

BERGER
Lizo
BENBOUBAKER
Souheil
16/06/2025

Rapport de Projet - Gestion Hospitalière



Rapport de PROJET

GESTION HOSPITALIERE

SOMMAIRE

INTRODUCTION	0
--------------------	---

BERGER
Lizo
BENBOUBAKER
Souheil
16/06/2025

Rapport de Projet - Gestion Hospitalière



DIAGRAMMES UML	1
CAHIER DE TEST	2
CHOIX TECHNIQUES	3
REPARTITION DES TACHES DANS LE BINOME	4
CONCLUSION	5

1. Introduction – Description de notre projet

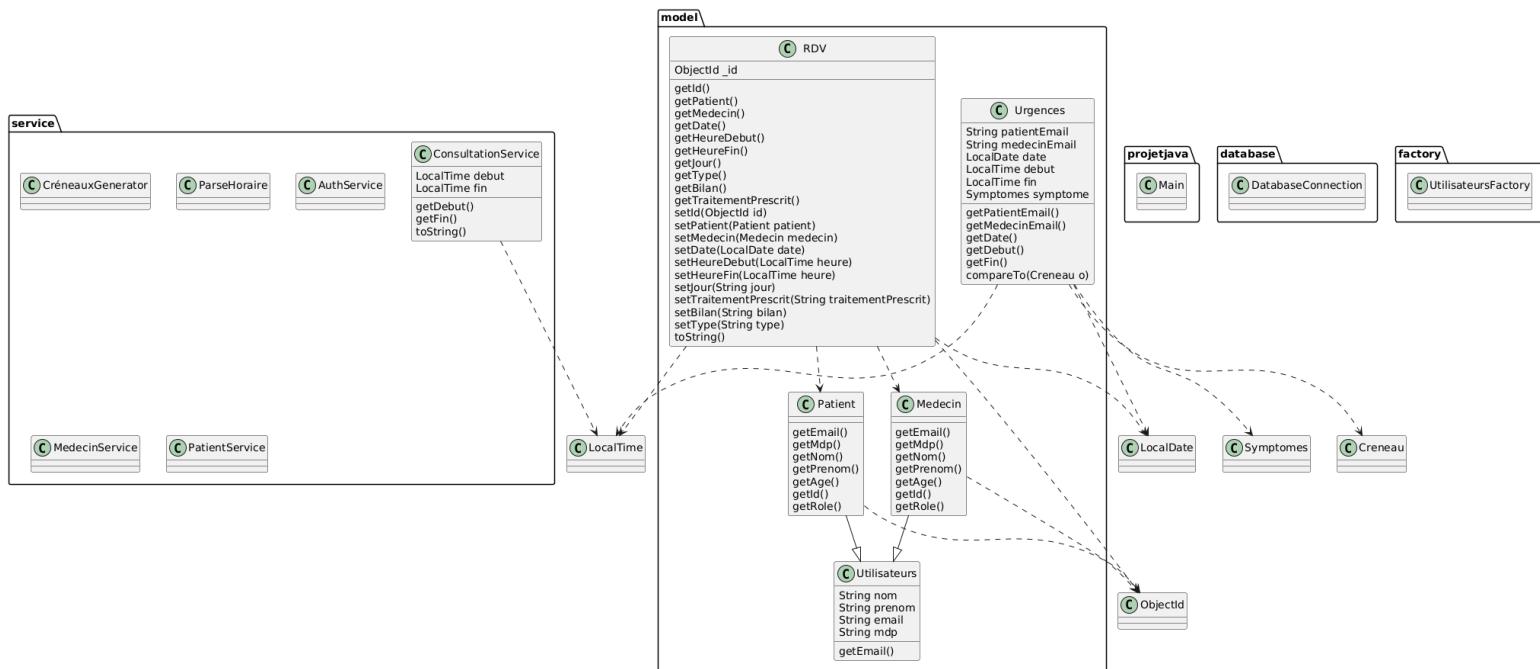
Dans le cadre du module de **développement logiciel avancé**, notre binôme a conçu une application Java permettant la **gestion hospitalière numérique**. Ce projet vise à simuler les interactions principales entre les **patients**, les **médecins**, et les **services médicaux**, en s'appuyant sur une architecture modulaire.

⌚ Objectifs principaux :

- Permettre aux patients de **prendre rendez-vous** avec un médecin en fonction des créneaux disponibles.
- Fournir aux médecins un accès à leur planning ainsi qu'un système de **bilan et traitement médical**.
- Gérer des situations d'**urgence médicale** en parallèle du système de rendez-vous classique.
- Assurer une **authentification sécurisée** des utilisateurs (patients ou médecins).
- Stocker toutes les données dans une **base NoSQL (MongoDB)** pour une plus grande flexibilité.

L'application est développée en **Java** en s'appuyant sur une architecture orientée objet, répartie en plusieurs packages fonctionnels (model, service, database, users, etc.). Le projet met en avant la logique métier, la gestion des données et le respect de bonnes pratiques de développement logiciel.

2. Diagrammes UML



3. Cahier de test

1. Tests du service RDV (ConsultationService)

Le service ConsultationService gère la création, la mise à jour, la suppression et la recherche des rendez-vous médicaux.

Objectifs des tests :

- Vérifier qu'un rendez-vous peut être correctement créé avec un patient, un médecin, une date et un créneau horaire valides.
- Vérifier la mise à jour du bilan et du traitement prescrit d'un rendez-vous existant.
- Valider la suppression d'un rendez-vous via son identifiant.
- Assurer la recherche des rendez-vous par médecin et par date, garantissant le bon filtrage des résultats.

