne sont pas conçues pour affronter .

### 1.2.2 La différence entre les BD SQL et No-SQL

Les deux types des bases de données se diffèrent en plusieurs points, parmi eux :

- (Schéma physique) SQL organise le stockage de données sur le principe de tables reliées entre elles. La structure et les types des données sont rigides, c'est-à-dire fixés à l'avance avant d'implémenter une logique métier. No-SQL stocke et manipule des documents qui correspondent à des collections d'objets.
- (Schéma logique) Pour les SQL la table qu'on appelle le schéma est fixe au début, ce qui implique que si on fait une erreur de conception, il faut tout refaire. Contrairement au NOSQL qui permet une flexibilité très utile.



- 
- (Normalistaion) Le SQL utilise des jointures pour établir le lien entre les tables, en utilisant des clés étrangères et des index .Pour le NOSQL la règle est totalement différente, la liaison se fait en intégrant des documents entièrement dans d'autres. Cela conduit à des requêtes beaucoup plus rapides en accès mais lents en mise à jour.

### 1.2.3 Mongo DB



#### Historique

MongoDB est développé depuis 2007 par MongoDB. Cette entreprise travaillait alors sur un système de Cloud computing, informatique à données largement réparties, similaire au service Google App Engine de Google. Sa première version considérée comme industriellement viable a été la 1.4, en 2010.

Il est ensuite devenu un des SGBD les plus utilisés, notamment pour les sites web comme eBay, Foursquare, New York Times 3.

#### Definition

MongoDB est un système de gestion de bases de données dans la mouvance NoSQL orientée documents écrit en langage C++.

#### Format des donnees JSON

Javascript Objet notation (JSON) est un format de données textuelles, dérivé de la notation des objets du langage Javascript. Il permet de représenter de l'information structurée comme le permet XML par exemple. Créé par Douglas Crockford entre 2002 et 2005.

la forme JSON comprend deux types d'éléments structurels :

- des ensembles de paires « clé » / « valeur »
- des listes ordonnées de valeurs.

Les valeurs représentent 3 types de données :

- Des objets.
- Des tableaux.
- Des types primitifs(bool,int,float,string).

---

## 1.2.4 Python



### Historique

Le langage Python a été créé en 1990/1991 par un certain Guido van Rossum à l'institution national de recherche des Pays-Bas Stichting Mathematisch (CWI laquelle a été fondée 1946 et est spécialisée dans les sciences de mathématiques et d'ordinateurs). Ce langage était destiné à remplacer le langage ABC.

Entre 1995 et 2000, Guido van Rossum poursuit le développement du langage dans un nouvel institut situé à Reston, en Virginie, aux États-Unis, à la Corporation for National Research Initiatives (CNRI) où il développera à partir de la version 1.2, les versions jusqu'à 1.5.2.

En mai 2000, le développement se poursuit chez BeOpen.com sous le nom de l'équipe BeOpen PythonLabs team. Puis en octobre 2000, l'équipe déménage chez Digital Creations (actuellement Zope Corporation). En 2001, Zope Corporation et certains partenaires crée l'organisme sans but lucratif Python Software Foundation, laquelle s'occupe de gérer la propriété intellectuelle du langage de programmation.

### Définition

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes qui est un sous-système informatique de gestion automatique de la mémoire. Il est responsable du recyclage de la mémoire préalablement allouée puis inutilisée et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.

# Chapitre 2

## Conception

### 2.1 Le langage UML

UML, c'est l'acronyme anglais pour « Unified Modeling Language ». On le traduit par « Langage de modélisation unifié ». La notation UML est un langage visuel constitué d'un ensemble de schémas, appelés des diagrammes, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour représenter le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel.

### 2.2 Modélisation

#### 2.2.1 Diagramme de cas d'utilisation

Ce diagramme est nommé un diagramme de «cas d'utilisations», son but ultime est de définir la relation entre un utilisateur et une application web. En effet, dans notre cas, cet utilisateur peut prendre trois formes majeures qui se présentent comme suit : un admin ou un secrétaire ou encore un médecin. Ces derniers doivent obligatoirement se connecter, en premier lieu, afin qu'ils puissent accéder à l'application et en manipuler par la suite. Néanmoins chacun des utilisateurs, cités au préalable, a un certain nombre de tâches et de missions à effectuer dans le but d'assurer la bonne conduite de l'application en question. D'ailleurs, ces tâches définissent les missions qui doivent être réalisées par chaque utilisateur. Dès lors, ces missions peuvent se décliner de la sorte selon la fonction de chacun de ces utilisateurs :

- **Admin** : Ajouter, supprimer et modifier les deux autres utilisateurs : le secrétaire et le médecin.
- **Secrétaire** : Il est dans la mesure de réaliser les missions suivantes :
  - Editer les rendez-vous ainsi que les patients.
  - Consulter les emplois du temps des médecins et de la journée à la fois.

— **Médecin** : Il se charge d'un ensemble de points tels que :

- Clôturer un soin d'un patient.
- Consulter les dossiers des patients ainsi que la table de leurs rendez-vous.
- Rédiger un rapport.

La figure ci-dessous « voir figure 2.1 » représente le diagramme de « cas d'utilisations » expliqués auparavant :

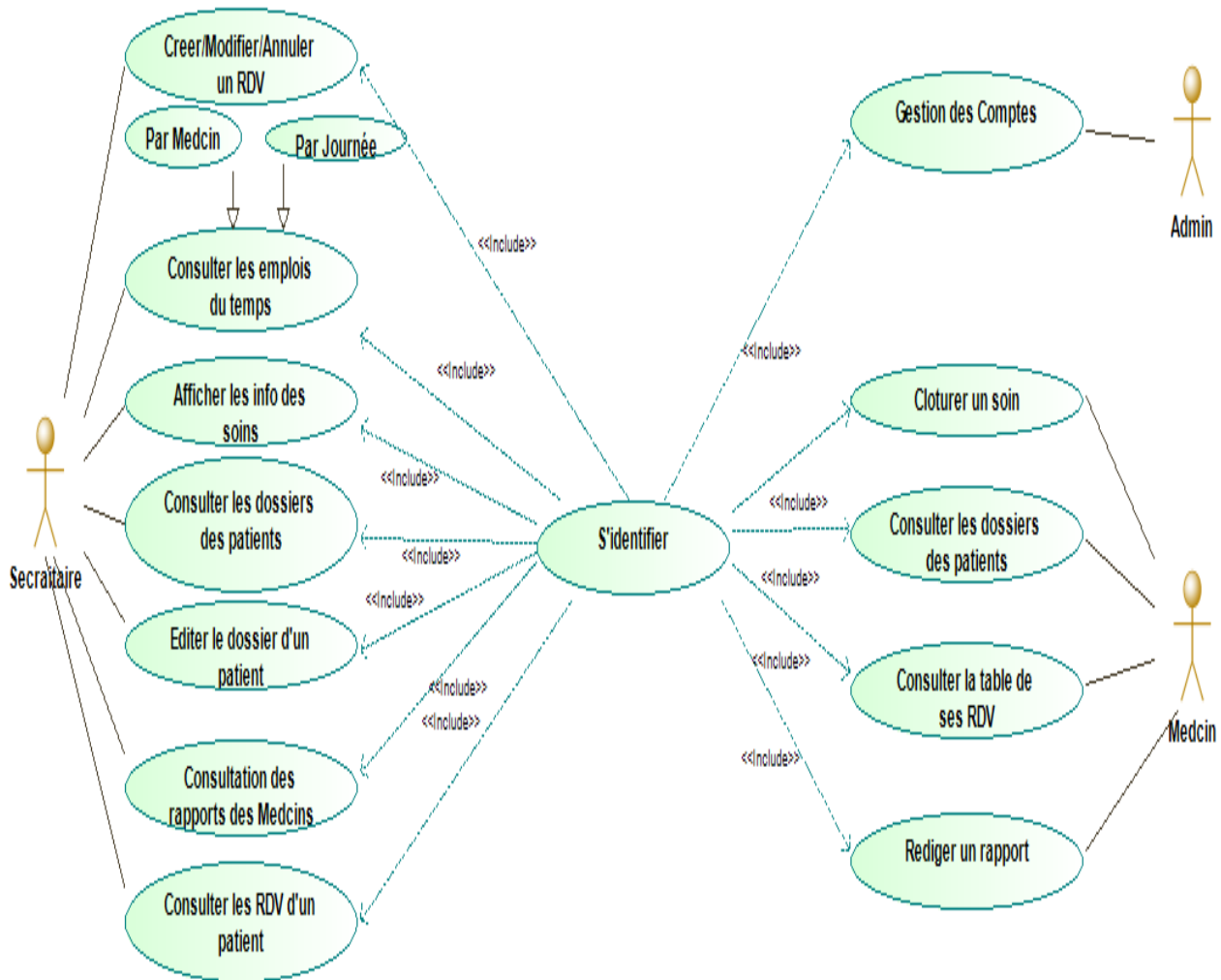


FIGURE 2.1 – Diagramme de Cas d'utilisation



---

Un rendez-vous comme indique la figure 2.2 a pour attribut : Numéro RDV et une Date.

Un rendez-vous est obligatoirement pris par une secrétaire qui est représentée par un Login et un mot de passe.

Cette dernière est chargée d'ajouter les patients, modifier et supprimer leurs informations, voir les soins disponibles et prendre des rendez-vous pour un patient et un médecin avec un motif donné, et qui a comme attribut numéro et prix. Ce dernier peut être un soin ou bien une consultation.

Un patient a comme attribut nom, Prénom, Date naissance, N Télé, Adresse, sexe, N Cin et N Assurance.

Le médecin est représenté par Login et un mot de passe et chargé de consulter son emploi, dossier des patients, clôturer les soins et rédiger des rapports.

Admin est représenté par Login et un mot de passe et chargé d'ajouter, modifier et supprimer les médecins et les secrétaires.

Rapport est une classe d'association entre le médecin et Rdv, qui a comme attribut N Rapport et lien.

## 2.3 Passage du Model Conceptuel au Model Physique

Un Modèle Conceptuel des Données (MCD) est utilisé par les concepteurs et les analystes, pour décrire sous forme d'un schéma les données relatives au sujet à traiter (en gros les entités, leurs attributs et les relations qu'elles entretiennent). Le MCD ne tient pas compte du SGBD et du langage de programmation à suivre.

Un Modèle Physique de Données (MPD) est un outil de conception de base de données, qui permet de définir la mise en oeuvre de structures physiques et de requêtes portant sur des données.

