المدرسة العليا للتكنولوجيا الناظور
École Supérieure de Technologie de Nador
ⵜⵉⵎⵙⵙⵓⵔⵉⵜ ⵜⴰⵣⴰⵔⵉⵜ ⵏ ⵜⵓⵜⵣⴽⵉⵏⵓⵍⵓⵊⵢⵜ ⵅⵓⴹⵓⵔ

جامعة محمد الأول بوجدة
UNIVERSITÉ MOHAMMED PREMIER OUJDA
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵎⵓⵃⴰⵎⵎⴷ ⴰⵣⵔⴼⴰⵏ ⵏ ⵡⵊⴷⴷⴰ

Diplôme Universitaire de Technologie (DUT)

Ingénierie Logicielle et Cybersécurité (ILCS)

Rapport de projet

Thème :

Développement d'une application web pour la gestion d'une clinique vétérinaire

Réalisé par :

MHARI Ilham

JAAI Malak

Encadré par :

Pr. ESBAI Redouane

Soutenu le 19 Mai 2025, sous la direction de :

Pr. ESBAI Redouane École Supérieure de Technologie de Nador (ESTN)

Année universitaire : 2024 - 2025

Remerciements

Nous tenons à exprimer notre profonde gratitude au Pr. ESBAI Redouane, professeur de la filière Ingénierie Logicielle et Cybersécurité (ILCS) à l'École Supérieure de Technologie de Nador (ESTN). Son soutien constant, ses recommandations avisées et ses orientations précieuses ont été indispensables à la réussite de ce projet. Sa disponibilité et son engagement ont non seulement facilité la surmontée des défis rencontrés, mais ont également enrichi considérablement la qualité du travail accompli. Nous lui sommes sincèrement reconnaissantes pour la confiance qu'il a placée en nous en nous proposant ce sujet, ainsi que pour sa précieuse contribution et son rôle déterminant tout au long de cette expérience académique.

Résumé

Ce rapport présente un projet portant sur le développement de **VetCare 360**, une application desktop complète basée sur JavaFX pour la gestion intégrée d'une clinique vétérinaire, à travers la gestion des propriétaires d'animaux, des vétérinaires, des animaux eux-mêmes et des visites médicales.

Table des matières

Remerciements **2**

Résumé **3**

Introduction générale **6**

I Introduction **7**

1 Contexte et motivations 7

2 Présentation du projet 7

3 Objectifs du projet 8

II Architecture et conception **10**

1 Architecture MVC 10

2 Données en mémoire 11

3 Diagramme de classes 12

III Développement et Implémentation **14**

1 Environnement de développement 14

1.1 Environnement de développement intégré (IDE) 14

1.2 Outils de gestion de projet 14

2 Technologies et outils de développement 15

2.1 Langages de balisage et de style 15

2.2 Langages de programmation 15

2.3 Frameworks & Bibliothèques 15

3 Présentation de l'application 16

IV Comparaison entre MERN et Java/JavaFX **18**

V Conclusion **20**

1 Récapitulation des objectifs atteints 20

2 Perspectives d'améliorations futures 20

Conclusion générale **22**

Références bibliographiques **22**

Introduction générale

Dans un monde de plus en plus tourné vers la transformation numérique, la plupart des secteurs d'activité, y compris celui de la médecine vétérinaire, doivent repenser leurs méthodes de gestion en intégrant des outils technologiques performants. Les cliniques vétérinaires ne font pas exception à cette tendance : elles sont aujourd'hui confrontées au défi de moderniser leurs processus administratifs et médicaux afin d'améliorer leur efficacité, renforcer la traçabilité des soins, et offrir un service de meilleure qualité aux propriétaires d'animaux.

C'est dans ce contexte que s'inscrit le projet **VetCare360**, une application desktop conçue pour répondre aux besoins concrets de gestion quotidienne d'une clinique vétérinaire. En centralisant les informations relatives aux propriétaires, aux animaux, aux visites médicales et au personnel soignant, cette solution logicielle vise à simplifier les tâches répétitives, à minimiser les risques d'erreurs et à améliorer l'expérience utilisateur, aussi bien pour les employés que pour les clients.

