Projet de gestion de rendez-vous médicaux

Epreuve E5 BTS SIO SLAM

Réalisé par Léa Giron
01/03/2025

Table des matières

1) Introduction

1.1) Plan

Pour ce projet de gestion de rendez-vous médicaux, je vais commencer par présenter brièvement le projet. Ensuite, je vais rédiger le cahier des charges et je continuerais avec la modélisation et la conception de la base de données. Puis, j'expliquerais le développement du projet ainsi que les tests effectués. Enfin, je présenterais la documentation technique et utilisateur.

2) Cahier des charges

2.1) Présentation du projet

- **Contexte**

Ce projet s'inscrit dans le cadre des problématiques rencontrées dans les établissements de santé. En effet, l'établissement doit posséder une gestion optimisée des rendez-vous médicaux afin de garantir un suivi des consultations et une meilleure organisation des soins. Dans de nombreux hôpitaux, la prise de rendez-vous repose souvent sur des outils existants et peuvent présenter des limites en termes d'ergonomie, de performance et de sécurité des données. Cela pénalise le personnel médical et administratif mais également les patients.
De plus, le domaine médical en général et la gestion des rendez-vous médicaux sont soumis à des exigences strictes en matière de protection des données personnelles et de confidentialité. Les informations de santé étant considérées comme sensibles, elles doivent être traitées conformément au Règlement Général sur la Protection des Données (RGPD). Il est donc essentiel de garantir un stockage sécurisé, un accès restreint aux données et de respecter le droit des patients, notamment l'accès, la rectification et la suppression de leurs données.

- **Objectifs**

L'objectif principal de ce projet est d'optimiser la prise et la gestion des rendez-vous médicaux entre les patients et les médecins. De plus, il est important que l'expérience utilisateur propose une interface intuitive et fonctionnelle.

2.2) Analyse des besoins

La phase d'analyse des besoins dans un cahier des charges permet d'identifier les différents types d'utilisateurs du système et de définir les besoins fonctionnels. C'est-à-dire ce que le système doit faire pour satisfaire les besoins des utilisateurs. Puis, définir les besoins non fonctionnels comme la performance ou la sécurité du système.

D'abord, il y aura deux types d'acteurs dans ce système, qui sont les médecins et les patients. Les médecins doivent pouvoir se connecter à leur compte afin de gérer les rendez-vous (accepter, annuler, restaurer). Les patients doivent pouvoir s'inscrire et se connecter à la plateforme afin de prendre et de consulter leurs rendez-vous.

De plus, le système devra être sécurisé afin de pouvoir se conformer au Règlement Général sur la Protection des Données (RGPD). Il faut aussi qu'il soit performant et ergonomique pour que les acteurs aient accès à une plateforme facile à utiliser.

2.3) Description des fonctionnalités

Les fonctionnalités principales du système sont l'inscription à la plateforme pour les patients et la connexion pour les patients et les médecins. Une fonctionnalité de prise de rendez-vous où les patients peuvent réserver un horaire disponible avec un médecin. Ensuite, les patients et médecins peuvent visualiser tous leurs rendez-vous. Enfin, le médecin peut gérer les rendez-vous.

Le patient peut réserver un rendez-vous, il choisit un médecin, une date, une heure selon les horaires disponibles. Il peut consulter son rendez-vous sur une page dédiée avec un rappel des détails du rendez-vous.
Pour un médecin, il peut consulter tous les rendez-vous pris et également les accepter, les annuler et les restaurer.

2.4) Contraintes

En ce qui concerne les contraintes techniques, le système doit être accessible via une application web locale. Le système doit être compatible avec différents navigateurs.
Pour les contraintes réglementaires, le système doit respecter le Règlement Général sur la Protection des Données (RGPD). L'accès doit être géré pour limiter les droits en fonction des rôles (patient ou médecin).
En termes de sécurité, l'authentification doit être sécurisée, une sauvegarde régulière des données, la protection contre les attaques courantes telles que les injections SQL.
Pour l'ergonomie, l'interface doit être intuitive et facile à utiliser, l'accès doit être simplifié.
Pour le budget, les technologies sont open source mais pourront évoluer vers des solutions payantes en fonction des besoins futurs.
Enfin, il y a aussi une contrainte de délai, le respect des échéances précisées dans le planning doit être respecté pour livrer le projet avant le mois d'avril.

2.5) Architecture technique

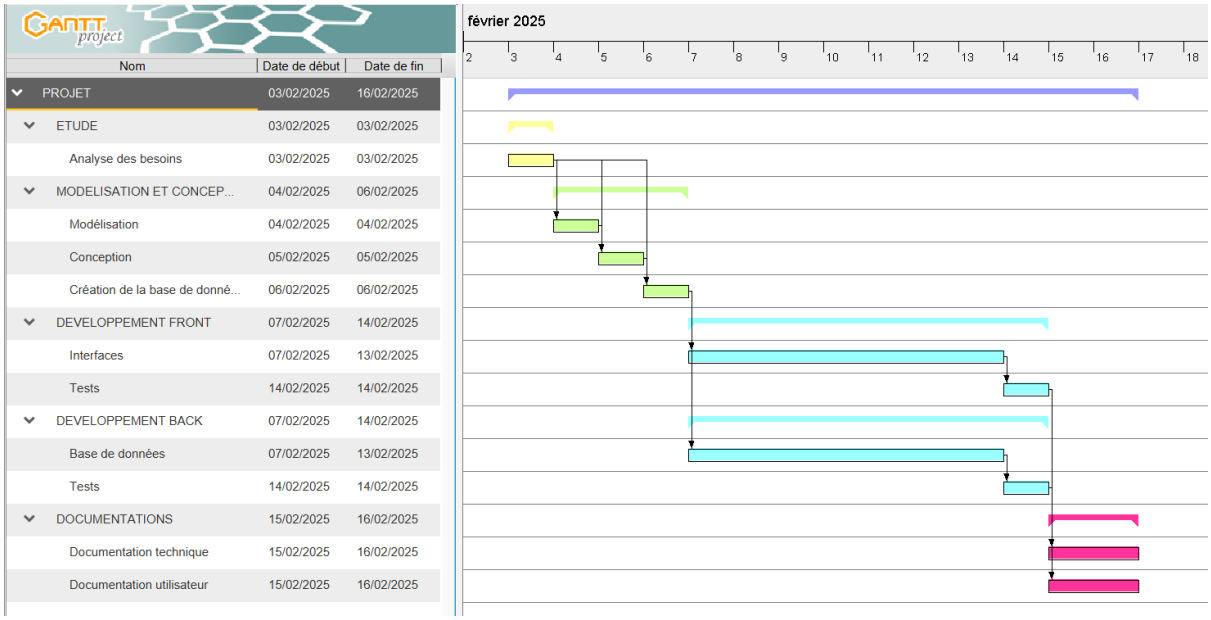
Le système est composé d'une architecture client-serveur avec une partie front-end pour l'interface utilisateur et d'une partie back-end pour le traitement des données.

Les technologies utilisées sont, pour la partie front-end, HTML et CSS. Pour la partie back-end, j'utilise PHP. Le développement s'effectue à l'aide de l'environnement de VS Code qui facilite l'écriture et le débogage du code. La base de données choisie est MySQL gérée via phpMyAdmin qui facilite la création, la modification et la suppression des tables et des données. Le serveur web local utilisé est XAMPP. Ce logiciel regroupe Apache, MySQL, PHP et Perl permettant de tester localement le projet et de reproduire les conditions d'un serveur de production.
En termes de sécurité, des requêtes préparées sont utilisées pour éviter les injections SQL. L'accès doit être géré avec un système d'authentification pour les patients et les médecins. Les composants communiquent entre eux, d'une part le front-end qui envoie les requêtes via

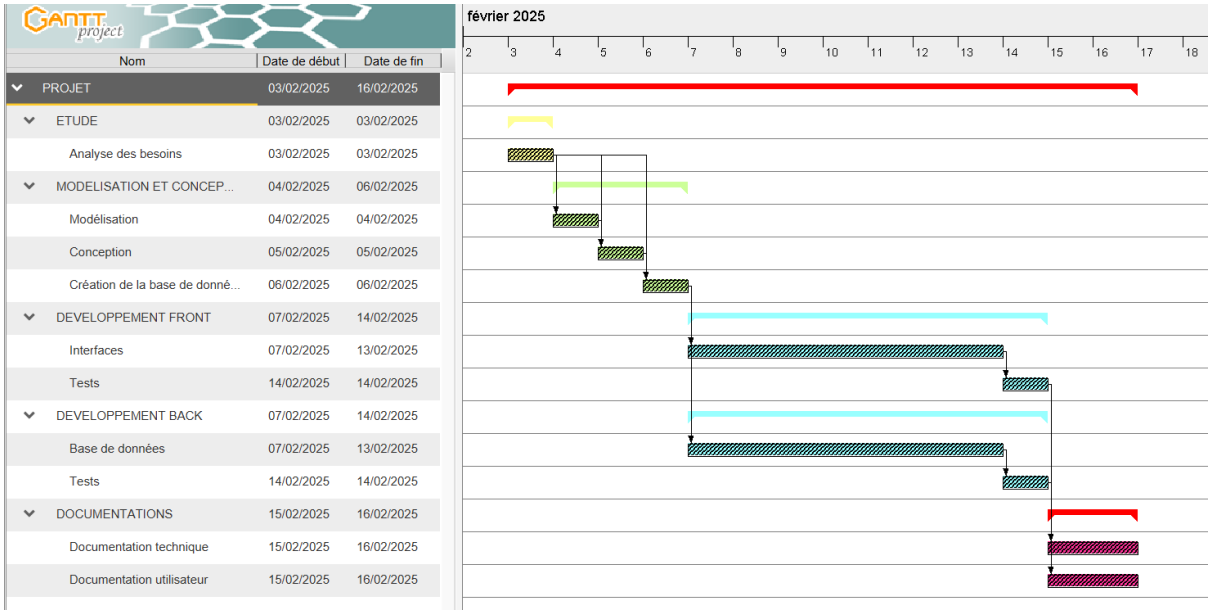
des formulaires HTML et traités d'autre part par le back-end PHP. Les données sont récupérées et affichées via PHP. Enfin, le projet est conçu pour être facilement modifiable si des évolutions sont nécessaires grâce à l'utilisation du logiciel de gestion de versions Git. La structure du code est organisée afin de permettre une maintenance simplifiée.

2.6) Planification du projet

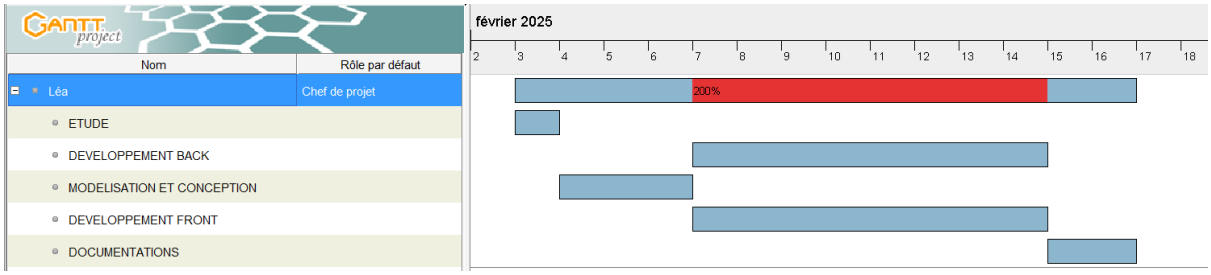
Le projet débute le 3 février 2025 et doit être finalisé le 20 février pour une livraison du projet pour le début du mois d'avril 2025.
Je réalise le planning prévisionnel à l'aide de GanttProject. L'utilité de ce planning est d'avoir une vue d'ensemble des étapes nécessaires à la réalisation de ce projet. En dessous, je visualise le chemin critique qui me permet de voir les tâches obligatoires. Grâce au tableau des ressources, je peux voir que je suis chef de projet et l'ensemble des groupes de tâches que je dois réaliser.



Nom	Date de début	Date de fin
PROJET	03/02/2025	16/02/2025
ETUDE	03/02/2025	03/02/2025
Analyse des besoins	03/02/2025	03/02/2025
MODELISATION ET CONCEP...	04/02/2025	06/02/2025
Modélisation	04/02/2025	04/02/2025
Conception	05/02/2025	05/02/2025
Création de la base de donné...	06/02/2025	06/02/2025
DEVELOPPEMENT FRONT	07/02/2025	14/02/2025
Interfaces	07/02/2025	13/02/2025
Tests	14/02/2025	14/02/2025
DEVELOPPEMENT BACK	07/02/2025	14/02/2025
Base de données	07/02/2025	13/02/2025
Tests	14/02/2025	14/02/2025
DOCUMENTATIONS	15/02/2025	16/02/2025
Documentation technique	15/02/2025	16/02/2025
Documentation utilisateur	15/02/2025	16/02/2025



Nom	Date de début	Date de fin
PROJET	03/02/2025	16/02/2025
ETUDE	03/02/2025	03/02/2025
Analyse des besoins	03/02/2025	03/02/2025
MODELISATION ET CONCEP...	04/02/2025	06/02/2025
Modélisation	04/02/2025	04/02/2025
Conception	05/02/2025	05/02/2025
Création de la base de donné...	06/02/2025	06/02/2025
DEVELOPPEMENT FRONT	07/02/2025	14/02/2025
Interfaces	07/02/2025	13/02/2025
Tests	14/02/2025	14/02/2025
DEVELOPPEMENT BACK	07/02/2025	14/02/2025
Base de données	07/02/2025	13/02/2025
Tests	14/02/2025	14/02/2025
DOCUMENTATIONS	15/02/2025	16/02/2025
Documentation technique	15/02/2025	16/02/2025
Documentation utilisateur	15/02/2025	16/02/2025



Les livrables attendus pour ce projet sont le cahier des charges, la modélisation et la conception de la base de données, le code source du projet, la documentation technique et d'utilisation, la documentation du projet, le rapport de tests et la plateforme fonctionnelle.

Les ressources mobilisées pour ce projet sont de type matérielles, logicielles et humaines. Elles se composent d'un ordinateur, d'un serveur local XAMPP, d'un logiciel de gestion de base de données phpMyAdmin ainsi qu'un éditeur de code VS Code, un outil collaboratif de gestion des versions Git, le langage de programmation PHP, un système de gestion de bases de données relationnelles MySQL, un navigateur web. Enfin, je serais seule à réaliser ce projet qui est individuel.

2.7) Critères de validation

Pour que le projet soit validé, il doit répondre aux exigences définies dans le cahier des charges. Les critères de validation incluent toutes les fonctionnalités prévues, le respect des contraintes techniques, les différents tests sont réussis. De plus, l'interface doit être intuitive et ergonomique et respecter les règles de sécurité.

2.8) Evolution

Ce projet est conçu pour être évolutif et pourra être amélioré selon les besoins futurs. Par exemple, le système pourra envoyer des notifications pour envoyer un rappel de rendez-vous au patient. Le patient pourra également annuler lui-même son rendez-vous et modifier ses informations personnelles.

3) Modélisation de la base de données

Ici, je vais visualiser toutes les données dont je vais avoir besoin et les représenter sous forme d'entités et de relations grâce aux méthodes Merise et UML.

3.1) Le dictionnaire de données

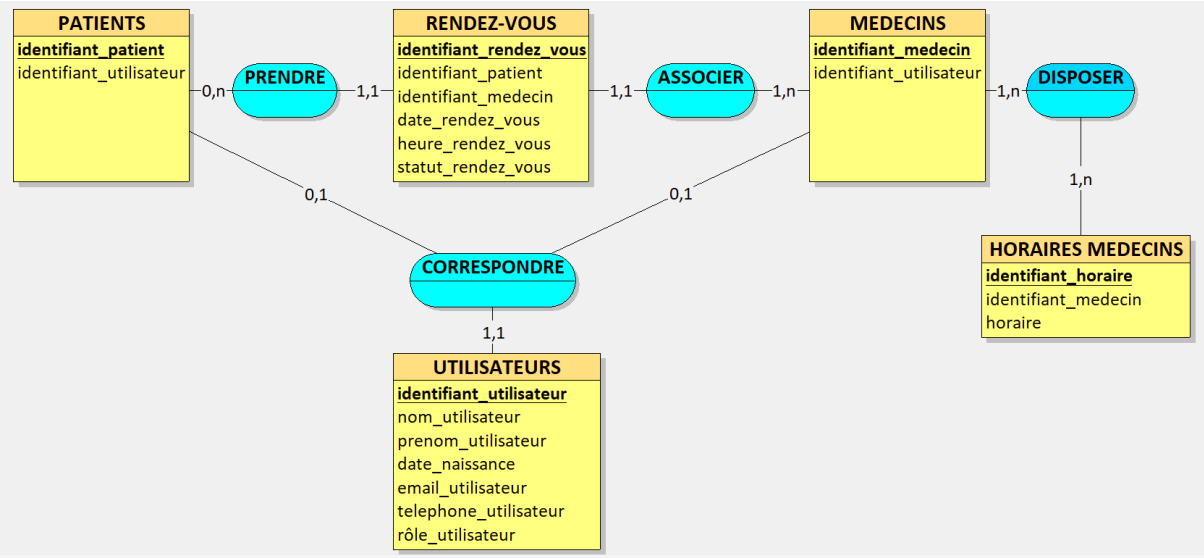
Je crée le dictionnaire de données à l'aide d'un tableau Excel. Il permet de concevoir et gérer une base de données. Il est utile pour décrire les données, faciliter la communication entre développeurs, standardiser et comprendre la structure des données.

TABLE	Nom	Nom logique	Type	Longueur	Observation
UTILISATEURS	identifiant_utilisateur	util_id_utilisateur	INT (numérique)	11	Clé primaire, identifiant unique pour chaque utilisateur
	nom_utilisateur	util_nom	VARCHAR (caractère)	50	Nom de famille de l'utilisateur
	prenom_utilisateur	util_prenom	VARCHAR (caractère)	50	Prénom de l'utilisateur
	date_naissance	util_date_naissance	DATE		Date de naissance de l'utilisateur
	email_utilisateur	util_email	VARCHAR (caractère)	100	Email de l'utilisateur
	telephone_utilisateur	util_telephone	VARCHAR (numérique)	10	Numéro de téléphone de l'utilisateur
	mot_de_passe_utilisateur	util_mot_de_passe	VARCHAR (numérique)	255	Mot de passe de l'utilisateur
	rôle_utilisateur	util_rôle	VARCHAR (caractère)	20	Rôle de l'utilisateur (patient ou médecin)
PATIENTS	identifiant_patient	pat_id_patient	INT (numérique)	11	Clé primaire, identifiant unique pour chaque patient
	identifiant_utilisateur	util_id_utilisateur	INT (numérique)	11	Clé étrangère, référence à l'identifiant de l'utilisateur dans la table "utilisateurs"
MÉDECINS	identifiant_medecin	med_id_medecin	INT (numérique)	11	Clé primaire, identifiant unique pour chaque médecin
	identifiant_utilisateur	util_id_utilisateur	INT (numérique)	11	Clé étrangère, référence à l'identifiant de l'utilisateur dans la table "utilisateurs"
HORAIRES MEDECINS	identifiant_horaire	hor_id_horaire	INT (numérique)	11	Clé primaire, identifiant unique pour chaque plage horaire
	identifiant_medecin	hor_id_medecin	INT (numérique)	11	Clé étrangère, référence à l'identifiant du médecin
	horaire	hor_horaire	TEMPS		Horaire du rendez-vous
RENDEZ-VOUS	identifiant_rendez_vous	rdv_id_rendez_vous	INT (numérique)	11	Clé primaire, identifiant unique pour chaque rendez-vous
	identifiant_patient	rdv_id_patient	INT (numérique)	11	Clé étrangère, référence à l'identifiant du patient
	identifiant_medecin	rdv_id_medecin	INT (numérique)	11	Clé étrangère, référence à l'identifiant du médecin
	date_rendez_vous	rdv_date_rendez_vous	DATE		Date du rendez-vous
	heure_rendez_vous	rdv_heure_rendez_vous	TEMPS		Heure du rendez-vous
	statut_rendez_vous	rdv_statut_rendez_vous	VARCHAR (caractère)	30	Statut du rendez-vous (en attente, accepté, refusé)

3.2) Modèle conceptuel de données et modèle physique de données

La méthode Merise est utilisée pour créer des bases de données et modéliser les processus en informatique. D'abord, je conçois le modèle conceptuel de données (MCD) qui représente les données du système d'information de manière abstraite sous la forme d'entités, d'associations, d'attributs et de cardinalités.
Enfin, je crée le modèle physique de données (MPD) qui est adapté aux caractéristiques techniques du système de gestion de bases de données.
Pour concevoir ces modèles, je vais utiliser le logiciel Looping.

Voici le modèle conceptuel de données (MCD) :



Ensuite, je vais expliciter les relations.

PATIENTS - RENDEZ-VOUS

Un patient peut avoir pris 0 ou plusieurs rendez-vous.

Un rendez-vous ne peut prendre qu'un seul patient.

MÉDECINS - RENDEZ-VOUS

Un médecin peut avoir un ou plusieurs rendez-vous.

Un rendez-vous est associé à un seul médecin.

HORAIRES-MÉDECINS - MÉDECINS

Un horaire est associé à un ou plusieurs médecins.

Un médecin peut avoir plusieurs horaires.