Pour passer d'un modèle conceptuel à un modèle physique, on doit passer par des règles nécessaires pour avoir une bonne automatisation du passage du MCD au MPD :

- **Règle n1** : toute entité doit être représentée par une table.
- **Règle n2** : Dans le cas d'entités reliées par des associations de type 1 :1, les tables doivent avoir la même clef.
- **Règle n3** : Dans le cas d'entités reliées par des associations de type 1 :n, chaque table possède sa propre clef, mais la clef de l'entité côté 0,n (ou 1,n) migre vers la table côté 0,1 (ou 1,1) et devient une clef étrangère (index secondaire).
- **Règle n4** : Dans le cas d'entités reliées par des associations de type n :m, une table intermédiaire dite table de jointure, doit être créée, et doit posséder comme clef primaire une conjonction des clefs primaires des deux tables pour lesquelles elle sert de jointure.

# Chapitre 3

## Réalisation

Après la phase d'explication de notre problématique, précisions ,définitions des outils utilisés et ensuite conception entière de système. Dans ce chapitre nous passons à la phase d'implémentation physique de notre solution proposée avec le langage python et MongoDB. Nous allons traiter en détails les différents cas d'utilisation de notre système pour les 3 comptes qui existent.

### 3.1 Partie Base de données

Dans ce qui suit on va héberger la partie base de données de notre solution, en allant de la création de la connexion et des collections jusqu'à l'insertion des données à la fin.

#### 3.1.1 Connexion à la Base de données

Tout d'abord il faut connecter le code avec la base de données . Pour connecter notre code python avec la base de données MongoDB, on va utiliser la librairie pymongo, qui permet d'établir la connexion entre ces derniers, pour interagir ensuite avec la base de données

```
import pymongo

# On cree tout d'abord le lien vers notre base de donnees

lienBD = pymongo.MongoClient("mongodb://localhost:27017/")

BD = lienBD["Cabinet"]
```

FIGURE 3.1 – Code Creation des collections.

---

### 3.1.2 Création des Collections

Notre base de données contient les collections suivantes :

- Admin : qui représente l'administrateur.
- Médecin.
- Secrétaire.
- Patient.
- Soin.
- Rapport.
- RDV.
- Consultation.
- Compteur : cette collection permet de générer des id pour les autres collections et stocker le prochaine id disponible.

```
collectionMedecin = BD["Medecin"]
collectionPatient = BD["Patient"]
collectionSecretaire = BD["Secretaire"]
collectionRDV = BD["RDV"]
collectionSoin = BD["Soin"]
collectionRapport = BD["Rapport"]
collectionConsultation = BD["Consultation"]
collectionDevis = BD["Devis"]
collectionFacture = BD["Facture"]
collectionPaieement = BD["Paieement"]
collectionAdmin = BD["Admin"]
collectionCompteur = BD["Compteur"]
```

FIGURE 3.2 – Code Création des collections.

### 3.1.3 Insertion dans la base de données

Pour la partie qui concerne l'insertion des informations dans notre base de données, On va utiliser les requêtes suivantes :

- Insertion d'un Médecin :

```
id = Fonction_Affichage_RDV.GenererID("Medecin")
requete = {"NumMed":id,"Login":pseudo,"Password":mdp,"Nom":nom,"Prenom":prenom,"Date-naissance":ddn,"Numero-tele":numeroT,"Sexe":sexe,"Adresse":addr}
testInsertion = CreationDB.collectionMedecin.insert_one(requete)
```

FIGURE 3.3 – Insertion d'un Médecin



---

- Insertion d'une Secrétaire :

```
id = Fonction_Affichage_RDV.GenererID("Secrétaire")
requete = {"NumSecrétaire":id,"Login":pseudo,"Password":mdp,"Nom":nom,"Prenom":prenom,"Date-naissance":ddn,"Numero-tele":numeroT,"Sexe":sexe,"Adresse":addr}
testInsertion = CreationDB.collectionSecrétaire.insert_one(requete)
```

FIGURE 3.4 – Insertion d'une Secrétaire

- Insertion d'un Rapport :

```
numRapport = Fonction_Affichage_RDV.GenererID("Rapport")
reponseRapport = CreationDB.collectionRapport.insert_one(
{
    "NumRapport": numRapport,
    "NumRDV": self.numrdv,
    "NumPat": self.numpat,
    "Chemine": cheminabsolue
})
```

FIGURE 3.5 – Insertion d'un Rapport

- Insertion d'un Rendez-vous :

```
numRDV = Fonction_Affichage_RDV.GenererID("RDV")
reponsePatient = CreationDB.collectionPatient.update_one(
{
    "NumPat": int(numeroPatient)},
{
    "$push":
    {
        "RDV":
        {
            "NumRDV": numRDV,
            "NumPatient": int(numeroPatient),
            "NumMed": numeroMedecin,
            "Motif": motif,
            "Date": date,
            "Heure": heure
        }
    }
})

reponseMedecin = CreationDB.collectionMedecin.update_one(
{
    "NumMed": int(numeroMedecin)},
{
    "$push":
    {
        "RDV":
        {
            "NumRDV": numRDV,
            "NumPatient": int(numeroPatient),
            "NumMed": numeroMedecin,
            "Motif": motif,
            "Date": date,
            "Heure": heure
        }
    }
})

reponseRDV = CreationDB.collectionRDV.insert_one(
{
    "NumRDV": numRDV,
    "NumPatient": int(numeroPatient),
    "NumMed": int(numeroMedecin),
    "Motif": motif,
    "Date": date,
    "Heure": heure
})
```

FIGURE 3.6 – Insertion d'un Rendez-vous

- Insertion d'un Patient :

```
idpatient = Fonction_Affichage_RDV.GenererID("Patient")
reponsePatient = CreationDB.collectionPatient.insert_one({"NumPat":idpatient,
                                                           "NumAssur":nassur,
                                                           "NumCIN":ncin,
                                                           "Nom":nom,
                                                           "Prenom":prenom,
                                                           "Numero-telephone":numero,
                                                           "Date-naissance":date,
                                                           "Sexe":sexe,
                                                           "Adresse":adresse,
                                                           })
```

FIGURE 3.7 – Insertion d'un Patient

## 3.2 Partie Traitements

Pour la partie du traitement on va traiter la création des fenêtres graphiques, la gestion des comptes et de login et finalement les principales tâches de notre système.

### 3.2.1 Création des Fenêtres

Pour Créer une fenêtre on a besoin d'un module qui s'appelle Tkinter, présent par défaut dans Python. Ce module permet de créer des interfaces graphiques, en offrant une passerelle entre Python et la bibliothèque Tk. Pour créer cette interface graphique avec Tkinter, il y a deux choses à faire :

- Créer une fenêtre racine.
- Lancer la boucle principale via la méthode `main Loop()`, l'appel à cette méthode bloque l'exécution de l'appelant.

```
from tkinter import Tk

fenetre = Tk() # Création de la fenêtre racine
fenetre.mainloop() # Lancement de la boucle principale
```

FIGURE 3.8 – Initialisation de la fenêtre

Lorsqu'on exécute le code de la «figure 3.2» on obtient une fenêtre avec le titre « tk » comme il est indiqué sur la «figure 3.3» ci-dessous

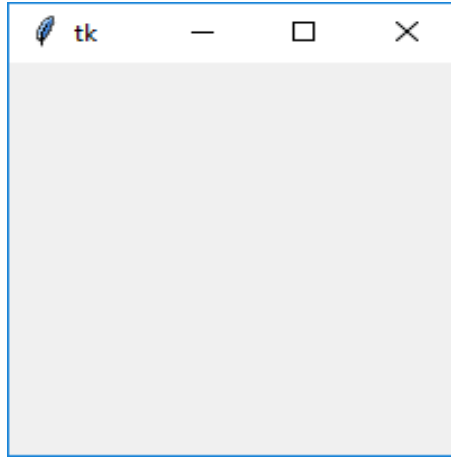


FIGURE 3.9 – Fenêtre Tk

Pour les fenêtres graphiques de notre système, on les a programmé chacune dans une classe indépendante. Ce qui permet un multifenêtrage qui offre une flexibilité et permet aussi aux utilisateurs de notre système d'organiser l'espace de travail selon la tâche à réaliser.

```
class FenetreLogin():
    def __init__(self):
        #initialisation de la fenetre
        self.fenetre = Tk()
        self.fenetre.title("Fenetre d'authentification")
        self.fenetre.iconbitmap("IHM\\logo.ico")
        largeurEcran = self.fenetre.winfo_screenwidth()
        hauteurEcran = self.fenetre.winfo_screenheight()
        geo = '{}x{}+{}+{}'.format(800,345,(largeurEcran // 2) - 400,(hauteurEcran // 2) - 235)
        self.fenetre.geometry(geo)
        self.fenetre.resizable(width=False, height=False)

        #fond d'ecran
        image_fond = PhotoImage(file="IHM\\login2.png")
        image_label = Label(self.fenetre,image=image_fond)
        image_label.place(x=0,y=0,relwidth=1,relheight=1)

        #Creation et insertion des composants
        color = '#C3e4f7'
        cadre = Frame(self.fenetre,bg=color,bd=30)
        cadre.grid(padx=40,pady=40)
        my_font = Font(family="Curlz MT",size=30, slant="italic")
        text_font = Font(family="fantasy",size=12,)
        self.label_titre = Label(cadre,text="LOGIN ",pady=10,bg=color,font=my_font)
        self.label_pseudo = Label(cadre,text="Pseudo :",bg=color,font=text_font)
        self.label_password = Label(cadre,text="Mot de passe :",bg=color,font=text_font)
        self.champ_pseudo = Entry(cadre)
        self.champ_password = Entry(cadre,show="*")
        self.boutton_confirmation = Button(cadre,text="Confirmer",command=self.verificationAuthe,font=text_font)
        self.boutton_creation = Button(cadre,text="Effacer",command=self.Effacer,font=text_font)
        self.label_titre.pack()
        self.label_pseudo.pack()
        self.champ_pseudo.pack()
        self.label_password.pack()
        self.champ_password.pack()
        self.boutton_confirmation.pack(pady=10,padx=10,side=LEFT)
```

FIGURE 3.10 – Code Fenêtre login

### 3.2.2 Liaison entre les Fenêtres

Puisque nos fenêtres sont encapsulées dans des classes, il suffit d'instancier un objet de la classe correspondante pour faire la liaison entre nos fenêtres. Par exemple, dans le menu de secrétaire dans la figure suivante, quand la secrétaire clique sur la gestion de rendez-vous, il sera affiché au-dessus de menu comme suit :

