## 

## **Rapport de Projet : *Pawly***

## ***Système de Gestion de Clinique Vétérinaire***

|  |
| --- |
| **Réalisé par** : El Achhab Khadija & Brahimi Douae  **Encadré par** : Mr Esbai Redouane  **Établissement** : L’école Supérieure De Technologie Nador |

***Remerciement*** :

Je tiens à vous remercier sincèrement pour votre précieuse contribution à la réussite de notre projet Pawly . Votre expertise, votre engagement et votre professionnalisme ont joué un rôle essentiel dans l’atteinte de nos objectifs.

Grâce à votre implication constante et à vos conseils toujours pertinents, nous avons pu avancer sereinement et obtenir des résultats concrets. Encore un grand merci pour votre présence et votre soutien tout au long de ce parcours.

**Table des matières :**

[Rapport de Projet : Pawly 1](#_Toc1909738091)

[Système de Gestion de Clinique Vétérinaire 1](#_Toc680462396)

[1.1 Contexte du projet 2](#_Toc968226672)

[1.2 Objectifs 2](#_Toc1045928986)

[1.3 Portée du projet 3](#_Toc1535812865)

[2. Analyse des besoins 3](#_Toc1677404364)

[2.1 Besoins fonctionnels 3](#_Toc1139546578)

[2.2 Besoins non fonctionnels 4](#_Toc79487290)

[2.3 Contraintes techniques 4](#_Toc2043164577)

[3. Architecture du système 5](#_Toc337620997)

[3.1 Architecture globale 5](#_Toc987421486)

[3.2 Modèle MVC 6](#_Toc856524045)

[4.2 Diagramme de classes 6](#_Toc190129427)

[5. Implémentation 7](#_Toc1762891894)

[5.1 Technologies utilisées 7](#_Toc309951095)

[5.2 Structure du projet 8](#_Toc2036608923)

[5.3 Description des modules principaux 10](#_Toc1162481347)

[6. Interfaces utilisateur 10](#_Toc2019453772)

[6.1 Maquettes d'écrans 11](#_Toc454050733)

[6.2 Navigation entre les écrans 12](#_Toc59965675)

[6.3 Expérience utilisateur 12](#_Toc1403814804)

[7.1 Gestion des propriétaires 13](#_Toc1854279708)

[7.3 Gestion des visites 15](#_Toc1075806475)

[7.4 Gestion des vétérinaires 16](#_Toc72611564)

[8. Déploiement 16](#_Toc779027542)

[8.1 Prérequis système 16](#_Toc387968362)

[8.2 Procédure d'installation 17](#_Toc1862164559)

[8.3 Configuration 17](#_Toc1442675056)

[9. Conclusion et perspectives 17](#_Toc1639426544)

[9.1 Bilan du projet 17](#_Toc341951690)

[9.2 Difficultés rencontrées 18](#_Toc692137647)

[9.3 Améliorations futures: 18](#_Toc519420660)

**1. Introduction**

### **1.1 Contexte du projet**

Le projet Pawly est une application de gestion de clinique vétérinaire développée pour répondre aux besoins des professionnels du secteur vétérinaire. Dans un contexte où la digitalisation des services de santé animale devient essentielle, Pawly propose une solution complète pour la gestion des dossiers médicaux, des rendez-vous et du suivi des patients animaux.

### **1.2 Objectifs**

L'objectif principal du projet Pawly est de créer une application desktop intuitive et performante permettant de :

* Gérer efficacement les dossiers des propriétaires d'animaux
* Suivre les dossiers médicaux des animaux de compagnie
* Planifier et gérer les rendez-vous et visites médicales
* Administrer les informations relatives aux vétérinaires
* Améliorer la qualité des soins par un meilleur suivi des patients

### **1.3Portée du projet**

Le projet couvre le développement complet d'une application de gestion de clinique vétérinaire, depuis la conception jusqu'au déploiement. L'application est développée en Java pour le backend (programmation orientée objet) et JavaFX pour l'interface utilisateur, avec une base de données relationnelle pour le stockage des données.

## **2. Analyse des besoins**

### **2.1 Besoins fonctionnels**

Les principales fonctionnalités requises pour l'application Pawly sont :

***Gestion des propriétaires d'animaux :***

* Ajouter un nouveau propriétaire
* Modifier les informations d'un propriétaire existant
* Supprimer un propriétaire
* Rechercher un propriétaire par nom ou autres critères
* Afficher les détails d'un propriétaire

***Gestion des animaux :***

* Enregistrer un nouvel animal associé à un propriétaire
* Modifier les informations d'un animal
* Supprimer un animal
* Visualiser l'historique médical d'un animal
* Afficher les détails d'un animal

***Gestion des visites médicales :***

* Planifier une nouvelle visite
* Enregistrer les détails d'une consultation
* Modifier les informations d'une visite
* Annuler une visite
* Visualiser l'historique des visites d'un animal

***Gestion des vétérinaires :***

* Afficher la liste statique des vétérinaires
* Assigner un vétérinaire à une visite