UTILISATEURS - PATIENTS

Un patient correspond à un utilisateur.

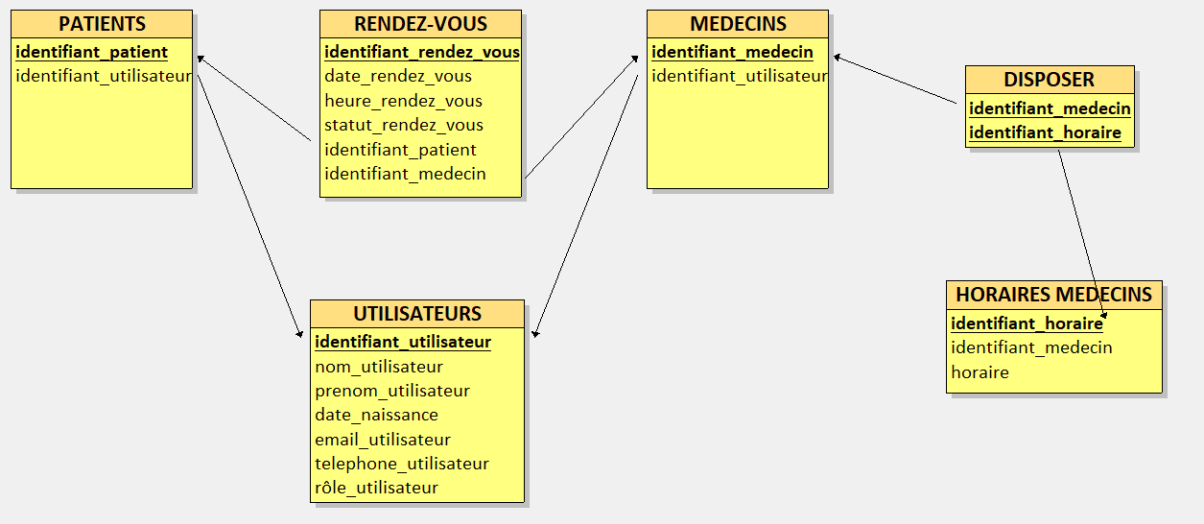
Un utilisateur correspond à un patient ou autre.

UTILISATEURS - MÉDECINS

Un médecin correspond à un utilisateur.

Un utilisateur correspond à un médecin ou autre.

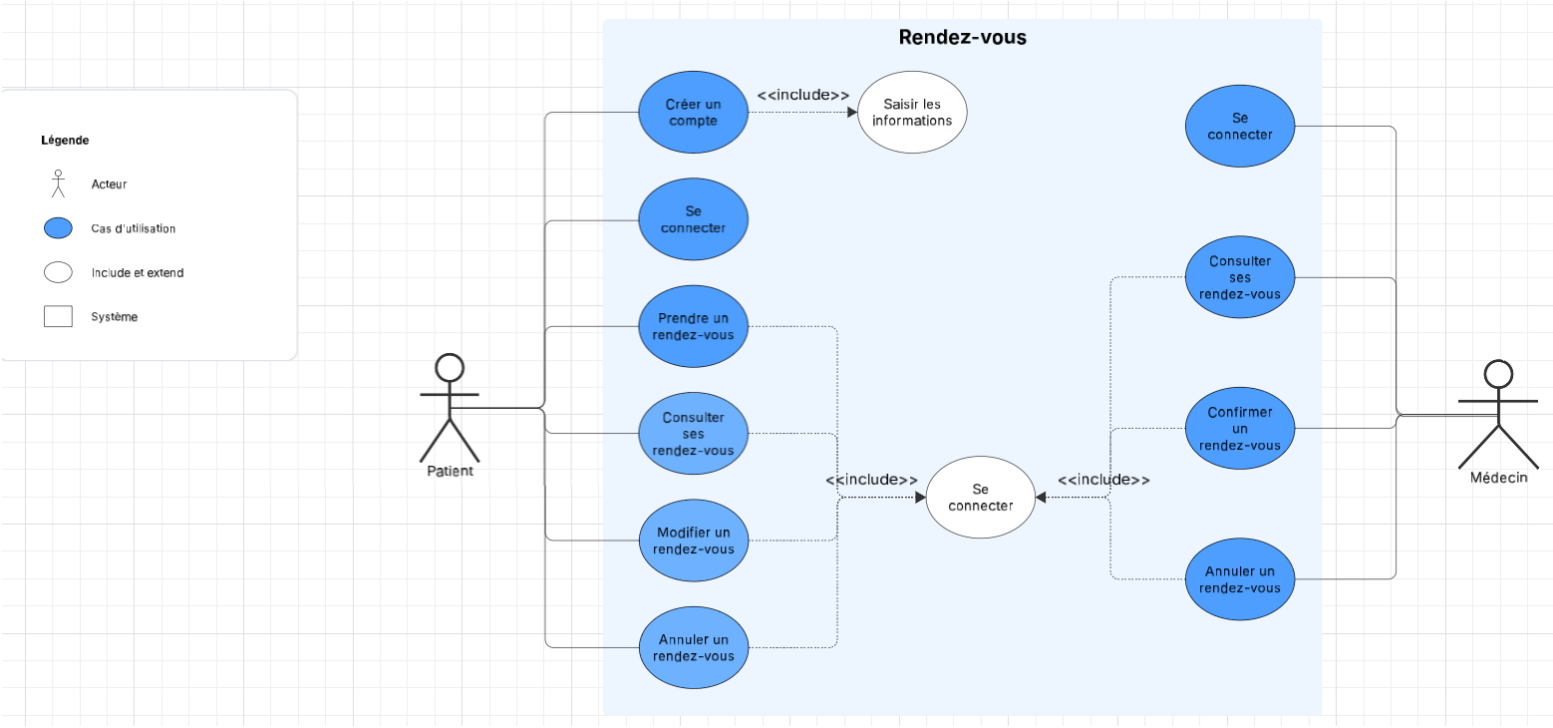
Je réalise le modèle physique de données (MPD) :



3.3) Diagramme de cas d'utilisation et diagramme de séquence

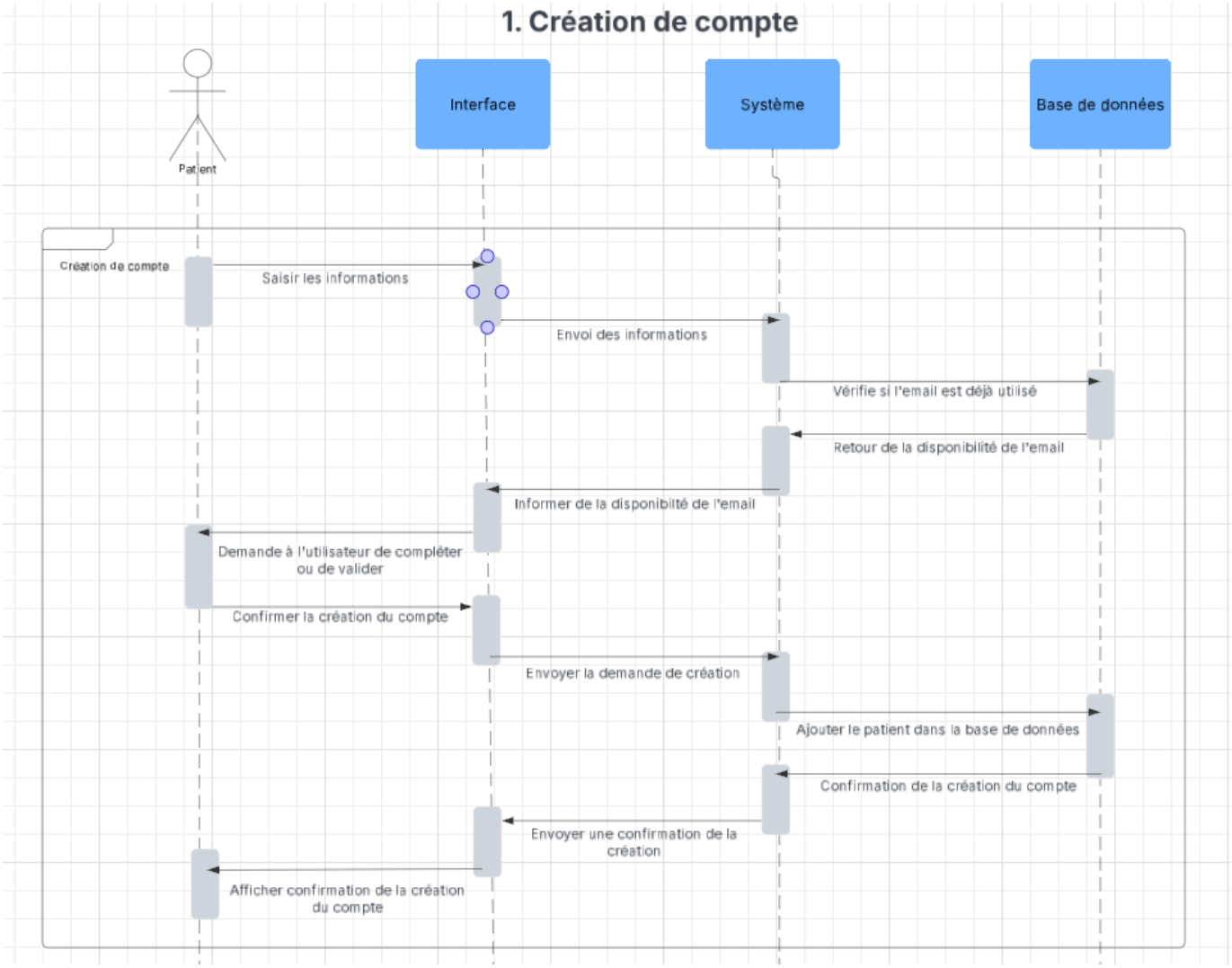
Le diagramme de cas d'utilisation est utilisé en langage de modélisation unifié (UML), il montre les interactions entre les utilisateurs et le système.

Je vais commencer par ce diagramme avec les patients et les médecins qui seront les utilisateurs. Pour cela, je m'aide de l'outil en ligne Lucidchart.

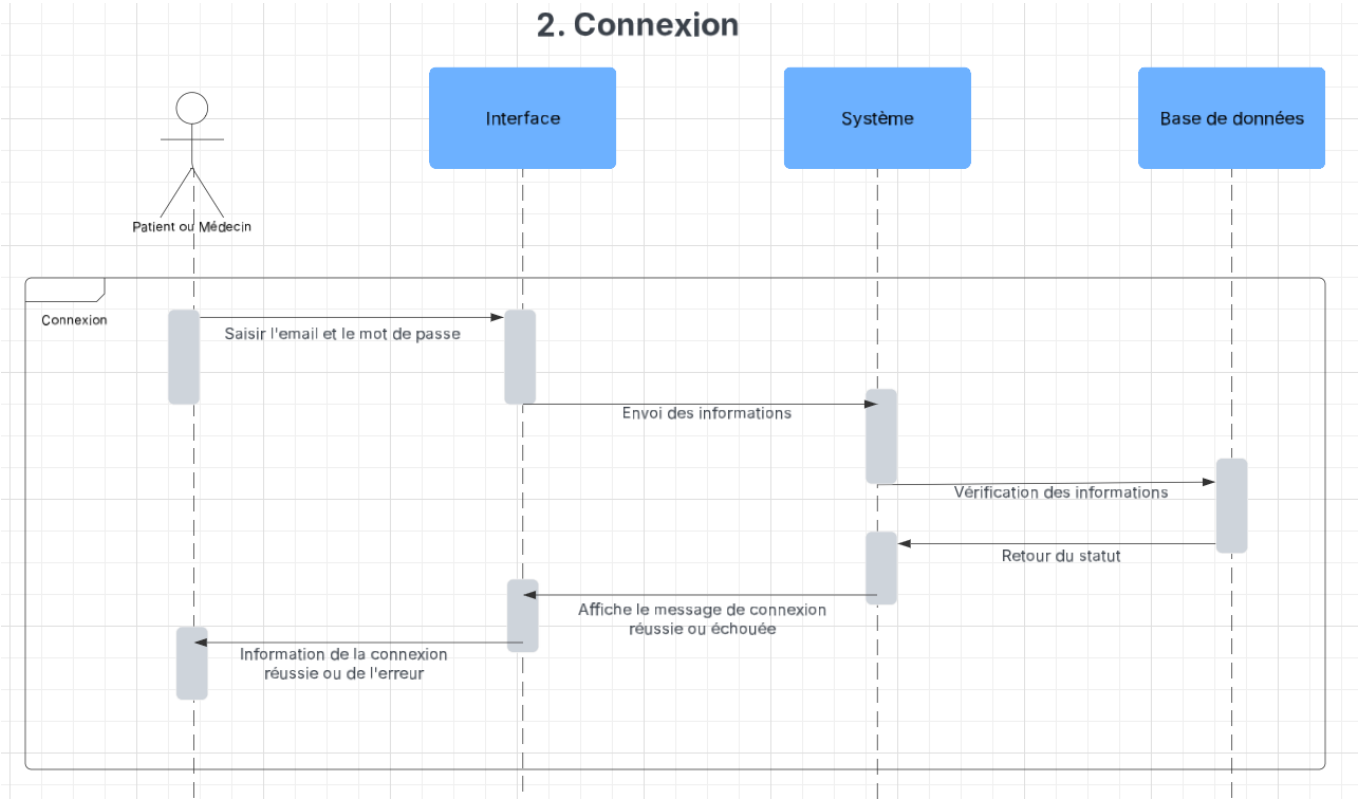


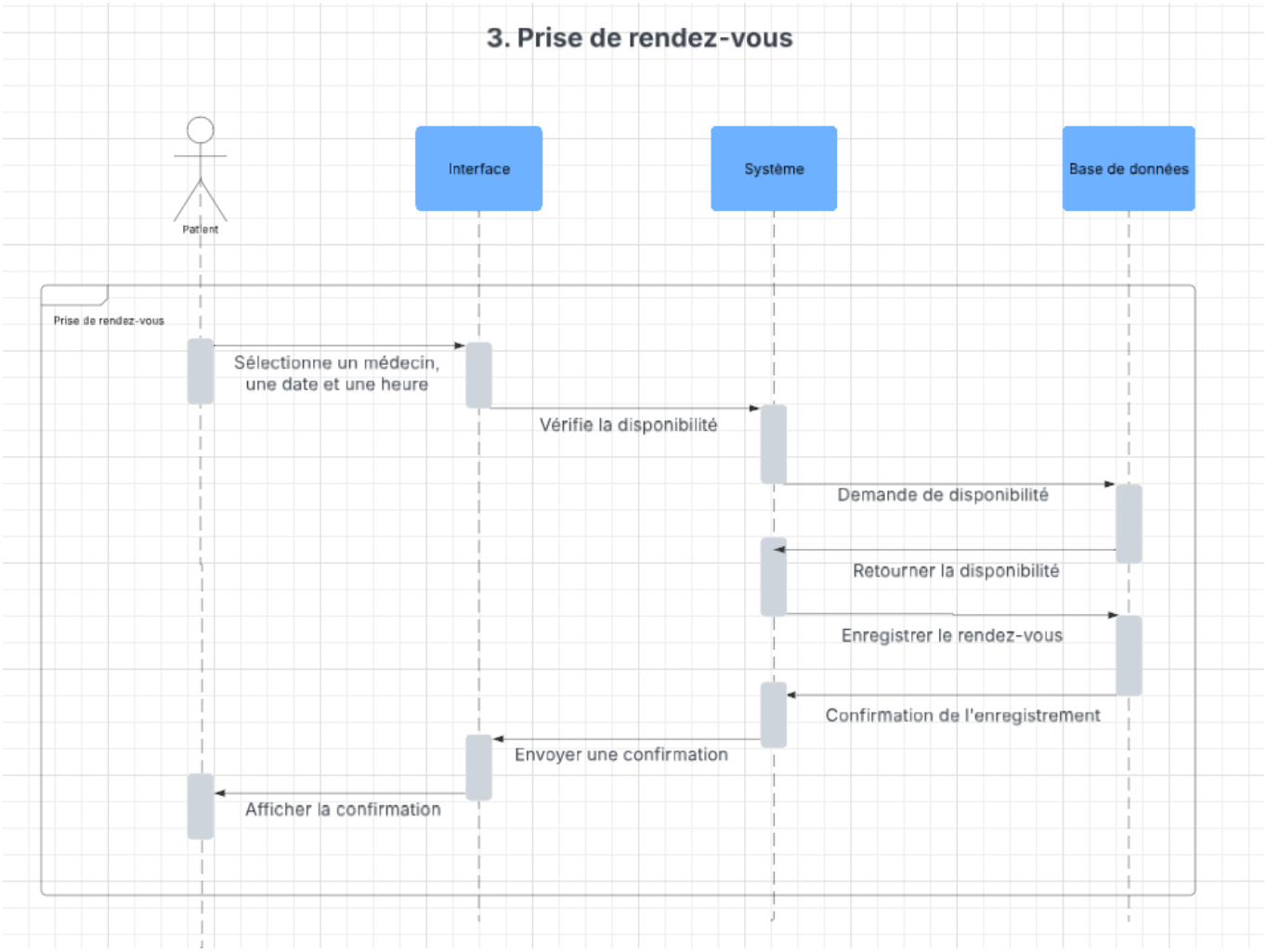
Le diagramme de séquence est utilisé en langage de modélisation unifié (UML), il représente le déroulement des interactions entre les acteurs et le système. Ici, il y a cinq acteurs qui sont le patient ou médecin, l'interface, le système et la base de données. Chaque diagramme représente une seule fonctionnalité dont la création de compte, la connexion, la prise de rendez-vous, la consultation, la modification et l'annulation des rendez-vous.