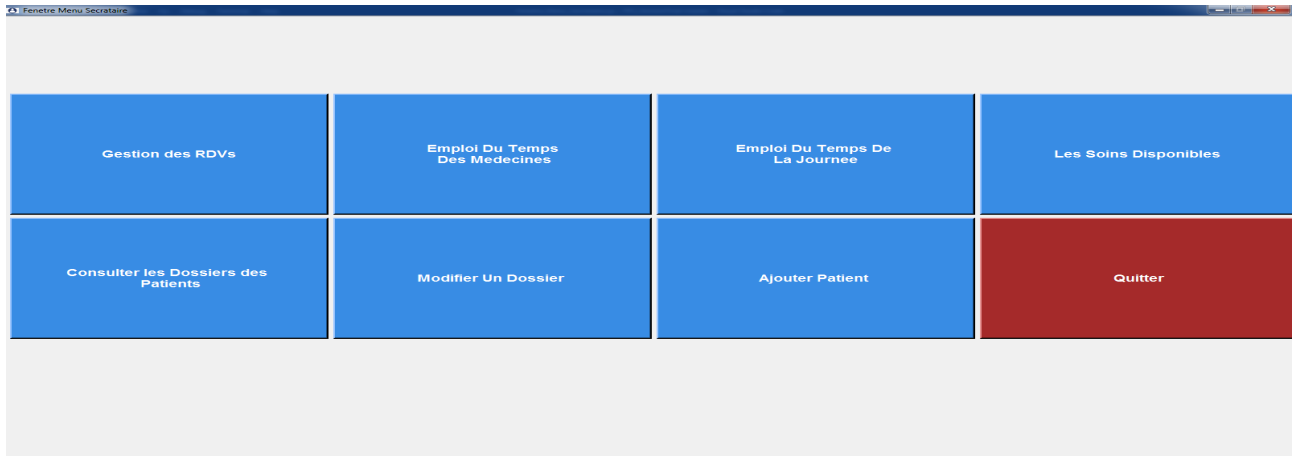


FIGURE 3.11 – Menu Secrétaire

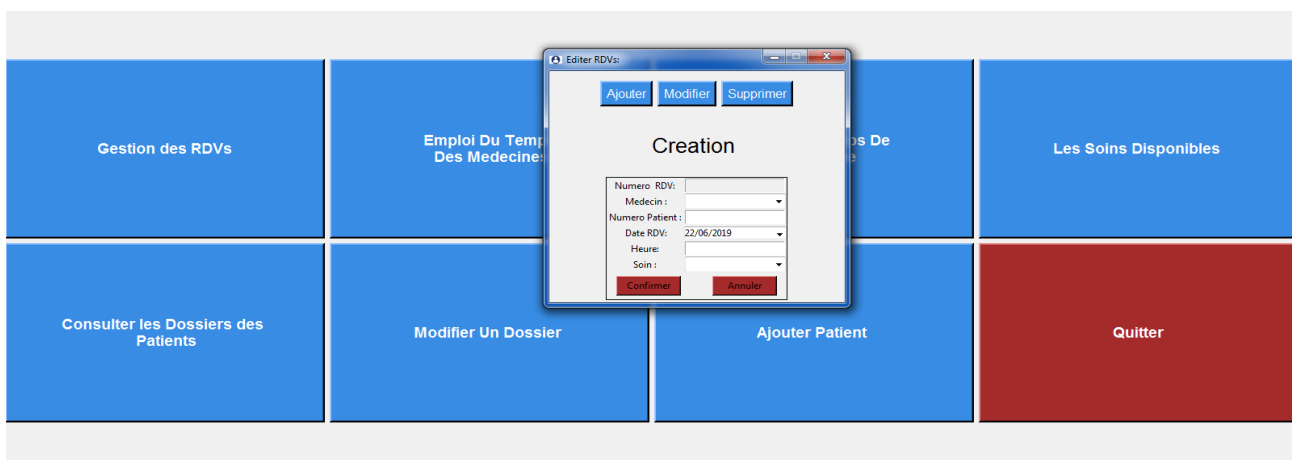


FIGURE 3.12 – Menu Secrétaire Après Clique Sur Gérer RDV

et ceci est réalisé avec le code suivant :

```
#Bouton pour creer une RDV
self.bouton_rdv = Button(self.cadre, text='Gestion des RDVs',width=(self.largeur//50),height=10,bg=color,fg='white',font=font,bd=3,command=Fenetre_Gerer_RDV.Fenetre_Gerer_RDV)
self.bouton_rdv.grid(row=1,column=1,padx=3,pady=3)
```

FIGURE 3.13 – Code Liaison entre fenêtres

### 3.2.3 Les fonctions Utilisées

On a utilisé plusieurs fonctions qui sont les suivantes :

- fonction affichage rendez-vous pour les médecins qui permet d’afficher les rendez-vous d’un médecin donné.

```
def Affichage_RDV_Medecin(cadreTab,reponserequete):  
  
    color='#99ffff'  
    label_Numero_RDV = Label(cadreTab, text="Numero RDV ",width=20,font=("bold", 11),bg=color)  
    label_Numero_Patient = Label(cadreTab, text=" N° Patient ",width=20,font=("bold", 11),bg=color)  
    label_Motif = Label(cadreTab, text=" Motif ",width=20,font=("bold", 11),bg=color)  
    label_Date = Label(cadreTab, text=" Date ",width=20,font=("bold", 11),bg=color)  
    label_Heure = Label(cadreTab, text=" Heure ",width=20,font=("bold", 11),bg=color)  
    label_Numero_RDV.grid(row=1,column=1)  
    label_Numero_Patient.grid(row=1,column=2)  
    label_Motif.grid(row=1,column=3)  
    label_Date.grid(row=1,column=4)  
    label_Heure.grid(row=1,column=5)  
    ligne =2  
    listRdv = multikeysort(reponserequete['RDV'],['Date','Heure'])  
    for RDV in listRdv :  
  
        champ_Numero_RDV = Entry(cadreTab)  
        champ_Numero_RDV.grid(row=ligne,column=1)  
        champ_Numero_RDV.insert(0,RDV['NumRDV'])  
        champ_Numero_RDV.configure(state='readonly')  
  
        champ_Numero_Patient = Entry(cadreTab)  
        champ_Numero_Patient.grid(row=ligne,column=2)  
        champ_Numero_Patient.insert(0,RDV['NumPatient'])  
        champ_Numero_Patient.configure(state='readonly')  
  
        champ_Motif = Entry(cadreTab)  
        champ_Motif.grid(row=ligne,column=3)  
        champ_Motif.insert(0,RDV['Motif'])  
        champ_Motif.configure(state='readonly')  
  
        champ_Date = Entry(cadreTab)  
        champ_Date.grid(row=ligne,column=4)  
        champ_Date.insert(0,RDV['Date'])  
        champ_Date.configure(state='readonly')  
  
        champ_Heure = Entry(cadreTab)  
        champ_Heure.grid(row=ligne,column=5)  
        champ_Heure.insert(0,RDV['Heure'])  
        champ_Heure.configure(state='readonly')  
        ligne = ligne + 1
```

FIGURE 3.14 – Fonction Affichage Rendez-vous Médecin

- fonction affichage rendez-vous pour les patients qui permet d’afficher les rendez-vous d’un patient donnée.

```
def Affichage_RDV_Patient(cadreTab,reponserequete,numpatient,medecin):  
  
    color='#99ffff'  
    label_Numero_RDV = Label(cadreTab, text="Numero RDV ",width=20,font=("bold", 11),bg=color)  
    label_Numero_Medecin = Label(cadreTab, text=" N° Medecin ",width=20,font=("bold", 11),bg=color)  
    label_Motif = Label(cadreTab, text=" Motif ",width=20,font=("bold", 11),bg=color)  
    label_Date = Label(cadreTab, text=" Date ",width=20,font=("bold", 11),bg=color)  
    label_Heure = Label(cadreTab, text=" Heure ",width=20,font=("bold", 11),bg=color)  
    label_Numero_RDV.grid(row=1,column=1)  
    label_Numero_Medecin.grid(row=1,column=2)  
    label_Motif.grid(row=1,column=3)  
    label_Date.grid(row=1,column=4)  
    label_Heure.grid(row=1,column=5)  
    ligne =2  
  
    listRdv = multikeysort(reponserequete['RDV'],['Date','Heure'])  
    listboutonajouter = list()  
    listboutonconsulter = list()  
    indicebutton = 0  
    for RDV in listRdv:  
  
        champ_Numero_RDV = Entry(cadreTab)  
        champ_Numero_RDV.grid(row=ligne,column=1)  
        champ_Numero_RDV.insert(0,RDV['NumRDV'])  
        champ_Numero_RDV.configure(state='readonly')  
  
        champ_Numero_Medecin = Entry(cadreTab)  
        champ_Numero_Medecin.grid(row=ligne,column=2)  
        champ_Numero_Medecin.insert(0,RDV['NumMed'])  
        champ_Numero_Medecin.configure(state='readonly')  
  
        champ_Motif = Entry(cadreTab)  
        champ_Motif.grid(row=ligne,column=3)  
        champ_Motif.insert(0,RDV['Motif'])  
        champ_Motif.configure(state='readonly')  
  
        champ_Date = Entry(cadreTab)  
        champ_Date.grid(row=ligne,column=4)  
        champ_Date.insert(0,RDV['Date'])  
        champ_Date.configure(state='readonly')  
  
        champ_Heure = Entry(cadreTab)  
        champ_Heure.grid(row=ligne,column=5)  
        champ_Heure.insert(0,RDV['Heure'])  
        champ_Heure.configure(state='readonly')  
        if medecin:  
            listboutonajouter.append(Button(cadreTab,text="Ajouter un Rapport",padx=3,  
                bg='brown',command=lambda numrdv=RDV['NumRDV'],numpat=numpatient,  
                nummed=RDV['NumMed']: appelCreation(numrdv,numpat,nummed)))  
            listboutonajouter[indicebutton].grid(row=ligne,column=6)  
            listboutonconsulter.append(Button(cadreTab,text="Consulter les rapports",  
                bg='brown',command=lambda numrdv=RDV['NumRDV'],numpat=numpatient,  
                nummed=RDV['NumMed']: appelConsultation(numrdv,numpat,nummed)))  
            listboutonconsulter[indicebutton].grid(row=ligne,column=7)  
            ligne = ligne + 1  
            indicebutton = indicebutton + 1  
    def appelCreation(numrdv,numpatient,nummed):  
        Fenetre_Creation_Rapport.Fenetre_Creation_Rapport(numpatient,nummed,numrdv).fenetre.mainloop()  
    def appelConsultation(numrdv,numpatient,nummed):  
        Fenetre_Consultation_Rapport(numrdv,numpatient,nummed)
```

