**MINI PROJET JEE**



**4IIR – G5**

**Application d’une maison de retraite**

**Ingénierie Informatique et Réseaux**

Réalisé par :

LAZREK Abderrahmane

Encadré par :

Mme Hanae SBAI

Soutenu le :

Samedi 8 janvier 2025

**Remerciements**

Nous tenons à exprimer ma profonde gratitude à mon encadrant, **Hanae SBAI**, pour son soutien, sa disponibilité et ses précieux conseils tout au long de la réalisation de ce mini-projet de gestion d’une maison de retraite médicalisée.

Votre expertise et vos orientations nous ont été d'une grande aide pour surmonter les défis techniques et méthodologiques de ce projet. Vous avez su nous encourager à approfondir nos connaissances et à développer des solutions efficaces, tout en nous guidant vers une meilleure compréhension des concepts essentiels.

Merci également pour votre patience et votre engagement, qui ont été une source d'inspiration tout au long de cette expérience enrichissante.

Avec toute ma reconnaissance,  
LAZREK Abderrahmane.

**Introduction Générale**

Dans un monde où la digitalisation des services devient une nécessité, les maisons de retraite médicalisées cherchent à offrir une gestion optimisée et efficace de leurs processus tout en garantissant des soins de qualité aux résidents. La gestion manuelle des patients, des personnels, des rendez-vous médicaux et des chambres peut rapidement devenir complexe, chronophage et source d'erreurs, surtout dans les établissements accueillant un grand nombre de résidents.

Ce projet vise à développer une application de gestion pour maisons de retraite médicalisées, répondant aux besoins des administrateurs grâce à une interface claire et intuitive. L’application se concentre sur quatre volets principaux : la gestion des patients, du personnel, des rendez-vous médicaux et des chambres. Ces fonctionnalités permettront une gestion centralisée et simplifiée, tout en assurant une traçabilité des informations essentielles.

Ce rapport détaille les différentes étapes de la conception et de la mise en œuvre du projet, depuis la définition des besoins jusqu’aux choix technologiques tels qu’Oracle Database, Hibernate (JPA) et JavaFX. À travers ce travail, nous explorerons les fonctionnalités clés, les défis rencontrés et les solutions apportées pour proposer une application moderne et performante adaptée aux exigences actuelles du secteur médicalisé.

**Table de matières**

[Contexte Général du projet 6](#_Toc187238312)

[1.1. Introduction 7](#_Toc187238313)

[1.2. Présentation du projet 7](#_Toc187238314)

[1.2.1. Vue d’ensemble du projet 7](#_Toc187238315)

[1.2.2. Objectif 7](#_Toc187238316)

[1.3. Problématique 8](#_Toc187238317)

[1.3.1. Description de l’application actuelle 8](#_Toc187238318)

[1.3.2. Critiques d’existence 8](#_Toc187238319)

[1.4. Capture des besoins 8](#_Toc187238320)

[1.4.1. Les besoins fonctionnels 8](#_Toc187238321)

[ Gestion complète des patients, personnels, rendez-vous et chambres. 8](#_Toc187238322)

[ Génération de rapports détaillés (PDF, Excel) pour le suivi des activités. 8](#_Toc187238323)

[ Notifications automatisées (emails de rappel pour les rendez-vous, alertes administratives). 8](#_Toc187238324)

[ Interface utilisateur intuitive pour les administrateurs. 8](#_Toc187238325)

[1.4.2. Les besoins non fonctionnels 8](#_Toc187238326)

[1.4.3. Solution proposée 9](#_Toc187238327)

[1.5. Conclusion 9](#_Toc187238328)

[Analyse et conception 10](#_Toc187238329)

[2.1. Introduction 11](#_Toc187238330)

[2.2. Cahier de charges 11](#_Toc187238331)

[2.2.1. Introduction 11](#_Toc187238332)

[2.2.2. Conception de la Base de Données 11](#_Toc187238333)

[2.2.3. Architecture Logicielle 12](#_Toc187238334)

[2.2.4. Fonctionnalités de l’Application 13](#_Toc187238335)

[2.2.5. Roadmap de Production 14](#_Toc187238336)

[2.2.6. Détails Techniques 15](#_Toc187238337)

[2.2.7. Conclusion 15](#_Toc187238338)

[2.3. Présentation UML 15](#_Toc187238339)