1. Création de compte

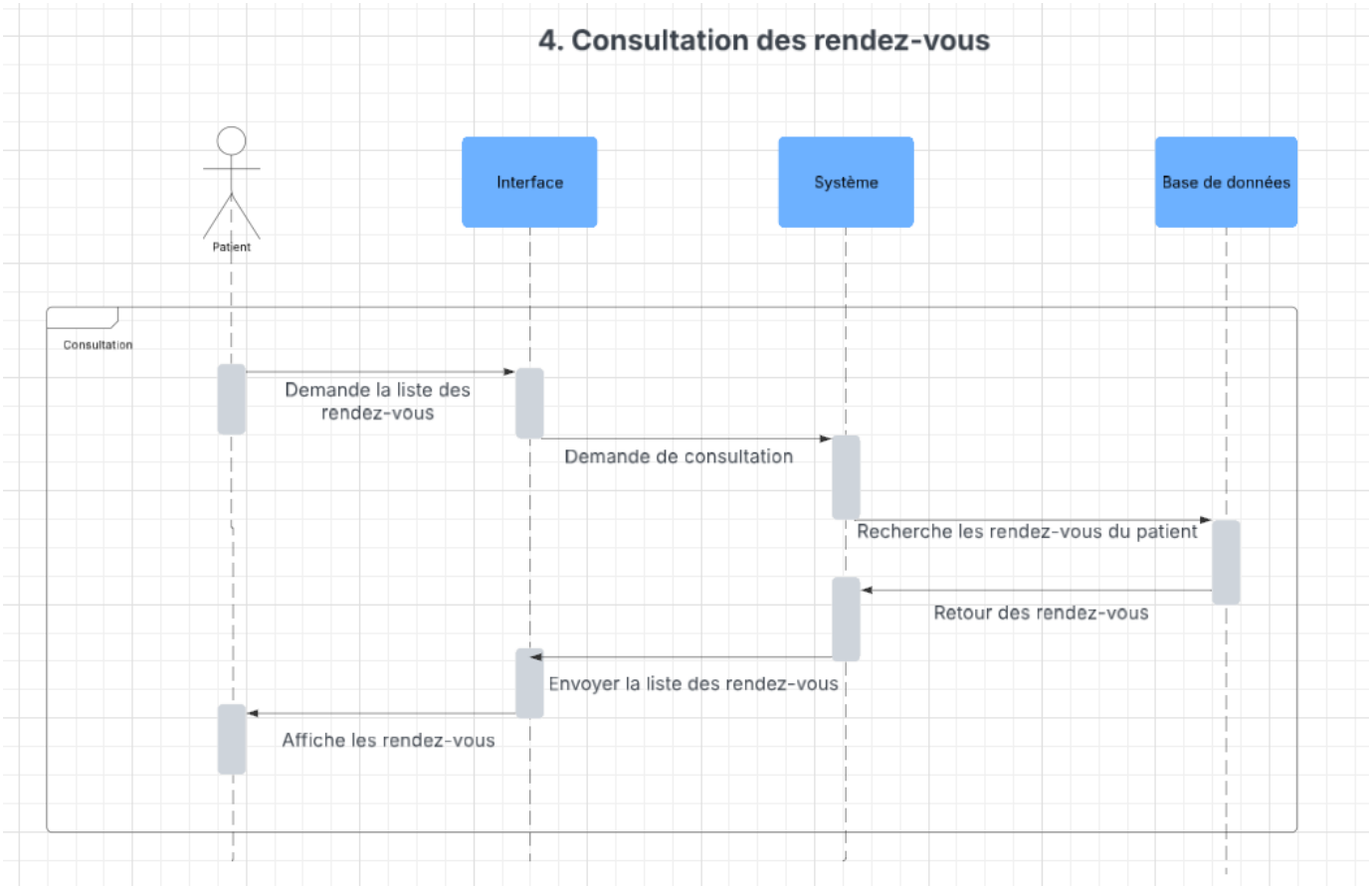


2. Connexion

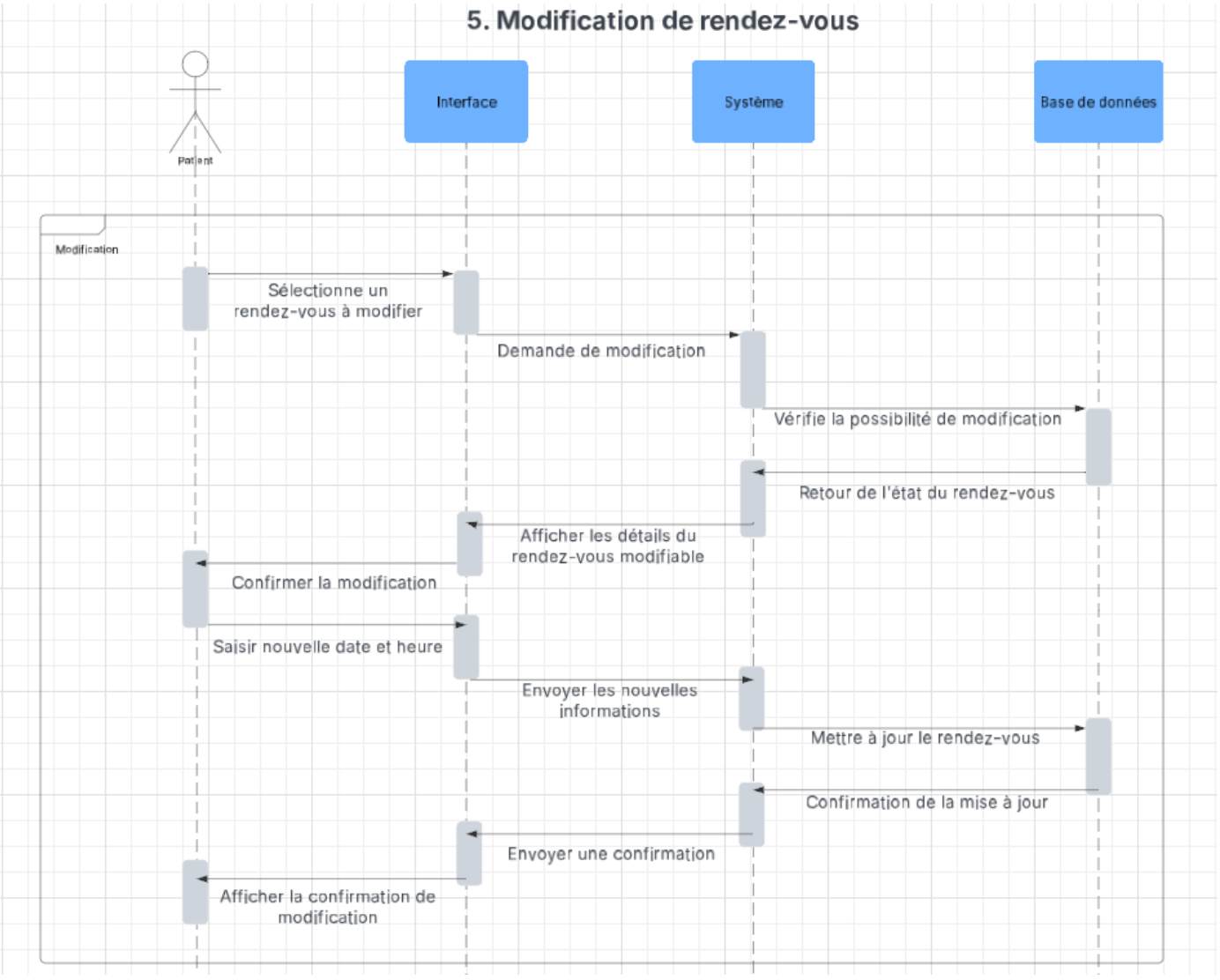


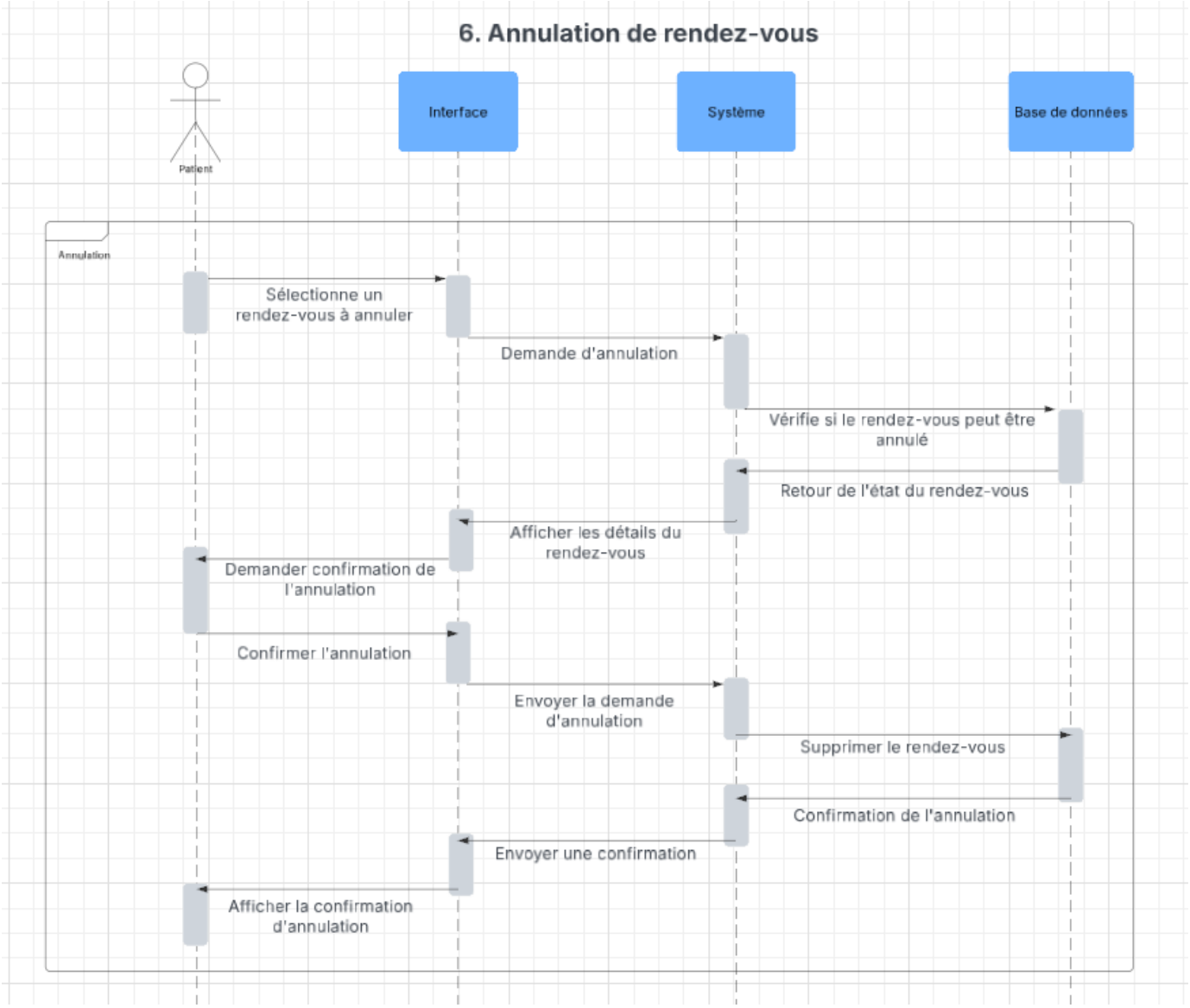

3. Prise de rendez-vous
Patient
Interface
Système
Base de données
Prise de rendez-vous
Sélectionne un médecin, une date et une heure
Vérifie la disponibilité
Demande de disponibilité
Retourner la disponibilité
Enregistrer le rendez-vous
Confirmation de l'enregistrement
Envoyer une confirmation
Afficher la confirmation

4. Consultation des rendez-vous



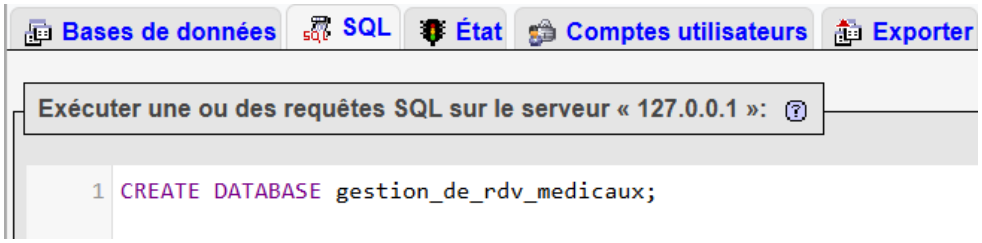
5. Modification de rendez-vous





4) Conception de la base de données

Maintenant, je vais créer la base de données MySQL sur phpMyAdmin. Pour cela, je crée une nouvelle base de données et j'y ajoute des tables grâce à des requêtes SQL.
Ici, je crée la base de données :



Ensuite, je crée la table “patients”. Chaque patient est associé à un utilisateur enregistré dans la table “utilisateurs” grâce à une clé étrangère (util_id_utilisateur). Cela permet de récupérer automatiquement les informations de la table “utilisateurs”. Grâce à “ON DELETE CASCADE”, un utilisateur supprimé de la table “utilisateurs” sera supprimé également dans la table “patients”.

```
CREATE TABLE Patients (
    pat_id_patient INT AUTO_INCREMENT PRIMARY KEY,
    util_id_utilisateur INT,
    FOREIGN KEY (util_id_utilisateur) REFERENCES Utilisateurs(util_id_utilisateur)
    ON DELETE CASCADE
);
```

Je crée la table “medecins”. Chaque médecin est associé à un utilisateur existant dans la table “utilisateurs” via une clé étrangère (util_id_utilisateur). Cela permet de récupérer automatiquement les informations du médecin (nom, prénom, email, mot de passe, numéro de téléphone, rôle) à partir de la table “utilisateurs”. Grâce à “ON DELETE CASCADE”, si un utilisateur est supprimé, il sera supprimé de la table “utilisateurs” et “medecins”.

```
CREATE TABLE Medecins (
    med_id_medecin INT AUTO_INCREMENT PRIMARY KEY,
    util_id_utilisateur INT NOT NULL,
    CONSTRAINT fk_utilisateur FOREIGN KEY (util_id_utilisateur)
        REFERENCES Utilisateurs(util_id_utilisateur)
        ON DELETE CASCADE
);
```

Je crée la table “horaires_medecins” pour stocker les horaires des médecins. Chaque horaire est lié à un médecin via hor_id_medecin qui est une clé étrangère référencée dans la table “medecins”. Grâce à ON DELETE CASCADE, si un médecin est supprimé, ses horaires sont supprimés également.

```
CREATE TABLE horaires_medecins (
    hor_id_horaire INT AUTO_INCREMENT PRIMARY KEY,
    hor_id_medecin INT NOT NULL,
    hor_heure TIME NOT NULL,
    CONSTRAINT fk_hor_medecin FOREIGN KEY (hor_id_medecin) REFERENCES medecins(med_id_medecin) ON DELETE CASCADE
);
```

La table “rendez_vous” enregistre les rendez-vous médicaux. Chaque rendez-vous est lié à un patient et un médecin via des clés étrangères. Elle stocke la date, l’heure et le statut du rendez-vous (en attente, confirmé, annulé).

```sql
CREATE TABLE RENDEZ_VOUS (
    rdv_id_rendez_vous INT AUTO_INCREMENT PRIMARY KEY,
    rdv_id_patient INT NOT NULL,
    rdv_id_medecin INT NOT NULL,
    rdv_date_rendez_vous DATE NOT NULL,
    rdv_heure_rendez_vous TIME NOT NULL,
    rdv_statut_rendez_vous VARCHAR(30),
    FOREIGN KEY (rdv_id_patient) REFERENCES PATIENTS (pat_id_patient) ON DELETE CASCADE,
    FOREIGN KEY (rdv_id_medecin) REFERENCES MEDECINS (med_id_medecin) ON DELETE CASCADE
);

```

```sql
ALTER TABLE RENDEZ_VOUS
MODIFY rdv_statut_rendez_vous ENUM('Confirmé', 'Annulé', 'En attente') NOT NULL;

```

Je crée la table “utilisateurs” pour stocker les informations des utilisateurs qui sont soit des patients soit des médecins (identifiant unique, nom, prénom, email, mot de passe, téléphone, rôle).

```sql
CREATE TABLE Utilisateurs (
    util_id_utilisateur INT AUTO_INCREMENT PRIMARY KEY,
    util_nom VARCHAR(50) NOT NULL,
    util_prenom VARCHAR(50) NOT NULL,
    util_email VARCHAR(100) NOT NULL UNIQUE,
    util_mot_de_passe VARCHAR(255) NOT NULL,
    util_telephone VARCHAR(10),
    util_role ENUM('patient', 'medecin') NOT NULL
);

```

4.1) Insérer des données de test

Voici toutes les tables créées, je vais insérer quelques données de test pour vérifier que tout fonctionne correctement.
D’abord, je crée des triggers qui se déclenchent après l’insertion d’un nouvel utilisateur dans la table “utilisateurs”. Ce déclencheur vérifie si l’utilisateur est un patient ou un médecin. Si c’est le cas, il insère l’identifiant de l’utilisateur du patient ou du médecin dans la table “patients” ou “medecins” de manière automatique.

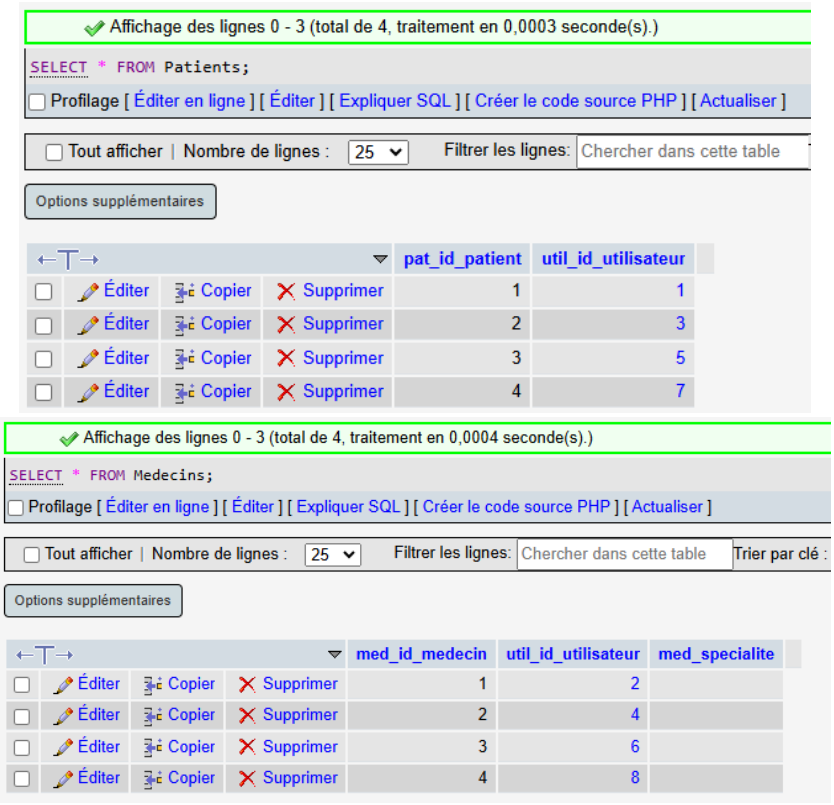
```sql
DELIMITER //
CREATE TRIGGER inserer_patient_apres_utilisateur
AFTER INSERT ON Utilisateurs
FOR EACH ROW
BEGIN
    IF NEW.util_role = 'patient' THEN
        INSERT INTO Patients (util_id_utilisateur)
        VALUES (NEW.util_id_utilisateur);
    END IF;
END;
//
DELIMITER ;
```

```sql
DELIMITER //
CREATE TRIGGER inserer_medecin_apres_utilisateur
AFTER INSERT ON Utilisateurs
FOR EACH ROW
BEGIN
    IF NEW.util_role = 'medecin' THEN
        INSERT INTO Medecins (util_id_utilisateur)
        VALUES (NEW.util_id_utilisateur);
    END IF;
END;
//
DELIMITER ;
```

J’ajoute ensuite quelques données de test pour la table “utilisateurs”. On retrouve ici les informations utiles pour la connexion à la plateforme tel que l’email et le mot de passe des patients et médecins.

```sql
INSERT INTO Utilisateurs (util_nom, util_prenom, util_email, util_mot_de_passe, util_telephone, util_role) VALUES
('Dupont', 'Jean', 'jean.dupont@email.com', 'mdp123', '0601020304', 'patient'),
('Martin', 'Alice', 'alice.martin@email.com', 'mdp456', '0611223344', 'medecin'),
('Lemoine', 'Sophie', 'sophie.lemoine@email.com', 'mdp789', '0622334455', 'patient'),
('Bertrand', 'Thomas', 'thomas.bertrand@email.com', 'mdp358', '0633445566', 'medecin'),
('Garnier', 'Camille', 'camille.garnier@email.com', 'mdp124', '0644556677', 'patient'),
('Morel', 'Nicolas', 'nicolas.morel@email.com', 'mdp756', '0655667788', 'medecin'),
('Lefevre', 'Chloé', 'chloe.lefevre@email.com', 'mdp486', '0666778899', 'patient'),
('Durand', 'Pierre', 'pierre.durand@email.com', 'mdp548', '0677889900', 'medecin');
```

Je vérifie que les patients et médecins sont ajoutés automatiquement dans leur table respective grâce aux triggers. J’utilise deux requêtes SQL dans les tables “patients” et “médecins” qui récupère l’identifiant du patient / médecin et son identifiant utilisateur correspondant.



Ensuite, je fais une jointure entre les tables “utilisateurs” et “médecins” pour obtenir l’identifiant du médecin stocké dans la table “medecins” et son nom et prénom dans la table “utilisateurs”.

```sql
SELECT
    medecins.med_id_medecin,
    utilisateurs.util_nom,
    utilisateurs.util_prenom
FROM
    medecins
JOIN
    utilisateurs
    ON medecins.util_id_utilisateur = utilisateurs.util_id_utilisateur;

```

J’ajoute des horaires pour les médecins avec son identifiant correspondant (med_id_medecin) dans la table “médecins”.

```sql
INSERT INTO horaires_medecins (hor_id_medecin, hor_heure) VALUES
(1, '08:00:00'),(1, '08:30:00'),(1, '09:00:00'),(1, '09:30:00'),(1, '10:00:00'),(1, '10:30:00'),(1, '11:00:00'),(1, '11:30:00'),(1, '12:00:00'),
(1, '13:30:00'),(1, '14:00:00'),(1, '14:30:00'),(1, '15:00:00'),(1, '15:30:00'),(1, '16:00:00'),(1, '16:30:00'),(1, '17:00:00'),(1, '17:30:00'),(1, '18:00:00'),
(2, '08:00:00'),(2, '08:30:00'),(2, '09:00:00'),(2, '09:30:00'),(2, '10:00:00'),(2, '10:30:00'),(2, '11:00:00'),(2, '11:30:00'),(2, '12:00:00'),
(2, '13:30:00'),(2, '14:00:00'),(2, '14:30:00'),(2, '15:00:00'),(2, '15:30:00'),(2, '16:00:00'),(2, '16:30:00'),(2, '17:00:00'),(2, '17:30:00'),(2, '18:00:00'),
(3, '08:00:00'),(3, '08:30:00'),(3, '09:00:00'),(3, '09:30:00'),(3, '10:00:00'),(3, '10:30:00'),(3, '11:00:00'),(3, '11:30:00'),(3, '12:00:00'),
(3, '13:30:00'),(3, '14:00:00'),(3, '14:30:00'),(3, '15:00:00'),(3, '15:30:00'),(3, '16:00:00'),(3, '16:30:00'),(3, '17:00:00'),(3, '17:30:00'),(3, '18:00:00'),
(4, '08:00:00'),(4, '08:30:00'),(4, '09:00:00'),(4, '09:30:00'),(4, '10:00:00'),(4, '10:30:00'),(4, '11:00:00'),(4, '11:30:00'),(4, '12:00:00'),
(4, '13:30:00'),(4, '14:00:00'),(4, '14:30:00'),(4, '15:00:00'),(4, '15:30:00'),(4, '16:00:00'),(4, '16:30:00'),(4, '17:00:00'),(4, '17:30:00'),
(4, '18:00:00');

```

Enfin, j'ajoute des rendez-vous avec un patient et un médecin associé grâce aux identifiants rdv_id_patient et rdv_id_medecin. Je précise une date, une heure et le statut.

```sql
INSERT INTO rendez_vous (rdv_id_patient, rdv_id_medecin, rdv_date_rendez_vous, rdv_heure_rendez_vous, rdv_statut_rendez_vous)
VALUES
(1, 1, '2025-02-20', '08:00:00', 'Confirmé'),
(2, 3, '2025-02-21', '10:00:00', 'En attente'),
(3, 4, '2025-02-22', '14:00:00', 'Confirmé'),
(4, 2, '2025-02-23', '16:00:00', 'Annulé');