FIGURE 3.15 – Fonction Affichage Rendez-vous Patient

- fonction affichage rendez-vous de la journée qui permet d’afficher les rendez-vous qui sont programmés dans la journée actuelle.

```
def Affichage_RDV_Journee(cadreTab,date_courante):
    color="#99ffff"
    label_Numero_RDV = Label(cadreTab, text="Numero RDV ",width=20,font=("bold", 11),bg=color)
    label_Numero_Patient = Label(cadreTab, text=" N° Patient ",width=20,font=("bold", 11),bg=c
    label_Numero_Medecin = Label(cadreTab, text=" N° Medecin ",width=20,font=("bold", 11),bg=c
    label_Motif = Label(cadreTab, text=" Motif ",width=20,font=("bold", 11),bg=color)
    label_Heure = Label(cadreTab, text=" Heure ",width=20,font=("bold", 11),bg=color)
    label_Numero_RDV.grid(row=1,column=1)
    label_Numero_Patient.grid(row=1,column=2)
    label_Numero_Medecin.grid(row=1,column=3)
    label_Motif.grid(row=1,column=4)
    label_Heure.grid(row=1,column=5)

    reponserequete = CreationDB.collectionRDV.find({"Date":str(date_courante)},{"_id":0})

    if reponserequete == None :
        messagebox.showerror("Error", "Erreur lors de l'acces a la BD Resseayer plus tard")
    if reponserequete.count() == 0 :
        messagebox.showerror("Error", "Aucune Rendez-Vous Prevu Pour Cette Journee")
    if reponserequete != None:
        ligne = 2
        for RDV in multikeysort(reponserequete,['Date','Heure']):
            champ_Numero_RDV = Entry(cadreTab)
            champ_Numero_RDV.grid(row=ligne,column=1)
            champ_Numero_RDV.insert(0,RDV['_id'])
            champ_Numero_RDV.configure(state='readonly')

            champ_Numero_Patient = Entry(cadreTab)
            champ_Numero_Patient.grid(row=ligne,column=2)
            champ_Numero_Patient.insert(0,RDV['NumPatient'])
            champ_Numero_Patient.configure(state='readonly')

            champ_Numero_Medecin = Entry(cadreTab)
            champ_Numero_Medecin.grid(row=ligne,column=3)
            champ_Numero_Medecin.insert(0,RDV['NumMed'])
            champ_Numero_Medecin.configure(state='readonly')

            champ_Motif = Entry(cadreTab)
            champ_Motif.grid(row=ligne,column=4)
            champ_Motif.insert(0,RDV['Motif'])
            champ_Motif.configure(state='readonly')

            champ_Heure = Entry(cadreTab)
            champ_Heure.grid(row=ligne,column=5)
            champ_Heure.insert(0,RDV['Heure'])
            champ_Heure.configure(state='readonly')

            ligne = ligne + 1
```

FIGURE 3.16 – Fonction Affichage Rendez-vous Journée

- fonction générer id permet de donner le prochain id disponible.

```
def GenererID(collection):
    idGenerer = False
    if collection == "Patient":
        reponse = CreationDB.collectionCompteur.find_one(
            {'entite':
             "Patient"},
            {'_id':0
            })
        if reponse != None:
            idGenerer = reponse['id_dispo']
            CreationDB.collectionCompteur.update(
                {'entite':"Patient"},
                {'$inc':
                 {'id_dispo': 1
                 }}
            )
        if collection == "Medecin":
            reponse = CreationDB.collectionCompteur.find_one(
                {'entite':
                 "Medecin"},
                {'_id':0
                })
            if reponse != None:
                CreationDB.collectionCompteur.update(
                    {'entite':"Medecin"
                    },
                    {'$inc':
                     {'id_dispo':1
                     }}
                )
            idGenerer = reponse['id_dispo']

    if collection == "RDV":
        reponse = CreationDB.collectionCompteur.find_one(
            {'entite':"RDV"
            },
            {'_id':0
            })
        if reponse != None:
            CreationDB.collectionCompteur.update(
                {'entite':"RDV"
                },
                {'$inc':
                 {'id_dispo': 1
                 }}
            )
            idGenerer = reponse['id_dispo']
        if collection == "Secretaire":
            reponse = CreationDB.collectionCompteur.find_one(
                {'entite':"Secretaire"
                },
                {'_id':0
                })
            if reponse != None:
                CreationDB.collectionCompteur.update(
                    {'entite':"Secretaire"
                    },
                    {'$inc':
                     {'id_dispo': 1
                     }}
                )
            idGenerer = reponse['id_dispo']

    if collection == "Rapport":
        reponse = CreationDB.collectionCompteur.find_one(
            {'entite':
             "Rapport"
            },
            {'_id':0
            })
        if reponse != None:
            CreationDB.collectionCompteur.update(
                {'entite':
                 "Rapport"},
                {'$inc':
                 {'id_dispo': 1
                 }}
            )
        idGenerer = reponse['id_dispo']

    return idGenerer
```

FIGURE 3.17 – Fonction Générer Id

### 3.2.4 Gestion des Comptes

La gestion des comptes est une tâche d'administrateur de bases de données s'est à lui de créer, modifier, et éventuellement supprimer un compte. On a créé une session administratrice qui offre une fenêtre graphique qui facilite la tâche à l'admin.

- **Pour ajouter un compte :**

Le formulaire ci-dessous permet d'ajouter un compte Médecin/Secrétaire avec toutes les informations nécessaires, et ensuite envoyer une requête d'insertion pour stocker le compte dans la collection correspondante, après les vérifications nécessaires des données entrées.

Ajouter Un Compte   Modifier Un Compte   Supprimer Un Compte

## Enregistrement

Nom :

Prenom :

Date Naissance : 22/06/2019

Sexe : ☒ Male ☐ Female

Numero Telephone :

Adresse :

Session : ☒ Medecin ☐ Secretaire

Pseudo :

Mot De Passe :

Ressaisie le Mot De Passe :

Confirmer   Annuler

FIGURE 3.18 – Fenêtre Ajout d'un compte

- **Pour modifier un compte :**

1. Chercher le compte : Avant de modifier un compte il faut d'abord chercher le compte. Ce qui est possible avec le formulaire suivant.

Ajouter Un Compte   Modifier Un Compte   Supprimer Un Compte

## Chercher Le Compte

Numero Compte :

Session : ☒ Medecin ☐ Secretaire

Chercher

FIGURE 3.19 – Fenêtre Recherche un Utilisateur

- 
2. modifier le compte : Après la recherche, si on trouve le compte, on affiche le formulaire avec les données récupérées de la base de données, avec la possibilité de modifier.

The screenshot shows a web interface with a top navigation bar containing three buttons: 'Ajouter Un Compte', 'Modifier Un Compte', and 'Supprimer Un Compte'. The main content area is titled 'Modifier Un Compte'. It contains a form with the following fields: 'Nom :' with value 'xxxxx', 'Prenom :' with value 'xxxxx', 'Date Naissance :' with a dropdown menu showing '1997-05-14', 'Numero Telephone :' with value '0909090909', 'Adresse :' with value 'daindoa', 'Pseudo :' with value 'xxxx', and 'Mot De Passe :' with value '12'. At the bottom of the form are two buttons: 'Modifier' and 'Annuler'.

FIGURE 3.20 – Fenêtre Modifier un Utilisateur

- Pour supprimer un compte :

Pour supprimer un compte il suffit d'entrer le numéro de compte.

The screenshot shows a web interface with a top navigation bar containing three buttons: 'Ajouter Un Compte', 'Modifier Un Compte', and 'Supprimer Un Compte'. The main content area is titled 'Supprimer Un Compte'. It contains a form with the following fields: 'Numero Compte :' with an empty text input field, and 'Session :' with two radio buttons labeled 'Medecin' and 'Secretaire'. At the bottom of the form is a button labeled 'Supprimer'.

FIGURE 3.21 – Fenêtre Supprimer un Utilisateur



### 3.2.5 Gestion de Login

Premièrement on affiche la fenêtre de login ci dessous :



FIGURE 3.22 – Fenêtre Login

Après la verification des champs, on envoie les requêtes au 3 collections : (Admin,Secrétaire,Médecin) comme il est indiquée dans la figure ci-dessous.

```
# pour verifier est ce que l'utilisateur a remplis tous les champs de saisie
if self.champ_pseudo.get() == "" or self.champ_password.get() == "":
    messagebox.showerror("Error", "Veuillez Remplir tous Les Champs !!")
else:
    # On Envoie une requete a la DataBase pour verifier les coordonnees de l'utilisateur
    login = self.champ_pseudo.get()
    password = self.champ_password.get()
    test_secretaire = CreationDB.collectionSecrétaire.find_one({"Login":login,"Password":password},{})
    test_medecin = CreationDB.collectionMédecin.find_one({"Login":login,"Password":password},{})
    test_Admin = CreationDB.collectionAdmin.find_one({"pseudo":login,"password":password},{})

    if test_secretaire != None:
        # si c'est la secretaire on ouvre la session de la secretaire
        self.fenetre.destroy()
        f = Fenetre_Menu_Secrétaire.Fenetre_Menu_Secrétaire()
        f.fenetre

    elif test_medecin !=None:
        # si c'est le medecin on ouvre la session du medecine , A continuer
        self.fenetre.destroy()
        f = Fenetre_Menu_Médecin.Fenetre_Menu_Médecin()
        f.fenetre

    elif test_Admin != None :
        self.fenetre.destroy()
        Fenetre_inscription.Fenetre_inscription().fenetre

    else:
        #si on trouve pas d'element on affiche une erreur
        messagebox.showerror("Error", "Pseudo/mot de passe incorrect !!")
```

FIGURE 3.23 – Code Connexion

Si le pseudo et mot de passe correspondent à une session des 3, on ouvre le menu de cette session. Sinon on affiche un message d'erreur.