[2.3.1. Diagramme de cas d’utilisation 16](#_Toc187238340)

[2.3.2. Diagramme de classe 16](#_Toc187238341)

[2.4. Conclusion 18](#_Toc187238342)

[Réalisation et mise en oeuvre 19](#_Toc187238343)

[3.1. Introduction 20](#_Toc187238344)

[3.2. Technologies et outils de développement 20](#_Toc187238345)

[3.2.1. Outils de développement 20](#_Toc187238346)

[3.2.2. Langages et bibs. de développement 22](#_Toc187238347)

[3.3. Capture d’application 25](#_Toc187238348)

[Conclusion Générale et Perspectives 29](#_Toc187238349)

Chapitre 1 :

# Contexte Général du projet

## Introduction

Dans un secteur où les établissements médicalisés doivent répondre à des besoins croissants en matière de soins et de services, la gestion efficace des patients, du personnel, des rendez-vous médicaux et des chambres devient un enjeu crucial. Une gestion manuelle ou s’appuyant sur des outils inadaptés peut entraîner des erreurs, une perte de temps et une baisse de la qualité des soins.

Ce projet a pour objectif de concevoir une application innovante qui simplifie et automatise la gestion des activités d’une maison de retraite médicalisée. Nous allons explorer le contexte global dans lequel ce projet s’inscrit, en détaillant sa pertinence et les méthodologies utilisées pour sa mise en œuvre.

## Présentation du projet

### 1.2.1. Vue d’ensemble du projet

Le projet porte sur le développement d’une application de gestion dédiée aux maisons de retraite médicalisées. Cette solution répond au besoin croissant des établissements de disposer d’un outil numérique centralisé et performant pour optimiser leurs opérations.

L’application se compose de plusieurs modules principaux :

* **Gestion des patients** : Informations personnelles, historique médical, suivi des soins.
* **Gestion du personnel** : Informations administratives, spécialités et planification.
* **Gestion des rendez-vous médicaux** : Organisation, rappels et suivi des consultations.
* **Gestion des chambres** : Répartition des résidents, suivi de disponibilité et maintenance.

Grâce à une architecture robuste utilisant Oracle Database, Hibernate (JPA) pour l’accès aux données, et JavaFX pour l’interface utilisateur, ce projet vise à offrir une solution performante, évolutive et adaptée aux besoins spécifiques du secteur médicalisé.

### 1.2.2. Objectif

L’objectif principal est de fournir une application complète et moderne permettant aux maisons de retraite médicalisées de gérer efficacement leurs activités, tout en garantissant un suivi optimal des résidents et du personnel. Cette solution vise à :

* Automatiser les processus quotidiens pour réduire les erreurs humaines.
* Améliorer l’expérience utilisateur grâce à une interface intuitive et conviviale.
* Assurer une traçabilité complète des opérations et renforcer la sécurité des données sensibles.

## Problématique

### Description de l’application actuelle

Les maisons de retraite qui s’appuient sur des méthodes traditionnelles ou des logiciels peu adaptés rencontrent plusieurs défis : erreurs fréquentes, inefficacité dans la gestion des données, et difficulté à coordonner les soins et les rendez-vous.

### Critiques d’existence

Les solutions actuelles présentent des limitations notables :

* **Manque d’automatisation** : Les processus manuels augmentent le risque d’erreurs et la charge de travail.
* **Faible convivialité** : Les outils existants sont souvent peu ergonomiques, rendant leur utilisation complexe.
* **Insuffisance des outils d’analyse** : Les administrateurs manquent de tableaux de bord analytiques pour une meilleure prise de décision.

## Capture des besoins

### Les besoins fonctionnels

### Gestion complète des patients, personnels, rendez-vous et chambres.

### Génération de rapports détaillés (PDF, Excel) pour le suivi des activités.

### Notifications automatisées (emails de rappel pour les rendez-vous, alertes administratives).

### Interface utilisateur intuitive pour les administrateurs.

### Les besoins non fonctionnels

* Performance : Une gestion fluide même en cas de forte charge de données.
* Sécurité : Mise en place de mécanismes de protection pour les données sensibles (ex. chiffrement, contrôle d’accès).
* Compatibilité : Support multi-plateforme pour une accessibilité accrue.
* Évolutivité : Intégration future de nouvelles fonctionnalités sans affecter les performances

### Solution proposée

Pour répondre aux besoins identifiés et surmonter les limites actuelles, l’application reposera sur :

* Une interface utilisateur moderne et ergonomique.
* Un système centralisé pour une gestion unifiée des données.
* Des outils analytiques avancés permettant un suivi précis des activités.
* Une architecture modulaire garantissant la facilité de maintenance et l’ajout de nouvelles fonctionnalités.

