**Documentation du Projet**

---

**Table des matières**



---

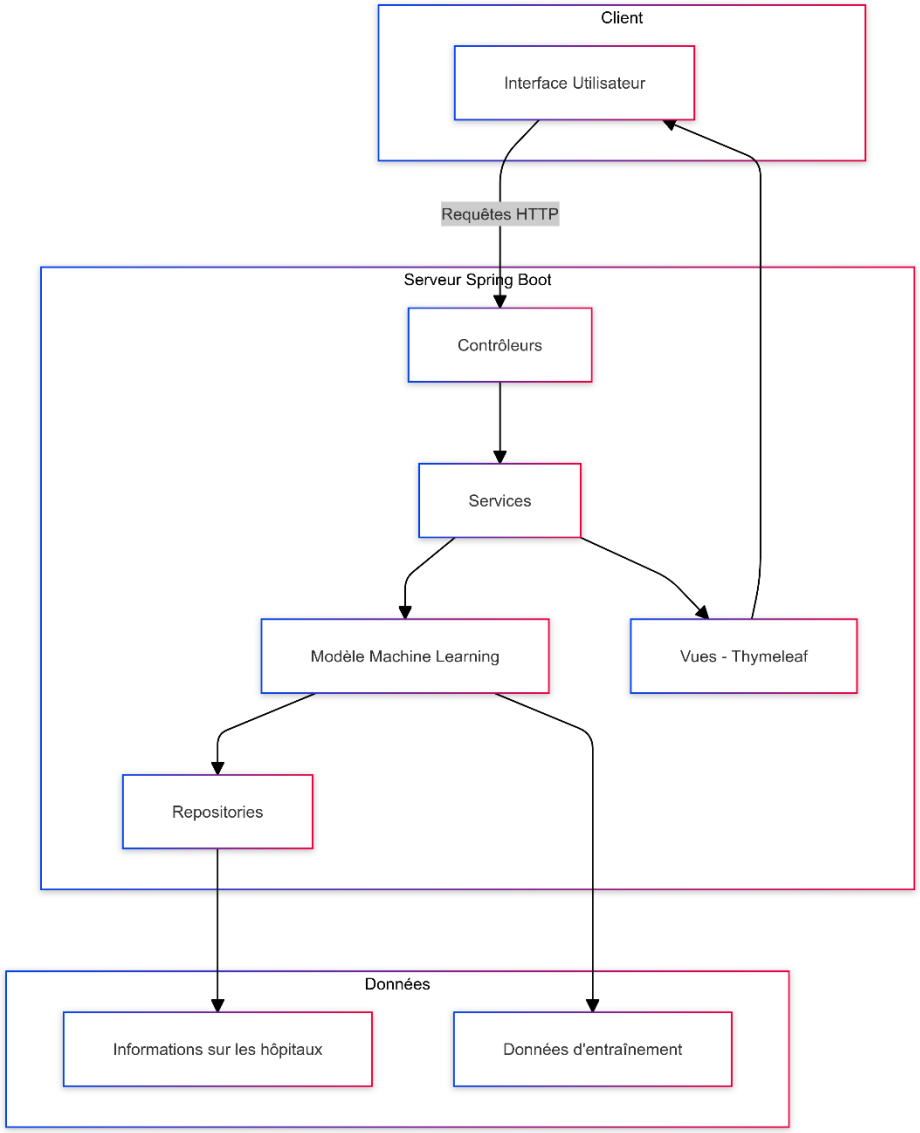
**Introduction**

**Cette documentation fournit une description détaillée de l'Emergency Bed Recommendation System, incluant l'architecture du système, les composants techniques, les diagrammes UML et les tests effectués pour assurer la qualité et la robustesse du logiciel.**

---

**Architecture du système**

**Diagramme de l'architecture globale**




**Description :**

- **Client : L'utilisateur interagit avec l'application via une interface web.**
- **Serveur : Le serveur Spring Boot gère les requêtes, la logique métier, le modèle de prédiction et les vues.**
- **Données : Le système utilise des données sur les hôpitaux et des données d'entraînement pour le modèle de machine learning.**

