



## Faculté d'Informatique

---

# RAPPORT DU PROJET TP SI2

---

### Réalisé par :

Nom : ANIK

Prénom : Sara

Matricule : 212131079010

Nom : ALLA

Prénom : Maya

Matricule : 212131070628

Section : Isil B

Groupe : 01

# Sommaire

## **I. Introduction**

## **II. Le Schéma de la base de données**

II.1 Les tables créées dans la base de données .

II.2 Le modèle conceptuel des données .

II.1 Le modèle relationnel des données .

II.3 Le dictionnaire des données.

## **III. L'architecture du système de gestion**

III.1 Les modules et les interaction

III.2 L'interaction entre les modules

## **IV. La structure du code**

## **V. Le processus de déploiement des pages web basées sur Django**

V.1 Environnement de développement

V.2 Processus de déploiement

V.3 Méthodologie de déploiement

## **VI. Annexe**

## **VII. Bibliographie**

## I. Introduction :

Le projet consiste à développer un système de gestion de services de santé pour une clinique chirurgicale qui offre des consultations médicales. L'objectif principal de ce système est de faciliter la gestion des rendez-vous entre les patients et les professionnels de santé, tout en informatisant les différentes étapes du processus d'entreprise.

En utilisant ce système, l'entreprise vise à optimiser ses processus de travail, à rationaliser ses activités et à améliorer la qualité des soins dispensés aux patients.

Ce rapport de synthèse présente les étapes clés de la conception, du développement et de la mise en œuvre du système, ainsi que les technologies utilisées. Nous abordons également les défis rencontrés tout au long du projet et les solutions mises en place pour les surmonter.

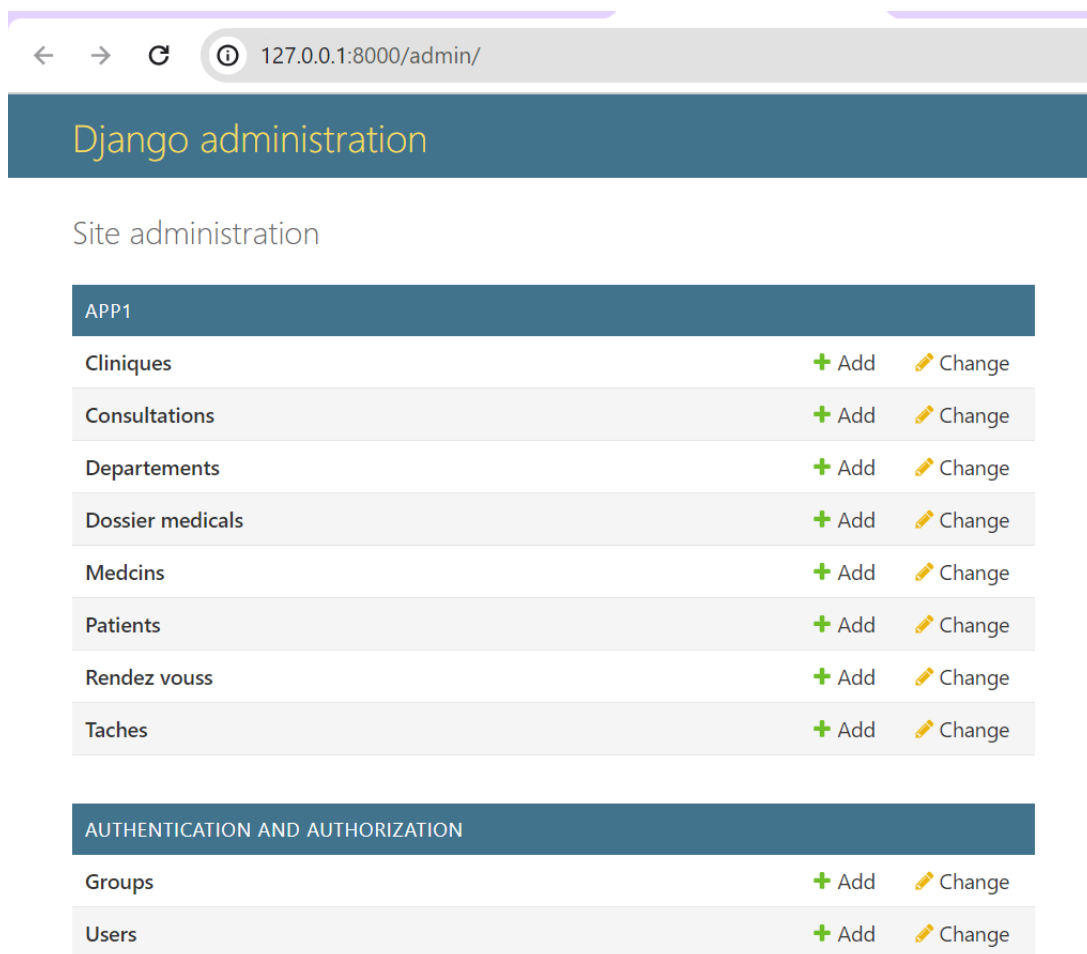
En résumé, ce système de gestion offre une solution complète pour la gestion des services de la clinique, en offrant aux médecins un moyen facile d'organiser les rendez-vous de leurs patients et de gérer leur emploi du temps d'une manière efficace.

## II. Le schéma de la base de données :

### II.1- Les tables créées dans la Base de données :

Ici , nous présentons une liste complète des tables que nous avons créées pour notre système de gestion de clinique ainsi que les types données qu'elles stockes qui sont représentées dans le Modèle conceptuelle de données « MCD » et dans le dictionnaire des données :

