Documentation Technique - Plateforme de Gestion de Clinique Médicale

1. Introduction

Cette documentation présente l'architecture technique du système de gestion de clinique médicale, basée sur une approche Spring Boot pour le backend et Angular pour le frontend. Le système permet la gestion complète des patients, rendez-vous, prescriptions et factures avec un système de rôles sécurisé.

**2. Diagramme des Entités Principales**

**3. Détail des Entités**

3.1. Entités de Base

EntityAbstracte

Classe mère abstraite contenant les champs communs à toutes les entités.

java

Copy

Download

public abstract class EntityAbstracte {

@Id @GeneratedValue(strategy = GenerationType.IDENTITY)

protected Integer id;

@CreationTimestamp

protected LocalDateTime creationDate;

@UpdateTimestamp

protected LocalDateTime modificationDate;

}

InfoPersonnel

Classe embarquée contenant les informations personnelles communes.

java

Copy

Download

@Embeddable

public class InfoPersonnel {

protected String nom;

protected String prenom;

@Column(unique = true) protected String email;

protected LocalDate dateNaissance;

protected String telephone;

protected String adresse;

protected String genre;

protected String photo;

}

3.2. Gestion Médicale

Patient

java

Copy

Download

@Table(name = "patient")

public class Patient extends EntityAbstracte {

@Embedded private InfoPersonnel infoPersonnel;

*// Relations:*

@OneToOne(mappedBy = "patient") private DossierMedical dossierMedical;

@OneToMany(mappedBy = "patient") private List<RendezVous> rendezVous;

@OneToMany(mappedBy = "patient") private List<Facture> factures;

@OneToMany(mappedBy = "patient") private List<Ordonnance> ordonnances;

@OneToMany(mappedBy = "patient") private List<Prescription> prescriptions;

}

DossierMedical

java

Copy

Download

@Table(name = "dossierMedical")

public class DossierMedical extends EntityAbstracte {

private String antecedents;

private String allergies;

private String traitementsEnCours;

private String observations;

*// Relations:*

@OneToOne() private Patient patient;

@OneToMany(mappedBy = "dossierMedical") private List<Consultation> consultations;

@OneToMany(mappedBy = "dossierMedical") private List<Ordonnance> ordonnances;

@OneToMany(mappedBy = "dossierMedical") private List<Prescription> prescriptions;

}

Consultation

java

Copy

Download

@Table(name = "consultation")

public class Consultation extends EntityAbstracte {

@NotNull private String motifs;

@NotNull private String tensionArterielle;

@NotNull private Float temperature;

@NotNull private Float poids;

@NotNull private Float taille;

@NotNull private String compteRendu;

@NotNull private String diagnostic;

*// Relations:*

@ManyToOne private DossierMedical dossierMedical;

@OneToMany(mappedBy = "consultation") private List<Prescription> prescriptions;

@OneToMany(mappedBy = "consultation") private List<Ordonnance> ordonnances;

@OneToOne private Facture facture;

@OneToOne private RendezVous rendezVous;

}

3.3. Gestion des Rendez-vous

RendezVous

java

Copy

Download

@Table(name = "rendezVous", uniqueConstraints = {

@UniqueConstraint(columnNames = {"dateHeure", "utilisateur\_id"}),

@UniqueConstraint(columnNames = {"dateHeure", "salle\_id"})

})

public class RendezVous extends EntityAbstracte {

@Column(unique = true, nullable = false)

private LocalDateTime dateHeure;

@Enumerated(EnumType.STRING)

private StatutRDV statut;

private String notes;

*// Relations:*

@ManyToOne private Patient patient;

@ManyToOne private Utilisateur utilisateur;

@ManyToOne private Salle salle;

@OneToOne(mappedBy = "rendezVous") private Consultation consultation;

}

Salle

java

Copy

Download

@Table(name = "salle")

public class Salle extends EntityAbstracte {

private String numero;

@Enumerated(EnumType.STRING)

@Column(nullable = false)

private StatutSalle statutSalle;

@OneToMany(mappedBy = "salle")

private List<RendezVous> rendezVous;

