

Date : 

**Manuel d'utilisation:**

**1- Installation des dépendances nécessaire:**

- Installer le module corsheaders avec la commande : **pip install corsheaders**

**2- Lancement du programme:**

- Lancer le serveur django avec la commande : **python SI/manage.py runserver** dans le port par défaut 8000

- ouvrir la page UI/main.html dans le navigateur, et c'est tout

**3- Utilisation:**

- Cliquer sur les boutons a gauches pour afficher toutes les instances enregistrées de ce model et afficher le menu des boutons de gestion (ajouter, modifier, supprimer)

- Ajouter : affichera un formulaire à remplir, apart les clés étrangères des client et médecin dans les services, les autres champs ne sont pas obligatoires, après avoir remplir le formulaire, appuyer sur Add pour créer l'objet

- Modifier : affichera le même formulaire que Ajouter avec un nouveau champs ID qui doit être remplis avec l' id d'objet que vous voulez modifier, pour le reste des champs, si vous avez une nouvelle valeur pour ce champs vous le remplissez, sinon vous le laissez vide pour concevoir la valeur ancienne de l'attribut, après appuyer sur Modify pour modifier l'objet

- Supprimer : affichera un formulaire à remplir avec la clé primaire de l'objet à supprimer, après appuyer sur Supprimer pour supprimer l'objet et to les objets des autres modèles qui le référencent (suppression en cascade)

Après chaque opération, appuyez sur les boutons du menu à gauche pour rafraîchir la vue.

Pour le champs ID médecin dans les services, vous pouvez saisir une listes des id des médecins séparé par virgule (2, 3, 2, 8) si ce rendez-vous de ce service requis plusieurs médecins

**Description du système:**

Le système conçu est une plateforme web pour la gestion d'une clinique hospitalière, on a pas choisis un département spécial mais on a pris le cas d'une clinique avec tous les services généraux : visite médicale, contrôle médical, hospitalisation, analyse, chirurgie et vaccination.

Le système permet de gérer (ajouter, modifier, supprimer, afficher) toutes les entités du système : patient, médecin, et les services au dessus

**Backend: Django and Python**

Le backend du système de gestion hospitalière est développé à l'aide de Django, un framework Web Python de haut niveau. Django fournit un moyen propre et efficace de créer des applications Web, en particulier celles qui nécessitent une structure de base de données robuste.

**Frontend : HTML, CSS, and JavaScript**

L'interface du système de gestion hospitalière est conçue en utilisant HTML pour structurer les pages Web, CSS pour le style et la mise en page, et JavaScript pour les fonctionnalités dynamiques et interactives. Cette combinaison garantit une interface conviviale et visuellement attrayante pour les utilisateurs.

**Database Model:**

Le projet implique les modèles de bases de données suivants pour gérer les données liées aux hôpitaux :