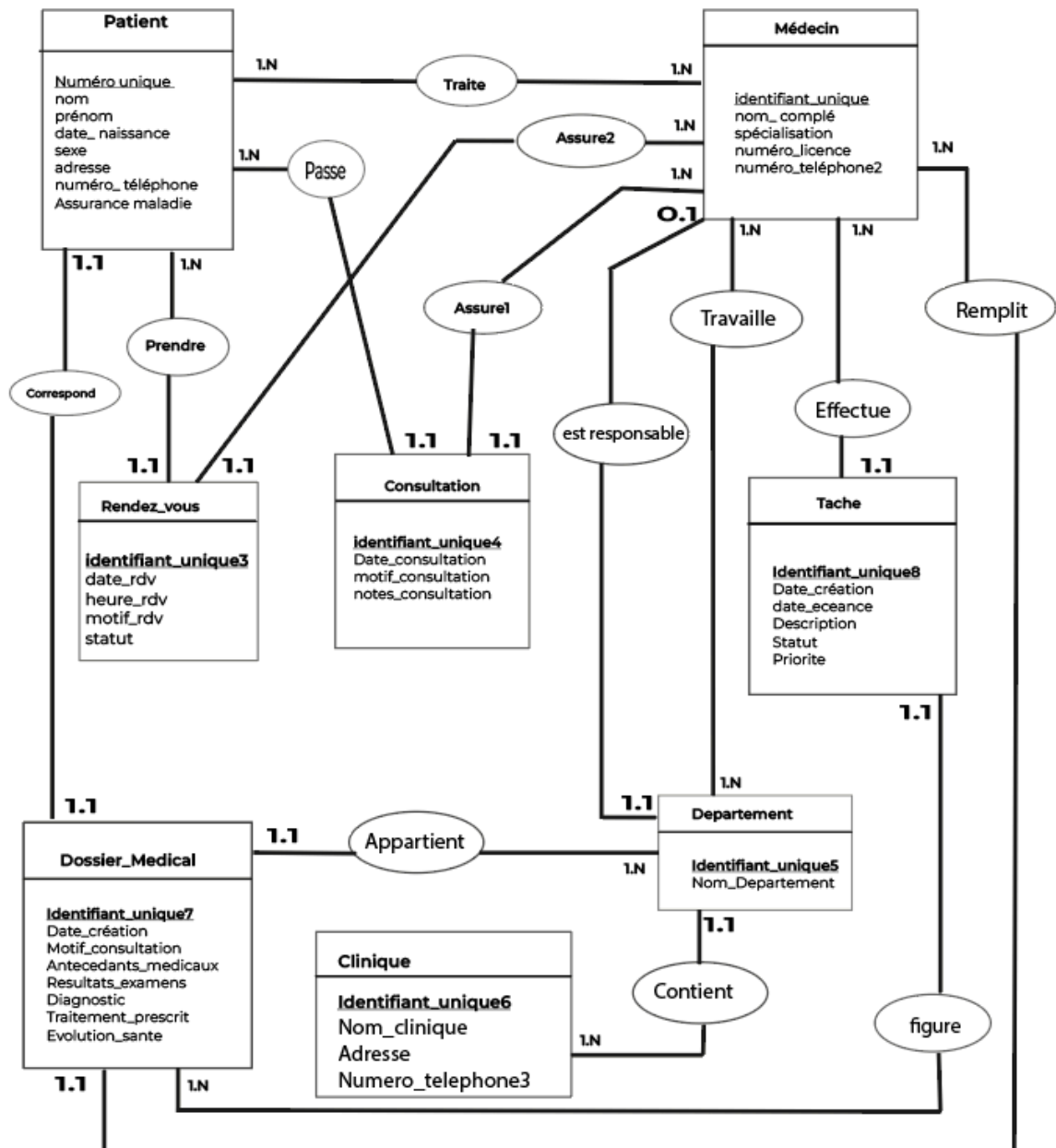
Voici les tables qu'on a créé dans la Base de données :



The screenshot shows the Django administration interface in a web browser. The address bar displays '127.0.0.1:8000/admin/'. The page title is 'Django administration'. Below the title, it says 'Site administration'. There are two main sections: 'APP1' and 'AUTHENTICATION AND AUTHORIZATION'. Each section contains a list of tables with 'Add' and 'Change' links.

APP1		
Cliniques	+ Add	Change
Consultations	+ Add	Change
Departements	+ Add	Change
Dossier médicaux	+ Add	Change
Medcins	+ Add	Change
Patients	+ Add	Change
Rendez vous	+ Add	Change
Taches	+ Add	Change
AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

## II.2-Le modèle conceptuelle des données :



## II.3-Le Modèle relationnelle des données :

**Patient** (identifiant\_unique , nom , prenom , date , sexe , adresse , numero\_telephone , assurance\_maladie) ;

**Medcin** (identifiant\_unique2 , nom\_complet , specialisation , numero\_licence , numero\_telephone2 ) ;

**Rendez-vous** (identifiant\_unique3 , date\_rdv , heure\_rdv , motif\_rdv , statut , identifiant\_unique\* , identifiant\_unique2\* ) ;

**Consultation** (identifiant\_unqiue4 , date\_consultation , motif\_consultation , identifiant\_unique\* , identifiant\_unique2\* ) ;

**Departement** ( identifiant\_unique5 , nom\_departement ,identifiant\_unique2\*,identifiant\_unique6\*) ;

**Clinique** ( identifiant\_unique6 , nom\_clinique , adresse , numero\_telephone3 ) ;

**DossierMedical** ( identifiant\_unique7 , date\_creation , motif\_consultation , antecedants\_medicaux , resultats\_examens , diagnostic , traitement\_prescrit , evolution\_sante , identifiant\_unique2\* , identifiant\_unique\* , identifiant\_unique5\* ) ;

**Tache** ( identifiant\_unique8 , date\_creation2 , date\_echeance , description , statut , priorite , identifiant\_unique2\* , identifiant\_unique7\*) ;

**Traite** (Num\_unique,Num\_unique2) ;

## II.4- Le dictionnaire des données :

<b>Donnée</b>	<b>Signification</b>	<b>Type</b>
Identifiant_unique	L'identifiant du patient	integer
nom	Nom du patient	Varchar2
prénom	Prénom du patient	Varchar2
Date_naissance	Date de naissance du patient	Date
sexe	Sexe du patient	Varchar2
adresse	Adresse du patient	Varchar2
Numero_telephone	Numero de téléphone du patient	integer
Assurance_maladie	L'assurance de maladie du patient	Varchar2
Identifiant_unique2	L'identifiant du médecin	integer
Nom_complet	Nom du médecin	Varchar2
Spécialisation	la spécialisation du médecin	Varchar2
Numero_licence	Numéro de licence du médecin	Integer
Numero_telephone2	Numéro de téléphone du médecin	Integer
Identifiant_unique3	L'identifiant du rendez-vous	integer
Date_rdv	Date du rendez-vous	Date
Heure_rdv	L'heure du rendez-vous	Integer
Motif_rdv	la raison du rendez-vous	Varchar2
Statut1	Statut du rendez-vous	Varchar2
Identifiant_unique4	L'identifiant du consultation	Integer
Date_consultation	La date de consultation	Date

Motif_consultation	la raison du consultation	varchar2
Notes_consultations	Les notes prises lors de la consultation	Varchar2
Identifiant_unique5	L'identifiant du département	Integer
Nom_departement	Le nom du département	Varchar2
Identifiant_unique6	L'identifiant de la clinique	Integer
Nom_clinique	Le nom de la clinique	Varchar2
Adressec	L'adresse de la clinique	Varcahar2
Numero_telephone3	Le numéro de téléphone de la clinique	Integer
Identifiant_unique7	L'identifiant du dossier médical	integer
Date_création	La date de création du dossier	Date
Antecedants_medicaux	Les antécédents médicaux du patient	Varchar2
Resultats_examens	Les resultats des tests du patient	Varchar2
Diagnostic	Le diagnostic médical du patient	Varchar2
Traitement_prescrit	Médicaments prescrits par un médecin à un patient	Varchar2
Evolution_sante	L'évolution de la santé du patient	Varchar2
Identifiant_unique8	L'identifiant de la tâche	Integer
Date_creation2	Date prise en charge de la tache	Date
Date_échéance	La date d'échéance de la tache	Date



Description	La description de la tâche	Varchar2
Statut	La statut de la tâche	Varchar2
priorité	La priorité de la tâche	Varchar2

## **III.L'architecture du système de gestion :**

### **III.1 Les modeles et les interaktions:**

Dans ce projet, l'architecture suit le modèle MVC (Modèle-Vue-Contrôleur), Cependant, Django utilise une variante de cette architecture qui appelée MVT (models-views-templates) , Dans cette structure :

#### **1. Modèle (Model):**

- Décrit la structure des données de l'application.
- Les modèles sont définis en Python et mappés aux tables de la base de données.
- Ils permettent de manipuler les données de manière abstraite et indépendante du système de base de données utilisé.