}

3.4. Gestion des Ordonnances et Prescriptions

Prescription

java

Copy

Download

@Table(name = "prescription")

public class Prescription extends EntityAbstracte {

private LocalDate datePrescription;

private String instructions;

private String dureeTraitement;

private String medicaments;

*// Relations:*

@ManyToOne private Consultation consultation;

@ManyToOne private Utilisateur utilisateur;

@ManyToOne private Patient patient;

@ManyToOne private DossierMedical dossierMedical;

}

Ordonnance

java

Copy

Download

@Table(name = "ordonnance")

public class Ordonnance extends EntityAbstracte {

private LocalDate dateOrdonnance;

private String medecinsConseils;

*// Relations:*

@ManyToOne private Consultation consultation;

@ManyToOne private Utilisateur utilisateur;

@ManyToOne private Patient patient;

@ManyToOne private DossierMedical dossierMedical;

}

3.5. Gestion Financière

Facture

java

Copy

Download

@Table(name = "facture")

public class Facture extends EntityAbstracte {

private Float montant;

private LocalDate dateEmission;

@Enumerated(EnumType.STRING)

private StatutPaiement statutPaiement;

@Enumerated(EnumType.STRING)

private ModePaiement modePaiement;

*// Relations:*

@ManyToOne private Patient patient;

@OneToOne private Consultation consultation;

}

3.6. Gestion des Utilisateurs

Utilisateur

java

Copy

Download

@Table(name = "utilisateurs")

public class Utilisateur extends EntityAbstracte {

@Embedded private InfoPersonnel infoPersonnel;

@Column(nullable = false) @JsonIgnore private String motDePasse;

private boolean actif;

*// Relations:*

@OneToMany(mappedBy = "utilisateur") private List<Role> roles;

@OneToMany(mappedBy = "utilisateur") private List<RendezVous> rendezVous;

@OneToMany(mappedBy = "utilisateur") private List<Prescription> prescriptions;

@OneToMany(mappedBy = "utilisateur") private List<Ordonnance> ordonnances;

@OneToMany(mappedBy = "expediteur") private List<Message> messagesEnvoyes;

@OneToMany(mappedBy = "destinataire") private List<Message> messagesRecus;

@OneToMany(mappedBy = "utilisateur") private List<HistoriqueAction> historiqueActions;

}

Role

java

Copy

Download

@Table(name = "role")

public class Role extends EntityAbstracte {

@Enumerated(EnumType.STRING)

private RoleType nomRole;

}

3.7. Communication

Message

java

Copy

Download

@Table(name = "message")

public class Message extends EntityAbstracte {

private String contenu;

private LocalDateTime dateEnvoi;

private boolean lu;

*// Relations:*

@ManyToOne private Utilisateur expediteur;

@ManyToOne private Utilisateur destinataire;

}

HistoriqueAction

java

Copy

Download

@Table(name = "historiqueAction")

public class HistoriqueAction extends EntityAbstracte {

private LocalDateTime date;

private String action;

@ManyToOne

private Utilisateur utilisateur;

}

4. Enums Utilisés

StatutSalle

java

Copy

Download

public enum StatutSalle {

DISPONIBLE, OCCUPEE, EN\_MAINTENANCE

}

StatutRDV

java

Copy

Download

public enum StatutRDV {

Confirme, Annule, Termine

}

StatutPaiement

java

Copy

Download

public enum StatutPaiement {

Paye, NonPaye, Annule

}

RoleType

java

Copy

Download

public enum RoleType {

ADMIN, MEDECIN, SECRETAIRE

}

ModePaiement

java

Copy

Download

public enum ModePaiement {

Espece, Carte, Cheque

}

**6. Workflows Principaux**

1. **Prise de Rendez-vous**:
   * Patient → RendezVous → Consultation → Prescription/Ordonnance → Facture
2. **Gestion Médicale**:
   * Patient → DossierMedical → Consultation → (Prescription/Ordonnance)
3. **Communication**:
   * Utilisateur ↔ Message

**7. Sécurité**

Le système utilise:

* Authentification JWT
* Rôles basés sur Spring Security (ADMIN, MEDECIN, SECRETAIRE)
* Protection des routes API

**8. Points Techniques Importants**

* Contraintes d'unicité sur les rendez-vous (médecin/salle)
* Historique complet des actions utilisateurs
* Système de messagerie interne
* Génération de documents (PDF) pour ordonnances et factures

Cette documentation couvre l'ensemble des entités et leur relations dans le système de gestion de clinique médicale. Pour des informations plus spécifiques sur l'implémentation des APIs ou des composants frontend, se référer à la documentation complémentaire.

