

# Gestion des Rendez-vous d'un Cabinet Médical

---

Réalisé par : Abla MARGHOUB

Encadré par : Pr Mohamed LACHGAR

Module : Techniques de programmation avancée

Établissement : Ecole Normale Supérieure- Université Cadi Ayyad

---

## 1. Description du projet

### 1.1 Contexte fonctionnel

Dans un cabinet médical, les secrétaires doivent gérer la planification, les confirmations, les annulations et la disponibilité des médecins. Une solution numérique permet de simplifier ce processus.

### 1.2 Objectif de l'application

L'application a pour objectif de faciliter la prise, la consultation et la modification des rendez-vous médicaux entre les patients et les médecins.

### 1.3 Public cible / cas d'usage

- **Assistante** : planifie, confirme ou annule les rendez-vous.
- **Médecin** : Gérer ses rendez-vous et consulter les informations des patients.

### 1.4 Fonction principale

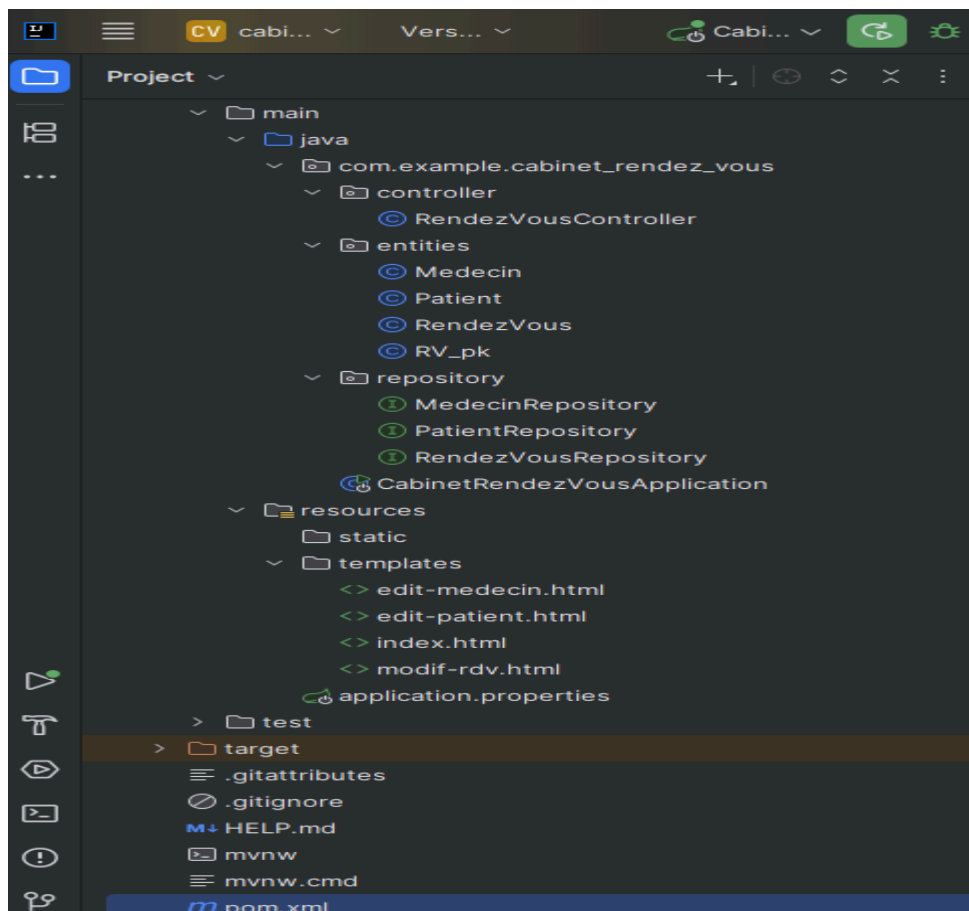
L'application permet de gérer les rendez-vous médicaux en ligne (ajout, modification, annulation, filtrage) via une interface web intuitive.

## 2. Architecture technique

### 2.1 Stack technologique

<b>Backend</b>	Spring Boot 3.5.6 Spring Web MVC Spring Data JPA / Hibernate Dev Tools
<b>Frontend</b>	Thymeleaf HTML5 CSS Bootstrap 5
<b>Base de données</b>	MySQL
<b>Build</b>	Maven

## 2.2 Structure du code



## 2.3 Diagramme d'architecture (flux)

L'application suit une architecture en couches (MVC : Model - View - Controller).