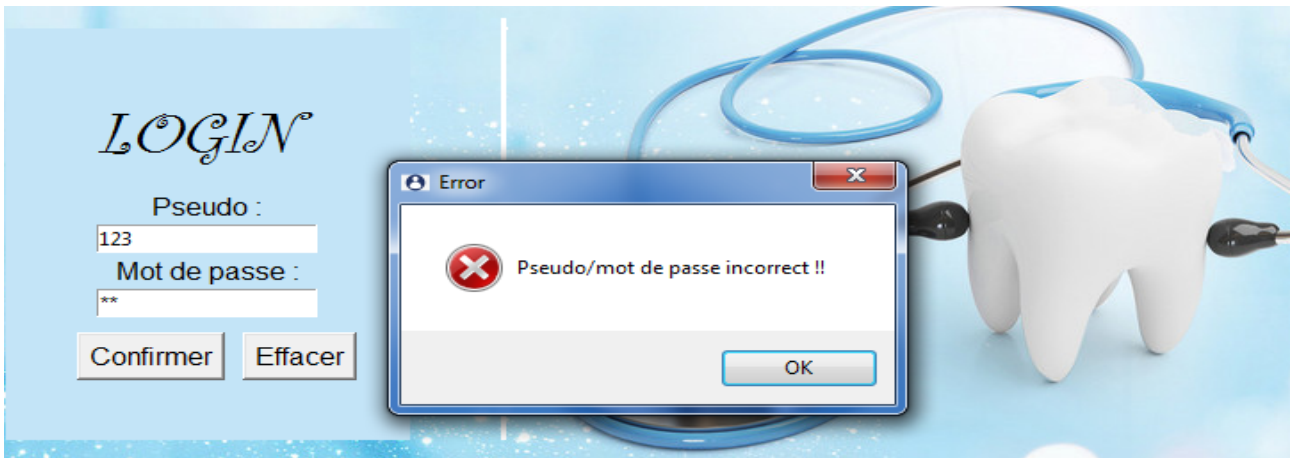


FIGURE 3.24 – Fenêtre d'erreur login

### 3.2.6 Gestion des Médecins

Pour la session Médecin, ce dernier peut faire les actions suivantes :

- **Clôturer un soin :**

Après la saisie du numéro de rendez-vous , cette option permet de clôturer un soin s'il est terminé, et interdit l'ajout dans le rapport du patient.



FIGURE 3.25 – Fenêtre Clôturer Soin

- **Emploi des rendez-vous :**

Cette fenêtre offre la possibilité de choisir un médecin parmi la liste des médecins disponibles dans la base de données, et après la validation, elle affiche l'emploi des rendez-vous programmés pour le médecin sous forme d'un tableau.

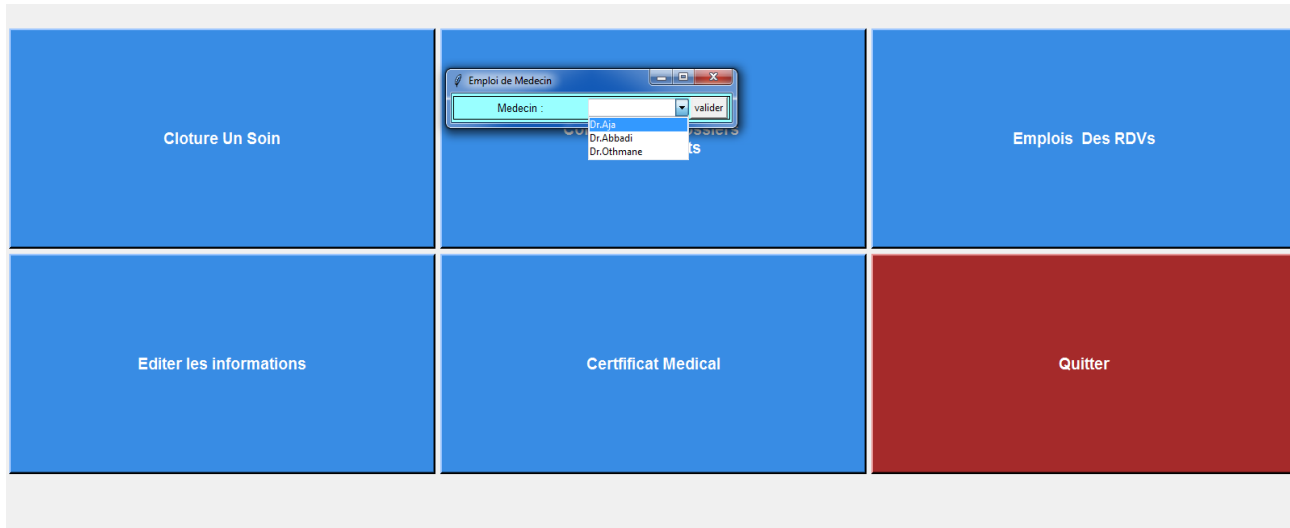


FIGURE 3.26 – Fenêtre Choix Médecin

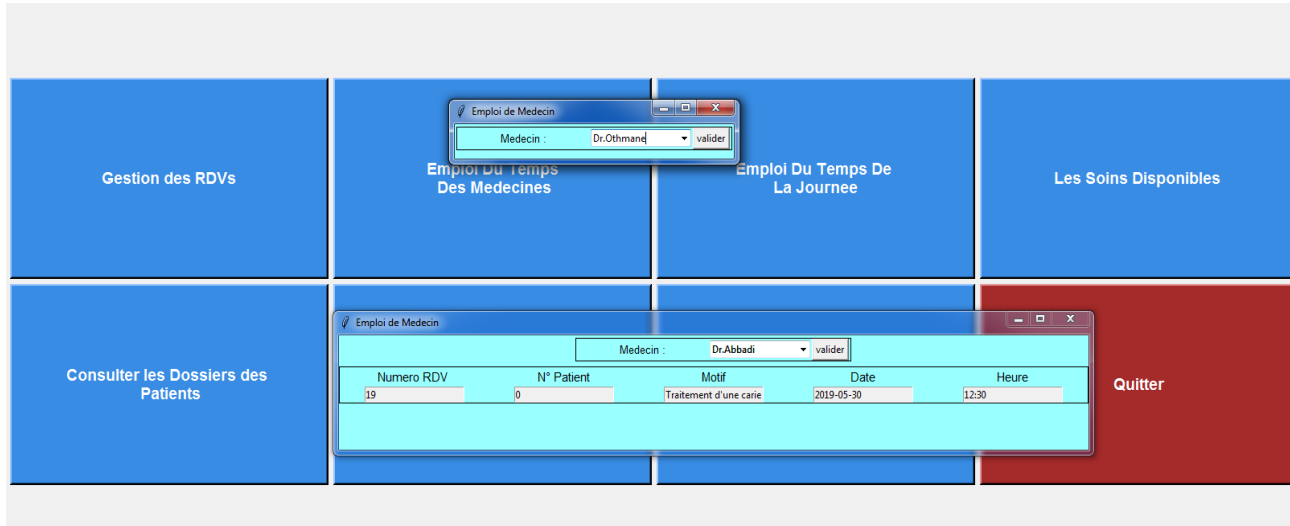


FIGURE 3.27 – Fenêtre Emploi des rendez-vous

- **Rediger un rapport :**

Le médecin et la secrétaire peuvent consulter les rapports, mais seul le médecin peut rédiger un nouveau rapport à l'aide de la fenêtre dans la figure ci-dessous.

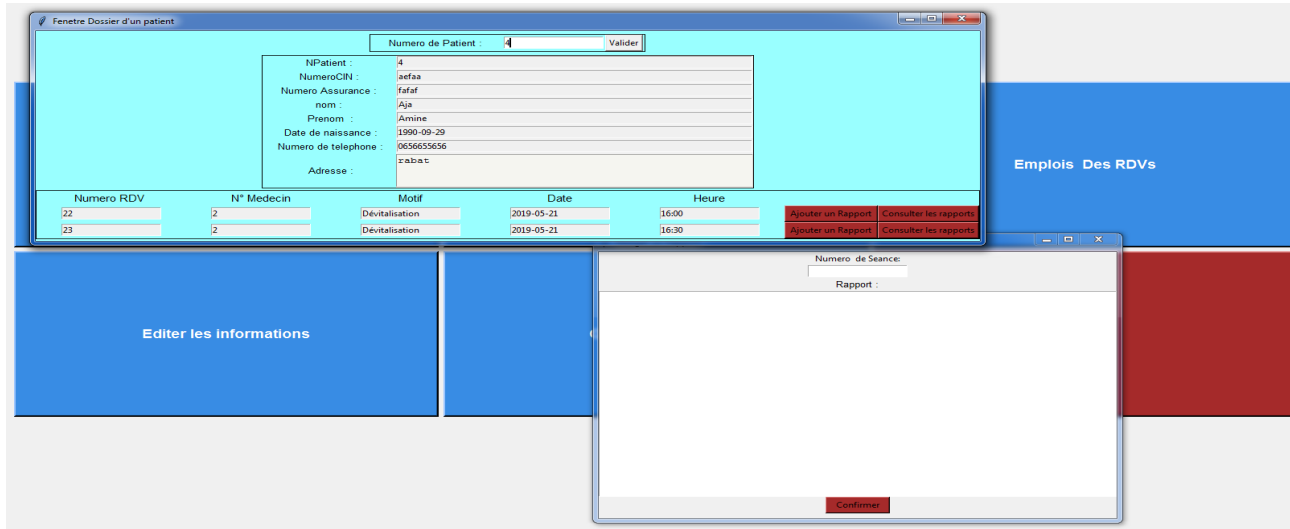


FIGURE 3.28 – Fenêtre Rédiger un rapport

- **Créer un certificat médical :**

Le médecin éventuellement peut rédiger un certificat pour un patient donné, à l'aide de la fenêtre dans la figure ci-dessous.

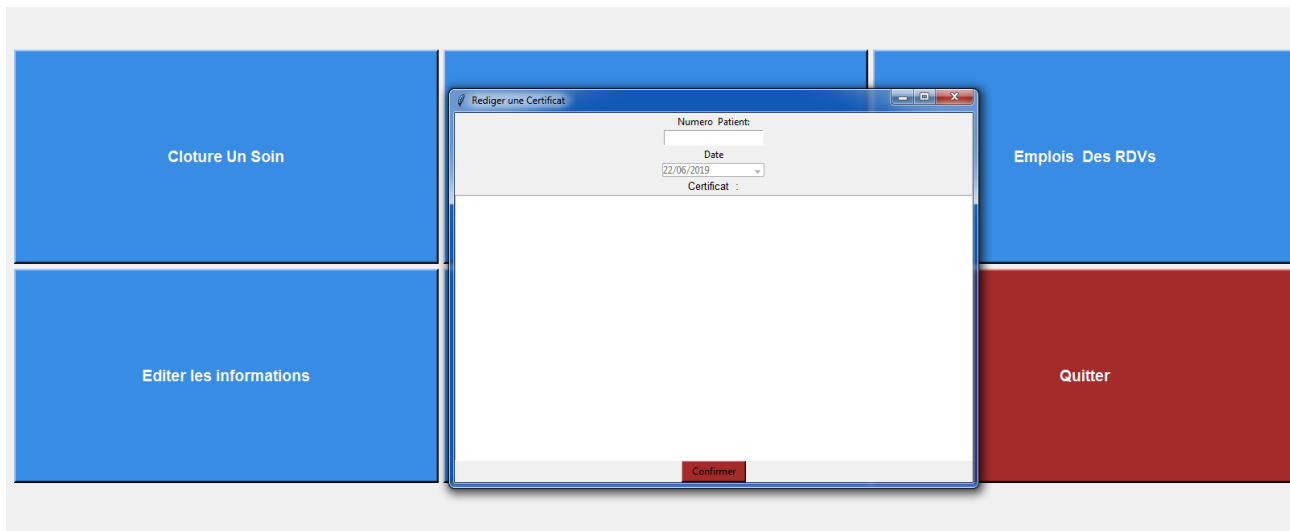


FIGURE 3.29 – Fenêtre Rédiger un Certificat

### 3.2.7 Gestion des patients

Pour la gestion d'un patient donné :

- **Ajouter un patient :**

La secrétaire peut ajouter un patient en entrant les informations du patient dans la fenêtre ci-dessous, qui affiche ensuite le numéro du patient dans la base de données.