Au-delà de son objectif fonctionnel, ce projet s'inscrit dans une démarche pédagogique ambitieuse. Il a pour but de mettre en pratique les connaissances acquises dans les domaines du développement d'applications desktop, de l'architecture logicielle et de la gestion de projet. À travers cette initiative, nous souhaitons non seulement proposer une solution technique adaptée à un besoin réel, mais aussi approfondir notre compréhension des choix technologiques, des modèles de conception et des architectures modernes utilisées dans le développement d'applications Java. Ce travail encourage ainsi une réflexion critique et structurée sur les bonnes pratiques de l'ingénierie logicielle, dans le cadre d'une approche réaliste et professionnelle.

Voici votre texte adapté pour JavaFX, en conservant strictement la même structure et le même fond :

I. Introduction

1. Contexte et motivations

Aujourd'hui encore, de nombreuses cliniques vétérinaires continuent d'utiliser des méthodes manuelles traditionnelles, telles que les dossiers papier ou les tableurs Excel, pour gérer les dossiers médicaux des animaux, les visites médicales, ainsi que les informations relatives aux vétérinaires et aux propriétaires.

Ces pratiques, bien qu'encore répandues, présentent de nombreuses limitations et posent plusieurs défis, notamment un risque élevé d'erreurs, des difficultés de traçabilité, une perte potentielle de données et un manque d'efficacité organisationnelle dans la gestion globale du cabinet.

Face à ces défis, nous avons conçu **VetCare360**, une application desktop moderne et intuitive, destinée à centraliser et automatiser les différentes tâches administratives et médicales au sein d'une clinique vétérinaire.

Au-delà de sa vocation fonctionnelle, ce projet constitue également une opportunité concrète de mettre en œuvre nos compétences acquises dans le développement d'applications Java, l'architecture logicielle, ainsi que la gestion de projet.

2. Présentation du projet

L'application **VetCare360** est une solution logicielle conçue pour la gestion d'une clinique vétérinaire. Son développement s'inspire du projet open source bien connu PetClinic Community, souvent utilisé dans un cadre pédagogique pour illustrer les bonnes pratiques de développement avec Java et Spring Boot. Ce dernier propose des fonctionnalités classiques telles que la gestion des propriétaires d'animaux, le suivi des visites médicales, ainsi que l'organisation des données des vétérinaires.

Cependant, **VetCare360** se distingue principalement par ses choix technologiques et architecturaux. Contrairement à PetClinic, qui repose sur une architecture web classique,

VetCare360 est entièrement développé avec JavaFX. Ce choix permet de proposer une solution desktop performante, avec une interface riche et réactive, tout en bénéficiant de la robustesse de la plateforme Java.

Ainsi, **VetCare360** ne vise pas à introduire de nouvelles fonctionnalités, mais plutôt à réadapter techniquement un concept éprouvé, en le modernisant et en l'adaptant aux besoins spécifiques des cliniques vétérinaires.

3. Objectifs du projet

Le projet **VetCare360** vise à atteindre plusieurs objectifs, articulés autour de trois axes principaux : fonctionnel, technique et pédagogique.

Objectifs fonctionnels

L'application vise à simplifier la gestion quotidienne d'une clinique vétérinaire grâce à une interface intuitive et conviviale. Les principales fonctionnalités attendues sont les suivantes :

- Permettre l'enregistrement des propriétaires et de leurs animaux.
- Gérer les visites médicales, avec la possibilité de consulter leur historique.
- Faciliter la recherche rapide d'un propriétaire par son nom de famille.
- Afficher dynamiquement la liste des vétérinaires depuis la base de données.

Objectifs techniques

Sur le plan technologique, **VetCare360** repose sur une architecture moderne et performante :

- Utiliser JavaFX pour le développement d'une interface desktop riche et réactive.
- Appliquer le modèle MVC (Modèle-Vue-Contrôleur) pour une meilleure organisation du code.
- Développer une architecture modulaire et maintenable avec Java 21+.
- Concevoir une interface utilisateur ergonomique avec FXML et CSS.
- Utiliser Git et GitHub pour la gestion des versions et la collaboration en équipe.