#### **2. Vue (View):**

- Gère les requêtes HTTP et génère les réponses.
- Les vues sont des fonctions Python qui accèdent aux données du modèle et les utilisent pour générer du contenu HTML, des flux JSON ou d'autres types de réponses.

#### **3. Template (Template):**

- Détermine la structure et la présentation du contenu généré par la vue.
- Les templates sont écrits en langage HTML et peuvent contenir des variables et des expressions Django pour afficher dynamiquement les données.

#### **4. URL (Uniform Resource Locator):**

- Définissent les correspondances entre les URL et les vues.
- Les URL sont configurées dans un fichier urls.py et permettent de diriger les requêtes vers les vues appropriées.

## 5. Formulaire (Form):

- Permettent de créer et de valider les données saisies par l'utilisateur.
- Les formulaires sont intégrés aux vues et peuvent être générés automatiquement à partir des modèles.

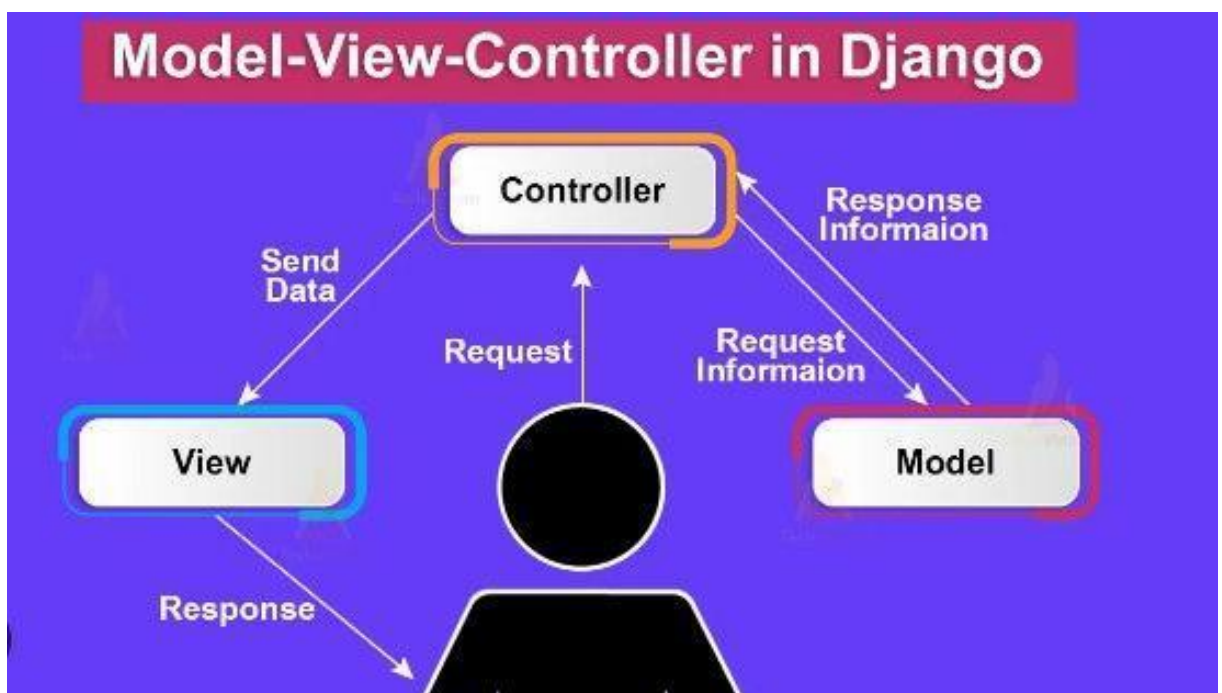


fig1. L'architecture utilisée dans les projet MVT

## III.2. Interactions entre les modules:

1. **Requête HTTP:** L'utilisateur envoie une requête HTTP à l'application.
2. **Moteur de routage:** Le moteur de routage analyse l'URL de la requête et la compare aux URL configurées.
3. **Vue:** La vue associée à l'URL est appelée.
4. **Modèle:** La vue peut accéder aux données du modèle pour les manipuler.
5. **Template:** La vue utilise un template pour générer le contenu de la

réponse.

6. **Réponse HTTP:** La réponse est envoyée à l'utilisateur.

## IV. La structure du code :

Voici la structure du code qu'on a établie pour organiser les différents composants de l'application :

**Models.py :** Ce fichier contient la définition des modèles de données de l'application qui seront stockées dans la base de données. On a créé des classes pour patient, médecin, consultation, rendez-vous, clinique, département, tâche, et dossier médical.

Voici un exemple sur la class Patient :

```
from django.db import models

class Patient(models.Model):
    identifiantunique = models.AutoField(primary_key=True)
    nom = models.CharField(max_length=255)
    prenom = models.CharField(max_length=255)
    date_naissance = models.DateField()
    sexe = models.CharField(max_length=10, choices=[('M', 'Masculin'), ('F', 'Féminin')])
    adresse = models.CharField(max_length=255)
    numero_telephone = models.CharField(max_length=10)
    assurance_maladie = models.CharField(max_length=50, unique=True)

    def __str__(self):
        return f"{self.nom} {self.prenom}"
```

**Views.py :** Ce fichier contient des classes et des fonctions implémentées qui définissent comment les données sont présentées à l'utilisateur et comment l'application réagit aux actions de l'utilisateur. Il contient des fonction pour :

**Pour l'affichage :** on a créé des fonctions qui permettent de faire l'affichage des départements, des rendez-vous,....., voici une fonction qui permet d'afficher les départements disponibles dans la base de données comme cet exemple:

```
def affiche_departement(request):
    d = Departement.objects.all()
    return render(request, 'index.html', {"departement": d})
```