**Record Model:**

**id\_r:** Primary key, CharField, max length of 50.

**type**: IntegerField.

**fait**: BooleanField.

**date**: DateTimeField with auto\_now\_add set to True.

**Patient Model:**

**id**: Primary key, CharField, max length of 50.

**nom**: CharField, max length of 50.

**prenom**: CharField, max length of 50.

**date\_naissance**: DateField.

**sexe:** CharField, max length of 1.

**information**: TextField.

**record**: ForeignKey to the Record model with on\_delete set to CASCADE.

**Hopitalisation Model:**

**id**: Primary key, CharField, max length of 50.

**duree**: DurationField.

**result**: TextField.

**record**: OneToOneField to the Record model with on\_delete set to CASCADE.

**medecin**: ManyToManyField to the "Medecin" model with on\_delete set to CASCADE.

**Visite Model:**

**id**: Primary key, CharField, max length of 50.

**result**: TextField.

**record**: OneToOneField to the Record model with on\_delete set to CASCADE.

**medecin**: ManyToManyField to the "Medecin" model with on\_delete set to CASCADE.

**Control Model:**

**id**: Primary key, CharField, max length of 50.

**result**: TextField.

**record**: OneToOneField to the Record model with on\_delete set to CASCADE.

**medecin**: ManyToManyField to the "Medecin" model with on\_delete set to CASCADE.

**Analyse Model:**

**id**: Primary key, CharField, max length of 50.

**result**: TextField.

**type**: CharField, max length of 30.

**record**: OneToOneField to the Record model with on\_delete set to CASCADE.

**medecin**: ManyToManyField to the "Medecin" model with on\_delete set to CASCADE.

**Chirugie Model:**

**id**: Primary key, CharField, max length of 50.

**detail**: TextField.

**duree**: DurationField.

**record**: OneToOneField to the Record model with on\_delete set to CASCADE.

**medecin**: ManyToManyField to the "Medecin" model with on\_delete set to CASCADE.

**Vaccination Model:**

**id**: Primary key, CharField, max length of 50.

**nom**: CharField, max length of 30.

**type**: CharField, max length of 30.

**record**: OneToOneField to the Record model with on\_delete set to CASCADE.

**medecin**: ManyToManyField to the "Medecin" model with on\_delete set to CASCADE.

**Medecin Model:**

**id**: Primary key, CharField, max length of 50.

**nom**: CharField, max length of 30.

**prenom**: CharField, max length of 30.

**specialite**: CharField, max length of 30.

Ce modèle de base de données est conçu pour gérer efficacement les informations relatives aux patients, aux hospitalisations, aux visites, aux contrôles, aux analyses, aux chirurgies, aux vaccinations et au personnel médical. Les relations entre ces modèles garantissent l'intégrité et la cohérence des données. L'utilisation de l'ORM (Object-Relational Mapping) de Django facilite les opérations et les interactions avec les bases de données.

le schéma de la base de données et du MCD se trouve à la fin du rapport

**Fonctionnalités du Projet Hospital Management System :**

Le système de gestion hospitalière comprend un ensemble complet de fonctionnalités permettant la gestion efficace des patients, des médecins et des rendez-vous médicaux.

**1. Affichage:**

Liste des Patients: Affiche la liste complète des patients enregistrés, avec des détails tels que le nom, le prénom, la date de naissance, le sexe, et l'information associée.

Liste des Médecins: Affiche la liste complète des médecins enregistrés, avec des détails tels que le nom, le prénom et la spécialité.

Liste des Rendez-vous: Affiche une liste complète des rendez-vous médicaux, regroupant hospitalisations, visites, contrôles, vaccinations, analyses, et chirurgies. Pour chaque rendez-vous, les détails tels que la durée, le résultat, le type (pour les analyses et vaccinations), la date, et le médecin associé sont affichés.

**2. Ajout:**

Ajout de Patient: Permet d'ajouter un nouveau patient en saisissant les informations nécessaires telles que le nom, le prénom, la date de naissance, le sexe, et l'information associée.

Ajout de Médecin: Permet d'ajouter un nouveau médecin en saisissant les informations nécessaires telles que le nom, le prénom, et la spécialité.

Ajout de Rendez-vous: Permet d'ajouter différents types de rendez-vous médicaux, y compris hospitalisations, visites, contrôles, vaccinations, analyses, et chirurgies. Les détails spécifiques à chaque type de rendez-vous sont saisis lors de l'ajout.

**3. Suppression:**

Suppression de Patient: Permet de supprimer un patient existant de la base de données. Une confirmation peut être demandée avant la suppression.

Suppression de Médecin: Permet de supprimer un médecin existant de la base de données. Une confirmation peut être demandée avant la suppression.

Suppression de Rendez-vous: Permet de supprimer un rendez-vous médical existant de la base de données. Une confirmation peut être demandée avant la suppression.

**4. Modification:**

Modification des Informations du Patient: Permet de mettre à jour les informations d'un patient existant, y compris le nom, le prénom, la date de naissance, le sexe, et l'information associée.

Modification des Informations du Médecin: Permet de mettre à jour les informations d'un médecin existant, y compris le nom, le prénom, et la spécialité.

Modification des Informations du Rendez-vous: Permet de mettre à jour les informations d'un rendez-vous médical existant, y compris la durée, le résultat, le type (pour les analyses et vaccinations), la date, et le médecin associé.

Ces fonctionnalités fournissent une interface complète pour la gestion des patients, des médecins, et des rendez-vous médicaux, garantissant ainsi un suivi précis et une administration efficace des opérations hospitalières.

**Fonctionnalités qu'on peut ajouter :**

- Affichage du dossier médicale d'un patient dans un seul tableau en récupérant tous ses records qui qu'on l'attribut fait = True, et les ordonner selon l'attribut date, et pour accéder à chaque service référencé par ce record, en vérifie l'attribut type pour récupérer l'entité du service

- Affichage de tous les services fait par un médecin dans un seul tableau en récupérant tous les objets de toutes les entités qui le référencent

**Fonctionement du systeme:**

On a trois partie :

1- le backend : qui comporte les models et les views pour chaque fonctionnalite, compose de python et django

2- le frontend : qui comporte les forms et l'interface graphique pour l'affichage des resultats, compose de HTML et CSS

3- La liason entre le backend et frontend : qui comporte l'envoie des donnees des forms au backend avec le bon url pour faire le traitement, et le control de l'affichage en fournand les tableaux avec les resultats retoures par la backend