```

4.2) Tester les relations entre les tables

Je vais tester les relations entre les tables à l'aide de requêtes SQL pour vérifier que tout fonctionne. J'utilise des alias dans chaque requête pour une meilleure lisibilité.
Je vais d'abord lister tous les rendez-vous avec les noms des patients et des médecins ainsi que l'identifiant du rendez-vous, la date, l'heure et le statut. Elle joint plusieurs tables entre elles, la table "rendez_vous", la table "patients" pour relier chaque rendez-vous à un patient. La table "utilisateurs" pour récupérer le nom et prénom du patient. Elle joint également la table "médecins" et la table "utilisateurs" pour récupérer les informations du médecin.
Par exemple, le patient Jean Dupont a rendez-vous avec le médecin Alice Martin le 20 février 2025 à 8h00 et son rendez-vous est confirmé par le médecin.

```sql
SELECT
    rendez_vous.rdv_id_rendez_vous,
    rendez_vous.rdv_date_rendez_vous,
    rendez_vous.rdv_heure_rendez_vous,
    rendez_vous.rdv_statut_rendez_vous,
    p.util_nom AS patient_nom,
    p.util_prenom AS patient_prenom,
    m.util_nom AS med_nom,
    m.util_prenom AS med_prenom
FROM
    rendez_vous
JOIN
    patients ON rendez_vous.rdv_id_patient = patients.pat_id_patient
JOIN
    utilisateurs AS p ON patients.util_id_utilisateur = p.util_id_utilisateur
JOIN
    medecins ON rendez_vous.rdv_id_medecin = medecins.med_id_medecin
JOIN
    utilisateurs AS m ON medecins.util_id_utilisateur = m.util_id_utilisateur;

```

rdv_id_rendez_vous	rdv_date_rendez_vous	rdv_heure_rendez_vous	rdv_statut_rendez_vous	patient_nom	patient_prenom	med_nom	med_prenom
1	2025-02-20	08:00:00	Confirmé	Dupont	Jean	Martin	Alice
2	2025-02-21	10:00:00	En attente	Lemoine	Sophie	Morel	Nicolas
3	2025-02-22	14:00:00	Confirmé	Garnier	Camille	Durand	Pierre
4	2025-02-23	16:00:00	Annulé	Lefevre	Chloé	Bertrand	Thomas

Je peux vérifier les horaires de chaque médecin. Elle joint la table “horaires_medecins” pour les horaires, la table “medecins” pour relier chaque horaire à un médecin, et la table “utilisateurs” pour récupérer le nom et prénom du médecin. Elle affiche également l’horaire.

```sql
SELECT
    u.util_nom AS nom_medecin,
    u.util_prenom AS prenom_medecin,
    h.hor_id_horaire,
    h.hor_heure
FROM
    horaires_medecins h
JOIN
    medecins m ON h.hor_id_medecin = m.med_id_medecin
JOIN
    utilisateurs u ON m.util_id_utilisateur = u.util_id_utilisateur
ORDER BY
    h.hor_heure ASC;

```

nom_medecin	prenom_medecin	hor_id_horaire ▲ 1	hor_heure
Martin	Alice	1	08:00:00
Martin	Alice	2	08:30:00
Martin	Alice	3	09:00:00
Martin	Alice	4	09:30:00
Martin	Alice	5	10:00:00
Martin	Alice	6	10:30:00
Martin	Alice	7	11:00:00
Martin	Alice	8	11:30:00
Martin	Alice	9	12:00:00
Martin	Alice	10	13:30:00
Martin	Alice	11	14:00:00
Martin	Alice	12	14:30:00
Martin	Alice	13	15:00:00
Martin	Alice	14	15:30:00
Martin	Alice	15	16:00:00
Martin	Alice	16	16:30:00
Martin	Alice	17	17:00:00
Martin	Alice	18	17:30:00
Martin	Alice	19	18:00:00
Bertrand	Thomas	20	08:00:00
Bertrand	Thomas	21	08:30:00
Bertrand	Thomas	22	09:00:00
Bertrand	Thomas	23	09:30:00
Bertrand	Thomas	24	10:00:00
Bertrand	Thomas	25	10:30:00

Une fois la base de données testée, je peux passer au développement de l'application web.

5) Développement

Pour rappel, l'objectif de cette plateforme de gestion de rendez-vous médicaux est d'optimiser la prise et la gestion des rendez-vous entre les patients et les médecins grâce à une interface intuitive et fonctionnelle qui va permettre aux patients de prendre facilement un rendez-vous et aux médecins de gérer et visualiser tous leurs rendez-vous.

J'utilise plusieurs outils définis dans le cahier des charges. D'abord, la plateforme est développée en HTML, CSS et PHP. Ensuite, la base de données choisie est MySQL gérée via phpMyAdmin. Cette base de données relationnelle est robuste et évolutive. Elle est idéale pour gérer les données tout en assurant la sécurité de celles-ci. De plus, phpMyAdmin facilite l'administration de ma base de données, l'interface est intuitive et efficace ce qui ne demande pas de compétences avancées.
Pour gérer les versions du code, j'utilise l'outil Git qui permet de suivre l'évolution du code et permet donc de développer de façon organisée et de revenir sur une version du code en particulier si besoin.
Ces choix technologiques garantissent la stabilité, la sécurité et l'évolutivité de la plateforme tout en facilitant le développement et sa maintenance à long terme.

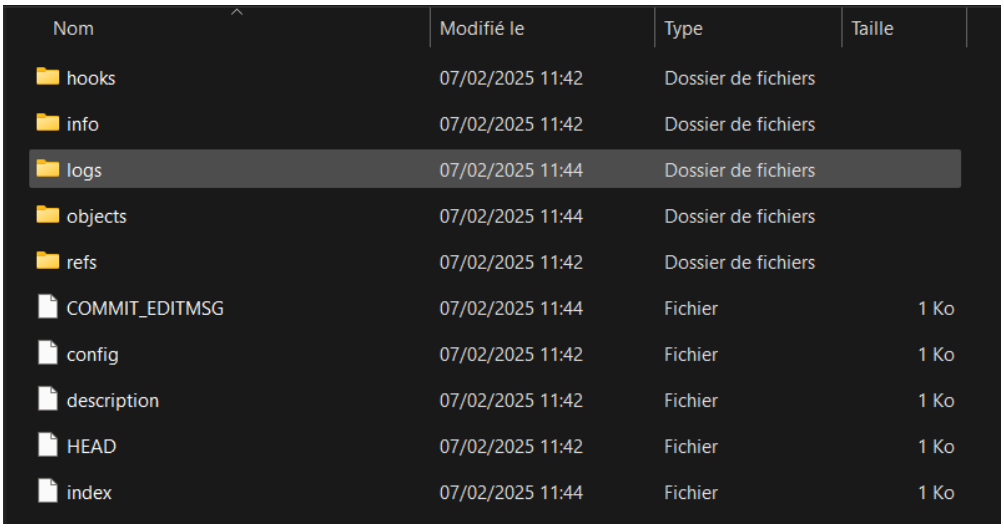
5.1) Présentation de Git

Comme défini dans le cahier des charges, un logiciel de gestion de versions doit être utilisé afin de garder une trace des modifications du code. Pour ce faire, je vais utiliser Git. Je vais pouvoir avoir un historique complet de mon projet, je pourrais revenir à la version précédente en cas d'erreurs et cela va me permettre de mieux m'organiser et également de retrouver l'ensemble de mes fichiers dans un répertoire GitHub.

Je vais ouvrir une fenêtre de terminal dans le dossier de mon projet, puis initialiser Git et y ajouter les fichiers. Je fais ensuite un commit dans le dépôt Git pour commenter les modifications.

```
C:\Users\giron>cd C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux

C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux>git init
Initialized empty Git repository in C:/Users/giron/OneDrive - CHARLES DE FOUCAULD/Bureau/BTS SIO 2/Bloc 2 SLAM/Projet BTS SLAM Gestion de rendez vous médica
ux/.git/

C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux>git add .

C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux>git commit -m "Ajout des fichiers
 au projet de gestion de rendez vous médicaux"
[master (root-commit) c797c26] Ajout des fichiers au projet de gestion de rendez vous médicaux
 8 files changed, 425 insertions(+)
 create mode 100644 connexion-bdd.php
 create mode 100644 connexion.html
 create mode 100644 index.html
 create mode 100644 prendre-rdv.html
 create mode 100644 styles.css
 create mode 100644 tableau-de-bord.html
 create mode 100644 voir-rdv-medecin.html
 create mode 100644 voir-rdv.html

C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux>
```

Les fichiers Git sont stockés dans un fichier caché appelé .git et contient tout l'historique du projet, les commits et la configuration.

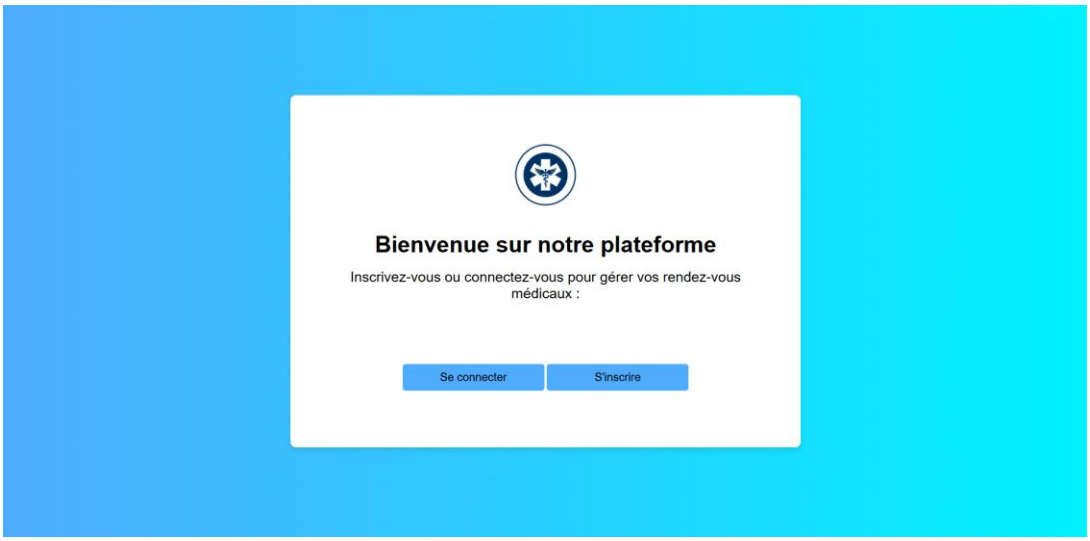


Nom	Modifié le	Type	Taille
hooks	07/02/2025 11:42	Dossier de fichiers	
info	07/02/2025 11:42	Dossier de fichiers	
logs	07/02/2025 11:44	Dossier de fichiers	
objects	07/02/2025 11:44	Dossier de fichiers	
refs	07/02/2025 11:42	Dossier de fichiers	
COMMIT_EDITMSG	07/02/2025 11:44	Fichier	1 Ko
config	07/02/2025 11:42	Fichier	1 Ko
description	07/02/2025 11:42	Fichier	1 Ko
HEAD	07/02/2025 11:42	Fichier	1 Ko
index	07/02/2025 11:44	Fichier	1 Ko

Ensuite, je décide d'envoyer ce projet sur GitHub afin de sauvegarder une fois de plus mon projet et de pouvoir m'organiser.
Je crée un nouveau dépôt GitHub qui s'intitule "gestion-rdv-medical" et j'utilise la commande "git push origin main" pour ajouter mes fichiers dans le dépôt.

5.2) La page d'accueil (index)

La page d'accueil permet aux patients et médecins de se connecter et aux patients de s'inscrire sur la plateforme. J'ajoute un logo médical, un bouton de connexion et d'inscription. Il suffit de moins de trois clics pour trouver l'information recherchée.
Voici un aperçu de la page.



Lorsque je termine ma page index.html ainsi que le CSS de cette page. Je me place dans le dossier de mon projet à l'aide du terminal et j'ajoute les deux fichiers modifiés à Git. Je fais également deux commits qui expliquent mes modifications et je pousse ces modifications sur GitHub.

```
PS C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux> git add index.html
PS C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux> git commit -m "Page index finie"
[main 291cb3c] Page index finie
 1 file changed, 11 insertions(+), 5 deletions(-)
PS C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux> git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 513 bytes | 513.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/LeaGiron/gestion-rdv-medical
   c797c26..291cb3c  main -> main
PS C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux> git add styles.css
PS C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux> git commit -m "CSS pour la page index OK"
[main 48a409e] CSS pour la page index OK
 1 file changed, 17 insertions(+), 192 deletions(-)
PS C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux> git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 548 bytes | 548.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/LeaGiron/gestion-rdv-medical
   291cb3c..48a409e  main -> main
PS C:\Users\giron\OneDrive - CHARLES DE FOUCAULD\Bureau\BTS SIO 2\Bloc 2 SLAM\Projet BTS SLAM Gestion de rendez vous médicaux>
```

5.3) La page d'inscription

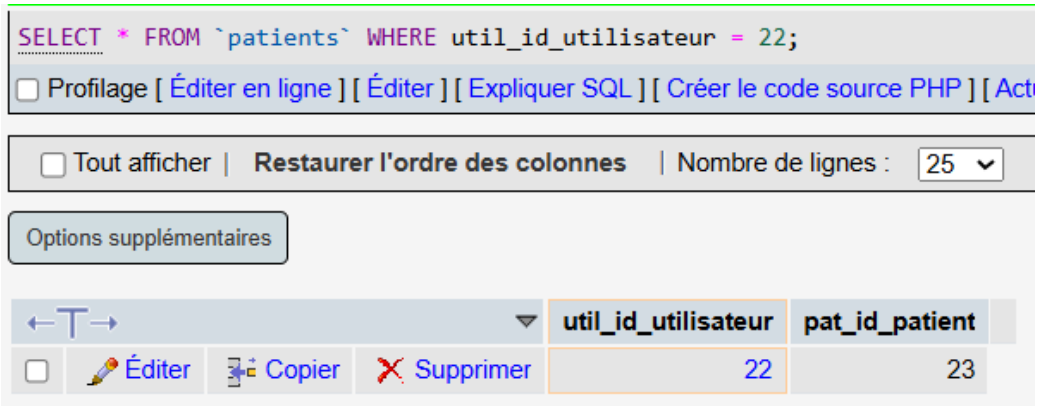
Pour accéder aux rendez-vous, les utilisateurs patients doivent s'inscrire s'ils n'ont pas de compte. Pour cela, ils remplissent un formulaire avec le prénom, le nom, l'email, le numéro de téléphone, la date de naissance et le mot de passe. J'ajoute un bouton qui permet de rediriger vers la page de connexion si besoin.

Je crée un fichier “config.php” qui contient la configuration de la base de données avec des informations comme le nom d’utilisateur et le mot de passe. Puis, je crée un fichier “connexion-bdd.php” qui contient la connexion à la base. J’utilise PDO (PHP Data Objects) pour la connexion à la base de données. L’avantage d’utiliser PDO est la sécurité car elle permet de protéger contre les attaques par injection SQL. Aussi, la gestion des erreurs est plus robuste car elle permet de rendre plus facile le débogage. Ce choix est approprié car le cahier des charges défini que la plateforme doit être sécurisée car elle contient des données personnelles et le projet doit être facilement modifiable.

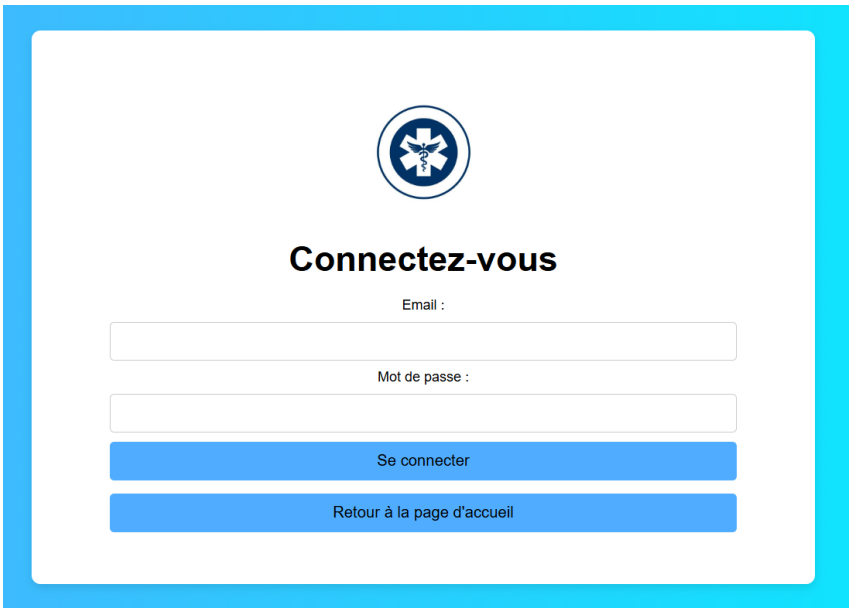
J’ajoute ces fichiers à Git et GitHub avec les mêmes commandes vu précédemment.

Je vais créer un compte pour tester la fonctionnalité. Je vérifie que le compte s’enregistre dans la table “utilisateurs” et que l’identifiant du patient s’enregistre automatiquement dans la table “patients”.

Pour vérifier que les informations du compte sont enregistrées dans la base, j’exécute une requête SQL dans la table “utilisateurs” et dans la table “patients”. Cela me permet de sélectionner, d’une part, l’identifiant “util_id_utilisateur” qui correspond à 22 dans la table “utilisateurs” et d’autre part, l’identifiant “util_id_utilisateur” qui est égal à 22 dans la table “patients”.



5.4) La page de connexion



Pour accéder aux rendez-vous, les utilisateurs doivent se connecter.
D'abord, je crée une page de connexion simple en PHP avec PDO (PHP Data Objects).

Je crée un utilisateur sécurisé avec des droits limités. Pour cela, je me rend sur phpMyAdmin et j'exécute ces requêtes. Je crée l'utilisateur 'administrateur' avec un mot de passe. La seconde requête permet de donner tous les droits en lecture, écriture, modification et suppression à toutes les tables de la base de données "gestion_de_rdv_medicaux". Je recharge la table des privilèges avec "FLUSH PRIVILEGES" pour prendre en compte les modifications.

Je vérifie ensuite dans l'onglet "Comptes utilisateurs" si mon utilisateur administrateur a tous les privilèges.

Dans ma base de données, j'ai précédemment inséré des données de test pour quelques utilisateurs. Cependant, les mots de passe sont stockés en clair dans la table "utilisateurs".

Donc, cela indique une erreur quand j'essaie de me connecter avec un mot de passe en clair. Je vais donc mettre à jour les données de test en exécutant un script PHP qui va hacher ces mots de passe de tests.

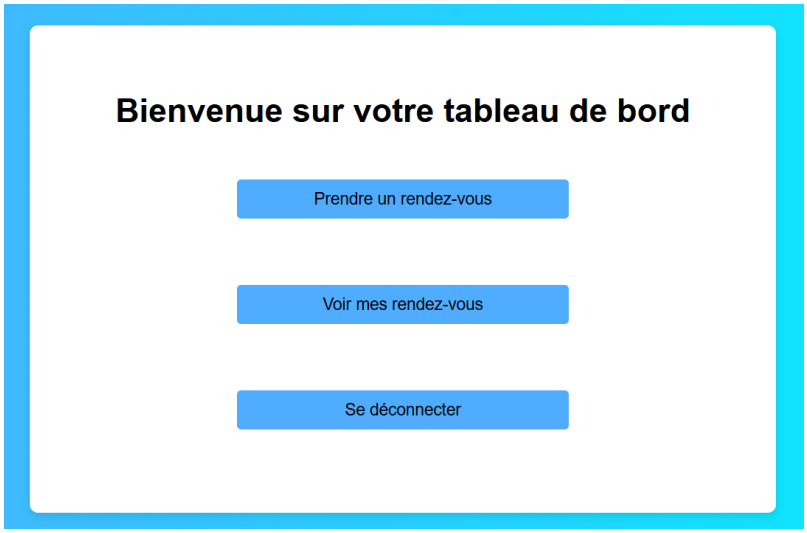
Je crée un script "maj_mot_de_passe" et j'y accède via cet url : http://localhost/gestion_de_rdv_medicaux/maj_mot_de_passe.php.

Je reçois une confirmation de mise à jour puis je vérifie en accédant à la table "utilisateurs".

	util_id_utilisateur	util_nom	util_prenom	util_email	util_mot_de_passe
Éditer Copier Supprimer	1	Dupont	Jean	jean.dupont@email.com	$2y$10$o9ZY/V2TDgmFZeN7e9O/ju2tzflyi9zVtd.DmqKrDWj...
Éditer Copier Supprimer	2	Martin	Alice	alice.martin@email.com	$2y$10$sCuIRv4Ky9J26.28U02jj.8eWeRx1RNVpDSEF1S8fmq...
Éditer Copier Supprimer	3	Lemoine	Sophie	sophie.lemoine@email.com	$2y$10$AyMpPEf4ZFEDJ6/qBj86o.Jiy3quOsxGRrdHolBvpYe...
Éditer Copier Supprimer	4	Bertrand	Thomas	thomas.bertrand@email.com	$2y$10$3yRuso7MwZKyCQV5n/VBfuHRg0J7s5HplZFsnd7/r7N...
Éditer Copier Supprimer	5	Garnier	Camille	camille.garnier@email.com	$2y$10$uTpFEV.8Vl2yLHoZax7vseG/sA92SnJ5vV/iPNkgRQ0...
Éditer Copier Supprimer	6	Morel	Nicolas	nicolas.morel@email.com	$2y$10$eF5cRJvf.4SYB9t6iMO.8ezw/0EP2InhAQwKflieq06...
Éditer Copier Supprimer	7	Lefevre	Chloé	chloe.lefevre@email.com	$2y$10$EbnNmEfrCheXuLkMHKLgburyBnSVbVElVf6YjXGwMpy...
Éditer Copier Supprimer	8	Durand	Pierre	pierre.durand@email.com	$2y$10$s.Kd8aFZ3Z/YecoKDRQIV.cC50NEEDsVbqQlVraJEwS...
Éditer Copier Supprimer	11	Test	Lea	lea.test@test.fr	$2y$10$F8rUK6dYK3z2Gob.ItmQSOcA3F0LJxFn5SzrTlhNZU0...