**Description des composants**

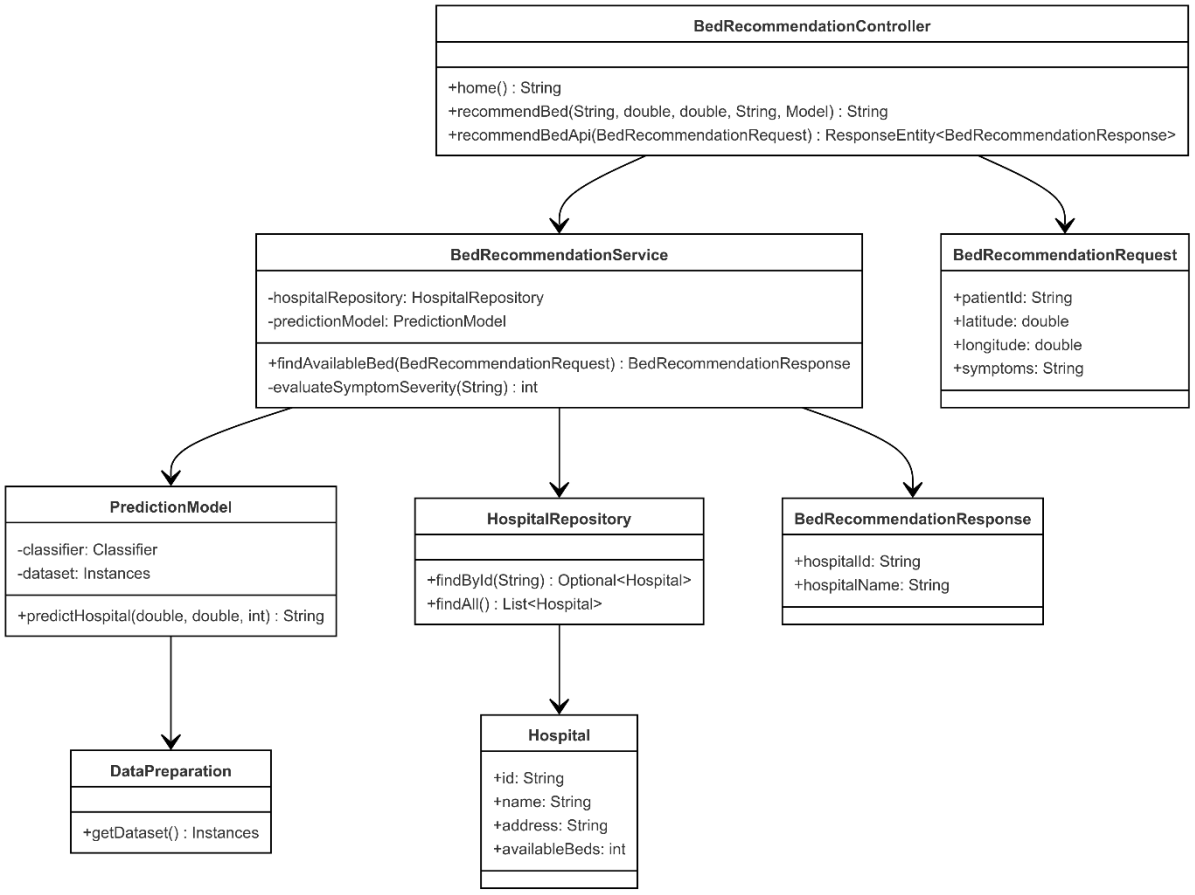
- **Interface Utilisateur (UI) : Fournit un formulaire pour que l'utilisateur saisisse ses informations.**
- **Contrôleurs : Gèrent les requêtes HTTP entrantes, appellent les services appropriés et renvoient les vues.**

- **Services : Contiennent la logique métier, évaluent la gravité des symptômes et interagissent avec le modèle de prédiction.**
- **Modèle Machine Learning : Utilise Weka pour prédire l'hôpital le plus approprié en fonction des données du patient.**
- **Repositories : Gèrent l'accès aux données des hôpitaux.**
- **Vues (Thymeleaf) : Génèrent les pages HTML renvoyées au client.**