Objectifs pédagogiques

Enfin, ce projet constitue une opportunité concrète d'apprentissage et de mise en pratique :

- Approfondir nos compétences en développement d'applications Java desktop.
- Travailler en équipe sur un projet logiciel réaliste, en respectant les bonnes pratiques.
- Apprendre à structurer, organiser et documenter un projet informatique de manière professionnelle.
- Se familiariser avec les différentes étapes du cycle de développement logiciel : conception, développement, tests et validation.

II. Architecture et conception

1. Architecture MVC

Le développement de l'application VetCare 360 repose sur l'architecture MVC, qui permet de structurer l'application en trois couches distinctes :

Modèle (Model) :

Cette couche contient la logique métier et les données manipulées par l'application. Les entités principales (Propriétaire, Animal, Visite, Vétérinaire) sont représentées par des classes Java.

Les données ne sont pas stockées dans une base de données ni dans des fichiers : elles sont gérées exclusivement en mémoire à l'aide de structures comme les ArrayList ou ObservableList de JavaFX. Ces listes sont créées au lancement de l'application et alimentées à mesure que l'utilisateur interagit avec l'interface.

Vue (View) :

La vue est l'interface utilisateur, construite avec FXML pour une organisation claire et maintenable du code. Elle contient les éléments d'interface tels que les champs de saisie, les boutons, les tableaux, etc.

Le style graphique est géré avec des fichiers CSS pour une interface fluide et moderne.

Contrôleur (Controller) :

Les contrôleurs (fichiers Java liés aux vues FXML) contiennent la logique de gestion des événements utilisateurs. Lorsqu'un utilisateur clique sur un bouton ou remplit un formulaire, le contrôleur interagit avec le modèle (par exemple en ajoutant un nouvel animal à la liste) puis met à jour la vue si nécessaire.

Ce découpage permet une séparation des responsabilités, un code plus lisible, et une meilleure maintenance du projet.

2. Données en mémoire

Dans cette version de l'application **VetCare 360**, les données sont entièrement gérées en mémoire, sans recours à une base de données externe ni à un système de fichiers. Toutes les informations relatives aux propriétaires, animaux, visites et vétérinaires sont stockées temporairement à l'aide de structures telles que les ArrayList ou ObservableList de Java.

Au lancement de l'application, les listes sont initialement vides, et l'utilisateur peut les alimenter dynamiquement en ajoutant, modifiant ou supprimant des éléments via l'interface. Cependant, ces données ne sont conservées que le temps de la session : elles sont perdues dès que l'application est fermée.

Ce choix technique simplifie le développement en évitant les contraintes liées à la persistance des données, tout en permettant une simulation réaliste du fonctionnement d'une clinique vétérinaire.

Conception orientée objet

L'application est conçue selon les principes de la programmation orientée objet, ce qui signifie que chaque élément important du domaine est représenté par une classe Java.

Correspondance entre entités métier et classes

Chaque classe reflète une entité réelle : `Propriétaire`, Animal, Visite, Vétérinaire. Cela permet de modéliser précisément les données et leur comportement.

Structure des classes

- Les attributs (comme le nom, la date de naissance, etc.) sont `privés` pour protéger les données.
- Des constructeurs permettent de créer facilement des objets avec des valeurs initiales.
- Les `getters` et `setters` donnent un accès contrôlé aux attributs.
- Des méthodes utilitaires peuvent être ajoutées pour des fonctions spécifiques.

Relations entre classes

▷ Un **Propriétaire** possède plusieurs **Animaux** (liste d'objets).

▷ Un **Animal** peut avoir plusieurs **Visites** (historique des consultations).

Ces liens sont réalisés par des références entre objets, ce qui permet de naviguer facilement entre les différentes entités.