Je supprime ensuite ce fichier pour qu'il ne soit pas accessible à des personnes malveillantes. Je peux maintenant me connecter avec l'email et le mot de passe d'un utilisateur de test sans avoir d'erreurs, ce qui me redirige vers le tableau de bord de l'utilisateur.

5.5) Le tableau de bord du patient et du médecin

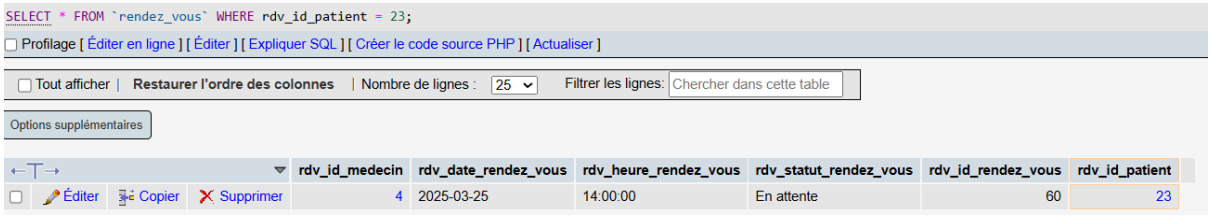


Ensuite, le tableau de bord permet d'afficher plusieurs options pour un patient ou un médecin. Pour le patient, il peut choisir entre prendre un rendez-vous, voir ses rendez-vous ou se déconnecter. Pour un médecin, il peut seulement voir tous les rendez-vous ou se déconnecter. Pour cela, je récupère le rôle de l'utilisateur depuis la table "utilisateurs" et je le stocke dans la session avec $_SESSION['role'] = $user['util_role'];
Les sections de la page s'affichent en fonction du rôle de l'utilisateur.

5.6) La page pour prendre un rendez-vous

Cette page permet aux patients, une fois connectés, de prendre un rendez-vous. Il faut choisir un médecin dans la liste déroulante et une date. Le patient clique sur "vérifier les disponibilités" pour afficher les horaires disponibles. Il peut ensuite prendre son rendez-vous sinon un message d'erreur s'affiche pour informer que le rendez-vous n'est pas disponible.

Je vérifie que le rendez-vous s'enregistre bien dans la base de données. Pour cela, je remplis le formulaire pour prendre mon rendez-vous et j'exécute une requête pour la vérification. Je choisis le patient "leane.test@gmail.com" qui possède l'identifiant utilisateur 22 et l'identifiant patient 23.



Le rendez-vous de mon patient s'enregistre correctement dans la table "rendez-vous" de la base de données.

5.7) La page pour visualiser tous les rendez-vous d'un patient

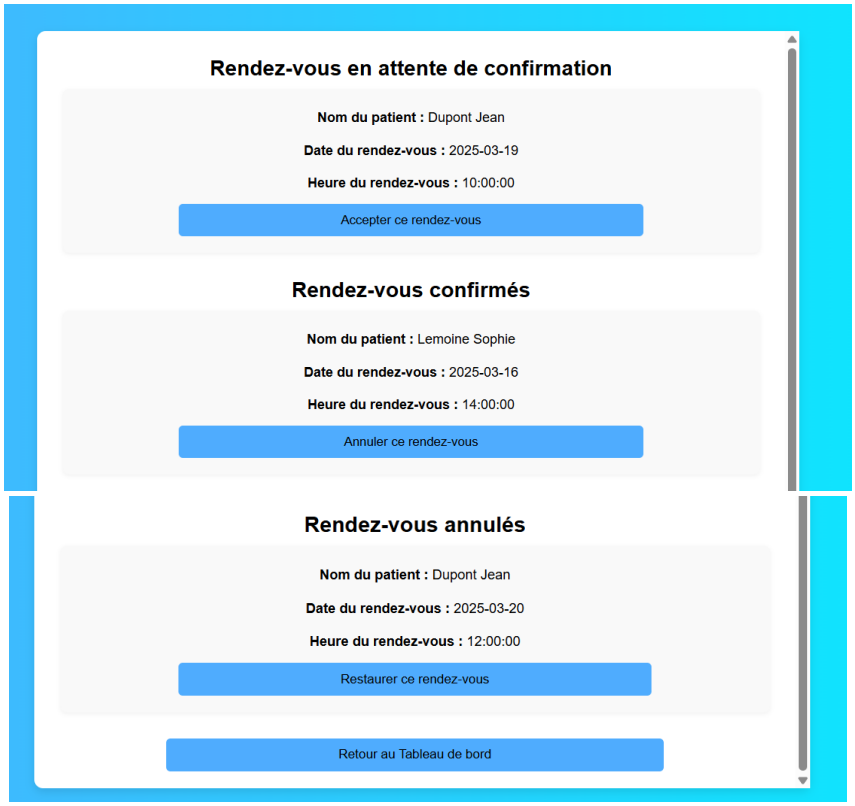
Cette page permet aux patients de visualiser tous les rendez-vous pris. Il peut récupérer le nom et prénom du médecin, la date et l'heure du rendez-vous ainsi que son statut ("En attente", "Confirmé", Annulé").

5.8) La page de déconnexion

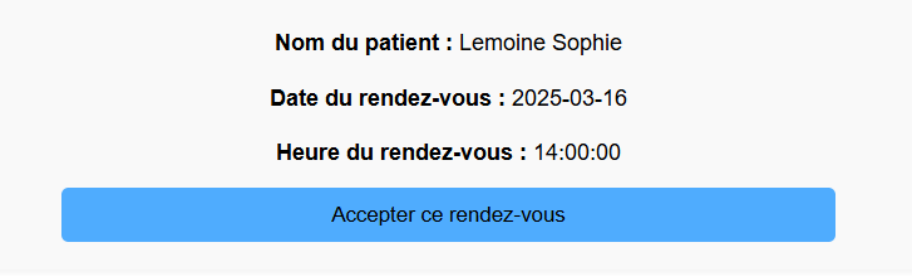
Ce fichier est un code PHP qui permet simplement de déconnecter l'utilisateur dès qu'il clique sur un bouton de déconnexion.

5.9) La page pour voir tous les rendez-vous d'un médecin

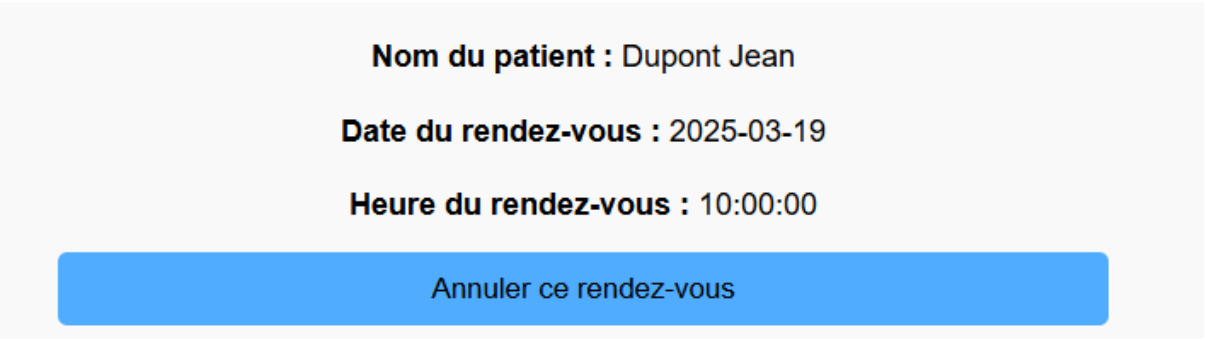
Les médecins peuvent visualiser l'ensemble de leurs rendez-vous en attente de confirmation, confirmés et annulés. Chaque case permet de trouver le nom du patient, la date et l'heure du rendez-vous. Il y a aussi un bouton qui permet de revenir au tableau de bord.
Le médecin accepte les rendez-vous, pour ce faire, il doit cliquer sur le bouton "accepter ce rendez-vous" et celui-ci se retrouve dans la partie des rendez-vous confirmés. Pour l'annuler, il clique sur le bouton "annuler ce rendez-vous" et celui-ci retourne dans la partie des rendez-vous annulés. Enfin, il peut cliquer sur le bouton "restaurer ce rendez-vous" et celui-ci se retrouve dans la partie des rendez-vous confirmés.



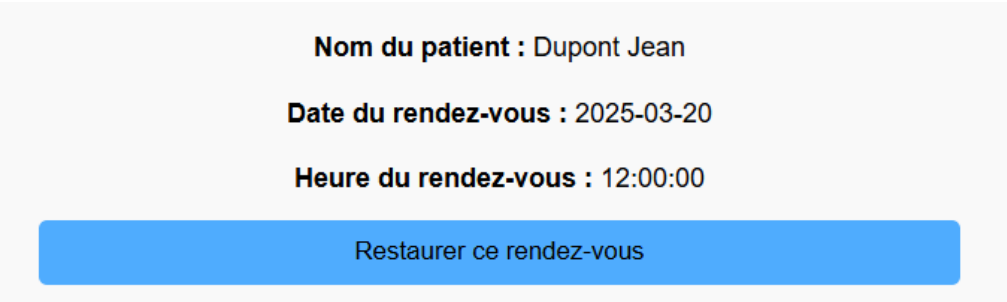
Cette page en PHP sert à accepter un rendez-vous.



Cette page en PHP sert à annuler un rendez-vous.



Cette page en PHP sert à restaurer un rendez-vous.



6) Tests

Dans cette partie, je vais tester manuellement mon application web ainsi que toutes les pages et fonctionnalités. Le but est de valider le bon fonctionnement entre l'interface et la base de données.
Mes tests seront organisés de la façon suivante : l'identifiant du test et une suite claire et explicite [ID] : [Page] - [Action] - [Résultat attendu] - [Résultat obtenu] - [Statut], des tests par catégories et un tableau de suivi des tests.

6.1) Tests d'interface

L'objectif des tests d'interface (UI) est de vérifier l'affichage des pages et l'ergonomie de l'application.

6.2) Tests fonctionnels

Tests d'interface

Test fonctionnels

ID	Page	Action	Résultat attendu	Résultat obtenu	Statut
FUNC-01	Connexion	Se connecter avec des identifiants valides	Connexion réussie	OK	Réussi
FUNC-02	Tableau de bord (médecin)	Cliquer sur "accepter ce rendez-vous"	Le rendez-vous est déplacé dans la "Rendez-vous confirmés"	OK	Réussi
FUNC-03	Tableau de bord (médecin)	Cliquer sur "annuler ce rendez-vous"	Le rendez-vous est déplacé dans la "Rendez-vous annulés"	OK	Réussi
FUNC-04	Tableau de bord (médecin)	Cliquer sur "restaurer ce rendez-vous"	Le rendez-vous est déplacé dans la "Rendez-vous confirmés"	OK	Réussi
FUNC-05	Inscription	Remplir le formulaire correctement et valider	Utilisateur enregistré	OK	Réussi
UI-09	Connexion	Cliquer sur le bouton "se connecter"	Redirection vers le tableau de bord	OK	Réussi
UI-10	Connexion	Cliquer sur le bouton "retour à la page d'accueil"	Redirection vers la page d'accueil	OK	Réussi
UI-11	Connexion	Passer la souris sur le bouton "se connecter"	Le bouton change de couleur	OK	Réussi
UI-12	Connexion	Passer la souris sur le bouton "retour à la page d'accueil"	Le bouton change de couleur	OK	Réussi
UI-13	Tableau de bord (médecin et patient)	Vérifier l'alignement des boutons	Boutons bien alignés	OK	Réussi
UI-14	Tableau de bord (médecin et patient)	Vérifier la lisibilité du texte	Texte lisible	OK	Réussi
UI-15	Tableau de bord (patient)	Cliquer sur le bouton "prendre un rendez-vous"	Redirection vers le formulaire de prise de rendez-vous	OK	Réussi
UI-16	Tableau de bord (patient)	Cliquer sur le bouton "voir mes rendez-vous"	Redirection vers la page de visualisation des rendez-vous	OK	Réussi
UI-17	Tableau de bord (médecin)	Cliquer sur le bouton "voir tous les rendez-vous"	Redirection vers la page de visualisation de tous les rendez-vous	OK	Réussi
UI-18	Tableau de bord (médecin et patient)	Cliquer sur le bouton "déconnexion"	Redirection vers la page d'accueil	OK	Réussi
UI-19	Tableau de bord (patient)	Passer la souris sur le bouton "prendre un rendez-vous"	Le bouton change de couleur	OK	Réussi
UI-20	Tableau de bord (médecin et patient)	Passer la souris sur le bouton "voir mes rendez-vous"	Le bouton change de couleur	OK	Réussi
UI-21	Tableau de bord (médecin)	Passer la souris sur le bouton "voir tous les rendez-vous"	Le bouton change de couleur	OK	Réussi
UI-22	Tableau de bord (médecin et patient)	Passer la souris sur le bouton "deconnexion"	Le bouton change de couleur	OK	Réussi
UI-23	Inscription	Vérifier l'alignement des boutons	Boutons bien alignés	OK	Réussi
UI-24	Inscription	Vérifier la lisibilité du texte	Texte lisible	OK	Réussi
UI-34	Prendre un rdv	Cliquer sur le bouton "retour au tableau de bord"	Redirection vers le tableau de bord	OK	Réussi
UI-35	Prendre un rdv	Cliquer sur le bouton "vérifier les disponibilités"	Affiche le champ pour choisir l'heure	OK	Réussi
UI-36	Prendre un rdv	Passer la souris sur le bouton "vérifier les disponibilités"	Le bouton change de couleur	OK	Réussi
UI-37	Prendre un rdv	Passer la souris sur le bouton "confirmer le rendez-vous"	Le bouton change de couleur	OK	Réussi
UI-38	Prendre un rdv	Passer la souris sur le bouton "retour au tableau de bord"	Le bouton change de couleur	OK	Réussi
UI-39	Mes rendez-vous (patient)	Vérifier l'alignement des boutons	Boutons bien alignés	OK	Réussi
UI-40	Mes rendez-vous (patient)	Vérifier la lisibilité du texte	Texte lisible	OK	Réussi
UI-41	Mes rendez-vous (patient)	Cliquer sur le bouton "retour au tableau de bord"	Redirection vers le tableau de bord	OK	Réussi
UI-42	Mes rendez-vous (patient)	Passer la souris sur le bouton "retour au tableau de bord"	Le bouton change de couleur	OK	Réussi
UI-43	Mes rendez-vous (médecin)	Vérifier l'alignement des boutons	Boutons bien alignés	OK	Réussi
UI-44	Mes rendez-vous (médecin)	Vérifier la lisibilité du texte	Texte lisible	OK	Réussi
UI-45	Mes rendez-vous (médecin)	Cliquer sur le bouton "accepter ce rendez-vous"	Le rendez-vous est redirigé vers la section "rendez-vous confirmés"	OK	Réussi
UI-46	Mes rendez-vous (médecin)	Cliquer sur le bouton "annuler ce rendez-vous"	Le rendez-vous est redirigé vers la section "rendez-vous annulés"	OK	Réussi
UI-47	Mes rendez-vous (médecin)	Cliquer sur le bouton "restaurer ce rendez-vous"	Le rendez-vous est redirigé vers la section "rendez-vous confirmés"	OK	Réussi

Les tests fonctionnels (FUNC) consistent à vérifier que chaque fonctionnalité remplit son rôle comme la connexion ou l'action attendue lorsque l'on clique sur un bouton.

6.3) Tests de base de données

Les tests de base de données (DB) consistent à vérifier que les données sont bien enregistrées, modifiées ou supprimées après une action sur le site.

Tests de base de données

ID	Page	Table	Action	Résultat attendu	Résultat	Statut
DB-01	Inscription	utilisateurs et patients	Remplir le formulaire d'inscription	Nouvel utilisateur	OK	Réussi
DB-02	Tableau de bord (médecin)	rendez-vous	Accepter un rendez-vous	Le statut du rendez-vous doit être "confirmé"	OK	Réussi
DB-03	Tableau de bord (médecin)	rendez-vous	Annuler un rendez-vous	Le statut du rendez-vous doit être "annulé"	OK	Réussi
DB-04	Tableau de bord (médecin)	rendez-vous	Restaurer un rendez-vous	Le statut du rendez-vous doit être "confirmé"	OK	Réussi
DB-05	Prendre un rendez-vous (patient)	rendez-vous	Remplir le formulaire pour prendre un rendez-vous	Rendez-vous enregistré	OK	Réussi

6.4) Tests d'erreurs

L'objectif des tests d'erreurs (ERR) est de vérifier la robustesse du système face aux erreurs et attaques potentielles.

Tests d'erreurs

ID	Page	Action	Résultat attendu	Résultat obtenu	Statut	Commentaire
ERR-01	Connexion	Se connecter avec un compte inexistant	Message d'erreur "Aucun utilisateur trouvé avec cet email."	KO	échoué	Pas de message d'erreur
ERR-02	Connexion	Laisser le champ "email" vide et valider	Message d'erreur "Veuillez renseigner ce champ."	OK	Réussi	
ERR-03	Connexion	Laisser le champ "mot de passe" vide et valider	Message d'erreur "Veuillez renseigner ce champ."	OK	Réussi	
ERR-04	Connexion	Saisir un mot de passe incorrect pour un email existant	Message d'erreur "Email ou mot de passe incorrect"	OK	Réussi	
ERR-05	Connexion	Saisir un email incorrect pour un mot de passe existant	Message d'erreur "Aucun utilisateur trouvé avec cet email."	OK	Réussi	
ERR-06	Connexion	Entrer " ' OR '1'='1 " dans le champ "email"	Veuillez inclure "@" dans l'adresse e-mail. Il manque un symbole "@" dans ' OR '1'='1	OK	Réussi	
ERR-07	Inscription	Laisser le champ "email" vide et valider	Message d'erreur "Veuillez renseigner ce champ."	OK	Réussi	
ERR-08	Inscription	Saisir une adresse email invalide (test@com)	Message d'erreur "L'email n'est pas valide."	OK	Réussi	
ERR-09	Inscription	Saisir un email déjà utilisé	Message d'erreur "Cet utilisateur est déjà inscrit en tant que patient."	OK	Réussi	
ERR-10	Inscription	Saisir un numéro de téléphone invalide (0123 et 0123abc)	Message d'erreur "Le numéro de téléphone doit contenir exactement 10 chiffres."	KO	échoué	Le numéro de téléphone "0123" et "0123abc" est validé et le compte est créé.
ERR-11	Inscription	Ne pas entrer de date de naissance et valider	Message d'erreur "Veuillez renseigner ce champ."	OK	Réussi	
ERR-12	Inscription	Entrer " ' OR '1'='1 " dans le champ "nom"	Message d'erreur "Le nom ne peut contenir que des lettres et des espaces".	KO	échoué	Le nom " OR '1'='1 est validé et le compte est créé.
ERR-13	Inscription	Entrer " ' OR '1'='1 " dans le champ "prénom"	Message d'erreur "Le prénom ne peut contenir que des lettres et des espaces".	OK	Réussi	
ERR-14	Inscription	Entrer '" OR '1'='1" dans le champ "email"	Message d'erreur "Veuillez inclure "@" dans l'adresse e-mail.	OK	Réussi	

6.4.1) [ERR-01] [Connexion] [Se connecter avec un compte inexistant] [Message d'erreur "Aucun utilisateur trouvé avec cet email"]

Ce test consiste à entrer un email inconnu. L'email testé est "leag@exemple.com" qui n'affiche pas l'erreur attendue sur la page de connexion.
Il a fallu modifier le code PHP de la page "connexion.php".

```
if (!$user) {
                $error_message = "Aucun utilisateur trouvé avec cet email.";
            } elseif (!password_verify($password, $user['util_mot_de_passe'])) {
                $error_message = "Email ou mot de passe incorrect.";
            } else {
                // Stocker les informations de session
                $_SESSION['util_id_utilisateur'] = $user['util_id_utilisateur'];
                $_SESSION['role'] = $user['util_role'];
```

```
// Stocker et recharger la page pour afficher l'erreur
    if (!empty($error_message)) {
        $_SESSION['error_message'] = $error_message;
        header("Location: connexion.php");
        exit;
    }
```

```
        <!-- Affichage du message d'erreur d'email introuvable -->
        <?php if (!empty($error_message)) : ?>
            <p><?php echo htmlspecialchars($error_message); ?></p>
        <?php endif; ?>
```

Après modification, un test d'erreur a de nouveau été effectué. Maintenant l'email testé affiche l'erreur "Aucun utilisateur trouvé avec cet email.".
Le test d'erreur 1 de la page de connexion pour entrer un email inconnu est résolu. Le test est maintenant concluant.