**Pour l'ajout :** on a créé des fonctions qui permettent de faire l'ajout des patients , des dossiers médicaux , des tâches , ..... , voici un exemple d'une fonction qui fait l'ajout d'un patient comme cet exemple:

```
def add(request):
    if request.method == 'POST':
        form = AddPatient(request.POST)
        if form.is_valid():
            form.save()
            mssg = "PATIENT AJOUTER AVEC SUCCES"
            return render(request, 'create.html', {'form' : form, 'mssg': mssg})
        else:
            form = AddPatient()
            return render(request, 'create.html', {'form': form})
```

**Pour la modification :** on a créé des fonctions qui permettent de faire la modifications des patients , des médecins ..... , voici un exemple d'une fonction qui fait la modification des informations d'un patient comme cet exemple:

```
def editP(request, id):

    p = Patient.objects.get(identifiantunique=id)

    if request.method == 'POST':
        form = AddPatient(request.POST, instance=p)
        if form.is_valid():
            # mettre à jour l'objet dans la base de données
            form.save()
            # rediriger vers la page liste patient
            return redirect('/gerer_patient')
        else:
            form = AddPatient(instance=p)
            # on pré-remplir le formulaire avec l'objet ayant l'id spécifié
            return render(request, 'patientU.html', {'form': form})
```

**Pour la suppression** : on a créé des fonctions qui font la suppression des patients , médecins , consultations ,.... , voici un exemple d'une fonction de suppression d'un patient par son ID comme cet exemple:

```
def deleteP(request, id):
    p = Patient.objects.get(identifiantunique=id)
    if request.method == 'POST':
        p.delete()
        return redirect('/gerer_patient')
    return render(request, 'patientS.html', {'p': p})
```

**Pour la recherche** : on a créé des fonctions qui permettent de faire la recherche des patients , des dossiers , des tâches .... , la recherche se fait pas leurs ID , voici un exemple d'une fonction qui fait la recherche du patient comme cet exemple:

```
def rechrcheP(request):
    patient = ''

    if request.method == "GET":
        query = request.GET.get('rechrcheP')
        if query:
            patient =
Patient.objects.filter(identifiantunique__contains=query)
    return render(request, 'searchp.html', {"patient": patient})
```

**Forms.py** : Ce fichier contient les définitions des formulaires utilisées pour la saisie et la validation des données dans l'application. on a créé des formulaires pour l'insertion , la modification et pour la suppression .

Par exemple pour l'ajout et edit :

```
from django.db.models import fields

from django import forms
from .models import Patient, Medcin, RendezVous, Consultation, Tache,
DossierMedical, Departement

class AddPatient(forms.ModelForm):
    class Meta:
        model = Patient
        fields = '__all__'
```

**Urls.py** : Dans ce fichier, on a mis les associations entre les URL de l'application et les vues correspondantes.

Par exemple :