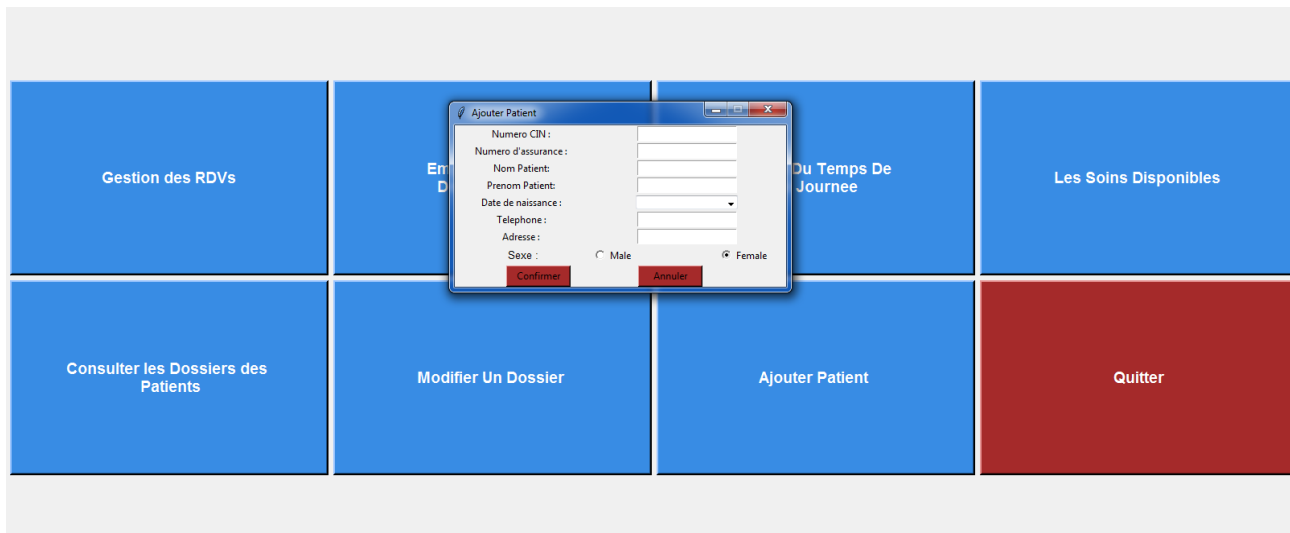


FIGURE 3.30 – Fenêtre Ajout d'un Patient

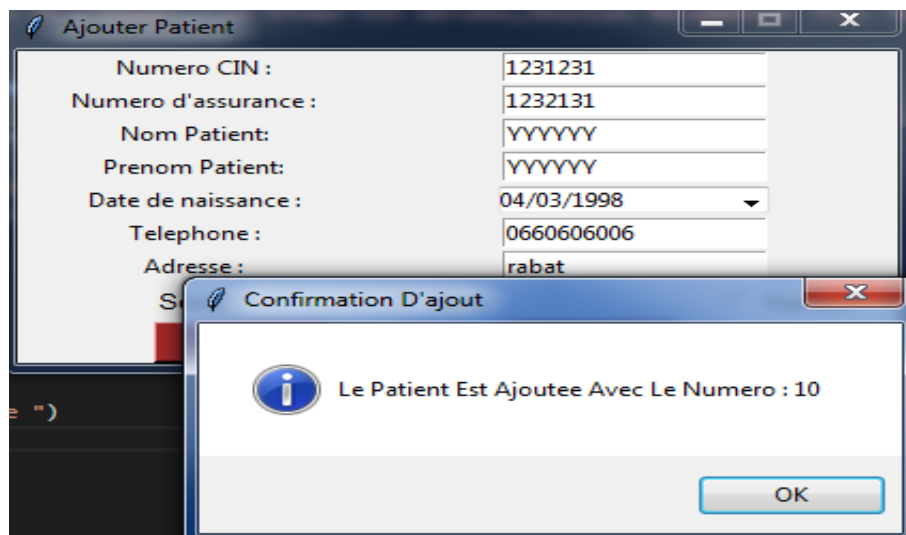


FIGURE 3.31 – Fenêtre Ajout d'un Patient

- **Modifier les informations d'un patient :**

La modification se passe en 2 étapes, on cherche le patient dans la base de données et ensuite les données seront affichées dans un formulaire avec la possibilité de les modifier.

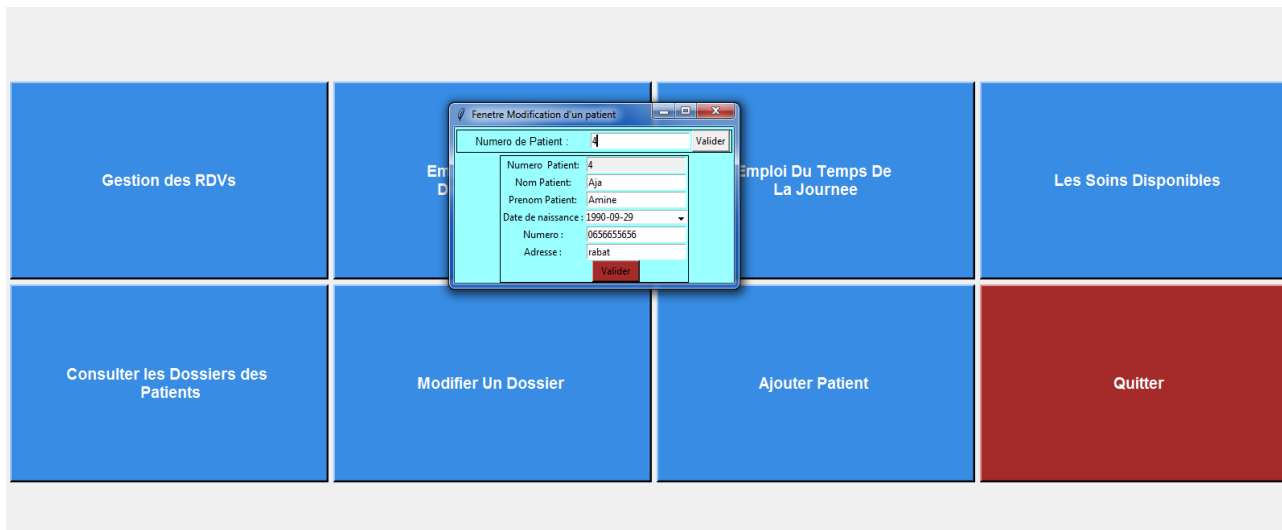


FIGURE 3.32 – Fenêtre Modification d'un Patient

- **Consulter le dossier d'un patient :**

Pour afficher le dossier d'un patient donné on entre son numéro, et il s'affiche avec tout les rendez-vous et les informations personnelles de ce dernier.

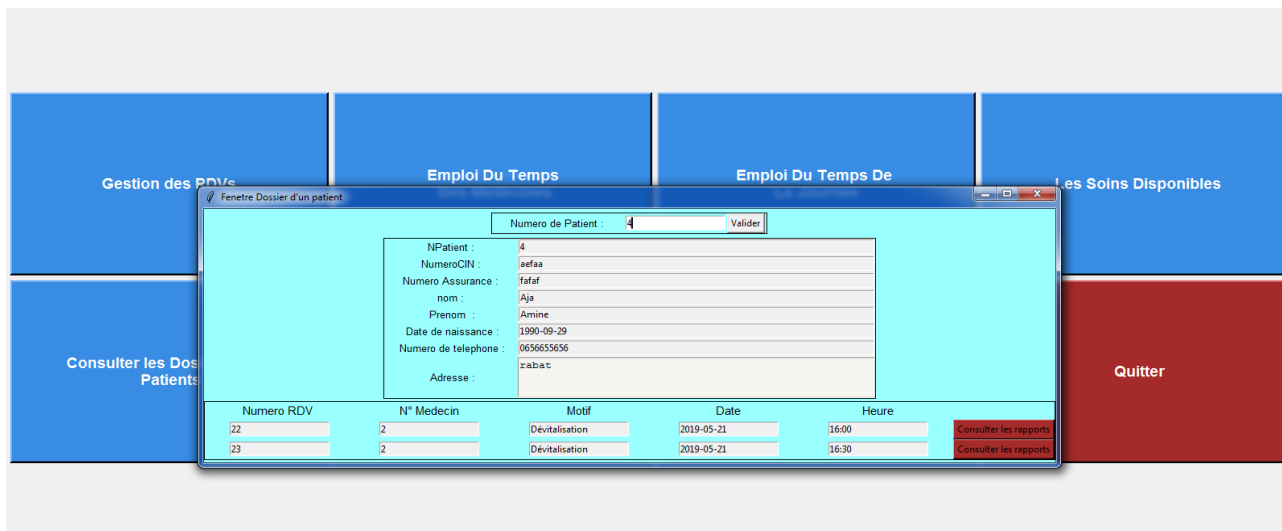


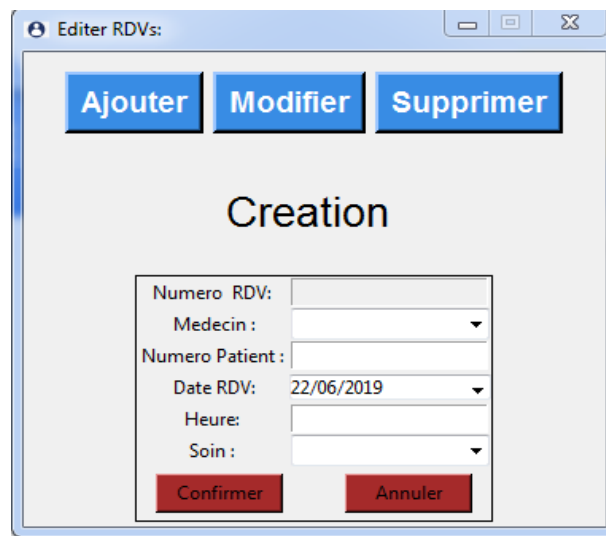
FIGURE 3.33 – Fenêtre Consultation d'un Patient

### 3.2.8 Gestion des Rdv

Pour la gestion des rendez-vous c'est une tâche de la secrétaire et cette dernière peut éventuellement :

- **Créer un rendez-vous :**

Cette fenêtre offre la possibilité de créer un rendez-vous pour un patient donné avec un médecin, dans une date et heure précises pour un certain soin. Après la confirmation, le rendez-vous s'ajoute à la base de données.