6.4.2) [ERR-10] [Inscription] [Saisir un numéro de téléphone invalide (0123 et 0123abc)] [Message d'erreur "Le numéro de téléphone doit contenir exactement 10 chiffres."]

Ce test consiste à entrer un numéro de téléphone incorrect. Les numéros testés sont “0123” et “0123abc” qui n’affichent pas les erreurs attendues sur la page d’inscription. Il valide et enregistre le numéro de téléphone dans la base de données.
Il a fallu modifier le code PHP de la page “inscription.php”.

```
    // Validation du numéro de téléphone : doit contenir exactement 10 chiffres
    if (!preg_match('/^\d{10}$/', $telephone)) {
        $errors['telephone'] = "Le numéro de téléphone doit contenir exactement 10 chiffres.";
    }
```

```
            <?php if (isset($errors['telephone'])): ?>
                <p><?php  echo  htmlspecialchars($errors['telephone']); ?></p>
            <?php endif; ?>
```

Cette expression régulière !preg_match('/^\d{10}$/', $part_telephone) valide un numéro de téléphone composé de 10 chiffres.
Le symbole ^ indique le début de la chaîne ce qui empêche d’avoir des caractères avant. Ensuite, le numéro de téléphone doit être composé de chiffres \d.
Il doit aussi y avoir exactement 10 chiffres {10}. Le symbole $ indique la fin de la chaîne et donc empêche d’avoir des caractères après.

Après modification, un test d’erreur a de nouveau été effectué. Maintenant les numéros de téléphone testés affichent l’erreur "Le numéro de téléphone doit contenir exactement 10 chiffres”.
Le test d’erreur 10 de la page d’inscription pour entrer un numéro de téléphone invalide est résolu. Le test est maintenant concluant.

6.4.3) [ERR-12] [Inscription] [Entrer " ' OR '1'='1 " dans le champ "nom"] [Message d'erreur "Le nom ne peut contenir que des lettres et des espaces".]

Ce test consiste à entrer un nom incorrect. Le nom testé est ‘ OR ‘1’=’1 qui n’affiche pas l’erreur attendue sur la page d’inscription. Il valide et enregistre le nom dans la base de données.
Il a fallu modifier le code PHP de la page “inscription.php”.

```
    // Vérifier que le nom ne contient que des lettres et des espaces
```

```
    if (!preg_match("/^[a-zA-Z\s]+$/", $nom)) {
        $errors['nom'] = "Le nom ne peut contenir que des lettres et des
espaces.";
    }

    // Vérifier que le nom ne contient que des lettres et des espaces
    if (!preg_match("/^[a-zA-Z\s]+$/", $prenom)) {
        $errors['prenom'] = "Le prénom ne peut contenir que des lettres
et des espaces.";
    }
```

```
            <?php if (isset($errors['prenom'])): ?>
                <p><?php    echo    htmlspecialchars($errors['prenom']);
?></p>
            <?php endif; ?>
```

```
            <?php if (isset($errors['nom'])): ?>
                <p><?php echo htmlspecialchars($errors['nom']); ?></p>
            <?php endif; ?>
```

Cette expression régulière !preg_match('/^[a-zA-Z\s-]+$/", $nom) ou !preg_match('/^[a-zA-Z\s-]+$/', $prenom) est utilisée pour valider un nom ou un prénom.
Le symbole ^ indique le début de la chaîne ce qui empêche d'avoir des caractères avant. Ensuite, l'expression accepte les lettres minuscules et majuscules de l'alphabet a-zA-Z, et autorise les espaces \s.
Le symbole + indique qu'il doit y avoir au moins un caractère parmi ceux autorisés et le symbole $ indique la fin de la chaîne ce qui évite d'ajouter des caractères interdits après.

Après modification, un test d'erreur a de nouveau été effectué. Maintenant le nom testé affiche l'erreur "Le nom ne peut contenir que des lettres et des espaces.".
Le test d'erreur 12 de la page d'inscription pour entrer un nom incorrect est résolu. Le test est maintenant concluant.

6.5) Tests de compatibilité des navigateurs

Ces tests (NAV) consistent à vérifier que le site fonctionne et s'affiche correctement sur différents navigateurs. Ici, les navigateurs testés sont Chrome, Firefox et Edge.

Tests de compatibilité

ID	Page	Navigateur	Résultat attendu	Résultat obtenu	Statut
NAV-01	Accueil	Chrome	La page d'accueil s'affiche et fonctionne correctement	OK	Réussi
NAV-02	Accueil	Firefox	La page d'accueil s'affiche et fonctionne correctement	OK	Réussi
NAV-03	Accueil	Edge	La page d'accueil s'affiche et fonctionne correctement	OK	Réussi
NAV-04	Connexion	Chrome	La page de connexion s'affiche et fonctionne correctement	OK	Réussi
NAV-05	Connexion	Firefox	La page de connexion s'affiche et fonctionne correctement	OK	Réussi
NAV-06	Connexion	Edge	La page de connexion s'affiche et fonctionne correctement	OK	Réussi
NAV-07	Inscription	Chrome	Le formulaire d'inscription s'affiche et fonctionne correctement	OK	Réussi
NAV-08	Inscription	Firefox	Le formulaire d'inscription s'affiche et fonctionne correctement	OK	Réussi
NAV-09	Inscription	Edge	Le formulaire d'inscription s'affiche et fonctionne correctement	OK	Réussi
NAV-10	Tableau de bord	Chrome	Le tableau de bord s'affiche et fonctionne correctement	OK	Réussi
NAV-11	Tableau de bord	Firefox	Le tableau de bord s'affiche et fonctionne correctement	OK	Réussi
NAV-12	Tableau de bord	Edge	Le tableau de bord s'affiche et fonctionne correctement	OK	Réussi
NAV-13	Prendre un rdv	Chrome	Le formulaire pour prendre un rendez-vous s'affiche et fonctionne correctement	OK	Réussi
NAV-14	Prendre un rdv	Firefox	Le formulaire pour prendre un rendez-vous s'affiche et fonctionne correctement	OK	Réussi
NAV-15	Prendre un rdv	Edge	Le formulaire pour prendre un rendez-vous s'affiche et fonctionne correctement	OK	Réussi
NAV-16	Mes rendez-vous (patient)	Chrome	La page pour prendre un rendez-vous (patient) s'affiche et fonctionne correctement	OK	Réussi
NAV-17	Mes rendez-vous (patient)	Firefox	La page pour prendre un rendez-vous (patient) s'affiche et fonctionne correctement	OK	Réussi
NAV-18	Mes rendez-vous (patient)	Edge	La page pour prendre un rendez-vous (patient) s'affiche et fonctionne correctement	OK	Réussi
NAV-19	Mes rendez-vous (médecin)	Chrome	La page pour prendre un rendez-vous (médecin) s'affiche et fonctionne correctement	OK	Réussi
NAV-20	Mes rendez-vous (médecin)	Firefox	La page pour prendre un rendez-vous (médecin) s'affiche et fonctionne correctement	OK	Réussi
NAV-21	Mes rendez-vous (médecin)	Edge	La page pour prendre un rendez-vous (médecin) s'affiche et fonctionne correctement	OK	Réussi

7) Documentation technique v1.0

1) Introduction

1.1) Contexte du projet

Ce projet s'inscrit dans le cadre des problématiques rencontrées dans les établissements de santé. En effet, l'établissement doit posséder une gestion optimisée des rendez-vous médicaux afin de garantir un suivi des consultations et une meilleure organisation des soins. Dans de nombreux hôpitaux, la prise de rendez-vous repose souvent sur des outils existants et peuvent présenter des limites en termes d'ergonomie, de performance et de sécurité des données. Cela pénalise le personnel médical et administratif mais également les patients.
De plus, le domaine médical en général et la gestion des rendez-vous médicaux sont soumis à des exigences strictes en matière de protection des données personnelles et de confidentialité. Les informations de santé étant considérées comme sensibles, elles doivent être traitées conformément au Règlement Général sur la Protection des Données (RGPD). Il est donc essentiel de garantir un stockage sécurisé, un accès restreint aux données et de respecter le droit des patients, notamment l'accès, la rectification et la suppression de leurs données.

1.2) Objectifs de la documentation

Cette documentation technique du projet de gestion de rendez-vous médicaux comporte plusieurs objectifs tels que l'architecture, l'installation et la configuration des différents éléments, le fonctionnement, les tests et le déploiement, la maintenance et les évolutions de l'application web.

2) Sécurité et conformité RGPD

Les mesures de sécurité mises en place sont l'utilisation de requêtes préparées pour éviter les injections SQL, l'accès est restreint aux médecins et patients pour la vérification et l'accès aux données, le stockage des informations de test fictives est sécurisé.
En termes de conformité du Règlement Général sur la Protection des Données (RGPD), les données personnelles fictives sont traitées en respectant ces normes. La plateforme permet aux utilisateurs d'exercer leurs droits d'accès, de rectification et de suppression de leurs données.

3) Architecture et conception

3.1) Technologies utilisées

Le système est composé d'une architecture client-serveur avec une partie front-end pour l'interface utilisateur et d'une partie back-end pour le traitement des données.

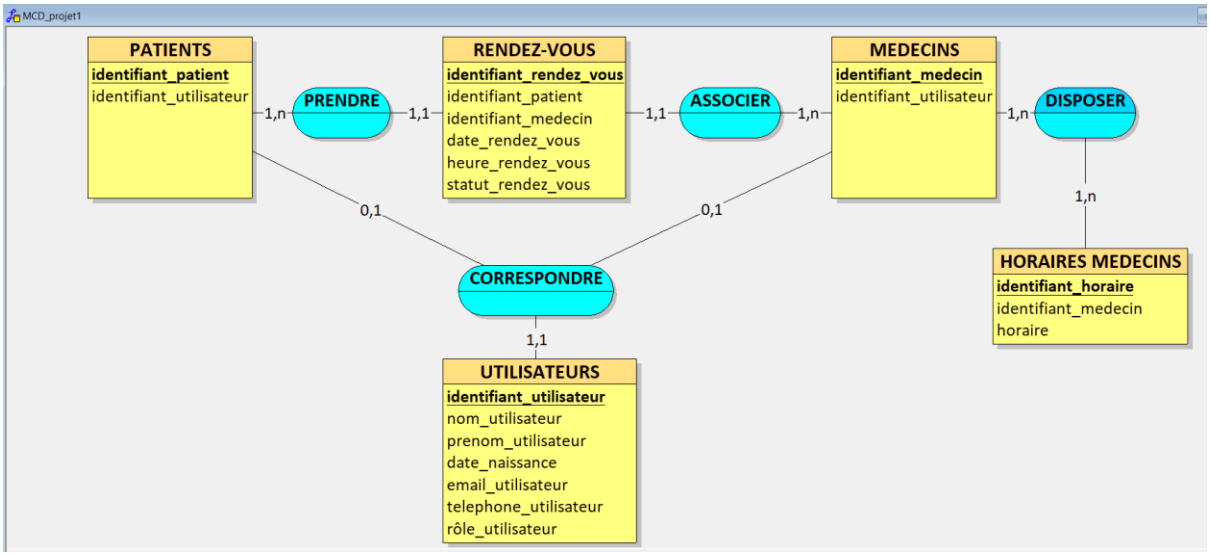
Les technologies utilisées sont, pour la partie front-end, HTML et CSS. Pour la partie back-end, j'utilise PHP. Le développement s'effectue à l'aide de l'environnement de VS Code qui facilite l'écriture et le débogage du code. La base de données choisie est MySQL gérée via phpMyAdmin qui facilite la création, la modification et la suppression des tables et des données. Le serveur web local utilisé est XAMPP. Ce logiciel regroupe Apache, MySQL, PHP et Perl permettant de tester localement le projet et de reproduire les conditions d'un serveur de production.
En termes de sécurité, des requêtes préparées sont utilisées pour éviter les injections SQL. L'accès doit être géré avec un système d'authentification pour les patients et les médecins. Les composants communiquent entre eux, d'une part le front-end qui envoie les requêtes via des formulaires HTML et traités d'autre part par le back-end PHP. Les données sont récupérées et affichées via PHP. Enfin, le projet est conçu pour être facilement modifiable si des évolutions sont nécessaires grâce à l'utilisation du logiciel de gestion de versions Git. La structure du code est organisée afin de permettre une maintenance simplifiée.

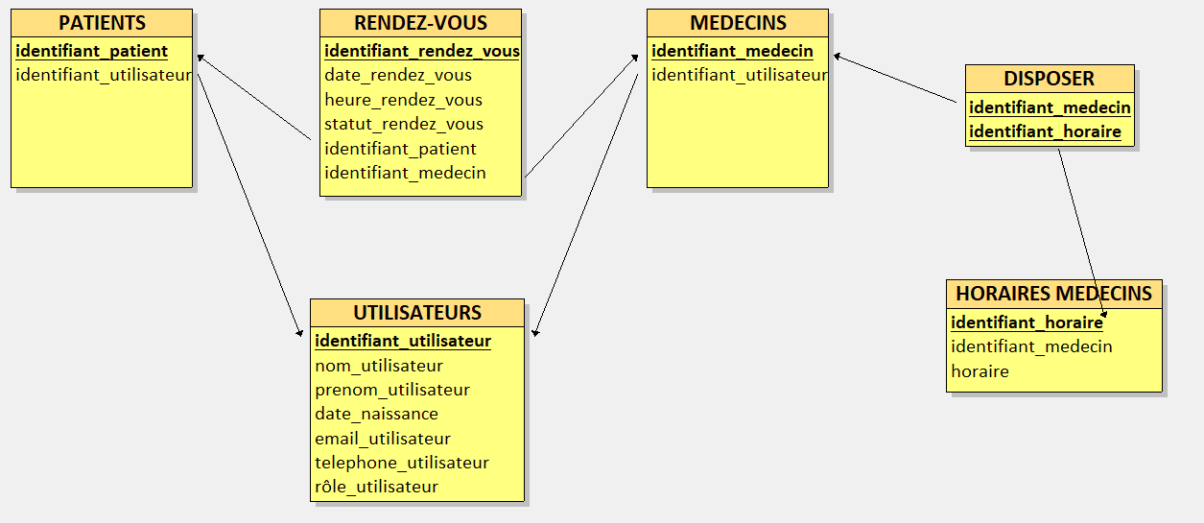
3.2) Modélisation et diagrammes

Pour concevoir et gérer la base de données, il faut créer le dictionnaire de données. Il y a cinq tables à créer dont la table utilisateurs, patients, médecins, horaires médecins et rendez-vous.

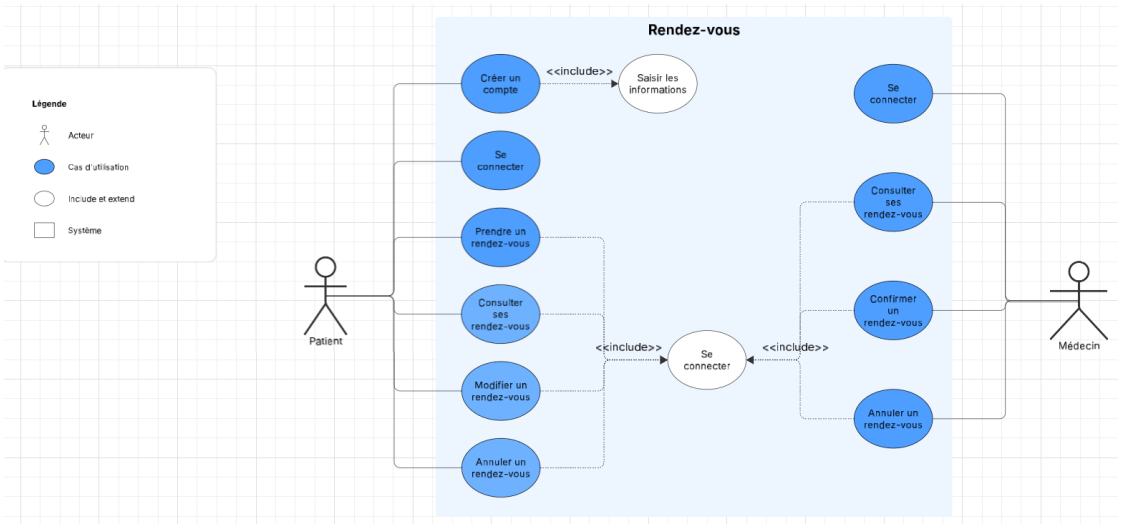
TABLE	Nom	Nom logique	Type	Longueur	Observation
UTILISATEURS	identifiant_utilisateur	util_id_utilisateur	INT (numérique)	11	Clé primaire, identifiant unique pour chaque utilisateur
	nom_utilisateur	util_nom	VARCHAR (caractère)	50	Nom de famille de l'utilisateur
	prenom_utilisateur	util_prenom	VARCHAR (caractère)	50	Prénom de l'utilisateur
	date_naissance	util_date_naissance	DATE		Date de naissance de l'utilisateur
	email_utilisateur	util_email	VARCHAR (caractère)	100	Email de l'utilisateur
	telephone_utilisateur	util_telephone	VARCHAR (numérique)	10	Numéro de téléphone de l'utilisateur
	mot_de_passe_utilisateur	util_mot_de_passe	VARCHAR (numérique)	255	Mot de passe de l'utilisateur
	rôle_utilisateur	util_rôle	VARCHAR (caractère)	20	Rôle de l'utilisateur (patient ou médecin)
PATIENTS	identifiant_patient	pat_id_patient	INT (numérique)	11	Clé primaire, identifiant unique pour chaque patient
	identifiant_utilisateur	util_id_utilisateur	INT (numérique)	11	Clé étrangère, référence à l'identifiant de l'utilisateur dans la table "utilisateurs"
MÉDECINS	identifiant_medecin	med_id_medecin	INT (numérique)	11	Clé primaire, identifiant unique pour chaque médecin
	identifiant_utilisateur	util_id_utilisateur	INT (numérique)	11	Clé étrangère, référence à l'identifiant de l'utilisateur dans la table "utilisateurs"
HORAIRES MEDECINS	identifiant_horaire	hor_id_horaire	INT (numérique)	11	Clé primaire, identifiant unique pour chaque plage horaire
	identifiant_medecin	hor_id_medecin	INT (numérique)	11	Clé étrangère, référence à l'identifiant du médecin
	horaire	hor_horaire	TEMPS		Horaire du rendez-vous
RENDEZ-VOUS	identifiant_rendez_vous	rdv_id_rendez_vous	INT (numérique)	11	Clé primaire, identifiant unique pour chaque rendez-vous
	identifiant_patient	rdv_id_patient	INT (numérique)	11	Clé étrangère, référence à l'identifiant du patient
	identifiant_medecin	rdv_id_medecin	INT (numérique)	11	Clé étrangère, référence à l'identifiant du médecin
	date_rendez_vous	rdv_date_rendez_vous	DATE		Date du rendez-vous
	heure_rendez_vous	rdv_heure_rendez_vous	TEMPS		Heure du rendez-vous
	statut_rendez_vous	rdv_statut_rendez_vous	VARCHAR (caractère)	30	Statut du rendez-vous (en attente, accepté, refusé)

Le modèle conceptuel de données (MCD) représente les données du système d'information de manière abstraite sous la forme d'entités, d'associations, d'attributs et de cardinalités. Enfin, je crée le modèle physique de données (MPD) qui est adapté aux caractéristiques techniques du système de gestion de bases de données.