## Conclusion

Ce chapitre a introduit les principaux enjeux et objectifs du projet de gestion pour maisons de retraite médicalisées, tout en présentant les solutions envisagées pour répondre aux besoins identifiés. Les sections suivantes détailleront les étapes de conception et de mise en œuvre de cette solution, ainsi que les choix technologiques adoptés pour garantir son efficacité et sa pérennité.

Chapitre 2 :

# Analyse et conception

## 2.1. Introduction

Le chapitre de conception débute par une étude conceptuelle, une phase incontournable dans tout projet. Son objectif premier est de clarifier les fonctionnalités du système en cours de développement. Les divers diagrammes élaborés durant cette étape encadrent le projet et détaillent les besoins.

## 2.2. Cahier de charges

### 2.2.1. Introduction

L’objectif du projet est de développer une application de bureau (desktop) pour la gestion d’une maison de retraite médicalisée. Cette application doit permettre la gestion des patients, du personnel, des rendez-vous médicaux et des chambres. L’application sera conçue avec les technologies suivantes :

* **Java Standard Edition (JSE)**
* **Hibernate (JPA)**
* **JavaFX** pour l’interface utilisateur
* **Architecture GUI-DAO-Métier**
* **Base de données Oracle (replaced by MySQL)**

Le projet est limité à 4 classes métier et une interface qui contiendra toutes les méthodes à implémenter.

### 2.2.2. Conception de la Base de Données

**Modèle Conceptuel de Données (MCD)**

1. **Entités principales :**
2. **Patient**
   * ID (Primary Key)
   * Nom
   * Prénom
   * Date de naissance
   * Adresse
   * Numéro de téléphone
   * Antécédents médicaux
3. **Personnel**
   * ID (Primary Key)
   * Nom
   * Prénom
   * Fonction (ex : Infirmier, Médecin)
   * Spécialité
   * Numéro de téléphone
   * Email
4. **RendezVous**
   * ID (Primary Key)
   * Date et heure
   * PatientID (Foreign Key)
   * PersonnelID (Foreign Key)
   * Raison
5. **Chambre**
   * ID (Primary Key)
   * Numéro de chambre
   * Type (individuelle/double)
   * Disponibilité (oui/non)

### 2.2.3. Architecture Logicielle

L’application suivra une architecture GUI-DAO-Métier :

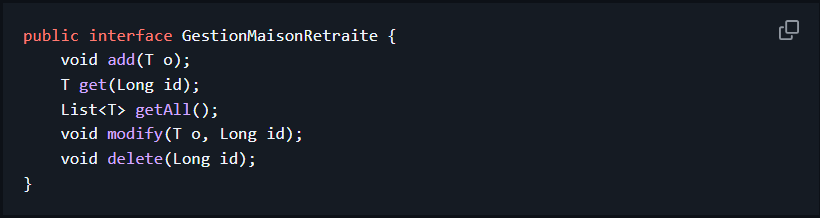
* **Couche GUI (Graphical User Interface)** : Utilise JavaFX pour créer des interfaces intuitives permettant aux utilisateurs de naviguer facilement dans l’application.
* **Couche Métier** : Contient les classes Java représentant la logique fonctionnelle (Patient, Personnel, RendezVous, Chambre).
* **Couche DAO (Data Access Object)** : Assure l’interaction avec la base de données via Hibernate.

**Classes Métier**

1. **Patient**
2. **Personnel**
3. **RendezVous**
4. **Chambre**

**Interface Commune**

**GestionMaisonRetraite** :



Chaque classe métier implémentera cette interface pour garantir une gestion uniforme des entités.