### 1. Navigateur (Frontend)

- L'utilisateur (assistante ou médecin) interagit via un navigateur web.
- Il envoie des requêtes HTTP (GET, POST, etc.) vers l'application, par exemple pour afficher la liste des rendez-vous ou en ajouter un.

### 2. Contrôleur Spring

- C'est la porte d'entrée du backend.
- Il reçoit la requête HTTP, récupère ou envoie les données nécessaires, et choisit quelle vue (page HTML) afficher.

### 3. Repository

- Communiquer directement avec la base de données via Spring Data JPA.
- Il exécute les requêtes SQL générées automatiquement (findAll, findById, save, delete...).
- 

### 4. Base de données (MySQL)

- Contient les tables : patient, medecin, rendez\_vous.
- Stocke toutes les données persistantes de l'application.

### 5. Vue Thymeleaf

- Le contrôleur renvoie un modèle (Model) contenant les données.
- Thymeleaf les affiche dynamiquement dans les pages HTML grâce à des expressions .

### 6. Flux global

- Navigateur → Contrôleur → Service → Repository → Base de données → Repository → Service → Contrôleur → Vue (Thymeleaf) → Navigateur.

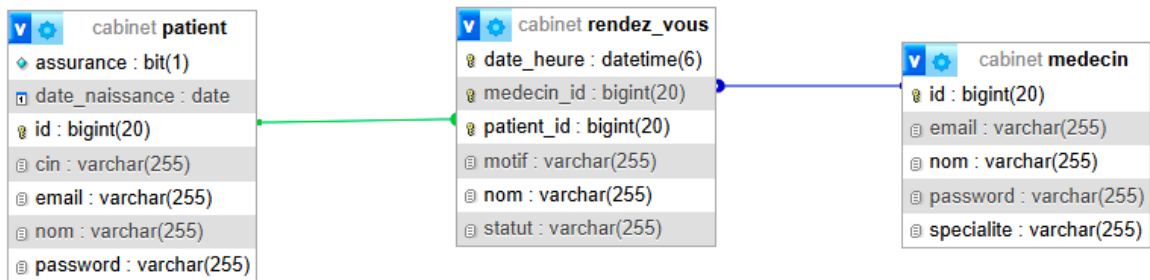
## 3. Fonctionnalités principales

- CRUD complet sur les rendez-vous, médecins et patients.
- Filtrage dynamique par date, spécialité, statut, médecin.

- Gestion des statuts : *Confirmé, Annulé.*
- Modification et suppression d'un rendez-vous.
- Interface responsive en Bootstrap

## 4. Modèle de données

### 4.1 Entités principales



### 4.2 Relations

Les relations entre les entités est comme se suit :

- Patient (OneToMany) RendezVous
- Médecin (OneToMany) RendezVous
- RendezVous (ManyToOne) Patient et Médecin

### 4.3 Configuration base de données

```

application.properties
1  spring.application.name=cabinet-rendez-vous
2  server.port=8080
3  spring.datasource.url=jdbc:mysql://localhost:3306/cabinet?serverTimezone=UTC
4  spring.datasource.username=root
5  spring.datasource.password=
6
7  spring.jpa.show-sql=true
8  spring.jpa.properties.hibernate.format_sql=true
9  spring.jpa.hibernate.ddl-auto=update
10
  
```

## 5. Lancer le projet

### 5.1 Prérequis

- Jdk 21
- Maven
- MySQL en service

## 5.2 Installation

```
git clone https://github.com/abmarghoub/cabinet-rendez-vous.git  
mvn spring-boot:run
```

## 5.3 Accès

- Page d'accueil : <http://localhost:8080>
- Page statistiques : <http://localhost:8080>

## 6. Démonstration (Vidéo)

[https://drive.google.com/drive/folders/13r9\\_Pc8GxxqnyTAy-mFx7vIZ3xrl7zCe?usp=sharing](https://drive.google.com/drive/folders/13r9_Pc8GxxqnyTAy-mFx7vIZ3xrl7zCe?usp=sharing)

.