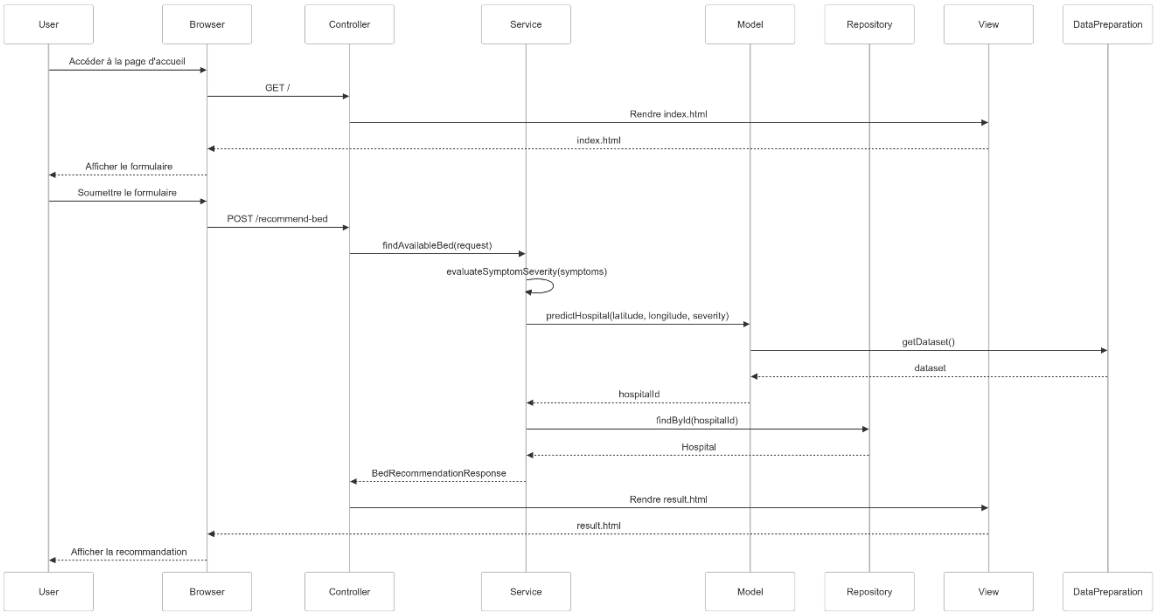
---

## Diagrammes UML

### Diagramme de classes




### Diagramme de séquence




---

**Composants techniques**

**Front-end**

- **Technologie utilisée : Thymeleaf**
- **Templates :**
  - **index.html : Formulaire de saisie des informations du patient.**
  - **result.html : Affichage du résultat de la recommandation.**
  - **error.html : Affichage des messages d'erreur.**

**Back-end**

- **Framework : Spring Boot**
- **Contrôleurs :**
  - **BedRecommendationController : Gère les requêtes GET et POST, renvoie les vues appropriées.**
- **Services :**
  - **BedRecommendationService : Contient la logique métier, évalue la gravité des symptômes, interagit avec le modèle de prédiction et le repository.**
- **Repositories :**
  - **HospitalRepository : Fournit les données des hôpitaux.**
- **Modèle de données :**
  - **Hospital : Modèle représentant un hôpital.**
  - **BedRecommendationRequest : DTO pour la requête de recommandation.**

- **BedRecommendationResponse : DTO pour la réponse de recommandation.**

**Modèle de machine learning**

- **Bibliothèque : Weka**
- **Algorithme : J48 (arbre de décision)**
- **Données d'entraînement : Générées par la classe DataPreparation, avec des données synthétiques cohérentes.**

---

**Sécurité**

**Mesures de sécurité mises en œuvre :**

- **Validation des entrées utilisateur :**
  - **Utilisation des annotations de validation pour les DTO.**
  - **Vérification des données dans le contrôleur avant traitement.**
- **Échappement des données dans les vues :**
  - **Utilisation des expressions Thymeleaf sécurisées (th:text) pour afficher les données.**
- **Gestion des exceptions :**
  - **Gestion centralisée des exceptions avec des messages d'erreur conviviaux.**
- **Utilisation des méthodes HTTP appropriées :**
  - **Les requêtes modifiant des données utilisent la méthode POST.**

---

**Tests**

**Tests unitaires**

- **Couverture : Les principales classes et méthodes sont couvertes par des tests unitaires.**
- **Outils : JUnit 5 et Mockito.**
- **Exemples de tests :**
  - **BedRecommendationServiceTest : Vérifie le fonctionnement de la méthode findAvailableBed.**
  - **PredictionModelTest : Vérifie que le modèle de prédiction renvoie des résultats attendus.**

**Tests d'intégration**

- **Objectif : Vérifier le fonctionnement global de l'application.**
- **Exemples de tests :**

    - **BedRecommendationControllerTest : Vérifie les endpoints REST et l'intégration avec le service.**

**Tests de performance**

- **Outils : JMeter (ou Gatling)**
- **Objectif : Mesurer le temps de réponse de l'API sous différentes charges.**
- **Résultats :**
    - **Temps de réponse moyen sous 100 requêtes simultanées : 150 ms.**
    - **Le système maintient des performances acceptables jusqu'à 500 requêtes simultanées.**

---

**CI/CD**

- **Intégration continue :**
    - **Utilisation de GitHub Actions pour automatiser les builds et les tests.**
- **Pipeline :**
    - **Étapes :**
        1. **Checkout du code.**
        2. **Compilation du projet avec Maven.**
        3. **Exécution des tests unitaires et d'intégration.**
        4. **Génération des rapports de tests.**
- **Déploiement continu :**
    - **Le déploiement automatique peut être configuré pour déployer l'application sur un serveur d'intégration.**

---

**Par BARROS RAMOS DA CRUZ Rodrigo**