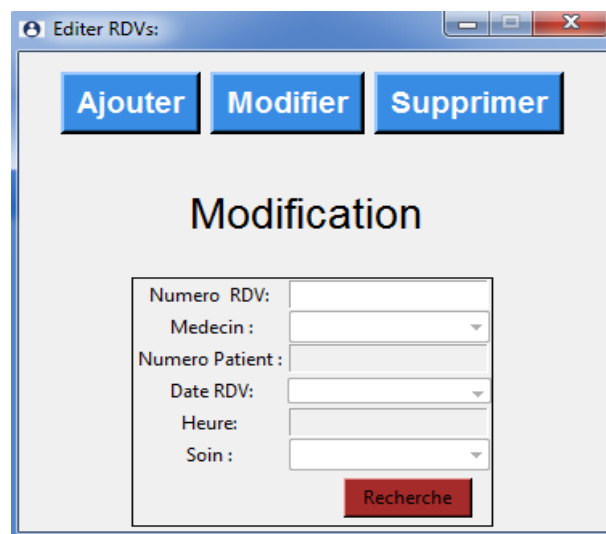


The screenshot shows a window titled "Editer RDVs:". At the top, there are three blue buttons: "Ajouter", "Modifier", and "Supprimer". Below these buttons, the word "Creation" is centered. Under "Creation", there is a form with the following fields: "Numero RDV:" (text input), "Medecin :" (dropdown menu), "Numero Patient :" (text input), "Date RDV:" (dropdown menu showing "22/06/2019"), "Heure:" (text input), and "Soin :" (dropdown menu). At the bottom of the form are two red buttons: "Confirmer" and "Annuler".

FIGURE 3.34 – Fenêtre Ajout d'un Rendez-vous

- **Modifier un rendez-vous :**

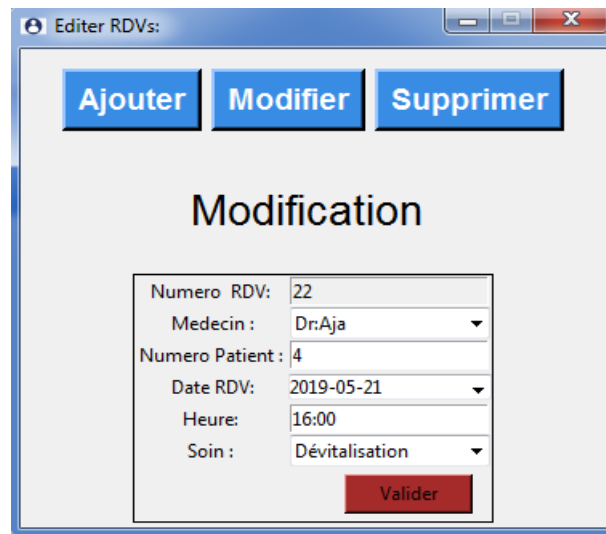
Cette Fenêtre permet de chercher un rendez-vous par Numero de Rendez-vous.



The screenshot shows a window titled "Editer RDVs:". At the top, there are three blue buttons: "Ajouter", "Modifier", and "Supprimer". Below these buttons, the word "Modification" is centered. Under "Modification", there is a form with the following fields: "Numero RDV:" (text input), "Medecin :" (dropdown menu), "Numero Patient :" (text input), "Date RDV:" (dropdown menu), "Heure:" (text input), and "Soin :" (dropdown menu). At the bottom of the form is a single red button: "Recherche".

FIGURE 3.35 – Fenêtre Recherche d'un Rendez-vous

Après la recherche du rendez-vous, on obtient toutes les données nécessaires du rendez-vous. Puis on peut modifier les informations de ce rendez-vous.

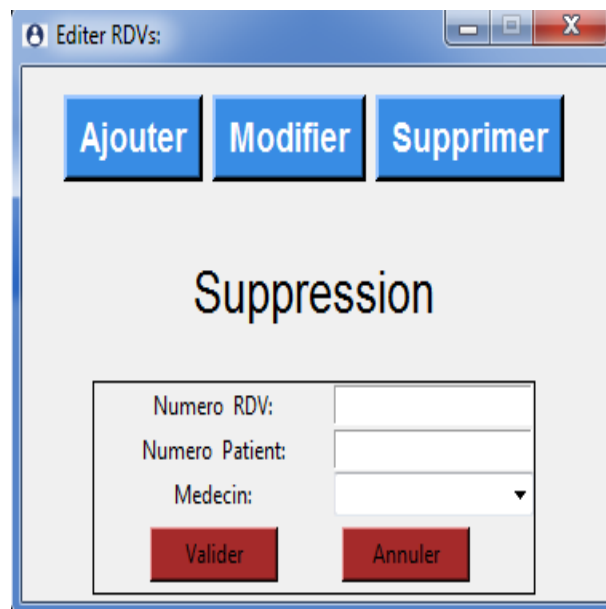


The screenshot shows a window titled "Editer RDVs:" with three buttons at the top: "Ajouter", "Modifier", and "Supprimer". The "Modifier" button is selected. Below the buttons, the word "Modification" is centered. A form contains the following fields: "Numero RDV:" with the value "22", "Medecin :" with a dropdown menu showing "Dr:Aja", "Numero Patient :" with the value "4", "Date RDV:" with a dropdown menu showing "2019-05-21", "Heure:" with the value "16:00", and "Soin :" with a dropdown menu showing "Dévitalisation". A red "Valider" button is at the bottom right of the form.

FIGURE 3.36 – Fenêtre Modification d'un Rendez-vous

- **Supprimer un rendez-vous :**

Pour supprimer un rendez-vous la secrétaire doit entrer le numéro du rendez-vous, médecin et numéro du patient.



The screenshot shows a window titled "Editer RDVs:" with three buttons at the top: "Ajouter", "Modifier", and "Supprimer". The "Supprimer" button is selected. Below the buttons, the word "Suppression" is centered. A form contains the following fields: "Numero RDV:" with an empty text box, "Numero Patient:" with an empty text box, and "Medecin:" with a dropdown menu. Two red buttons, "Valider" and "Annuler", are at the bottom of the form.

FIGURE 3.37 – Fenêtre Suppression d'un Rendez-vous



### 3.2.9 Gestion des Rapports Certificats

En ce qui concerne la gestion des rapports, seul le Medecin peut rediger un rapport. Quant à la secrétaire, elle ne peut que l'utiliser.

- **hiérarchie des dossiers :**

Après l'ajout d'un patient, un dossier se crée avec le nom et le numéro du patient qui contient 2 sous-dossiers(rapport, certificat). Dans ces derniers, on stocke les rapports et les certificats de tous les patients dans la base des données.




 Aja_Aja	06/06/2019 15:40	Dossier de fichiers
 Ha_Hamid	20/06/2019 16:41	Dossier de fichiers
 xxxxx_xxxxx	22/06/2019 17:52	Dossier de fichiers

FIGURE 3.38 – Dossier Des Patients



 Certificat	22/06/2019 17:52	Dossier de fichiers
 Rapport	22/06/2019 17:53	Dossier de fichiers

FIGURE 3.39 – Dossier D'un Patient

- **Rapport :**

1. Rediger Rapport :

Quand le médecin rédige un rapport, il sera enregistré dans la base de données avec le chemin de fichier dans la machine et les autres caractéristiques. Comme l'indique la figure suivante .

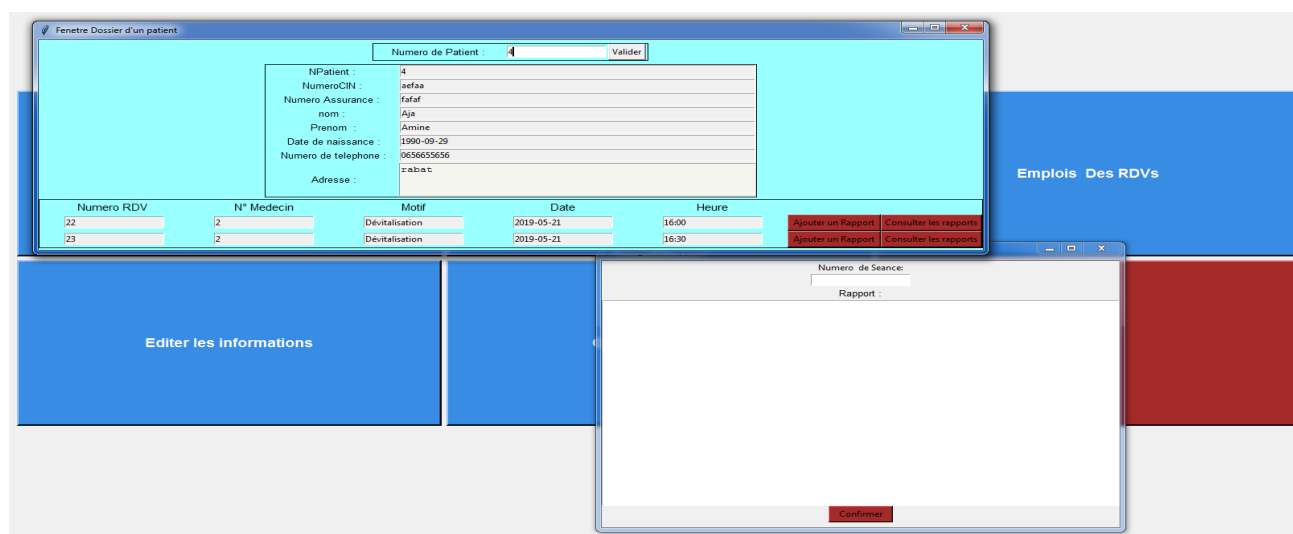


FIGURE 3.40 – Fenêtre Rédaction d'un Rapport

```

{
  "_id" : ObjectId<"5ce36bf3c941425fc53462e">,
  "NumRapport" : 11,
  "NumRDV" : 22,
  "NumPat" : 4,
  "Chemine" : "C:\\Users\\Othmane\\Desktop\\PFE_VersionFinal-master\\Dossi
erdesPatients\\Aja_Amine\\Rapport\\Aja_Amine_22.pdf"
}
{
  "_id" : ObjectId<"5ce465243c941404c4ed2b08">,
  "NumRapport" : 12,
  "NumRDV" : 23,
  "NumPat" : 4,
  "Chemine" : "C:\\Users\\Othmane\\Desktop\\PFE_VersionFinal-master\\Dossi
erdesPatients\\Aja_Amine\\Rapport\\Aja_Amine_23.pdf"
}
{
  "_id" : ObjectId<"5d0e6b083c94141a54b9eca3">,
  "NumRapport" : 13,
  "NumRDV" : 24,
  "NumPat" : 9,
  "Chemine" : "C:\\Users\\Othmane\\Desktop\\PFE_VersionFinal-master\\Dossi
erdesPatients\\xxxxx_xxxxx\\Rapport\\xxxxx_xxxxx_24.pdf"
}
}

```

FIGURE 3.41 – Bases données du rapport

## 2. Consulter Rapport :

Quand la secrétaire ou le médecin clique sur consulter, le rapport sera ouvert dans un lecteur PDF.

