**Plan du Rapport pour le Projet de Gestion de Rendez-vous Médicaux**

**1. Introduction**

* **1.1 Contexte du Projet**
  + Présentez le besoin d’un système de gestion de rendez-vous dans un contexte médical, en expliquant les défis tels que la coordination entre médecins, patients, salles et créneaux horaires.
  + Mentionnez brièvement l’importance d’une solution automatisée pour améliorer l’efficacité et réduire les erreurs humaines.
* **1.2 Objectifs du Projet**
  + Listez les objectifs principaux, par exemple :
    - Permettre la gestion des utilisateurs avec différents rôles (administrateurs, patients, médecins, planificateurs).
    - Faciliter la réservation, l’annulation et le report des rendez-vous.
    - Gérer les disponibilités des médecins et des salles.
    - Offrir une interface graphique conviviale pour les utilisateurs.
* **1.3 Fonctionnalités Clés**
  + Résumez les fonctionnalités principales déduites du diagramme de classe :
    - Gestion des profils utilisateurs et de leurs rôles.
    - Planification et suivi des rendez-vous avec statut (planifié, annulé, etc.).
    - Vérification de la disponibilité des salles et des créneaux horaires.
    - Interaction via une interface graphique intuitive.

**2. Architecture du Système**

* **2.1 Vue d’Ensemble**
  + Décrivez la structure générale du système en vous basant sur le diagramme de classe, en mettant en avant les entités principales : utilisateurs, rôles, médecins, patients, salles, rendez-vous et créneaux horaires.
  + Mentionnez que le système suit une approche orientée objet avec des relations bien définies.
* **2.2 Relations entre les Classes**
  + Expliquez les relations clés illustrées dans le diagramme :
    - **Utilisateur Role** : Une relation plusieurs-à-un (0..\* à 1), chaque utilisateur ayant un rôle spécifique (ADMIN, PATIENT, DOCTEUR, SCHEDULER).
    - **RendezVous Docteur, Salle, Patient, TimeSlot** : Une relation plusieurs-à-un (0..\* à 1), un rendez-vous étant associé à un médecin, une salle, un patient et un créneau horaire.
    - **Scheduler RendezVous** : Une relation un-à-plusieurs (1 à 0..\*), le planificateur gérant plusieurs rendez-vous.
  + Utilisez des exemples pour clarifier les associations, agrégations ou compositions.
* **2.3 Flux de Données**
  + Décrivez comment les données circulent dans le système, par exemple :
    - Un patient réserve un rendez-vous en sélectionnant un créneau horaire disponible via la classe TimeSlot.
    - Le planificateur consulte le planning via Scheduler et met à jour les statuts des rendez-vous.

**3. Composants Principaux**

* **3.1 Utilisateur**
  + Présentez la classe Utilisateur avec ses attributs (id, nom, adresse, telephone, dateNaissance) et son rôle comme entité de base pour tous les acteurs du système.
* **3.2 Rôles**
  + Décrivez l’énumération Role (ADMIN, PATIENT, DOCTEUR, SCHEDULER) et son rôle dans la gestion des permissions et des fonctionnalités accessibles à chaque utilisateur.
* **3.3 Docteur**
  + Détaillez la classe Docteur avec ses attributs (specialite, matricule) et ses méthodes (definirDisponibilites, ajouterCommentaire), en expliquant son rôle dans la définition des disponibilités et la gestion des rendez-vous.
* **3.4 Scheduler**
  + Expliquez la classe Scheduler avec ses attributs (fonction) et ses méthodes (consulterPlanning, annulerRendezVous), en soulignant son rôle central dans la coordination des rendez-vous.
* **3.5 Salle**
  + Décrivez la classe Salle avec ses attributs (id, numero) et sa méthode estDisponible, qui vérifie la disponibilité pour un rendez-vous donné.
* **3.6 Patient**
  + Présentez la classe Patient avec ses attributs (numeroSecu, dossierMedical, sexe) et ses méthodes (reserverRendezVous, annulerReservation), en expliquant son interaction avec les rendez-vous.
* **3.7 RendezVous**
  + Détaillez la classe RendezVous avec ses attributs (id, date, comment, statut, duree, institutRendezVous) et ses méthodes (annuler, reporter, terminer), en insistant sur son importance comme cœur du système.
* **3.8 TimeSlot**
  + Décrivez la classe TimeSlot avec ses attributs (id, start, end, statut) et ses méthodes (liberer, reserver), qui gèrent les créneaux horaires et leur disponibilité.
* **3.9 Énumérations**
  + Présentez les énumérations :
    - StatutRendezVous (PLANIFIE, CONFIRME, EN\_COURS, TERMINE, ANNULE, REPORTE) pour les états des rendez-vous.
    - StatutTimeSlot (DISPONIBLE, INDISPONIBLE, RESERVE) pour les créneaux horaires.
    - SexePatient pour définir le genre des patients.

**4. Implémentation**

* **4.1 Technologies Utilisées**
  + **4.1.1 JDBC et MySQL**
    - Expliquez l’utilisation de JDBC avec MySQL Connector pour connecter l’application à une base de données MySQL, en détaillant la gestion des requêtes (SELECT, INSERT, UPDATE) et des connexions.
  + **4.1.2 Pattern DAO**
    - Décrivez le pattern DAO (Data Access Object) utilisé pour séparer la logique métier de l’accès aux données, avec des exemples comme UtilisateurDAO ou RendezVousDAO.
  + **4.1.3 Interface Graphique Swing**
    - Présentez l’interface utilisateur développée avec Swing, en mentionnant les composants utilisés (JFrame, JButton, JTable, etc.) pour afficher les plannings, réserver des rendez-vous ou gérer les utilisateurs.
* **4.2 Exemples de Code**
  + Incluez des extraits de code illustratifs, par exemple :
    - Une connexion JDBC à la base de données.
    - Une méthode DAO pour récupérer les rendez-vous.
    - Un composant Swing affichant une liste de créneaux disponibles.

**5. Conclusion**

* **5.1 Récapitulatif**
  + Résumez les points forts du projet : une architecture claire, une gestion efficace des rendez-vous, une interface utilisateur intuitive et une implémentation robuste avec JDBC, DAO et Swing.
* **5.2 Perspectives**
  + Proposez des améliorations futures, comme :
    - Ajouter des notifications par email ou SMS pour les rendez-vous.
    - Développer une version web avec Java EE.
    - Intégrer un historique des rendez-vous pour les patients et les médecins.