Avantage de cette modélisation

Cette approche reflète fidèlement le fonctionnement d'une clinique vétérinaire. Même sans utiliser de base de données, les données sont organisées en mémoire via les objets et leurs associations, ce qui facilite la gestion et le traitement des informations.

3. Diagramme de classes

Le modèle de l'application VetCare 360 repose sur une structure orientée objet claire, avec les classes principales suivantes :

▷ **Owner** : représente un propriétaire d'animaux, avec les attributs `id`, `name`, `phone`, et `email`.

▷ **Pet** : représente un animal, avec `id`, `name`, `birthDate`, `type`, une référence vers son `owner`, et une liste de `visits`.

▷ **Visit** : correspond à une consultation vétérinaire, avec `id`, `date`, `description`, et une référence vers le `pet` concerné.

▷ **Veterinarian** : représente un vétérinaire, avec `id`, `name` et `specialty`.

Relations entre les classes

- Un `Owner` peut posséder plusieurs `Pets` (relation 1 à plusieurs).
- Un `Pet` peut avoir plusieurs `Visits` (relation 1 à plusieurs).
- Chaque `Visit` est associée à un seul `Pet`.
- Chaque `Pet` appartient à un seul `Owner`.

Ce modèle relationnel est entièrement géré en mémoire, sans base de données, et reflète fidèlement l'organisation d'une clinique vétérinaire. Il permet une gestion fluide des entités et de leurs interactions, tout en assurant une bonne lisibilité du code et une maintenance simplifiée.

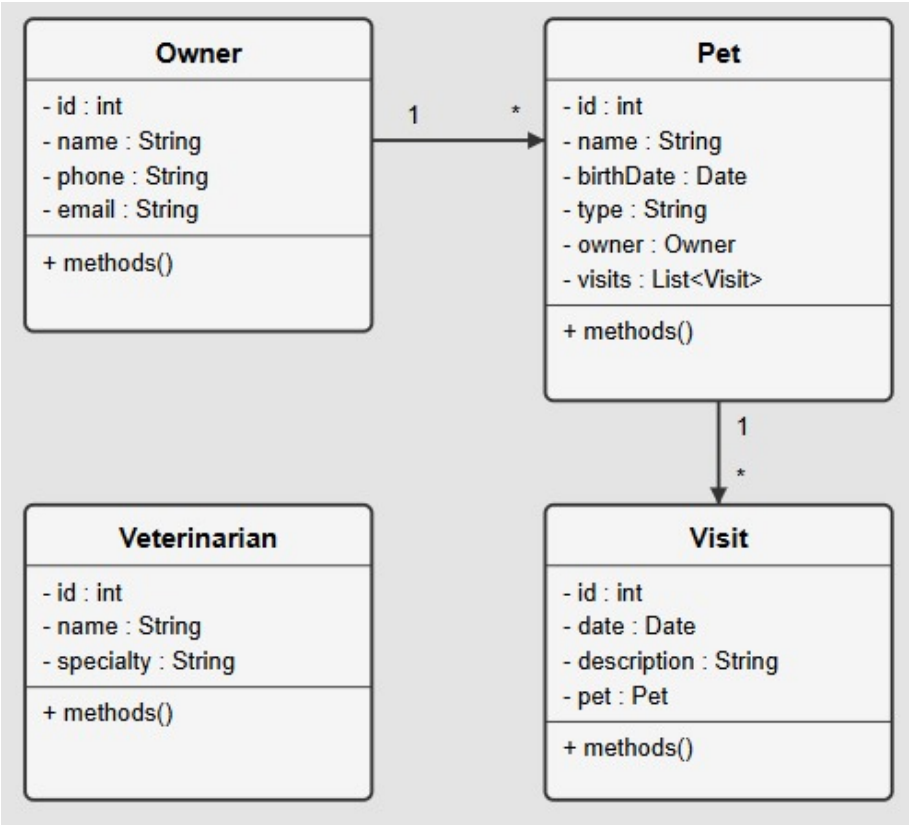

Diagramme de classes de **VetCare360**

III. Développement et Implémentation

1. Environnement de développement

1.1. Environnement de développement intégré (IDE)

Le développement de l'application **VetCare 360** a été réalisé à l'aide de IntelliJ IDEA Community Edition, un environnement de développement intégré puissant et ergonomique. Ce choix a été motivé par les nombreux avantages offerts par IntelliJ IDEA, notamment :