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index1),
    path('list_departement/', views.affiche_departement),
```

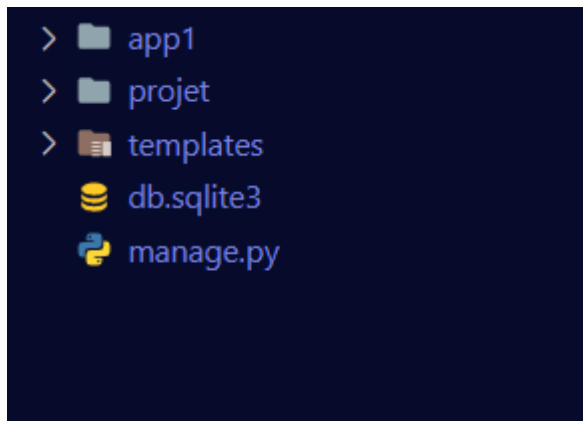
**Répertoire templates/** : on trouve les fichiers HTML qui définissent l'interface utilisateur de l'application, on a créé des templates pour chaque classe créée dans la base de données , comme on a créé des templates pour l'insertion, la modification, l'affichage, l'insertion et pour la recherche .

**Remarque :**

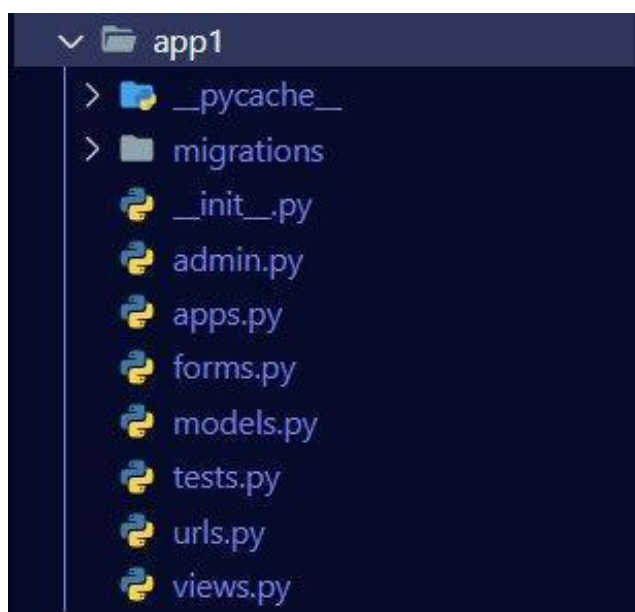
En ce qui concerne les fichiers HTML qu'on a créé :

- **Les fichiers de modification**, ils se terminent par "U ", par exemple vous trouvez : ConsultationU.html, PatientU.html, etc ....,
- **Les fichiers destinés à la suppressions** se terminent par "s" , par exemple patientS.html, medecinS.html ,etc ....
- **Les fichiers de gestion** comme gérerPatient , gérerMedecin ,etc, ... , on les a nommé d'après le nom du module par exemple pour gérerPatient vous trouvez Patient.html , medecin.html, etc ....
- **Les fichiers de recherche** sont nommés Search...html , par exemple vous trouvez searchp.html c'est pour la recherche des patients ;

□ La structure des répertoires est indiqué dans les figures ci-dessous :



**fig2.** structure des repertoires 1



**fig3.** structure de repertoire application



fig4. structure de répertoire templates

## V. Le processus de déploiement des pages web basées sur Django :

### V.1. Environnement de développement:

- **WSGI:** Interface standard pour l'interaction entre les serveurs web et les applications web Python.
- **Nginx:** Serveur web performant et largement utilisé, utilisé pour servir les pages web statiques et dynamiques.

### V.2. Processus de déploiement:

- **Hébergement:** Hébergement web mutualisé



- **Système d'exploitation:** Windows
- **Serveur web:** IIS (Internet Information Services)
- **Intergiciel WSGI:** WSGIHandler
- **Langage de programmation:** Python
- **Module Python pour Windows:** Installé
- **Fichiers statiques:** Non disponibles

### **V.3.Méthodologie de déploiement:**

- **Déploiement:** Manuel
- **Outils de déploiement:** Non utilisés

Le processus de déploiement actuel est simple mais peut être amélioré pour gagner en efficacité et en fiabilité.

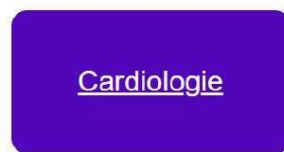
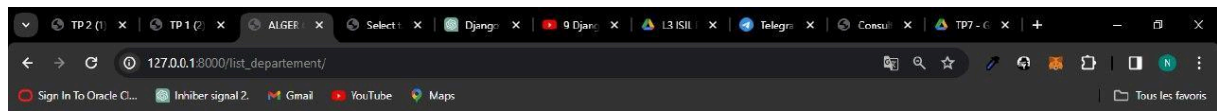
## **VI. Annexe:**

Voici à ce que ressemble le système :

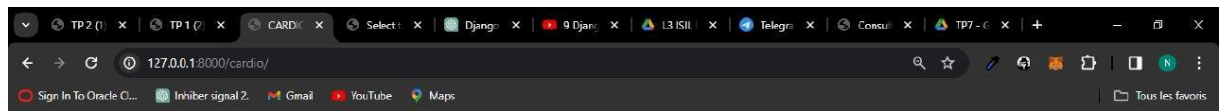


ALGER CLINIQUE

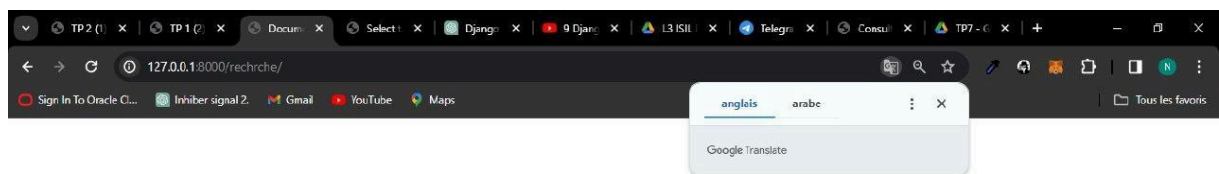
Quand on clique sur la clinique 'Alger clinique ', on accède directement au département cardiologie, il nous affiche :



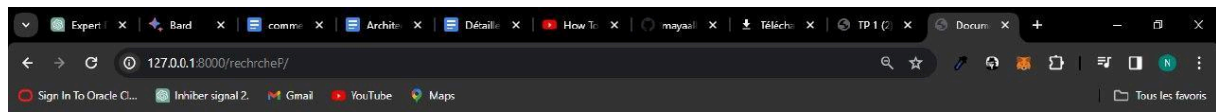
Quand on clique sur le département 'cardiologie ', il nous affiche cette interface ou on peut faire la gestion des médecins, des rendez-vous, des consultation, des taches , des dossiers médicaux et on peut aussi faire la recherche :



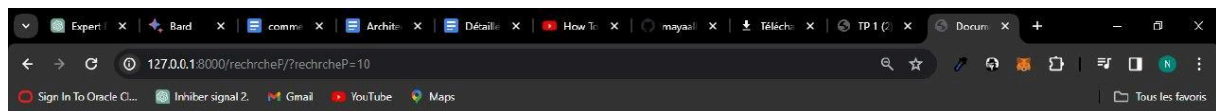
Quand on clique sur 'recherche', l'interface suivante sera affichée ou on peut faire la recherche des patients, des médecins, des dossiers médicaux, des consultation, des rendez-vous et des taches , comme on peut revenir au 'home' .



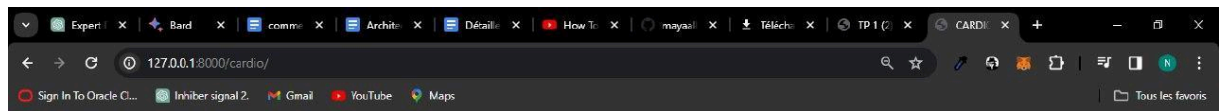
la recherche se fait par le id, par exemple pour la recherche des patients :  
on saisit l'ID du patient qu'on cherche comme le montre la figure  
suivante :