### **2.2 Besoins non fonctionnels**

***Performance :***

* Temps de réponse rapide
* Capacité à gérer efficacement une base de données de plusieurs milliers

D'enregistrements

***Fiabilité :***

* Sauvegarde automatique des données
* Mécanismes de récupération en cas d'erreur
* Validation des données saisies

***Utilisabilité :***

* Interface intuitive et conviviale
* Aide contextuelle
* Navigation fluide entre les différentes fonctionnalités

### **2.3 Contraintes techniques**

* Utilisation de Java (POO) pour le développement backend
* Utilisation de JavaFX pour l'interface utilisateur (frontend)
* Base de données (Fichiers)
* Compatibilité avec Windows
* Performances acceptables sur des configurations matérielles modestes

## **3. Architecture du système**

### **3.1 Architecture globale**

L'application Pawly suit une architecture à trois niveaux :

1. ***Couche Présentation*** : Interface utilisateur développée avec JavaFX, responsable de l'affichage et de l'interaction avec l'utilisateur.
2. ***Couche Métier***: Logique applicative développée en Java, implémentant les fonctionnalités du système.
3. ***Couche Données*** : Accès aux données via des objets d'accès aux données (DAO) avec deux approches principales :

* Fichiers texte (.txt) pour la persistance des données des propriétaires
* Collections en mémoire pour les vétérinaires et autres entités

VetCare3600/

│

├── resources/ # CSS, FXML...

│ ├── css/

│ └── view/

│

└── src/

├── controller/ # Les contrôleurs JavaFX pour les interfaces

├── dao/ # Classes d’accès à la base de données

├── model/ # Les classes métier (Proprietaire, Animal, Visite...)

└── view/

└──Main.java # Point de départ de l’application

Cette architecture permet une séparation claire des responsabilités et facilite la maintenance et l'évolution du système

### **3.2 Modèle MVC**

Le projet Pawly adopte le modèle d'architecture Modèle-Vue-Contrôleur (MVC) :

* ***Modèle***: Représente les données et la logique métier de l'application
  + Classes d'entités (Proprietaire, Animal, Visite, Veterinaire)
* **Vue** : Interface utilisateur développée avec JavaFX
  + Fichiers FXML pour la définition des écrans
  + CSS pour la personnalisation de l'apparence
* **Contrôleur** : Gère les interactions entre le modèle et la vue
  + Contrôleurs JavaFX associés à chaque écran
  + Traitement des événements utilisateur

**4. Conception du système**

### **4.2 Diagramme de classes**

Le diagramme de classes simplifié du système Pawly comprend :

***Classes du modèle :***

* Proprietaire
* Animal
* Visite
* Veterinaire

***Classes DAO :***

* ProprietaireDAO
* AnimalDAO
* VisiteDAO
* VeterinaireDAO

***Contrôleurs :***

* AccueilController.java
* AjouterAnimalController.java
* AjouterVisiteController.java
* AnimalModificationCallback.java
* ContactController.java
* InfosProprietaireController.java
* ListeProprietairesController.java
* ListeVetoController.java
* ListesVisitesController.java
* ModifierAnimalController.java
* ModifierProprietaireController.java
* NouveauProprietaireController.java
* ProprietaireModificationCallback.java
* RechercheProprietaireController.java

## **5. Implémentation**

### **5.1 Technologies utilisées**

***Backend :***

* Java 11 ou supérieur recommandé
* Gestion de fichiers Java standard pour la persistance des données
* Collections Java pour le stockage en mémoire
* Maven pour la gestion des dépendances

***Frontend :***

* JavaFX Windows SDK
* FXML pour la définition des interfaces
* CSS pour la personnalisation de l'apparence

***Base de données :***

* Fichiers texte (.txt) pour le stockage des propriétaires
* Collections en mémoire pour les vétérinaires et autres entités

***Outils de développement :***

* IntelliJ IDEA
* Run 'RunJavaFX' pour la conception des interfaces JavaFX
* Git pour le contrôle de version

### **5.2 Structure du projet**

La structure du projet Pawly est organisée comme suit :

VetCare3600/

│

└── resources/

│ ├── css/

│ │ ├── ajouter-proprietaire.css

│ │ ├── contact.css

│ │ └── style.css

│ │ # Feuille de style personnalisée

│ └── view/ # Interfaces graphiques (JavaFX)

│ ├── accueil.fxml

│ ├── ajouter\_animal.fxml

│ ├── ajouter\_visite.fxml

│ ├── contact.fxml

│ ├── infos\_proprietaire.fxml

│ ├── liste\_proprietaires.fxml

│ ├── liste\_veto.fxml

│ ├── liste\_visites.fxml

│ ├── modifier\_animal.fxml

│ ├── modifier\_proprietaire.fxml

│ ├── nouveau\_proprietaire.fxml

│ └── recherche\_proprietaire.fxml

│

└── src/ # Tous les contrôleurs JavaFX (1 par interface)