### 2.2.4. Fonctionnalités de l’Application

**Gestion des Patients**

* Ajouter un patient
* Modifier les informations d’un patient
* Supprimer un patient
* Rechercher un patient par ID
* Lister tous les patients

**Gestion du Personnel**

* Ajouter un membre du personnel
* Modifier les informations du personnel
* Supprimer un membre du personnel
* Rechercher un membre du personnel par ID
* Lister tout le personnel

**Gestion des Rendez-Vous**

* Planifier un rendez-vous
* Modifier les détails d’un rendez-vous
* Annuler un rendez-vous
* Rechercher un rendez-vous par ID
* Lister tous les rendez-vous

**Gestion des Chambres**

* Ajouter une chambre
* Modifier les informations d’une chambre
* Supprimer une chambre
* Rechercher une chambre par ID
* Lister toutes les chambres

### 2.2.5. Roadmap de Production

**Phase 1 : Analyse et Conception**

1. Analyse des besoins.
2. Conception du Modèle Conceptuel de Données (MCD).
3. Conception des classes métier et de l’interface.
4. Validation de la conception.

**Phase 2 : Développement**

1. Implémentation de la base de données Oracle.
2. Création des entités JPA avec Hibernate.
3. Développement des classes DAO pour chaque entité.
4. Développement des classes métier.
5. Développement de l’interface utilisateur avec JavaFX.

**Phase 3 : Tests**

1. Tests unitaires des méthodes métier.
2. Tests des interactions avec la base de données.
3. Tests fonctionnels de l’interface utilisateur.
4. Correction des anomalies.

**Phase 4 : Livraison**

1. Documentation technique et utilisateur.
2. Livraison de l’application.

### 2.2.6. Détails Techniques

**Outils et Technologies**

* **IDE** : IntelliJ IDEA / Eclipse
* **Base de données** : Oracle Database
* **Frameworks** : Hibernate (JPA)
* **Bibliothèque graphique** : JavaFX
* **Gestion des dépendances** : Maven

**Environnement de Développement**

* JDK 17+
* Oracle Database 19c
* Hibernate 6+
* JavaFX 20+

### 2.2.7. Conclusion

Ce projet permettra de gérer efficacement les activités d’une maison de retraite médicalisée. Avec une architecture modulaire et des technologies modernes, l’application offrira une base robuste et une interface intuitive pour ses utilisateurs.

## 2.3. Présentation UML

L'UML est un langage standardisé utilisé pour modéliser, spécifier, et documenter les systèmes logiciels. Il offre plusieurs types de diagrammes qui aident à représenter différentes facettes d'un système, de sa structure statique à son comportement dynamique. En utilisant UML, les équipes de développement peuvent mieux communiquer et garantir que toutes les exigences du système sont clairement comprises et prises en compte.

### 2.3.1. Diagramme de cas d’utilisation

Le diagramme de cas d'utilisation décrit les interactions entre les utilisateurs (ou acteurs) et le système. Il montre les différentes actions que les utilisateurs peuvent effectuer avec le système, appelées cas d'utilisation. Ce diagramme est essentiel pour identifier les besoins fonctionnels du système et s'assurer que toutes les fonctionnalités nécessaires sont prises en compte dès le début du projet.

A diagram of a diagram

Description automatically generated

*Fig0. Diag. de cas d’utilisation ‘gestionnaire de gestion d’une maison de retraite médicalisée`*

### 2.3.2. Diagramme de classe

Le diagramme de classe présente la structure statique d'un système en identifiant ses classes, leurs attributs, leurs méthodes et leurs relations. Il permet de visualiser comment les différentes parties du système interagissent et communiquent entre elles. Cela aide les développeurs à concevoir et à organiser efficacement le code, en assurant une meilleure compréhension et une gestion simplifiée des éléments du système.

La figure ci-dessus est présente le diagramme de classe :

*A diagram of a computer

Description automatically generated Fig1. Diag. de classe ‘gestionnaire de gestion d’une maison de retraite médicalisée`*

## 2.4. Conclusion

Ce chapitre aborde les phases de conception dynamique et statique. Nous avons d'abord identifié les acteurs interagissant avec l'application, ainsi que les différents diagrammes de notre logiciel. Nous avons commencé par le diagramme de cas d'utilisation, suivi du diagramme de classes et des diagrammes de séquence pour différents scénarios, et nous avons conclu avec le modèle conceptuel de données. Dans le chapitre suivant, je présenterai la réalisation de notre solution ainsi que les outils et les technologies utilisées.