Quand on clique sur le bouton « search » les informations du patient  
recherché s'affichent :



Si on clique sur « GO HOME » l'interface suivante sera affichée :



Puis Lorsqu'on clique sur "gérer Patient " une interface **affichant la liste des patients** apparait, on y trouve également un lien pour ajouter un nouveau patient , ainsi que des boutons permettant de modifier ou de supprimer un patient :



LISTE DES PATIENT [GO HOME](#)

[ADD PATIENT](#)

**Id:** 22  
**Nom:** AIT ALIA  
**Prenom:** Serine  
**Date naissance:** Oct. 5, 2004  
**Sexe:** F  
**Adresse:** ORAN  
**Numero de telephone:** 554249424  
**Assurance maladie:** 03533326605

[UPDATE](#)

[DELETE](#)

[GO HOME](#)

Quand on **clique sur "ADD Patient "** une interface comme la suivante apparait :

**AJOUTER UN PATIENT**

[GO HOME](#) - [LIST PATIENT](#)

Capture d'écran

**Nom:**

**Prenom:**

**Date naissance:**

**Sexe:**

**Adresse:**

**Numero telephone:**

**Assurance maladie:**

Ajoute d'un patient:

**Nom:**

**Prenom:**

**Date naissance:**

**Sexe:**

**Adresse:**

**Numero telephone:**

**Assurance maladie:**

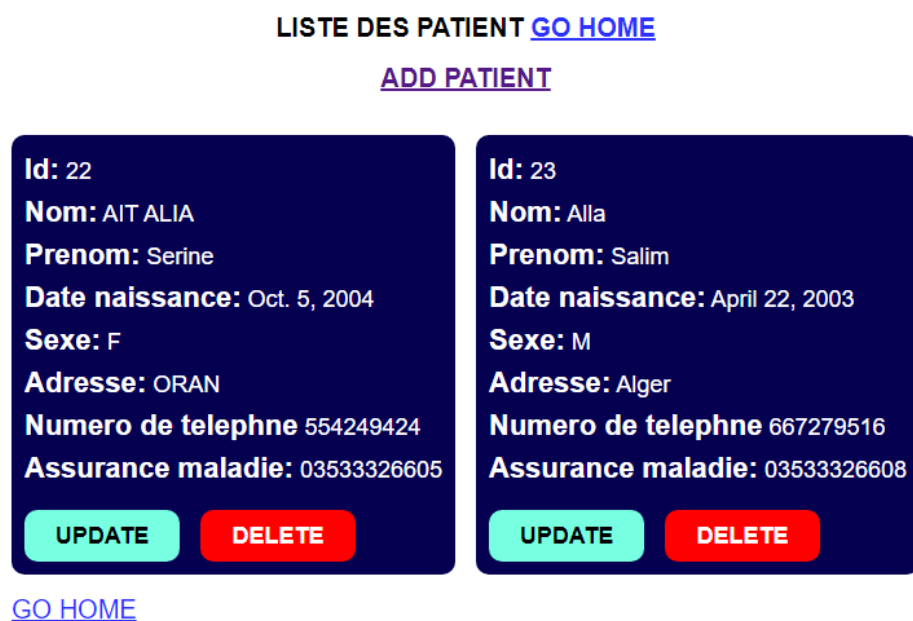
**Ajouter**

Puis lorsqu'on remplit les informations du patient et on **clique sur le bouton Ajouter** , le message " patient ajouter avec succès " sera affichée, comme ça :



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/add/". The page title is "AJOUTER UN PATIENT". Below the title, there are two links: "GO HOME" and "LIST PATIENT". A red message "PATIENT AJOUTER AVEC SUCCES" is displayed. Below the message, there is a form with the following fields: "Nom:" (Alla), "Prenom:" (Salim), "Date naissance:" (2003-04-22), "Sexe:" (Masculin), and "Adresse:" (Alger).

On peut vérifier si le patient a été vraiment ajouté en cliquant sur "Liste Patient " et on revient à cette interface :



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/list-patient/". The page title is "LISTE DES PATIENT". Below the title, there are two links: "GO HOME" and "ADD PATIENT". Below the links, there are two patient cards. Each card displays the following information: "Id:", "Nom:", "Prenom:", "Date naissance:", "Sexe:", "Adresse:", "Numero de telephone", and "Assurance maladie". At the bottom of each card, there are two buttons: "UPDATE" and "DELETE".

Id	Nom	Prenom	Date naissance	Sexe	Adresse	Numero de telephone	Assurance maladie
22	AIT ALIA	Serine	Oct. 5, 2004	F	ORAN	554249424	03533326605
23	Alla	Salim	April 22, 2003	M	Alger	667279516	03533326608

Dans l'interface "gérer patient" il est possible de **supprimer** un patient en **cliquant sur le bouton "Delete"** associé au patient concerné :

Par exemple : on veut supprimer le patient "AIT ALIA Serine" :

LISTE DES PATIENT [GO HOME](#)

[ADD PATIENT](#)

<p><b>Id:</b> 22</p> <p><b>Nom:</b> AIT ALIA</p> <p><b>Prenom:</b> Serine</p> <p><b>Date naissance:</b> Oct. 5, 2004</p> <p><b>Sexe:</b> F</p> <p><b>Adresse:</b> ORAN</p> <p><b>Numero de telephone</b> 554249424</p> <p><b>Assurance maladie:</b> 03533326605</p> <p><a href="#">UPDATE</a> <a href="#">DELETE</a></p>	<p><b>Id:</b> 23</p> <p><b>Nom:</b> Alla</p> <p><b>Prenom:</b> Salim</p> <p><b>Date naissance:</b> April 22, 2003</p> <p><b>Sexe:</b> M</p> <p><b>Adresse:</b> Alger</p> <p><b>Numero de telephone</b> 667279516</p> <p><b>Assurance maladie:</b> 03533326608</p> <p><a href="#">UPDATE</a> <a href="#">DELETE</a></p>