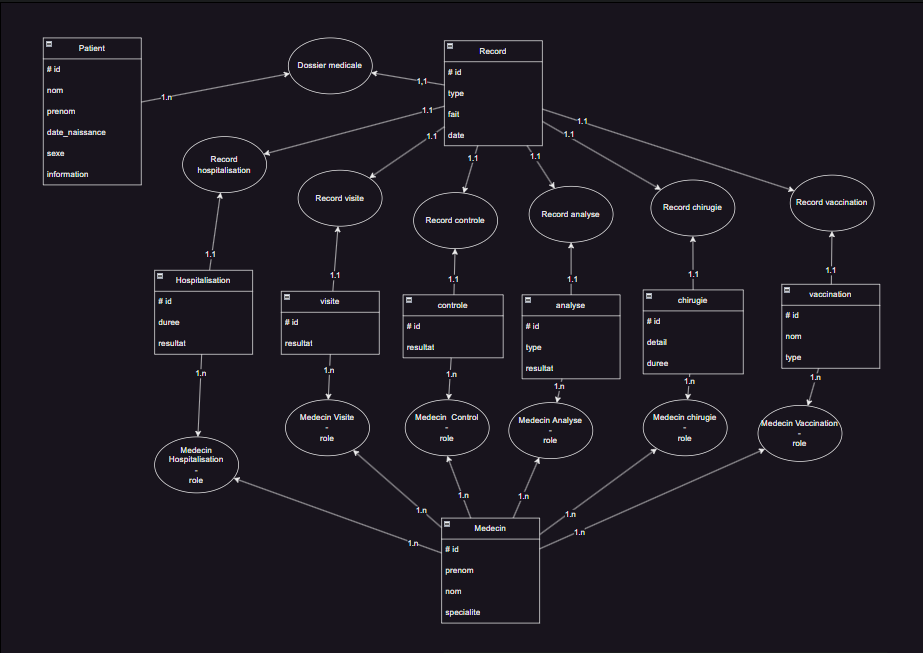


figure: MCD

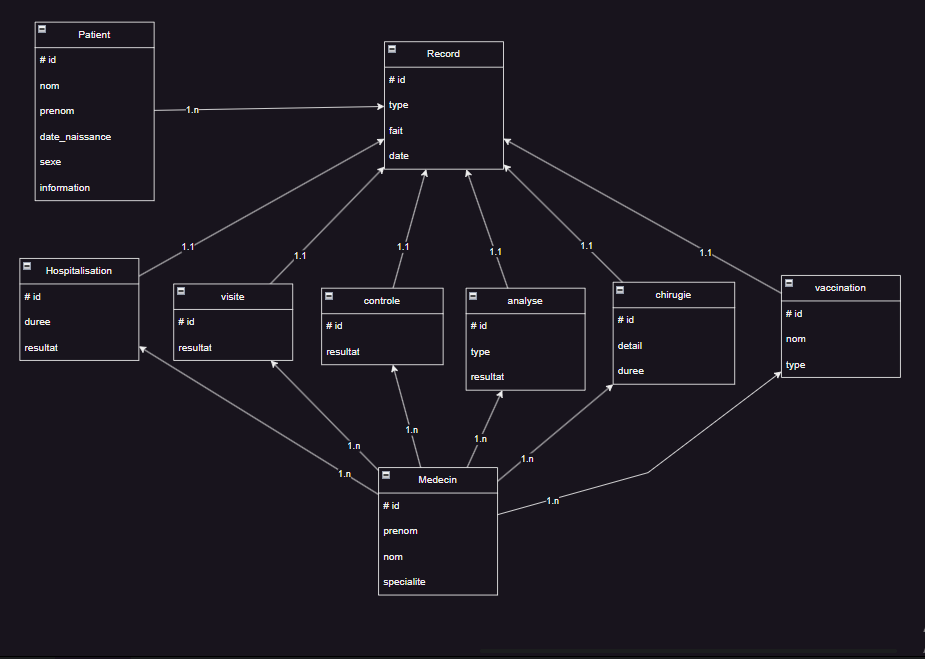


figure: diagramme de base donnée

**MLD:**

Patient( #id, nom, prenom, date\_naissance, sexe, information,

Medecin( #id, nom, prenom, specialite,

Record( #id, fait, date, type, \*patient)

Visite( #id, resultat, \*record)

Hospitalisation( #id, resultat, duree, \*record)

Control( #id, resultat, \*record)

Vaccination( #id, nom, type, \*record)

Chirurgie( #id, detail, duree, \*record)

Analyse( #id, resultat, type, \*record)

**Structure de donnée du projet:**

dans le système, on 2 type d'entités :

Acteur : qui sont Patient et Médecin qui représentent respectivement les patients qui demandent des services médicaux, et les médecins chacun avec une spécialité qui travaillent dans la clinique.

Service médicaux : sont les entités : visite, contrôle, hospitalisation, analyse, chirurgie, vaccination. Se sont les services offert par la clinique, chaque service contient des propriétés et référence un record

Un record et juste une entité commune qui contient la date du rendez-vous, son type, et est-ce que c'est fait ou pas encore

Chaque client a plusieurs record médicaux (des différents services) parmi celle qui sont fait (représenter son dossier médicale) et d'autre pas encore (présente ses rendez-vous de prochain services)

Il a aussi des informations médicales stockées dans l'attribut information (comme les allergies, les Maladies chroniques ...)