Rapport de Projet Application Mobile de Gestion d'un Cabinet Médical

Malouche Mohamed Rayen

Molka Toubale

Ines Tmimi

1. Contexte et Objectifs

1.1 Contexte

Ce projet a pour but de développer une application mobile multiplateforme destinée à un cabinet médical. L'application est conçue pour faciliter la gestion des rendez-vous, des consultations, de l'historique médical et des échanges de documents entre les patients et le personnel médical.

1.2 Objectifs Fonctionnels

Pour les patients :

- Prendre un rendez-vous avec un médecin (choix de date, heure, médecin).
- Envoyer des documents médicaux (résultats, radios…).
- Consulter leur historique médical (diagnostics, prescriptions).
- Recevoir des notifications (rappels de rendez-vous, lecture de documents).

Pour les médecins :

- Gérer les demandes de rendez-vous (accepter, refuser, replanifier).
- Visualiser les documents reçus et y ajouter des annotations.
- Enregistrer les consultations (diagnostics, prescriptions).
- Gérer leur profil et disponibilités.

2. Description de l'Application

2.1 Fonctionnalités Principales

1. Inscription & Connexion

- Authentification sécurisée par email/mot de passe (OAuth), avec différenciation des rôles (patient, médecin, administrateur).
- Redirection vers une page d'accueil adaptée au rôle.

2. Accueil & Navigation

- Page d'accueil par défaut.
- Interface fluide et responsive grâce à Ionic.

- Navigation entre les pages via une barre de menu et des onglets.

3. Gestion des Documents Médicaux

- Upload de fichiers par les patients.
- Visualisation et gestion des documents (consulté ou non par le médecin).
- Téléchargement et export en PDF.

4. Gestion des Rendez-vous

- Consultation des créneaux disponibles.
- Prise, modification, annulation de rendez-vous.
- Vue calendrier pour les médecins.

5. Messagerie

- Système de messagerie interne sécurisé.
- Accès restreint aux conversations selon le rôle.

6. Gestion des Profils

- Mise à jour des informations personnelles.
- Pages de profil personnalisées selon le rôle utilisateur.

7. Administration

- Ajout/suppression de médecins.
- Gestion de publications.
- Gestion des utilisateurs.

8. Recherche et Filtres

- Recherche dynamique de médecins par spécialité/localisation.
- Filtres pour affiner les résultats.

9. Notifications & Paramètres

- Notifications via Firebase Cloud Messaging (FCM) et par mail.

- Paramètres utilisateurs pour personnaliser l'expérience.

10. Statistiques

- Statistiques sur les rendez-vous, documents, délais de réponse.

3. Architecture Technique

3.1 Technologies Utilisées

- **Frontend** : Angular + Ionic 6+
- **Backend** : Python Flask
- **Base de données** : MongoDB (NoSQL)
- **Notifications** : Firebase Cloud Messaging (FCM)

3.2 Sécurité

- Authentification OAuth (email/mot de passe, Google, etc.).
- Guards Angular (AdminGuard, DoctorGuard, PatientGuard) pour sécuriser l'accès aux pages selon le rôle.

3.3 Performances

- Chargement paresseux (loadChildren) pour optimiser les performances de navigation.
- UI réactive adaptée à tous les écrans (responsive design).

4. Guide d'Installation et d'Exécution

4.1 Prérequis

- Node.js et npm
- Python 3.x et pip
- Angular CLI : npm install -g @angular/cli

```
git clone https://github.com/ines312692/CabinetMedicalProject_Rayen_Molka_Ines-Tmimi.git
cd CabinetMedical/CabinetMedical
npm install
cd CabinetMedical/BackendCabinetMedical/pythonProject
pip install -r requirements.txt
```

4.3 Exécution

Lancer le frontend

ng serve

Lancer le backend

flask run

5. Défis Rencontrés et Solutions

5.1 Conflits de fusion sur GitHub

- **Problème** : Lors du travail collaboratif ou en gestion de différentes branches (dev, main), plusieurs conflits de fusion (*merge conflicts*) sont apparus, notamment sur les fichiers de configuration (ex: package.json, environment.ts) et certaines interfaces partagées.
- **Solution** : Mise en place d'une **stratégie Git claire** :
  - Création de branches nommées par fonctionnalité (feature/rdv, feature/messaging, etc.).
  - Fusion systématique via **pull requests** (PR) avec **revue de code**.
  - Résolution manuelle des conflits dans les fichiers critiques en identifiant les versions à conserver.
  - Ajout de règles dans .gitignore pour éviter les conflits inutiles.

5.2 Gestion des rôles utilisateurs

- **Problème** : Sécuriser les routes et l'interface selon le rôle utilisateur (patient, médecin, admin).
- **Solution** : Implémentation de guards spécifiques dans Angular (AdminGuard, DoctorGuard, PatientGuard) pour interdire l'accès aux pages non autorisées.

5.3 Notifications

- **Problème** : Notifier les utilisateurs en cas de nouveaux messages, rendez-vous ou mise à jour de documents.
- **Solution** : Utilisation de **Firebase Cloud Messaging (FCM)** pour les notifications push + envoi d'**emails automatiques** en complément.