Fonctionnement Général de l'Application de Gestion de Clinique Médicale

Architecture Globale

L'application suit une architecture modulaire organisée autour de trois couches principales :

1. **Frontend** (Angular) : Interface utilisateur avec tableaux de bord, calendrier et formulaires
2. **Backend** (Spring Boot) : Logique métier et API REST sécurisée
3. **Base de données** (PostgreSQL) : Stockage des données relationnelles

Flux Principal de l'Application

**1. Authentification et Autorisation**

* Les utilisateurs (admin, médecin, secrétaire) se connectent via JWT
* Le système vérifie leur rôle (RoleType) pour déterminer les permissions
* Chaque action est enregistrée dans HistoriqueAction

**2. Gestion des Patients**

**3. Prise de Rendez-vous**

Diagram

Code

Download

PatientMédecinSalleSystèmeSecrétairePatientMédecinSalleSystèmeSecrétaireCrée RendezVousVérifie disponibilitéVérifie disponibilitéConfirmationNotification

Relations Clés Entre Entités

1. Cœur Médical

* **Patient** (1) ↔ (1) **DossierMedical** : Relation un-à-un
  + Le dossier contient l'historique médical complet
  + Exemple : Antécédents, allergies, observations
* **DossierMedical** (1) ↔ (n) **Consultation** : Relation un-à-plusieurs
  + Chaque consultation ajoute des données au dossier
  + Contient : motifs, diagnostic, mesures cliniques
* **Consultation** (1) ↔ (n) **Prescription/Ordonnance** : Relation un-à-plusieurs
  + Les prescriptions sont liées à une consultation spécifique
  + Incluent : médicaments, durée traitement, instructions

2. Gestion des Rendez-vous

* **RendezVous** : Entité centrale avec multiples contraintes
  + Unique par médecin (utilisateur) et salle sur un créneau horaire
  + Statut modifiable (Confirmé/Annulé/Terminé)
  + Converti en Consultation après réalisation
* Relations :
  + (n) **RendezVous** ↔ (1) **Patient**
  + (n) **RendezVous** ↔ (1) **Utilisateur** (médecin)
  + (n) **RendezVous** ↔ (1) **Salle**

3. Aspect Financier

* **Facture** : Générée après consultation
  + Liée à un patient et une consultation
  + Statut de paiement (Payé/Non payé/Annulé)
  + Mode de paiement (Espèce/Carte/Chèque)

4. Communication Interne

* **Message** : Système de chat entre utilisateurs
  + Relation (n) **Message** ↔ (1) **Utilisateur** (expéditeur)
  + Relation (n) **Message** ↔ (1) **Utilisateur** (destinataire)
  + Suivi de l'état (lu/non lu)

Workflows Métiers Détailés

Pour un Médecin

1. Consulter son calendrier de rendez-vous (RendezVous)
2. Accéder au dossier médical (DossierMedical) du patient
3. Créer une nouvelle consultation (Consultation)
4. Rédiger prescriptions (Prescription) ou ordonnances (Ordonnance)

Pour la Secrétaire

1. Créer/modifier des rendez-vous avec vérification automatique :
   * Disponibilité du médecin
   * Disponibilité de la salle
   * Unicité pour le patient
2. gérer les factures

Pour l'Administrateur

1. Gestion des comptes utilisateurs (Utilisateur + Role)
2. Supervision des statistiques :
   * Nombre de consultations
   * Revenus (via Facture)
   * Occupation des salles