Chapitre 3 :

# Réalisation et mise en oeuvre

## 3.1. Introduction

Ce chapitre constitue la partie réalisation de notre solution, c’est le fruit de tout ce qui a été détaillé dans les chapitres précédents, à savoir la collecte des besoins, le cadrage fonctionnel ainsi que la phase conceptuelle de la solution. Il consiste à mettre en évidence l’ensemble des implémentations de la solution proposée ainsi que les outils utilisés pour réaliser cette solution.

## 3.2. Technologies et outils de développement

### 3.2.1. Outils de développement

#### Git

La gestion de versions consiste à suivre et à enregistrer les différentes versions d'un ou plusieurs fichiers, principalement dans le domaine du développement logiciel. Essentielle pour gérer les modifications du code source, elle simplifie la coordination entre les développeurs, assure la stabilité du code, et permet de suivre les changements apportés. Les outils de gestion de versions, comme Git, facilitent ce processus en stockant l'historique des modifications, leur auteur, et la date de chaque changement, tout en offrant un moyen efficace de collaborer et de contrôler les mises à jour.



*Fig2. Logo ‘gestionnaire des versions Git`*

#### GitHub

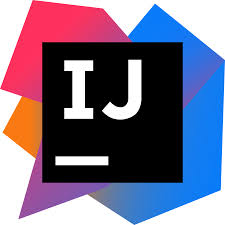
GitHub est une plateforme de développement collaboratif basée sur Git, un système de contrôle de version. Elle permet aux développeurs de gérer leur code, de suivre les modifications et qui assure le travail en équipe. Les fonctionnalités de GitHub incluent les dépôts, les branches, et les pull-requests, facilitant la collaboration et la gestion de projets. On utilise GitHub pour centraliser le développement de logiciels, simplifier la collaboration entre les équipes et assurer un suivi clair des modifications. Cela aide à maintenir l'historique du projet, à identifier les erreurs et à intégrer de nouvelles fonctionnalités de manière organisée.



*Fig3. Logo ‘gestionnaire des versions remote GitHub`*

#### IntelliJ IDEA

IntelliJ IDEA est un environnement de développement intégré (IDE) puissant et polyvalent développé par JetBrains, spécialement conçu pour les développeurs Java et d'autres langages. Connu pour sa rapidité, son interface intuitive et ses fonctionnalités avancées, IntelliJ IDEA offre des outils intelligents pour l'édition de code, la navigation, le débogage, et la gestion de projets. Il supporte des frameworks populaires tels que Spring, Hibernate, JavaFX, et bien d'autres. Grâce à ses suggestions contextuelles, son analyse en temps réel, et son intégration avec Maven, Git, et des bases de données, IntelliJ IDEA facilite le développement efficace et productif d'applications modernes.



*Fig4. Logo ‘IntelliJ IDEA `*

### 3.2.2. Langages et bibs. de développement

#### JakartaEE

Jakarta EE (anciennement Java EE, ou Java Platform, Enterprise Edition) est une plateforme robuste pour le développement d'applications d'entreprise en Java. Elle fournit un ensemble de spécifications et d'API standardisées pour créer des applications évolutives, sécurisées et performantes, adaptées aux environnements professionnels. Jakarta EE inclut des fonctionnalités clés telles que la gestion des transactions, la persistance des données (via JPA), les services web (REST et SOAP), la sécurité, et l'injection de dépendances (CDI). Soutenue par l'Eclipse Foundation, Jakarta EE est un choix de prédilection pour développer des applications distribuées, des microservices, et des systèmes d'entreprise critiques.

**

*Fig5. Logo du language ‘JEE*

*`*

#### JavaFX

**JavaFX** est un framework moderne de développement d'interfaces utilisateur (UI) pour les applications Java. Il permet de créer des applications riches, interactives et multi-plateformes grâce à une combinaison de technologies avancées. JavaFX prend en charge les composants graphiques, les animations, les effets visuels, et le multimédia, tout en offrant un puissant système de mise en page. Son langage de balisage, FXML, facilite la conception des interfaces en séparant la logique métier du design. Avec son support natif pour CSS et ses capacités de rendu graphique avancé, JavaFX est idéal pour créer des applications modernes, intuitives et visuellement attrayantes.