- Un excellent support pour Java et la programmation orientée objet, avec une assistance intelligente à la saisie du code, la vérification d'erreurs en temps réel, et des outils de refactoring efficaces.
- Une intégration native avec JavaFX, qui facilite la création et la gestion des interfaces graphiques.
- Un outil de débogage visuel performant, permettant de suivre pas à pas l'exécution du programme et de diagnostiquer rapidement les anomalies.
- La compatibilité avec Maven et Gradle (même si ces outils n'ont pas été utilisés dans ce projet), offrant une possibilité d'extension pour une future gestion automatique des dépendances.

1.2. Outils de gestion de projet

Le projet a été versionné à l'aide de Git, un système de contrôle de version distribué, et hébergé sur GitHub. Cette organisation a permis de :

- Suivre l'historique complet de développement, avec un enregistrement clair de chaque modification du code source.
- Organiser le travail par fonctionnalités, grâce à l'usage de branches séparées pour chaque tâche ou amélioration.
- Faciliter la collaboration, même si le projet a été mené individuellement, en rendant possible une extension future en équipe.

2. Technologies et outils de développement

2.1. Langages de balisage et de style

- FXML est utilisé pour définir la structure des interfaces graphiques de manière déclarative. Cela permet de séparer la présentation de la logique métier, ce qui améliore la lisibilité du code et facilite sa maintenance.
- CSS est employé pour personnaliser l'apparence des composants graphiques (boutons, champs, menus, etc.). Grâce au CSS, il est possible de modifier les couleurs, les bordures, les polices ou encore les espacements afin d'améliorer l'ergonomie et l'esthétique de l'application.

2.2. Langages de programmation

Le langage principal utilisé est **Java**, qui a servi à développer :

- La logique métier de l'application : gestion des entités (Propriétaires, Animaux, Visites, Vétérinaires), traitement des données, vérifications, etc.
- Les contrôleurs JavaFX, qui gèrent le lien entre les fichiers FXML (interface utilisateur) et les classes métier.
- La gestion des événements déclenchés par l'utilisateur (clics, saisies, sélections...).

Les principes fondamentaux de la programmation orientée objet (POO) ont été rigoureusement appliqués, notamment :

— L'encapsulation pour protéger les données.

— La modularité pour organiser le code par fonctionnalités.

— La réutilisabilité à travers des méthodes et des classes bien structurées.

2.3. Frameworks & Bibliothèques

- JavaFX est la bibliothèque principale utilisée pour le développement de l'interface graphique.
- Collections Java (List, ObservableList, etc.) ont été utilisées pour gérer temporairement les données. Aucun système de gestion de base de données n'est utilisé dans

cette version : toutes les informations sont conservées en mémoire pendant la session d'exécution.

3. Présentation de l'application

L'application **VetCare360** est une solution de gestion clinique vétérinaire, développée en Java avec JavaFX, sans base de données externe (données stockées en mémoire via `ObservableList`). L'interface est divisée en plusieurs modules fonctionnels, chacun correspondant à une tâche spécifique et accessible via un menu de navigation intuitif.

1. **Accueil**
   - ▷ Page d'introduction de l'application.
   - ▷ Affiche un message de bienvenue et les crédits des développeurs.
   - ▷ Sert de point d'entrée vers les modules fonctionnels via la barre de navigation.
2. **Recherche des propriétaires**
   - ▷ Permet de rechercher un propriétaire par son nom de famille (Last Name).
   - ▷ Les résultats sont affichés dans un tableau interactif, montrant : Nom, Adresse, Ville, Téléphone, Animaux.
   - ▷ Bouton "Add Owner" : redirige vers un formulaire d'ajout.
   - ▷ Possibilité d'accéder à :
     - "Edit Owner" pour modifier les informations personnelles.
     - "Add Pet" / "Edit Pet" pour gérer les animaux du propriétaire.
     - "Add Visit" pour gérer les visites médicales.
3. **Formulaire d'ajout / modification d'un propriétaire**
   - ▷ Champs : Nom, Prénom, Adresse, Ville, Téléphone.
   - ▷ Validation des champs avec gestion d'erreurs (ex : numéro de téléphone invalide).
4. **Détail d'un propriétaire**
   - ▷ Affiche les informations détaillées du propriétaire sélectionné.
   - ▷ Liste tous ses animaux.
   - ▷ Pour chaque animal :