The screenshot shows a medical software interface. A central window titled "Fenêtre Dossier d'un patient" is open, displaying patient information and a table of appointments (RDV). The background shows a menu with options like "Gestion des RDV", "Emploi Du Temps", "Les Soins Disponibles", and "Consulter les Dossiers Patients".

**Patient Information:**

Numero de Patient :	4
NPatient :	4
NumeroCIN :	sefaa
Numero Assurance :	sfaf
nom :	Aja
Prenom :	Amine
Date de naissance :	1990-09-29
Numero de telephone :	0656655656
Adresse :	rabat

**Appointment Table (RDV):**

Numero RDV	N° Medecin	Motif	Date	Heure	Action
22	2	Dévitalisation	2019-05-21	16:00	Consulter les rapports
23	2	Dévitalisation	2019-05-21	16:30	Consulter les rapports

FIGURE 3.42 – Fenêtre Consulter un Rapport

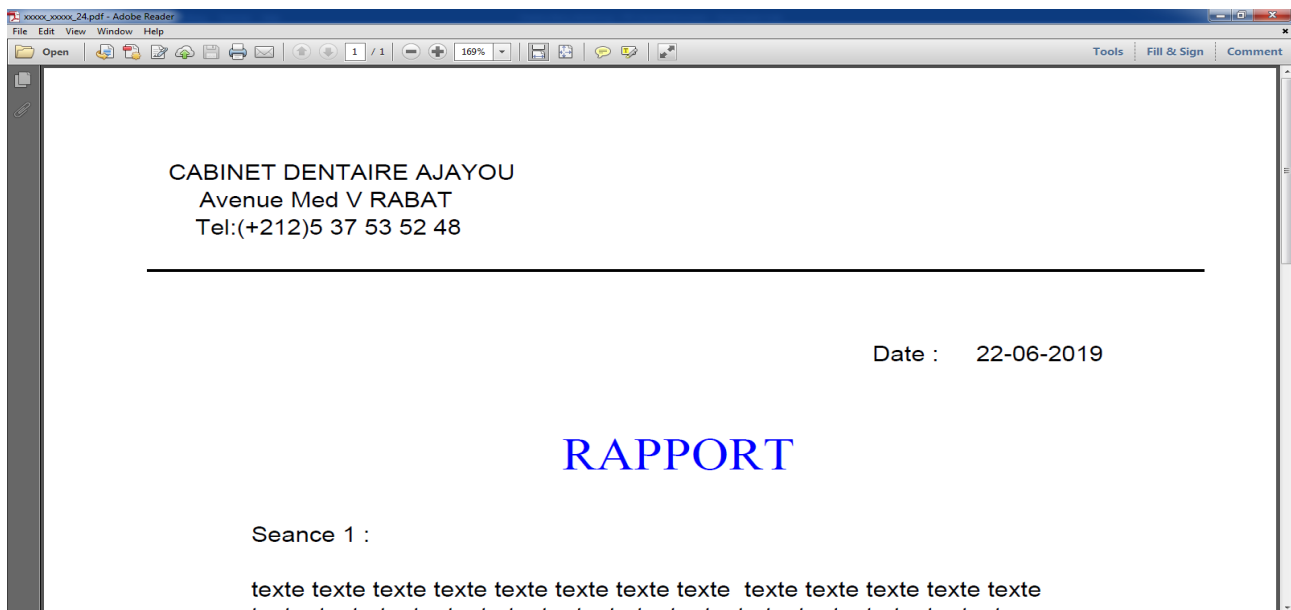


FIGURE 3.43 – Affichage d'un Rapport

- **Certificat :**

1. Rediger Certificat : Le certificat est rédigé à partir de la fenêtre de la figure ci-dessous

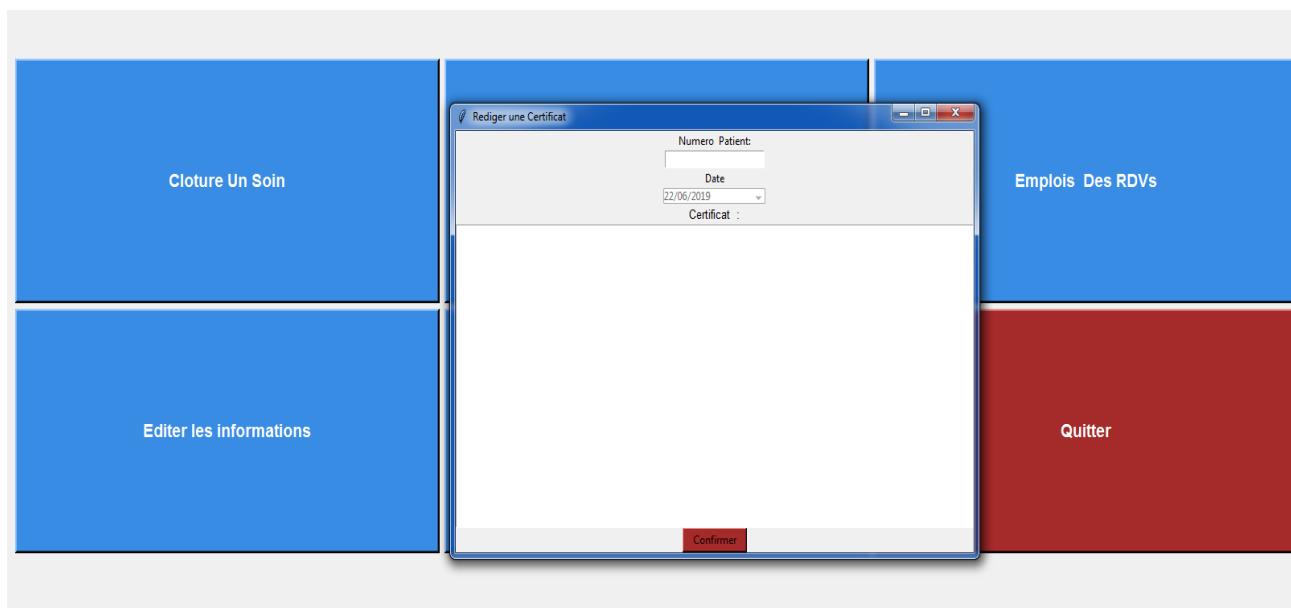


FIGURE 3.44 – Fenêtre Rédaction d'un Certificat

2. Consulter Certificat : Pour la consultation des certificats, ils sont enregistrés sur la machine, donc on a l'accès à partir du dossier du patient.

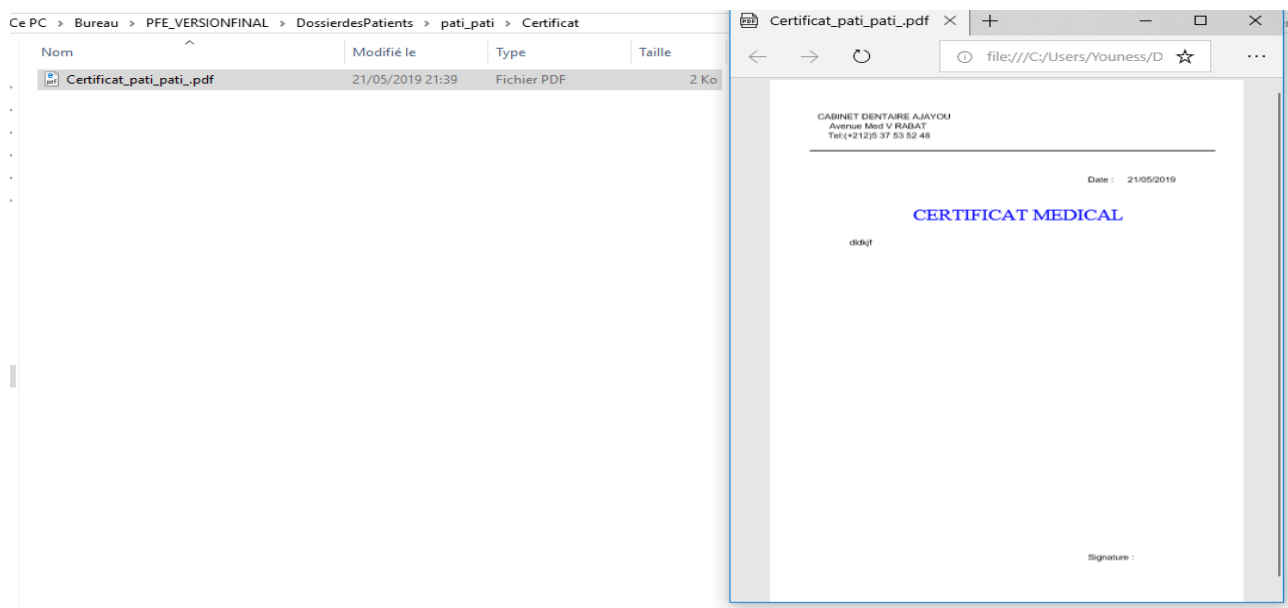


FIGURE 3.45 – Affichage d'un Certificat

# Bibliographie

# Annexes