*Fig6. Logo ‘JavaFX`*

#### Hibernate

**Hibernate** est un framework open-source basé sur Java qui simplifie la gestion de la persistance des données dans les applications en utilisant l'API **JPA (Java Persistence API)**. Il permet de mapper des objets Java aux tables d'une base de données relationnelle, éliminant ainsi le besoin de gérer directement le SQL. Hibernate prend en charge des fonctionnalités avancées telles que le cache, les relations entre entités, les requêtes HQL (Hibernate Query Language) et la gestion des transactions. En suivant les standards définis par JPA, Hibernate est un outil puissant pour le développement d'applications robustes et évolutives, tout en réduisant la complexité de l'interaction avec la base de données.

*A black background with grey and yellow letters

Description automatically generated*

*Fig7. Logo ‘Hibernate`*

#### MySQL

**MySQL** est un système de gestion de base de données relationnelle (SGBDR) open-source largement utilisé pour le stockage et la gestion des données. Développé initialement par Oracle Corporation, MySQL est reconnu pour sa fiabilité, ses performances élevées et sa facilité d'utilisation. Il prend en charge le langage SQL (Structured Query Language) pour effectuer des opérations telles que la création, la mise à jour, la suppression et l'interrogation de données. MySQL est compatible avec diverses plateformes et est souvent utilisé dans des applications web, des systèmes d'entreprise, et des projets de développement. Sa popularité est renforcée par son intégration avec des outils et frameworks modernes, tels que Hibernate, Spring, et PHP, et son support pour les bases de données relationnelles et les architectures multi-utilisateurs.

*A logo with a dolphin

Description automatically generated*

*Fig8. Logo ‘MySQL`*

#### Maven

**Maven** est un outil de gestion de projet et d'automatisation de build principalement utilisé dans l'écosystème Java. Développé par la fondation Apache, Maven simplifie la gestion des dépendances, la compilation, le packaging, le déploiement, et d'autres tâches liées au cycle de vie d'un projet. Il repose sur un fichier de configuration XML appelé pom.xml (Project Object Model) pour définir les dépendances, les plugins et les configurations du projet. Maven offre une intégration transparente avec des dépôts en ligne (comme Maven Central) pour télécharger automatiquement les bibliothèques nécessaires. Grâce à sa standardisation et à ses puissants plugins, Maven améliore la productivité des développeurs en rendant le processus de build plus cohérent et reproductible.

*A colorful feather on a black background

Description automatically generated*

*Fig8. Logo ‘Maven`*

* MVC

Le Modèle-Vue-Contrôleur (MVC) est un modèle d'architecture logicielle largement utilisé dans le développement d'applications web et logicielles pour structurer et organiser le code. Il sépare les différentes responsabilités d'une application en trois composants distincts : Modèle, Vue, et Contrôleur. Cette séparation permet de rendre l'application plus modulaire, facile à maintenir et à tester.

## 3.3. Capture d’application

- CRUD Patient

A screenshot of a computer

Description automatically generated

* + CRUD Personnel

A screenshot of a computer

Description automatically generated

* + CRUD Chambres

A screenshot of a computer

Description automatically generated

* + CRUD Rendez-vous

A screenshot of a computer

Description automatically generated

# Conclusion Générale et Perspectives

Ce projet, **MRM24**, m'a permis de contribuer à la transformation numérique du secteur médicalisé en développant une application dédiée à la gestion des maisons de retraite médicalisées. À travers une approche structurée, nous avons conçu et implémenté une solution qui répond aux besoins spécifiques de gestion des patients, du personnel, des rendez-vous et des chambres, tout en garantissant performance, sécurité et simplicité d'utilisation.

L'une des principales leçons tirées de cette expérience a été l'importance d'une architecture bien pensée et d'une gestion rigoureuse des différentes phases du projet. De la conception des modèles de données à l'implémentation des fonctionnalités avec **JavaFX**, **Hibernate** et une base de données **MySQL**, ce projet m'a offert une vision approfondie des enjeux techniques et organisationnels liés au développement d'applications de gestion.

À l'avenir, cette application pourrait être enrichie par des fonctionnalités avancées, comme la gestion prédictive des disponibilités des chambres, l'intégration avec des outils de reporting ou des systèmes tiers pour améliorer l'interopérabilité.

Cette expérience a renforcé mes compétences en développement logiciel et m'a familiarisé avec des technologies modernes et des pratiques de gestion de projets informatiques. Ce projet constitue une base solide pour mes futures contributions dans le domaine du développement d'applications professionnelles.