- Voir / Modifier ses informations.
- Ajouter une nouvelle visite médicale.

5. **Ajout / modification d'un animal**
   - ▷ Champs : Nom, Espèce, Race, Date de naissance, Remarques médicales.
   - ▷ L'animal est automatiquement lié à son propriétaire.
6. **Liste et gestion des visites médicales**
   - ▷ Affiche l'historique des visites d'un animal.
   - ▷ Ajout d'une visite via un formulaire : Date, Motif, Vétérinaire, Diagnostic.
   - ▷ Possibilité de modifier les visites existantes.
7. **Liste des vétérinaires**
   - ▷ Affiche une liste dynamique des vétérinaires de la clinique.
   - ▷ L'utilisateur peut effectuer une recherche ou trier la liste.

Aspects techniques

- ◇ Interface graphique conçue avec JavaFX : utilisation de TableView, ListView, TextField, ComboBox, etc.
- ◇ Données stockées en mémoire grâce à `ObservableList`, ce qui permet une interaction fluide avec les composants graphiques.
- ◇ Gestion des erreurs conviviale : messages clairs, champs obligatoires, alertes en cas de saisie incorrecte.

IV. Comparaison entre MERN et Java/JavaFX

Critère	**Version MERN**	**Version Java / JavaFX**
Technologie	Full-stack JavaScript (MongoDB, Express, React, Node.js)	Application Java standalone avec interface JavaFX
Type d'application	Application web (accessible via navigateur)	Application desktop (exécutable sur poste local)
Stockage des données	Base de données MongoDB (persistante)	Données en mémoire (ObservableList) – non persistantes
Interface utilisateur	UI dynamique et responsive en React (HTML/CSS/JSX)	UI native en JavaFX (FXML ou Java pur, avec CSS)
Communication avec le backend	API REST via Express / Node.js	Pas de backend séparé – toute la logique est embarquée
Déploiement	Accessible via un serveur web (cloud, local ou distant)	Nécessite une installation locale (JDK et JavaFX Runtime)
Structure modulaire	Frontend / Backend séparés avec appels HTTP	Modules intégrés dans une seule application Java
Ajout et gestion des données	Ajout, modification, suppression via appels à l'API	Manipulation directe d'objets en mémoire
Recherche et navigation	Recherche en temps réel (via React et appels API)	Recherche locale dans des listes (ObservableList)
Gestion des erreurs	Messages d'erreurs via frontend/express-validation	Alertes JavaFX (boîtes de dialogue, labels d'erreur)
Évolutivité	Haute – peut être connectée à une vraie BDD, API externe...	Limitée – nécessite ajout de persistance ou migration serveur

Performances	Peut varier selon le backend et la connexion	Très rapide (application locale, pas d'appel réseau)
Expérience utilisateur	Interface moderne et responsive (adaptée à tous les écrans)	Interface simple, réactive mais non responsive
Utilisation de bibliothèques	React, Axios, Mongoose, Express, Bootstrap, etc.	JavaFX, collections Java, utilitaires Java natifs
Persistance des données	Oui, sauvegarde en BDD MongoDB	Non, les données disparaissent à la fermeture de l'application

Comparaison entre la version MERN et la version Java/JavaFX de l'application

Points forts de chaque version

MERN

- ▷ Architecture moderne, scalable.
- ▷ Données persistantes en base MongoDB.
- ▷ Accès multi-utilisateur via le web.
- ▷ Responsive design.