Méthodologie :

- Utilisation d'une base MongoDB de test pré-remplie avec des données patients et médecins.
- Simulation de scénarios complets d'utilisation, de la prise de rendez-vous à l'édition des informations médicales.
- Vérification des réponses API et de la cohérence des données en base après chaque opération.

Résultats attendus :

- Création réussie avec génération d'un identifiant unique.
- Mise à jour effective et persistée du traitement et bilan.
- Suppression confirmée par absence de l'enregistrement.
- Recherche renvoyant uniquement les rendez-vous correspondant aux critères.

2. Tests du service d'authentification (AuthService)

Le service AuthService est responsable de la gestion des connexions des utilisateurs (patients et médecins).

Objectifs des tests :

- Vérifier l'authentification réussie avec un email et mot de passe corrects.
- Tester la gestion des erreurs d'identifiants (mauvais email ou mot de passe).
- Valider la création d'un nouveau compte utilisateur avec tous les champs requis.
- Vérifier la persistance sécurisée du mot de passe (hashé).

Méthodologie :

- Tests sur un environnement isolé avec des utilisateurs pré-enregistrés.
- Scénarios incluant tentatives de connexion valides et invalides.
- Vérification de la gestion des sessions et tokens d'authentification si applicable.

Résultats attendus :

- Accès autorisé pour les utilisateurs valides.
- Refus d'accès avec messages d'erreur adaptés pour les tentatives invalides.
- Nouveau compte correctement enregistré dans la base.
- Mot de passe stocké de manière sécurisée.

3. Tests du service Médecin (MédecinService)

Le service MédecinService permet la gestion des données des médecins, notamment leurs horaires de consultation.

Objectifs des tests :

- Vérifier la création et la mise à jour des médecins avec leurs horaires.
- Contrôler la récupération des horaires de consultation par jour.
- Assurer la cohérence des données en cas de modification des plages horaires.
- Tester la recherche de médecins disponibles sur une plage horaire donnée.

Méthodologie :

- Base de données de test avec plusieurs médecins et horaires variés.
- Scénarios de mise à jour des horaires et consultation des disponibilités.
- Tests des fonctions de recherche par plage horaire.

Résultats attendus :

- Données médecins mises à jour avec succès.
- Horaires correctement retournés et formatés.
- Recherche de disponibilités fiable et précise.

4. Explications de nos choix techniques

Langage et structure logicielle

Nous avons choisi Java pour sa structure fortement typée, sa lisibilité en contexte académique et sa compatibilité avec les bibliothèques MongoDB.

Le code est organisé selon une architecture MVC allégée, avec séparation des responsabilités entre :

- model : entités métier (Patient, Medecin, RDV, Urgences)
- service : logique applicative (ConsultationService, AuthService, etc.)
- database : gestion de la persistance (DatabaseConnection)
- users : gestion des rôles et de l'authentification
- factory : création dynamique d'objets utilisateurs

Base de données

Nous avons opté pour MongoDB comme base de données. Son modèle orienté document (NoSQL) est bien adapté à notre application :

- plus de flexibilité dans la structure des objets
- simplicité de mise en œuvre via le driver MongoDB Java
- pas besoin de schéma rigide à l'avance

Tests

Des tests manuels ont été réalisés pour :

- La création et la suppression de rendez-vous
- La génération de créneaux
- L'authentification des utilisateurs

Une extension future pourrait consister à intégrer JUnit pour automatiser les tests unitaires.

5. Répartition du travail dans le binôme

Nous avons réparti le travail selon les compétences et affinités de chacun :

- **Lizo :**
 - Développement des entités métiers RDV, Urgences
 - Connexion et manipulation des données avec MongoDB
 - Écriture des méthodes de traitement (ConsultationService, etc.)
 - **Souheil :**
 - Création des entités utilisateurs (Patient, Medecin, Utilisateurs)
 - Développement de l'authentification (AuthService)
 - Génération automatique des créneaux horaires disponibles (CreneauxGenerator)
- La rédaction du rapport, la génération du diagramme UML, ainsi que la vérification fonctionnelle de l'ensemble de l'application ont été réalisées en commun.

6. Conclusion

Qui a fait quoi dans le binôme (synthèse)

- **Lizo :** gestion des rendez-vous, traitement, urgence, base de données
- **Souheil :** authentification, utilisateurs, génération de créneaux

Analyse des résultats

Le projet est fonctionnel. Nous avons réussi à :

- Gérer la création et l'affichage des rendez-vous
- Intégrer une gestion simple des urgences
- Sécuriser l'accès via un système de connexion
- Sauvegarder l'ensemble dans MongoDB de manière fluide

Des améliorations restent possibles :

- Ajouter une interface graphique (Swing, JavaFX ou web)
- Mettre en place un système de conflit de créneaux
- Étendre les rôles (ex. : personnel administratif)

Utilisation de l'IA

Aucune technologie d'intelligence artificielle n'a été utilisée dans ce projet. Cependant, nous envisageons que des versions futures puissent intégrer des fonctionnalités d'IA telles que :

- Priorisation des urgences selon les symptômes détectés
- Suggestions intelligentes de créneaux horaires en fonction des habitudes du patient ou des disponibilités du médecin

BERGER
Lizo
BENBOUBAKER
Souheil
16/06/2025

Rapport de Projet - Gestion **Hospitalière**