--	--

[GO HOME](#)

Et après avoir cliqué sur "delete", cette interface sera affichée :

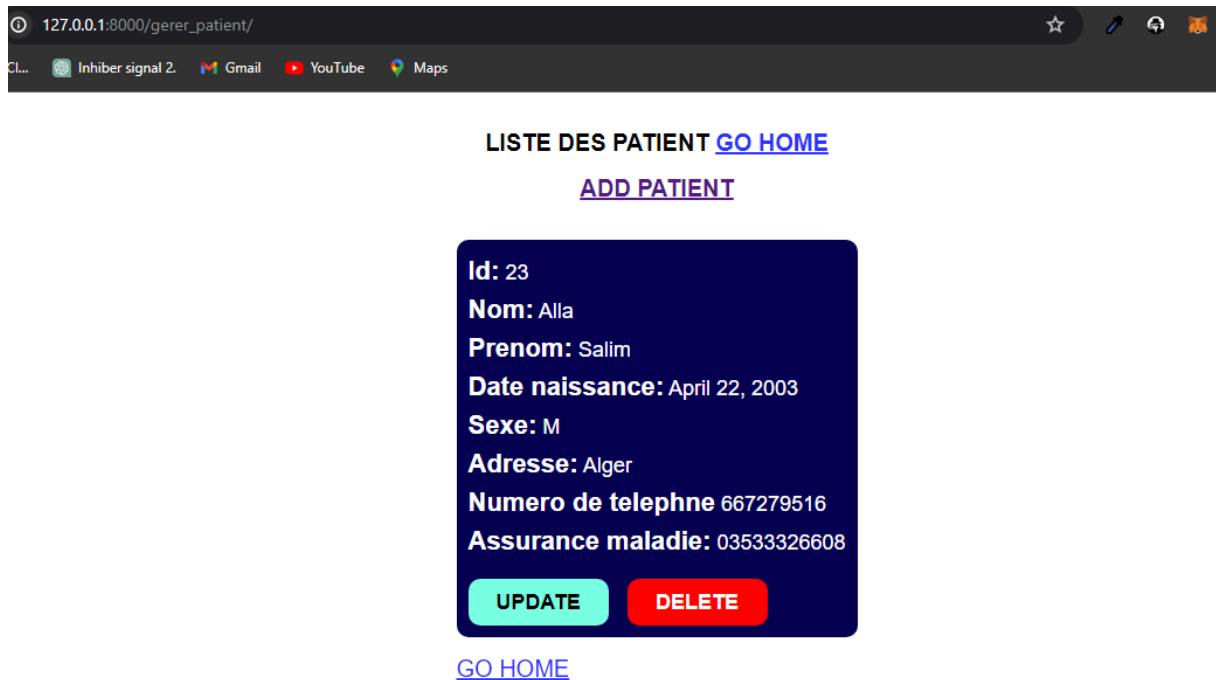
**Supprimer le pateint**

Etes-vous sure de vouloir supprimer ce patient de id 22?

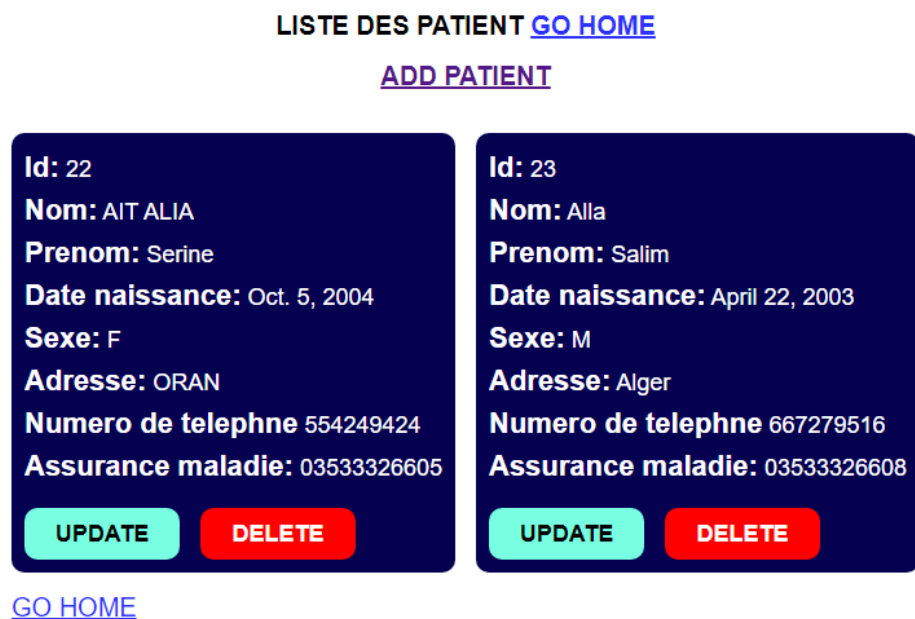
[Delete](#) [No, take me back](#)



Si on **clique sur "Delete"** le patient sera supprimé comme le montre cette interface :



Sinon si on clique sur **"no, take me back"** le patient ne sera pas effacé et nous retournons à la page "gérer patient" qui affiche la liste des patients ,celle-là :



Il est également possible de faire La modification des informations d'un patient en **cliquant sur le bouton "Update"**, ce qui fera apparaître l'interface suivante , Par exemple on veut modifier l'adresse du patient "ALLA Salim " a "Oran" :

### Mise à jour d'un patient

**Nom:**  
Alla

**Prenom:**  
Salim

**Date naissance:**  
2003-04-22

**Sexe:**  
Masculin

**Adresse:**  
ORAN

**Numero telephone:**  
667279516

**Assurance maladie:**  
03533326608

→ ↻ 127.0.0.1:8000/editP/23 ☆ 🔧 🔄 📄 📥 📂

Sign In To Oracle CL... Inhiber signal 2. Gmail YouTube Maps | Tous les

**Nom:**  
Alla

**Prenom:**  
Salim

**Date naissance:**  
2003-04-22

**Sexe:**  
Masculin

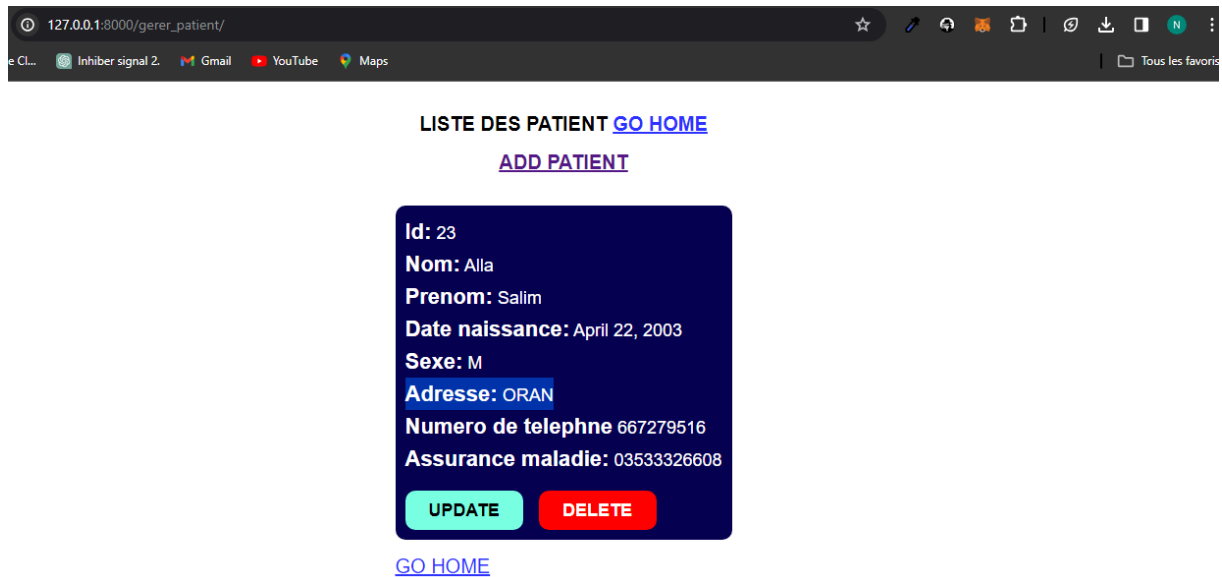
**Adresse:**  
ORAN

**Numero telephone:**  
667279516

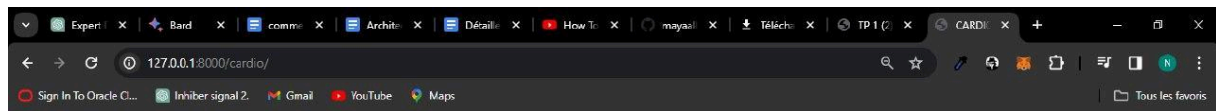
**Assurance maladie:**  
03533326608

**Update**

Et lorsqu'on clique sur **"Update "** on revient à la page **"gérer patient "**, ou on peut vérifier si les informations ont été modifiées :



Et à partir de cette interface qui affiche la liste des patients, on peut revenir à la page de gérer le département en cliquant sur **"Go home "**, on revient à cette interface :



Si par exemple on veut faire la **gestion des rendez-vous**, on clique sur **“Gérer RDV ”**, ce qui fera apparaitre cette interface qui affiche la liste des rendez-vous avec **la possibilité d’ajouter un nouveau rendez-vous** :



On peut ajouter un rendez-vous en cliquant sur **“ADD Rendez-vous ”**, et cette interface apparait :

127.0.0.1:8000/addr/

Sign In To Oracle CL... Inhiber signal 2. Gmail YouTube Maps

Tous les favoris

## AJOUTER UN RENDEZ-VOUS

[GO HOME](#) -- [LIST RENDEZ-VOUS](#)

**Date rdv:**

**Heure rdv:**

**Motif rdv:**

**Statut:**

**Patient:**

On remplit les informations du rendez-vous ,et on clique sur **“Ajouter”** :

**Date rdv:**

2024-01-30

**Heure rdv:**

16:00:00

**Motif rdv:**

Douleur thoracique persistante

Description : Mme Martin éprouve une douleur thoracique persistante depuis quelques semaines. Elle a décidé de consulter le département de cardiologie

**Statut:**

En attente

**Patient:**


Alla Salim

**Medcin:**

ALLA Maya

**Ajouter**

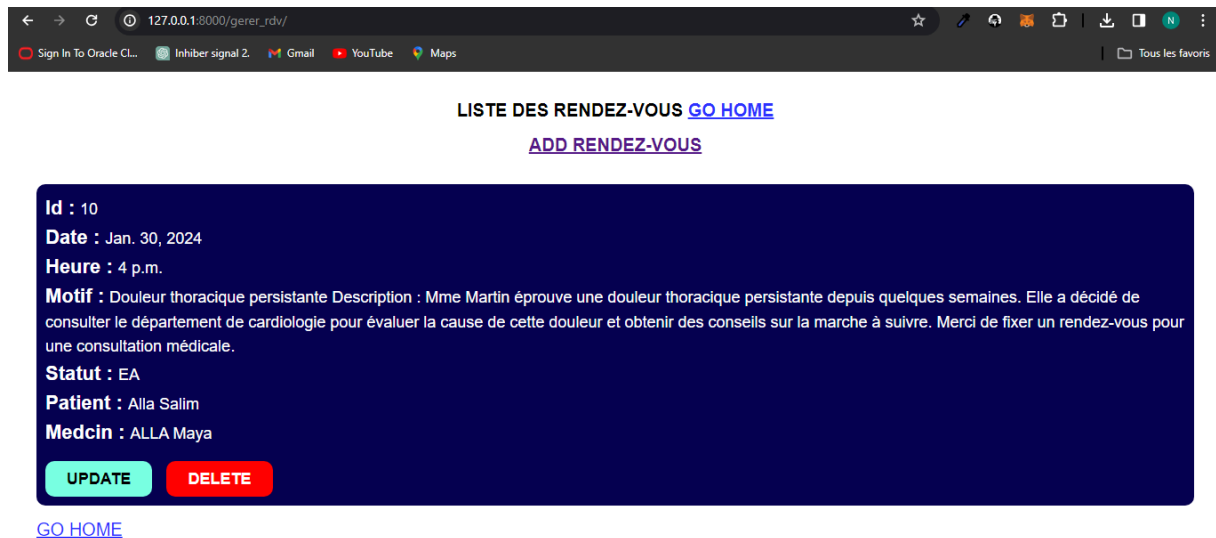
Puis le message “Rendez-vous ajouter avec succès ” sera affiché , comme ça :



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/addr/". The page title is "AJOUTER UN RENDEZ-VOUS". Below the title, there are two links: "GO HOME" and "LIST RENDEZ-VOUS". A red message "RENDEZ VOUS AJOUTER AVEC SUCCES" is displayed. Below the message, there is a form with the following fields:

- Date rdv:** 2024-01-30
- Heure rdv:** 16:00:00
- Motif rdv:** Douleur thoracique persistante
- Description :** Mme Martin éprouve une douleur thoracique persistante depuis quelques semaines. Elle a décidé de consulter le département de cardiologie