Fonctionnalités Spéciales

Contrôles Métiers Implémentés

1. **Rendez-vous** :
   * Empêche les doublons (médecin/salle)
   * Vérifie qu'un patient n'a qu'un RDV par jour
2. **Prescriptions** :
   * Lien obligatoire avec une consultation
   * Historique dans le dossier médical
3. **Facturation** :
   * Génération automatique après consultation
   * Calcul du montant basé sur les actes

Système de Notification

* Basé sur WebSocket pour :
  + Nouveaux messages (Message)
  + Rappels de rendez-vous
  + Changements de statut (RDV/factures)

Cycle de Vie Typique d'une Consultation

1. **Prise de RDV** (Secrétaire) → RendezVous
2. **Réalisation** (Médecin) → Converti en Consultation
3. **Documentation** :
   * Ajout au DossierMedical
   * Création de Prescription si nécessaire
4. **Facturation** → Génération de Facture
5. **Suivi** :
   * Paiement (statut Facture)
   * Respect des Prescriptions

Explications des Relations Clés

1. **Patient-DossierMedical** : Relation 1-1
   * Chaque patient a un seul dossier médical complet
   * Le dossier référence toutes les consultations du patient
2. **RendezVous-Consultation** : Relation 1-1
   * Un rendez-vous se transforme en consultation après réalisation
   * La consultation conserve le lien vers le RDV original
3. **Utilisateur-Rôle** : Relation 1-n
   * Un utilisateur peut avoir plusieurs rôles (admin peut aussi être médecin)
   * Détermine les permissions dans l'application
4. **Consultation-Prescription/Ordonnance** : Relation 1-n
   * Une consultation peut générer plusieurs prescriptions
   * Les ordonnances sont optionnelles selon les cas
5. **Salle-RendezVous** : Relation 1-n
   * Une salle peut accueillir plusieurs RDV (à des dates différentes)
   * Contrôle de disponibilité via le statut

Points Techniques Importants

1. **Contraintes d'Intégrité** :
   * Unicité des RDV (médecin + créneau horaire)
   * Unicité des RDV (salle + créneau horaire)
   * Un patient ne peut avoir qu'un RDV par jour
2. **Héritage** :
   * Toutes les entités concrètes héritent de EntityAbstracte
   * Fournit l'ID et les dates de création/modification
3. **Design Pattern** :
   * Composition forte pour InfoPersonnel (embeddable)
   * DTOs utilisés pour la couche API (non montrés ici)

Cas d’Utilisation et Explications Détaillées

1. Acteurs Principaux

**A. Administrateur**

* Gestion complète du système
* Droits d'accès à toutes les fonctionnalités
* Cas d'utilisation :
  + Création/suspension des comptes
  + Attribution des rôles (médecin/secrétaire)
  + Supervision des activités (logs)
  + Analyse financière

**B. Médecin**

* Activités cliniques principales
* Cas d'utilisation :
  + Consultation des dossiers patients
  + Saisie des observations médicales
  + Prescription des traitements
  + Communication avec le staff

**C. Secrétaire**

* Gestion opérationnelle
* Cas d'utilisation :
  + Organisation des rendez-vous
  + Interface avec les patients
  + Suivi financier élémentaire
  + Communication interne

**D. Patient** (externe)

* Interaction limitée
* Cas d'utilisation :
  + Demande de rendez-vous

2. Scénarios Types

**Scenario 1 : Prise de Rendez-vous**

1. Secrétaire vérifie disponibilité (médecin/salle)
2. Système vérifie unicité pour le patient
3. Enregistrement avec confirmation immédiate
4. Notification automatique au patient

**Scenario 2 : Consultation Médicale**

1. Médecin accède au dossier patient
2. Saisie des paramètres cliniques
3. Génération d'ordonnance si nécessaire
4. Clôture avec génération automatique de facture

**Scenario 3 : Gestion Financière**

1. Secrétaire consulte les factures impayées
2. Envoi de rappels aux patients
3. Enregistrement des paiements
4. Mise à jour automatique du statut

3. Sécurité et Contrôles

* **Contrôle d'accès** : Basé sur les rôles (RBAC)
* **Validation métier** :
  + Un médecin ne peut modifier que ses propres consultations
  + Une secrétaire ne peut accéder aux données médicales sensibles
  + Impossible de supprimer les historiques des actions