Cahier des charges : Système de gestion des rendez-vous pour cabinet médical

1. Contexte et objectifs

1.1 Contexte

Le cabinet médical souhaite automatiser la gestion des rendez-vous pour améliorer l'efficacité, réduire les erreurs administratives et offrir une meilleure expérience aux patients. Actuellement, les rendez-vous sont gérés manuellement (par téléphone ou sur place), ce qui entraîne des conflits d'horaires, des pertes de temps et des erreurs.

1.2 Objectifs

- Développer une application mobile pour permettre aux patients de réserver et gérer des rendez-vous, aux médecins de consulter leur agenda, et au personnel administratif de gérer les plannings.
- Automatiser les processus pour réduire le temps consacré à la gestion manuelle.
- Assurer une interface intuitive et accessible sur iOS et Android.
- Garantir la fiabilité (pas de doubles réservations) et la sécurité des données.
- Respecter les réglementations sur la protection des données (ex. RGPD).

2. Description du projet

Le projet consiste à développer une application mobile en **Flutter** pour iOS et Android, avec un backend basé sur des **APIs REST** développées en **Spring Boot**. L'application sera utilisée par :

- **Patients** : Pour réserver, modifier ou annuler des rendez-vous.
- **Médecins** : Pour consulter leur agenda et recevoir des notifications.
- **Personnel administratif** : Pour gérer les plannings, valider les rendez-vous et communiquer avec les patients.

3. Fonctionnalités

3.1 Pour les patients

- **Inscription et connexion** :
  - Création de compte avec e-mail, mot de passe et informations personnelles (nom, prénom, téléphone).
  - Connexion sécurisée.

- Option de réinitialisation du mot de passe.

- Prise de rendez-vous :

- Consultation des médecins disponibles (par nom ou spécialité).
- Affichage des créneaux horaires libres (par jour/semaine).
- Réservation d'un créneau avec confirmation dans l'application.

- Gestion des rendez-vous :

- Modification ou annulation d'un rendez-vous.
- Historique des rendez-vous passés et à venir.

- Notifications :

- Rappels automatiques avant le rendez-vous (push, e-mail ou SMS).

- Profil utilisateur :

- Mise à jour des informations personnelles.
- Gestion des préférences de notification.

3.2 Pour les médecins

- Connexion:

- Connexion sécurisée avec identifiants spécifiques.

- Agenda :

- Vue personnalisée des rendez-vous (journalier, hebdomadaire).
- Détails des patients pour chaque rendez-vous (nom, motif si fourni).

- Notifications:

- Alertes push pour nouveaux rendez-vous, modifications ou annulations.

- **Synchronisation :**

- Intégration avec Google Calendar ou Outlook (exportation des rendez-vous).

3.3 Pour le personnel administratif

- Connexion:

- Connexion avec rôle "ADMIN" pour accéder aux fonctionnalités de gestion.

- **Gestion des plannings :**

- Création et modification des créneaux horaires des médecins.
- Gestion des absences ou congés.

- **Validation des rendez-vous :**

- Approbation ou rejet des demandes de rendez-vous.

- Tableau de bord :

- Vue d’ensemble des rendez-vous (par jour, semaine, mois).
- Statistiques : nombre de rendez-vous, taux d’annulation, etc.

- Communication :

- Envoi de messages aux patients (confirmation, annulation, informations) via l’application, e-mail .

4. Contraintes techniques

- Frontend :

- Application mobile en **Flutter** pour iOS et Android.
- Interface responsive et intuitive (Material Design).
- Dépendances : `http` (appels API), `provider` (gestion d’état), `firebase_messaging` (notifications push).

- Backend :

- APIs REST en **Spring Boot**.
- Base de données **: PostgreSQL** pour la persistance (H2 pour les tests locaux).
- Endpoints principaux :
  - `POST /auth/login` : Authentification.
  - `POST /users` : Inscription.
  - `GET /doctors` : Liste des médecins.
  - `GET /slots` : Créneaux disponibles.
  - `POST /appointments` : Réserver un rendez-vous.
  - `PUT /appointments/{id}` : Modifier un rendez-vous.
  - `DELETE /appointments/{id}` : Annuler un rendez-vous.
  - `GET /appointments` : Liste des rendez-vous (par utilisateur).
  - `POST /slots` : Ajouter des créneaux (admin).
  - `PUT /appointments/{id}/status` : Valider/rejeter un rendez-vous (admin).
- Dépendances : `spring-boot-starter-web`, `spring-boot-starter-data-jpa`, `spring-boot-starter-security`, `postgresql`.

- Sécurité :

- Authentification via **JWT** (JSON Web Tokens).
- Chiffrement des données (HTTPS, SSL/TLS).
- Conformité RGPD : consentement utilisateur, anonymisation des données sensibles.
- Mots de passe hachés (BCrypt).

- Notifications :

  - Push via **Firebase Cloud Messaging.**
  - E-mail/SMS via **SendGrid ou Twilio.**

- Hébergement :

  - Backend sur un cloud (AWS, Azure, ou équivalent).
  - Sauvegardes régulières de la base de données.

- Interopérabilité :

  - APIs documentées avec **Swagger/OpenAPI.**
  - Intégration possible avec des systèmes externes (ex. dossiers médicaux).

- Maintenance :

  - Surveillance des performances (logs, monitoring).
  - Mises à jour régulières de l'application et du backend.

5. Contraintes organisationnelles

- Délai : Développement complet en une semaine (bien que très ambitieux, priorisation implicite des fonctionnalités patients/médecins).

- **Ressources** : Un développeur unique (Flutter + Spring Boot).

- Formation : Documentation pour le personnel administratif et les médecins.

- **Support** : FAQ pour les patients, support par e-mail pour les utilisateurs.

- Tests :

  - Tests unitaires (backend : JUnit, frontend : Flutter tests).
  - Tests d'intégration pour les APIs.
  - Tests manuels pour l'interface.

6. Livrables

- Application mobile Flutter fonctionnelle (iOS/Android, testable sur émulateur ou APK).
- Backend Spring Boot avec APIs REST déployées localement (ou cloud si possible).
- Base de données PostgreSQL configurée (H2 pour tests locaux).
- Code source (GitHub ou dossier local).
- Documentation :
  - Guide utilisateur (patients, médecins, admin).
  - Documentation technique (APIs Swagger, architecture).

7. Critères de succès

- Les patients peuvent s’inscrire, se connecter, réserver, modifier et annuler des rendez-vous.
- Les médecins peuvent consulter leur agenda et recevoir des notifications.
- Le personnel administratif peut gérer les plannings et valider les rendez-vous.
- Réduction d’au moins 50 % du temps de gestion manuelle.
- Taux d’erreurs (doubles réservations, etc.) inférieur à 1 %.
- Conformité RGPD complète.

9. Acteurs du projet

- **Développeur** : Responsable Flutter/Spring Boot.
- **Utilisateurs finaux** : Patients, médecins, personnel administratif.
- **Responsable RGPD** : Pour valider la conformité des données.