- Statut:** En attente

On peut vérifier si le rendez-vous a été ajouté en **cliquant sur “LIST RENDEZ-VOUS ”** :



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/gerer\_rdv/". The page title is "LISTE DES RENDEZ-VOUS". Below the title, there are two links: "GO HOME" and "ADD RENDEZ-VOUS". Below the links, there is a card displaying the details of an appointment:

- Id :** 10
- Date :** Jan. 30, 2024
- Heure :** 4 p.m.
- Motif :** Douleur thoracique persistante
- Description :** Mme Martin éprouve une douleur thoracique persistante depuis quelques semaines. Elle a décidé de consulter le département de cardiologie pour évaluer la cause de cette douleur et obtenir des conseils sur la marche à suivre. Merci de fixer un rendez-vous pour une consultation médicale.
- Statut :** EA
- Patient :** Alla Salim
- Medcin :** ALLA Maya

Below the card, there are two buttons: "UPDATE" and "DELETE".

On peut faire **la modification d’un rendez-vous en cliquant sur “Update”**, ce qui amènera à l’affichage de l’interface suivante :

Par exemple on a **modifié l'heure du rendez-vous qui a le ID=10**, en **cliquant sur "update"** on revient à l'interface qui affiche la liste des rendez-vous ou on peut s'assurer que la modification a été bien effectuée :



Mise à jour d'une Rendez-vous

**Date rdv:**  
2024-01-30

**Heure rdv:**  
19:00:00

**Motif rdv:**  
Suivi cardiaque annuel

Description : M. Dupont demande un rendez-vous pour son examen de suivi cardiaque annuel en raison de son hypertension diagnostiquée il y a deux ans.

**Statut:**  
En attente

**Patient:**  
Alla Salim

**Medcin:**  
ALLA Maya

#### [ADD RENDEZ-VOUS](#)

**Id :** 10

**Date :** Jan. 30, 2024

**Heure :** 7 p.m.

**Motif :** Douleur thoracique persistante Description : Mme Martin éprouve une douleur thoracique persistante depuis quelques semaines. Elle a décidé de consulter le département de cardiologie pour évaluer la cause de cette douleur et obtenir des conseils sur la marche à suivre. Merci de fixer un rendez-vous pour une consultation médicale.

**Statut :** EA

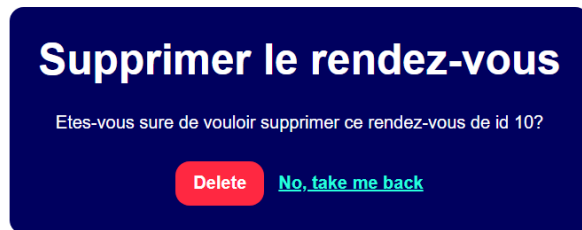
**Patient :** Alla Salim

**Medcin :** ALLA Maya

[UPDATE](#) [DELETE](#)

[GO HOME](#)

Et en ce qui concerne la **suppression d'un rendez-vous**, elle peut être **effectuée en cliquant sur « Delete »** du rendez-vous concerné, et cette interface de confirmation apparait :



Si on clique sur "delete" le rendez-vous sera effacé de la liste des rendez-vous, comme le montre cette interface :



LISTE DES RENDEZ-VOUS [GO HOME](#)  
[ADD RENDEZ-VOUS](#)  
[GO HOME](#)

Sinon **si on clique sur "no, take me back "** on revient à la liste des rendez-vous .



[ADD RENDEZ-VOUS](#)

**Id** : 10

**Date** : Jan. 30, 2024

**Heure** : 7 p.m.

**Motif** : Douleur thoracique persistante Description : Mme Martin éprouve une douleur thoracique persistante depuis quelques semaines. Elle a décidé de consulter le département de cardiologie pour évaluer la cause de cette douleur et obtenir des conseils sur la marche à suivre. Merci de fixer un rendez-vous pour une consultation médicale.

**Statut** : EA

**Patient** : Alla Salim

**Medcin** : ALLA Maya

UPDATE

DELETE

[GO HOME](#)

## **VII. Bibliographie :**

<https://docs.djangoproject.com/fr/5.0/>

<https://learn.microsoft.com/fr-fr/visualstudio/windows/?view=vs-2022>

<https://docs.github.com/fr>