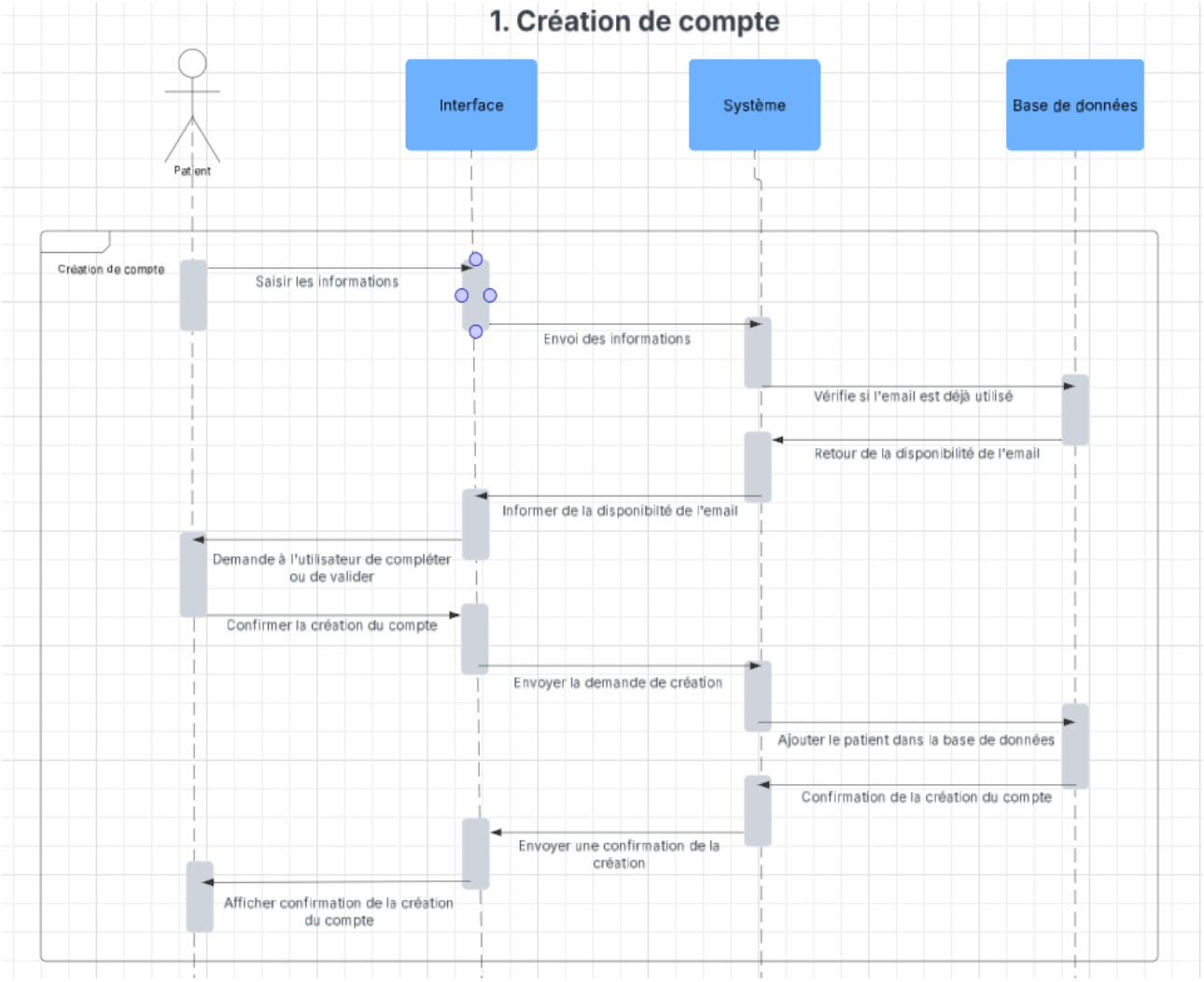


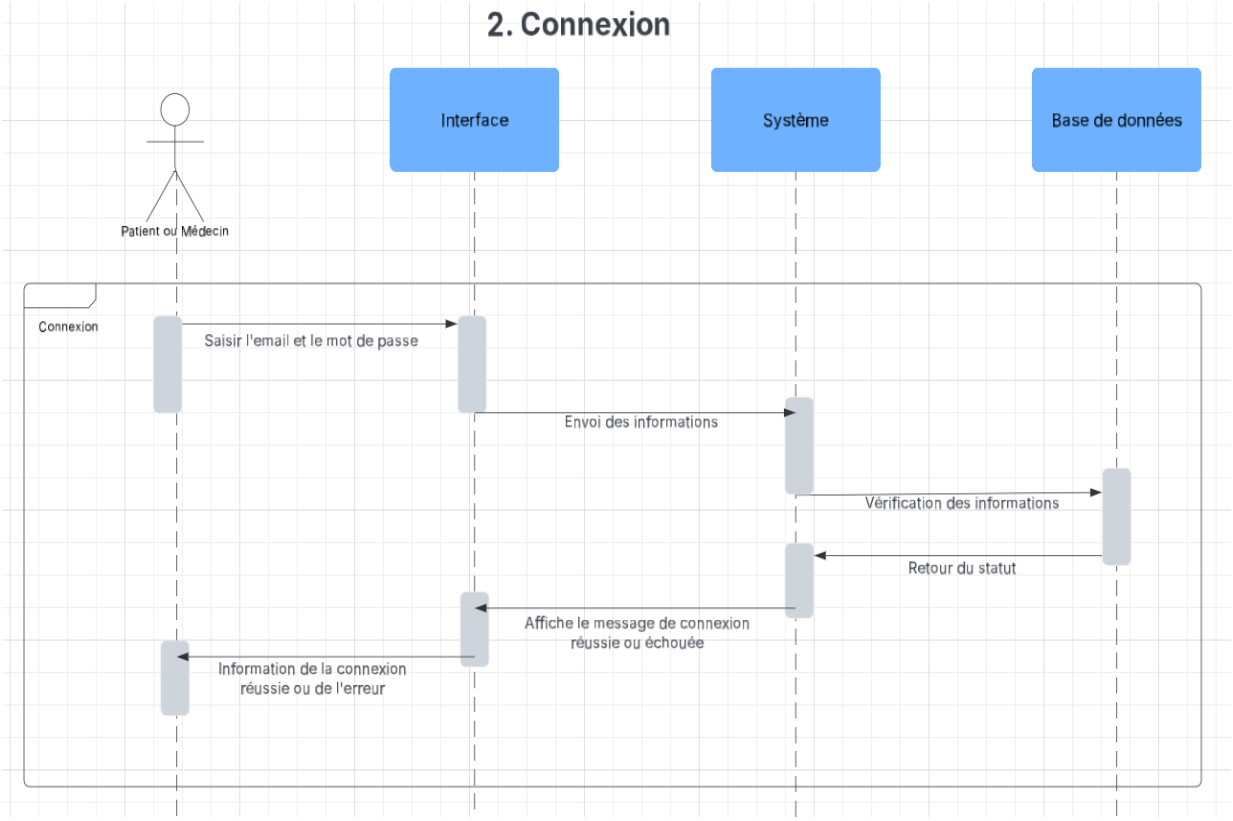


Le diagramme de cas d'utilisation est utilisé en langage de modélisation unifié (UML), il montre les interactions entre les utilisateurs et le système. Ici, les acteurs sont le patient et le médecin.

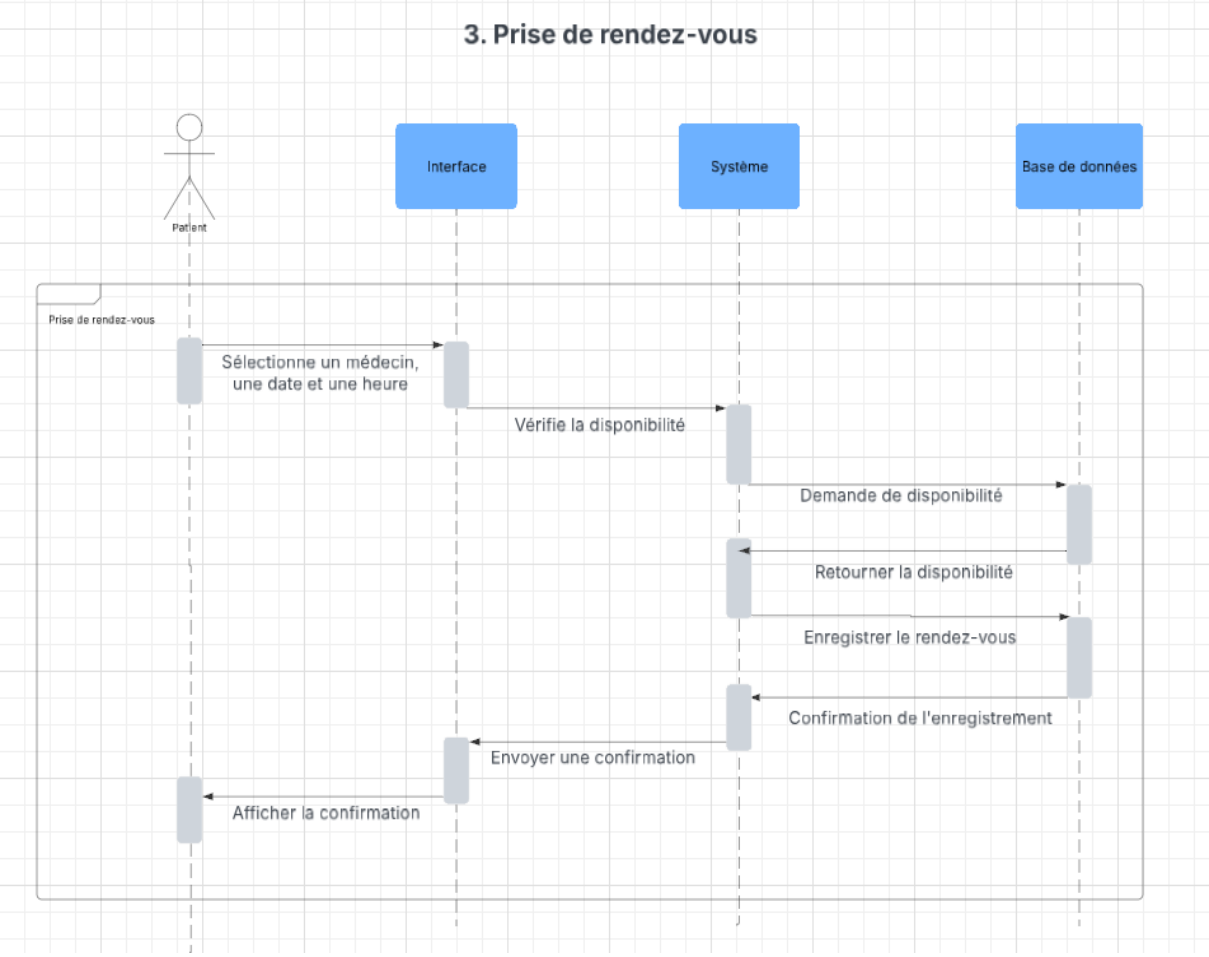


Le diagramme de séquence est utilisé en langage de modélisation unifié (UML), il représente le déroulement des interactions entre les acteurs et le système. Ici, il y a cinq acteurs qui sont le patient ou médecin, l'interface, le système et la base de données. Chaque diagramme représente une seule fonctionnalité dont la création de compte, la connexion, la prise de rendez-vous, la consultation, la modification et l'annulation des rendez-vous.

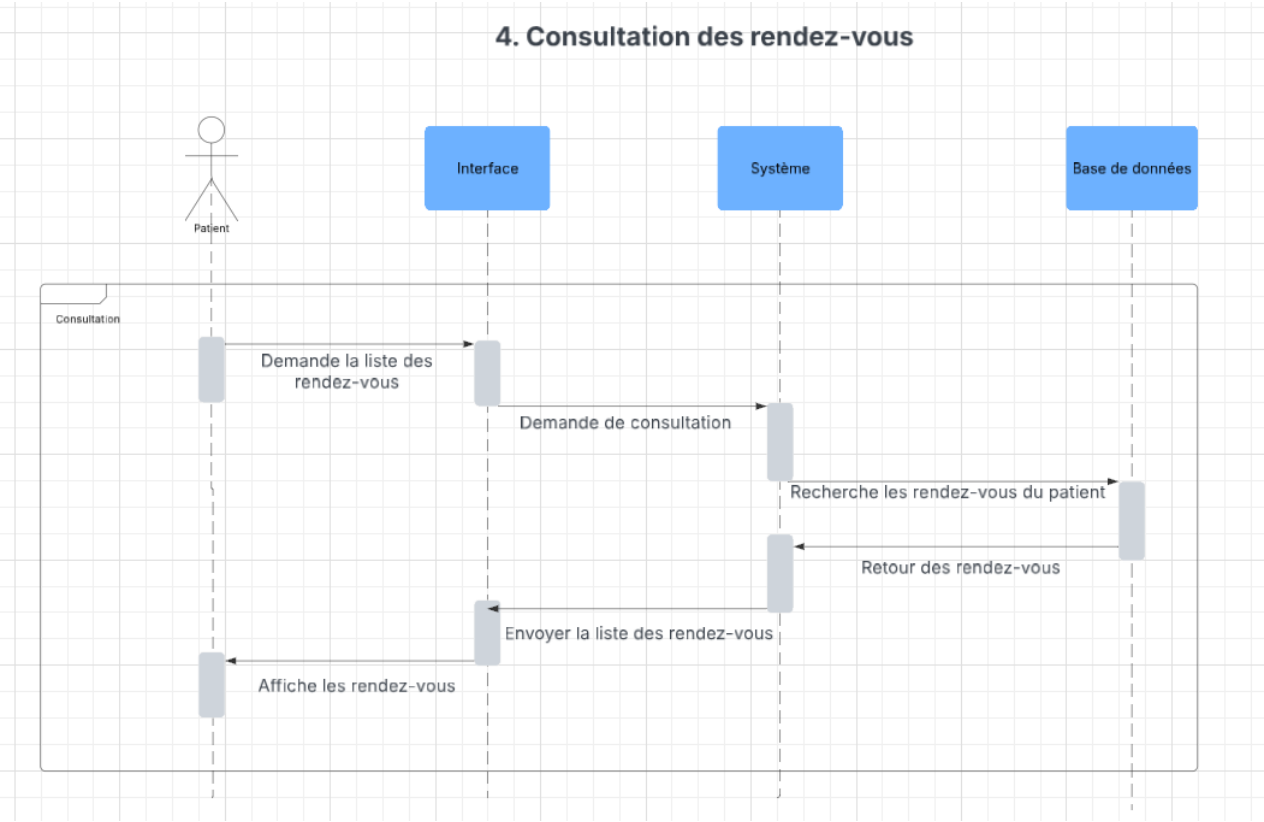

1. Création de compte
Patient
Interface
Système
Base de données
Création de compte
Saisir les informations
Envoi des informations
Vérifie si l'email est déjà utilisé
Retour de la disponibilité de l'email
Informer de la disponibilité de l'email
Demande à l'utilisateur de compléter ou de valider
Confirmer la création du compte
Envoyer la demande de création
Ajouter le patient dans la base de données
Confirmation de la création du compte
Envoyer une confirmation de la création
Afficher confirmation de la création du compte

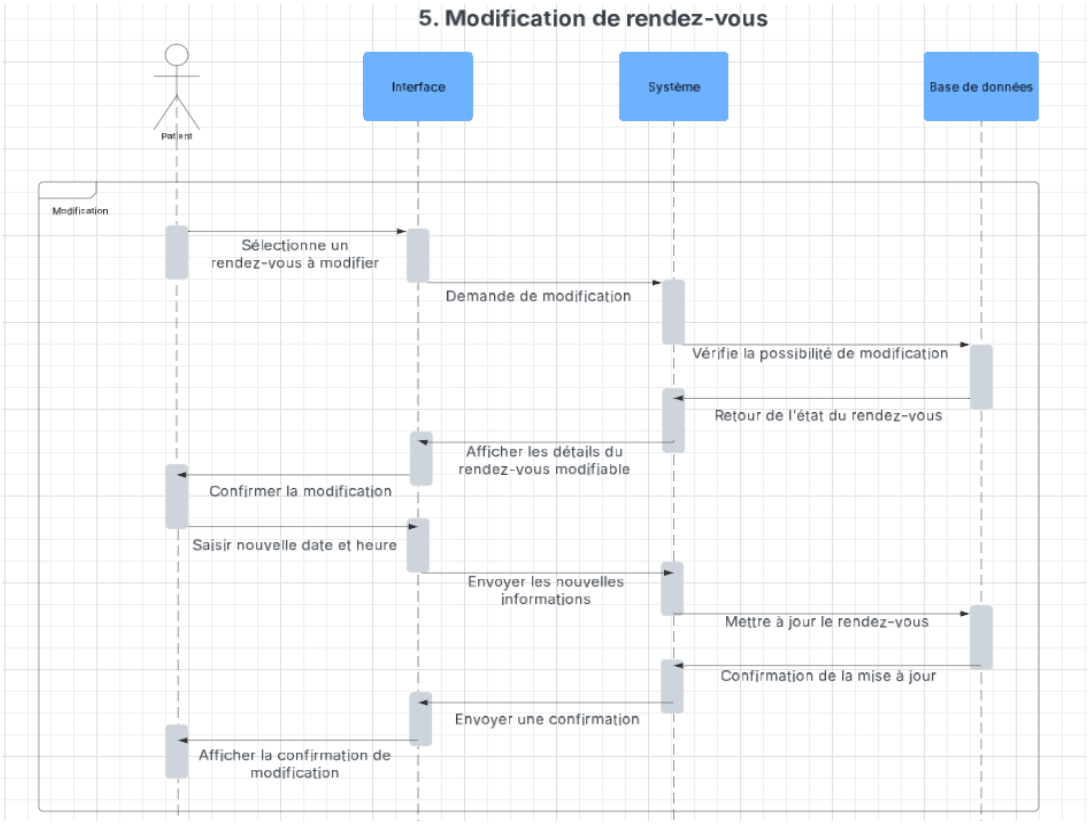

2. Connexion
Interface
Système
Base de données
Patient ou Médecin
Connexion
Saisir l'email et le mot de passe
Envoi des informations
Vérification des informations
Retour du statut
Affiche le message de connexion réussie ou échouée
Information de la connexion réussie ou de l'erreur

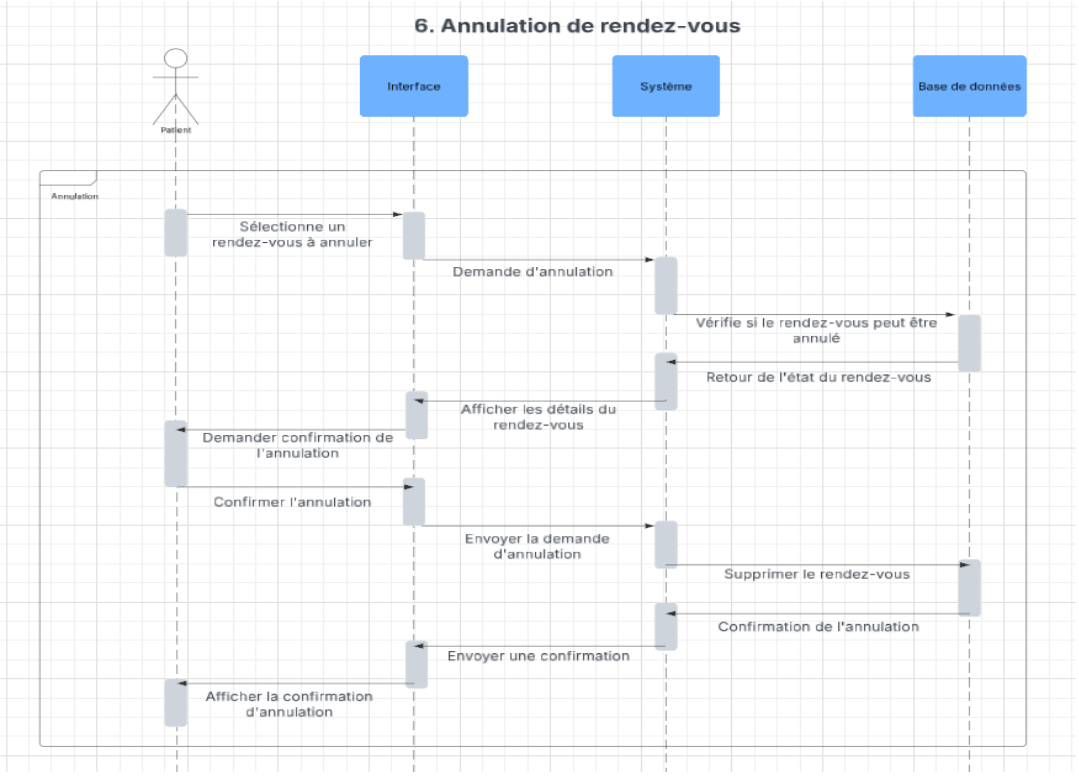
3. Prise de rendez-vous



4. Consultation des rendez-vous







4) Structure du projet

Voici la liste des principaux fichiers de code créés :

- Dossier “images” : ce dossier contient le logo
- accepter-rdv.php : accepte le rendez-vous

- annuler-rdv.php : annuler le rendez-vous
- config.php : configuration de la base de données
- connexion.php : connexion à la plateforme
- connexion-bdd.php : connexion à la base de données
- deconnexion.php : déconnexion de la plateforme
- index.html : page d'accueil
- inscription.php : formulaire d'inscription à la plateforme
- prendre-rdv.php : prendre un rendez-vous
- restaurer-rdv.php : restaurer un rendez-vous
- styles.css : styles des différentes pages
- tableau-de-bord.php : tableau de bord des patients et médecins
- voir-rdv.php : voir les rendez-vous d'un patient
- voir-tous-rdv.php : voir tous les rendez-vous d'un médecin

5) Prérequis

Dans le cadre de ce projet, le langage principal utilisé est HTML, CSS et PHP. Le développement s'effectue à l'aide de l'environnement de VS Code qui facilite l'écriture et le débogage du code. La base de données choisie est MySQL gérée via phpMyAdmin qui facilite la création, la modification et la suppression des tables et des données. Le serveur web local utilisé est XAMPP. Ce logiciel regroupe Apache, MySQL, PHP et Perl permettant de tester localement le projet et de reproduire les conditions d'un serveur de production. Enfin, le projet est conçu pour être facilement modifiable si des évolutions sont nécessaires grâce à l'utilisation du logiciel de gestion de versions Git.

6) Tests

6.1) Stratégie de tests

Les tests ont été réalisés sur l'ensemble de la plateforme afin d'assurer son bon fonctionnement et sa conformité.

L'objectif des tests d'interface (UI) est de vérifier l'affichage des pages et l'ergonomie de l'application. La méthode utilisée était de vérifier les couleurs, la lisibilité du texte et les boutons.
Un exemple de test d'interface pourrait être : [ID : UI-01] [Page : Accueil] [Action : Vérifier l'alignement des boutons] [Résultat attendu : Boutons bien alignés] [Résultat obtenu : OK] [Statut : Réussi].