Java / JavaFX

- ▷ Simplicité de déploiement (tout-en-un).
- ▷ Très rapide en local.
- ▷ Adapté pour démos ou projets sans base de données.
- ▷ Moins de dépendances externes.

V. Conclusion

1. Récapitulation des objectifs atteints

Le projet **VetCare360** visait à concevoir une application de gestion pour une clinique vétérinaire, en mettant l'accent sur une architecture claire, une interface utilisateur intuitive et une organisation logicielle maintenable.

Voici les principaux objectifs atteints :

— Mise en place d'un modèle orienté objet complet (Propriétaires, Animaux, Visites, Vétérinaires) géré entièrement en mémoire, sans base de données.

— Création d'une interface graphique moderne et fonctionnelle avec **JavaFX** et **FXML**, respectant les principes de séparation entre présentation et logique.

— Intégration d'un système de navigation clair entre les différentes sections : accueil, gestion des propriétaires, animaux, visites, et vétérinaires.

— Implémentation de la logique métier (ajout, modification, recherche) avec une gestion des erreurs conviviale.

— Application du modèle MVC pour structurer le code, améliorant ainsi sa lisibilité, modularité et maintenance.

— Utilisation de Git et GitHub pour assurer le versionnement du projet et faciliter le suivi du développement.

2. Perspectives d'améliorations futures

Même si l'application répond aux besoins fonctionnels définis, plusieurs pistes d'amélioration peuvent être envisagées pour les versions futures :

— **Persistance des données** : Intégrer une base de données (comme SQLite, MySQL ou MongoDB) pour sauvegarder les informations entre les sessions.

— **Authentification utilisateur** : Ajouter un système de connexion sécurisé pour différents profils (secrétaire, vétérinaire, admin).

— **Gestion des rendez-vous** : Étendre la gestion des visites pour inclure la planification à l'avance et l'envoi de rappels.

— **Exportation des données** : Offrir une option pour exporter les informations sous format PDF ou Excel.

— **Tests automatisés** : Ajouter des tests unitaires et d'intégration pour améliorer la fiabilité et la robustesse de l'application.

Conclusion générale

La réalisation du projet **VetCare360** a été pour nous une expérience à la fois enrichissante sur le plan technique et humain. En partant d'un besoin concret — la gestion numérique d'une clinique vétérinaire — et en nous inspirant d'une solution existante, nous avons réussi à concevoir une application complète, moderne et conforme aux standards actuels du développement logiciel.

Ce projet nous a permis de mettre en pratique de manière concrète les connaissances acquises durant notre formation : développement orienté objet, architecture logicielle (MVC), conception d'interfaces graphiques, mais aussi collaboration en équipe, gestion de versions avec Git et résolution de problèmes techniques.

Le choix de JavaFX comme framework d'interface nous a offert l'opportunité de travailler avec un outil puissant et largement utilisé dans le développement d'applications desktop, tout en maîtrisant les différentes étapes de la création d'une application fonctionnelle et modulaire.

Au-delà d'un simple exercice académique, **VetCare360** nous a permis de mieux comprendre les exigences du métier de développeur, de renforcer nos compétences pratiques et de poser les bases d'éventuelles améliorations futures.

Références bibliographiques

[1] Oracle Corporation. (2024). *JavaFX Documentation*. https://openjfx.io/

[2] Spring Team. (2020). *Spring PetClinic Community Edition*. GitHub Repository. https://github.com/spring-projects/spring-petclinic

[3] Git Team. (2024). *Pro Git Book – Git Documentation*. https://git-scm.com/book/fr/v2

[4] GitHub, Inc. (2024). *Git Documentation*. https://git-scm.com/doc

[5] Oracle Corporation. (2024). *Introduction to FXML – JavaFX Documentation*. https://openjfx.io/javadoc/17/javafx.fxml/javafx/fxml/doc-files/introduction_to_fxml.html

[6] Oracle Corporation. (2024). *Cascading Style Sheets Reference Guide – JavaFX CSS*. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html