L'objectif des tests fonctionnels (FUNC) est de vérifier que chaque fonctionnalité remplit son rôle. La méthode était de tester chaque action utilisateur tel que la connexion, le bon fonctionnement des formulaires, et s'assurer que les boutons déclenchent les actions attendues.
Un exemple de test fonctionnel pourrait être : [ID : FUNC-01] [Page : Connexion] [Action : Se connecter avec des identifiants valides] [Résultat attendu : Connexion réussie] [Résultat obtenu : OK] [Statut : Réussi].

L'objectif des tests de base de données (DB) est de vérifier la cohérence des données stockées. La méthode était de vérifier l'insertion, la suppression des données dans la base, tester les relations entre les tables, vérifier la gestion des erreurs.
Un exemple de test de base de données pourrait être : [ID : DB-01] [Page : Inscription] [Table : utilisateurs et patients] [Action : Remplir le formulaire d'inscription] [Résultat attendu : Nouvel utilisateur] [Résultat obtenu : OK] [Statut : Réussi].

L'objectif des tests d'erreurs (ERR) est de vérifier la robustesse du système face aux erreurs et attaques potentielles. La méthode était de laisser des champs obligatoires vides, injecter des requêtes SQL dans les formulaires et effectuer des tentatives d'accès à la connexion sans identifiants valides.
Un exemple de test d'erreur pourrait être : [ID : ERR-01] [Page : Connexion] [Action : Se connecter avec un compte inexistant.] [Résultat attendu : Message d'erreur "Aucun utilisateur trouvé avec cet email"] [Résultat obtenu : OK] [Statut : Réussi] [Commentaire : Email testé "leag@exemple.com"]

L'objectif des tests de compatibilité de navigateurs (NAV) est de vérifier qu'un site web fonctionne et s'affiche correctement sur différents navigateurs. La méthode était d'accéder aux différentes pages du site sur Chrome, Firefox et Edge.
Un exemple de test de compatibilité de navigateur pourrait être : [ID : NAV-01] [Page : Accueil] [Navigateur : Chrome] [Résultat attendu : La page d'accueil s'affiche et fonctionne correctement] [Résultat obtenu : OK] [Statut : Réussi].

6.2) Convention de nommage des tests

L'ensemble des tests sont réunis dans un fichier Excel.
La convention de nommage choisie est la suivante :

- Pour les tests d'interface (UI), et fonctionnels (FUNC)
    - [ID] [Page] [Action] [Résultat attendu] [Résultat obtenu] [Statut]
- Pour les tests de base de données (DB)
    - [ID] [Page] [Table] [Action] [Résultat attendu] [Résultat obtenu] [Statut]
- Pour les tests d'erreurs (ERR)
    - [ID] [Page] [Action] [Résultat attendu] [Résultat obtenu] [Statut] [Commentaire]
- Pour les tests de compatibilité navigateur (NAV)
    - [ID] [Page] [Navigateur] [Résultat attendu] [Résultat obtenu] [Statut]

De plus, les tests d'erreurs sont détaillés pour chaque erreur trouvée avec les étapes mises en place à sa résolution ainsi que le nouveau test effectué.

7) Evolutions

Ce projet est conçu pour être évolutif et pourra être amélioré selon les besoins futurs. Par exemple, le système pourra envoyer des notifications pour envoyer un rappel de rendez-vous au patient. Le patient pourra également annuler lui-même son rendez-vous et modifier ses informations personnelles.

8) Documentation utilisateur v1.0

1) Introduction

1.1) Présentation de l'application

Cette application web permet aux patients de prendre des rendez-vous médicaux avec des médecins.

Les fonctionnalités principales sont :

- Inscription à la plateforme pour les patients
- Connexion à la plateforme pour les patients et médecins
- Prise de rendez-vous médicaux pour les patients
- Gestion des rendez-vous pour les médecins (accepter, annuler, restaurer)

2) Accès à l'application web

2.1) Accès

L'application est accessible en local pour ce projet.
Les navigateurs compatibles sont : Chrome, Firefox et Edge.

3) Guide de l'utilisateur

3.1) Interface globale

En accédant à l'application, plusieurs actions sont possibles selon le type d'utilisateur :

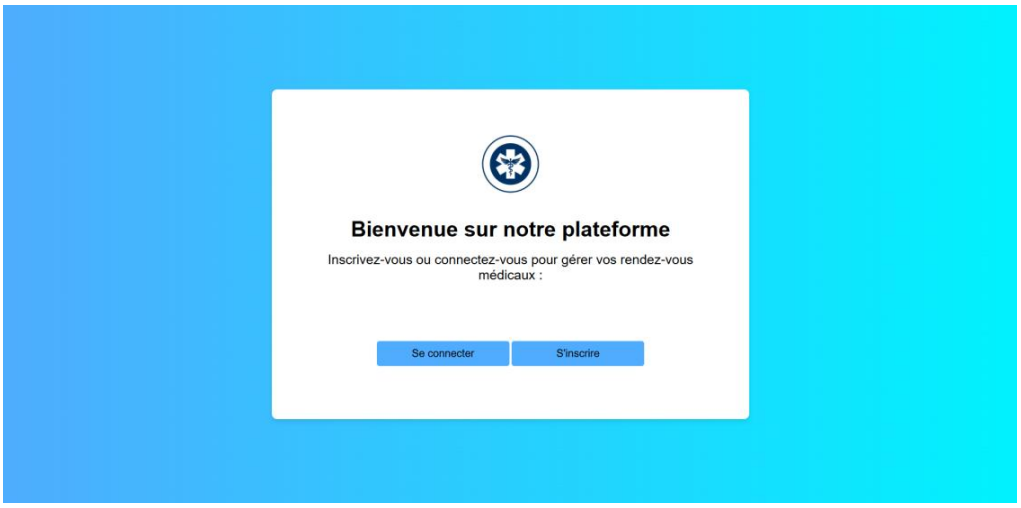
- Médecin : se connecter et gérer les rendez-vous
- Patient : créer un compte, se connecter, prendre un rendez-vous, voir ses rendez-vous pris

3.2) Fonctionnalités détaillées

Ici, les actions courantes de l'application web vont être décrites étapes par étapes.

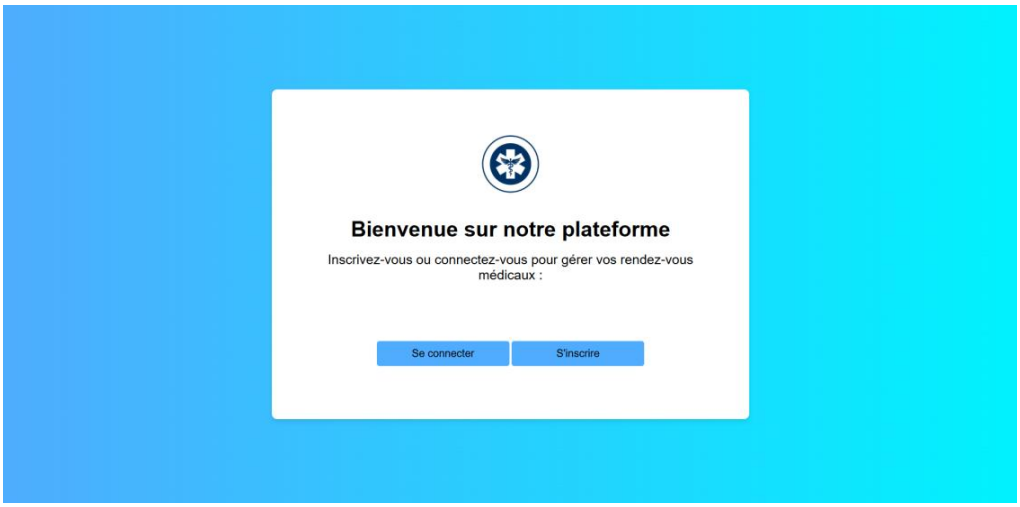
S’inscrire sur la plateforme (patient)

1. Accédez à l’accueil
2. Cliquez sur “s’inscrire”
3. Remplir le formulaire
4. Cliquez sur “créer un compte”



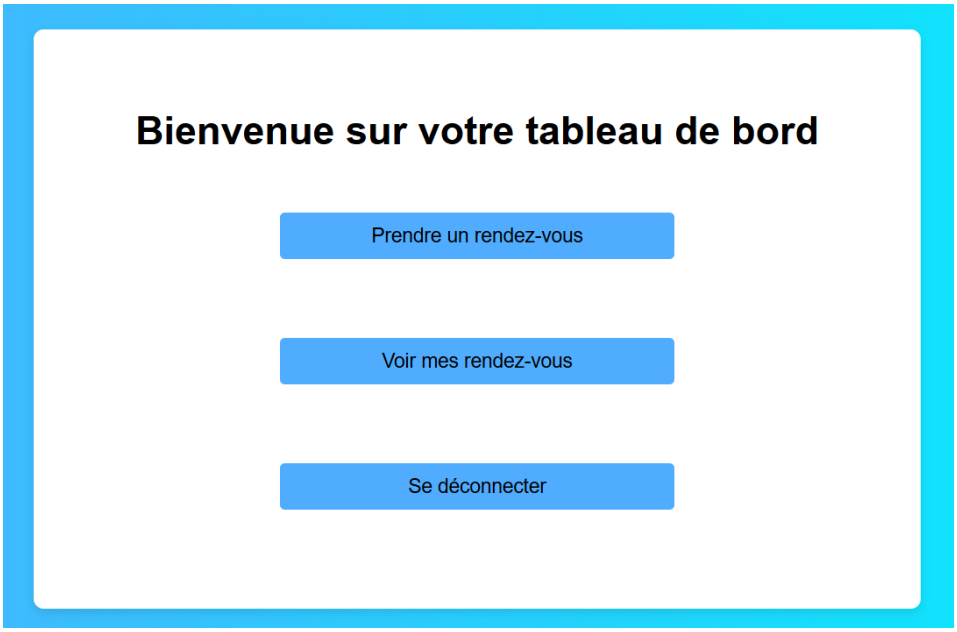
Se connecter (patient et médecin)

1. Accédez à la page d’accueil
2. Cliquez sur “se connecter”
3. Entrez vos identifiants
4. Cliquez sur “se connecter”



Prendre un rendez-vous (patient)

1. Connectez-vous
2. Cliquez sur “se connecter”
3. Vous accédez à votre tableau de bord de patient
4. Cliquez sur “prendre un rendez-vous”
5. Choisissez un médecin et une date
6. Cliquez sur “vérifier les disponibilités”
7. Choisissez une heure
8. Cliquez sur “confirmer le rendez-vous”

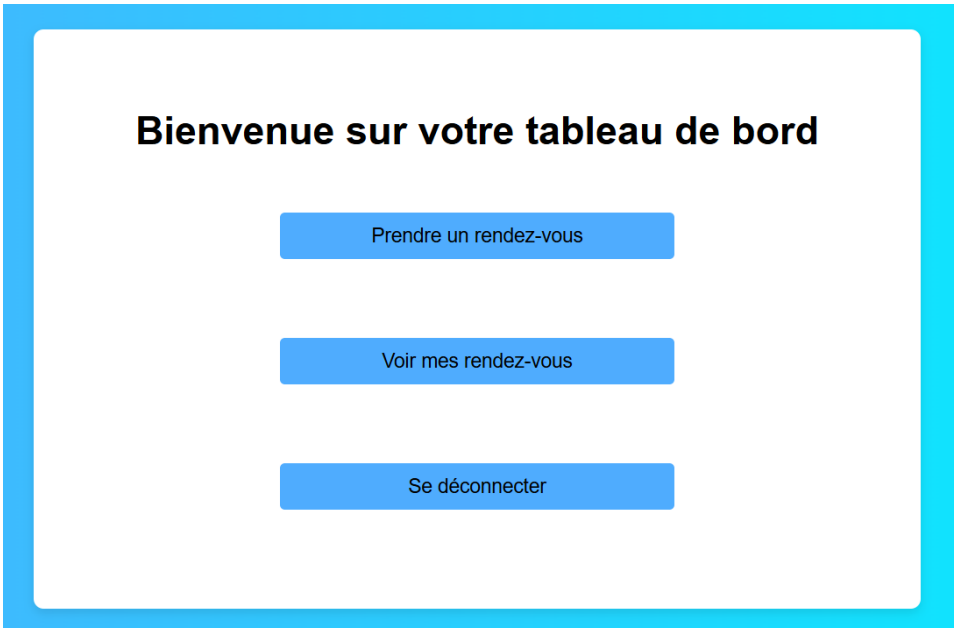



Prendre un Rendez-vous
Médecin :
Sélectionnez un médecin
Date :
jj/mm/aaaa
Vérifier les disponibilités
Retour au Tableau de Bord


Prendre un Rendez-vous
Médecin :
Durand Pierre
Date :
25/03/2025
Vérifier les disponibilités
Heure :
14h00
Confirmer le rendez-vous
Retour au Tableau de Bord

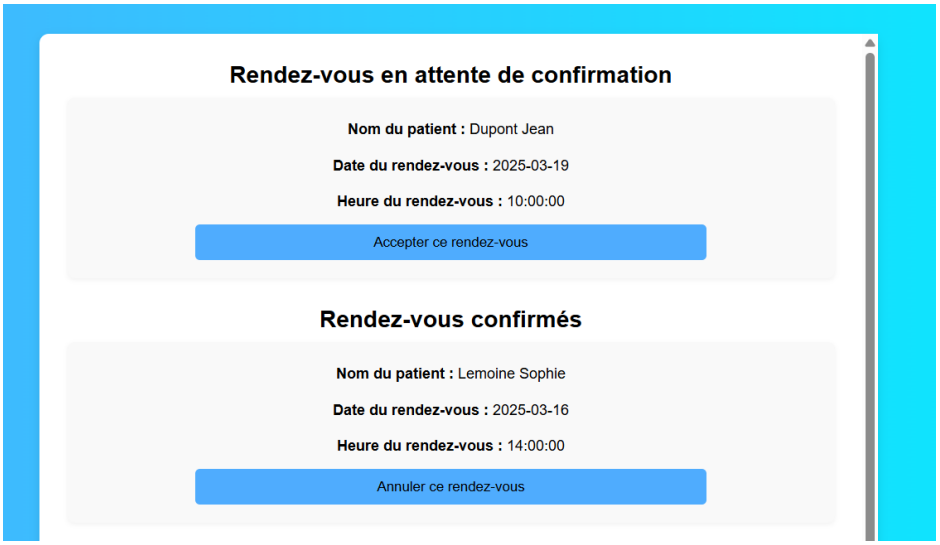
Voir ses rendez-vous (patient)

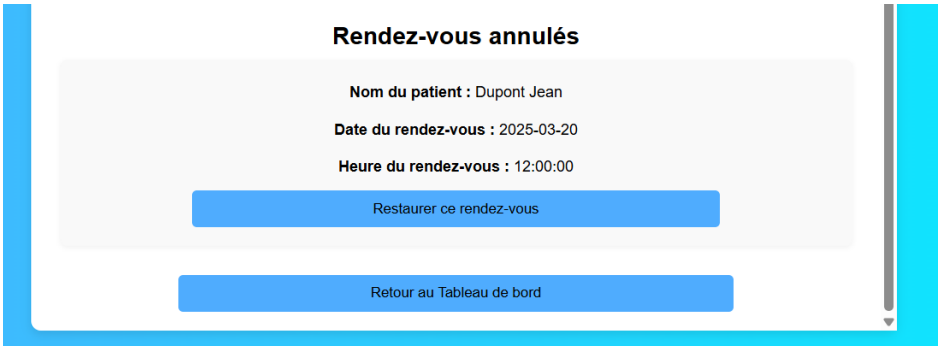
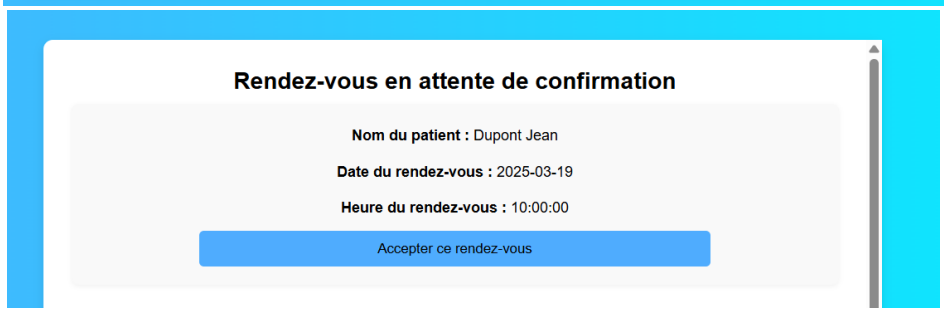
1. Connectez-vous
2. Cliquez sur “se connecter”
3. Vous accédez à votre tableau de bord de patient
4. Cliquez sur “voir mes rendez-vous”

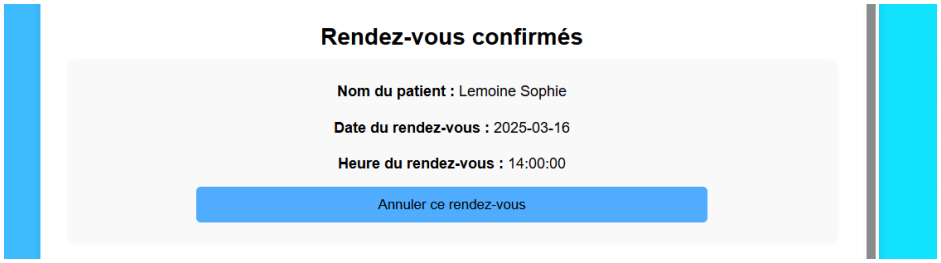
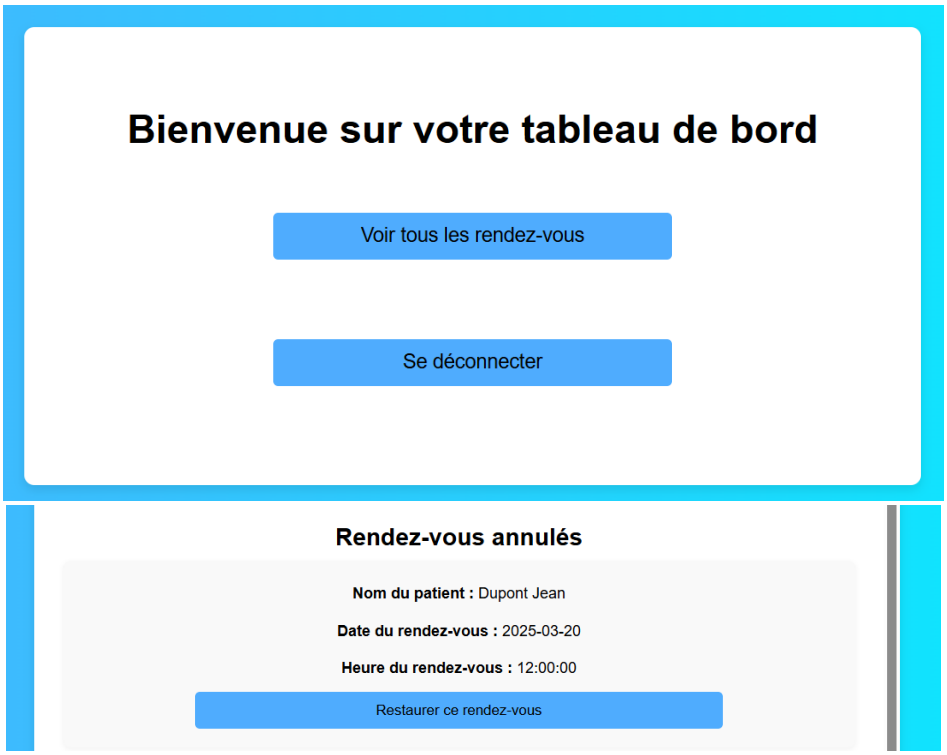


Voir ses rendez-vous (médecin)

1. Connectez-vous
2. Cliquez sur “se connecter”
3. Vous accédez à votre tableau de bord de médecin
4. Cliquez sur “voir tous les rendez-vous”



	
Accepter un rendez-vous (médecin) 1. Accédez à votre tableau de bord de médecin 2. Cliquez sur “voir tous les rendez-vous” 3. Cliquez sur “accepter ce rendez-vous”	 
Annuler un rendez-vous (médecin) 1. Accédez à votre tableau de bord de médecin 2. Cliquez sur “voir tous les rendez-vous” 3. Cliquez sur “annuler ce rendez-vous”	

	
Restaurer un rendez-vous (médecin) 1. Accédez à votre tableau de bord de médecin 2. Cliquez sur “voir tous les rendez-vous” 3. Cliquez sur “restaurer ce rendez-vous”	

4) Dépannage

4.1) Problèmes fréquents et solutions

Problème	Solution
Pourquoi mon rendez-vous est en attente ?	Un médecin doit confirmer le rendez-vous pour que celui-ci soit confirmé.
Je veux annuler mon rendez-vous.	Veuillez contacter votre médecin pour que celui-ci annule le rendez-vous.
Je veux restaurer mon rendez-vous.	Veuillez contacter votre médecin pour que celui-ci restaure le rendez-vous.

4.2) Contact pour le support

Email de contact : alice.martin@email.com (médecin référent)

Numéro de téléphone : 06 11 22 33 44
Disponibilité : du lundi au samedi de 9h00 à 17h