├── controller/

│ ├── AccueilController.java

│ ├── AjouterAnimalController.java

│ ├── AjouterVisiteController.java

│ ├── AnimalModificationCallback.java

│ ├── ContactController.java

│ ├── InfosProprietaireController.java

│ ├── ListeProprietairesController.java

│ ├── ListeVetoController.java

│ ├── ListesVisitesController.java

│ ├── ModifierAnimalController.java

│ ├── ModifierProprietaireController.java

│ ├── NouveauProprietaireController.java

│ ├── ProprietaireModificationCallback.java

│ └── RechercheProprietaireController.java

│

├── model/ # Les classes métier (POO)

│ ├── Animal.java

│ ├── Proprietaire.java

│ ├── Veterinaire.java

│ └── Visite.java

│

├── dao/ # DAO simulé ou connecté à une BDD (ArrayList)

│ ├── AnimalDAO.java

│ ├── ProprietaireDAO.java

│ ├──VeterinaireDAO.java

│ └── VisiteDAO.java

│

└── view/ # Point d’entrée Java

└── Main.java

### **5.3 Description des modules principaux**

***Module Modèle :***

Ce module contient les classes représentant les entités métier du système. Chaque classe modèle encapsule les données et comportements spécifiques à une entité.

***Module DAO :***

Ce module implémente le pattern DAO pour l'accès aux données. Chaque classe DAO fournit les méthodes CRUD (Create, Read, Update, Delete) pour une entité spécifique, mais avec deux approches distinctes :

* ProprietaireDAO : Utilise des fichiers texte (proprietaires.txt) pour la persistance des données. Les méthodes incluent la lecture/écriture de fichiers, la recherche par nom et la mise à jour des données.
* VeterinaireDAO : Utilise une collection en mémoire (ArrayList statique) pour stocker les données
* Les autres DAOs suivent l'une de ces deux approches selon les besoins.

***Module Contrôleur*** *:*

Ce module contient les contrôleurs JavaFX qui gèrent les interactions entre l'interface utilisateur et les objets DAO. Chaque contrôleur est associé à un fichier FXML définissant l'interface utilisateur correspondante.

## **6. Interfaces utilisateur**

### **6.1 Maquettes d'écrans**

L'application Pawly comprend les écrans suivants :

***Écran d'accueil* :**

Page principale avec les fonctionnalités de navigation vers les différentes parties de l'application.

**Écran de recherche de propriétaire :**

Permet de rechercher un propriétaire par son nom et d'accéder à la création d'un nouveau propriétaire.

**Écran de création de propriétaire :**

Formulaire pour l'ajout d'un nouveau propriétaire dans le système.

**Écran de modification de propriétaire** :

Formulaire pour modifier les informations d'un propriétaire existant.

**Écran d'information propriétaire** :

Affiche les détails d'un propriétaire ainsi que la liste de ses animaux.

**Écran d'ajout d'animal :**

Formulaire pour ajouter un nouvel animal associé à un propriétaire.

**Écran de modification d'animal :**

Formulaire pour modifier les informations d'un animal existant.

**Écran d'ajout de visite** :

Formulaire pour enregistrer une nouvelle visite médicale pour un animal.

**Écran de liste des vétérinaires** :

Affiche la liste des vétérinaires disponibles dans la clinique.

### **6.2 Navigation entre les écrans**

La navigation entre les écrans de l'application est conçue pour être intuitive et cohérente :

1. L'écran d'accueil permet d'accéder à la recherche de propriétaires ou à la liste des vétérinaires.
2. La recherche de propriétaires mène soit à la liste des résultats, soit à la création d'un nouveau propriétaire.
3. À partir de la liste des propriétaires, on peut accéder aux détails d'un propriétaire spécifique.
4. À partir des détails d'un propriétaire, on peut modifier ses informations ou gérer ses animaux.
5. Pour chaque animal, on peut voir ses détails, modifier ses informations ou ajouter une visite.

### **6.3 Expérience utilisateur**

L'interface utilisateur de Pawly est conçue avec les principes suivants :

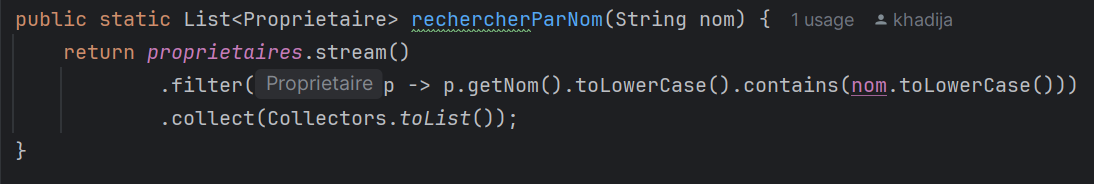
* ***Simplicité***: Interface épurée, sans surcharge d'informations
* ***Cohérence*** : Éléments d'interface et comportements cohérents à travers l'application
* ***Intuitivité*** : Flux de travail naturel correspondant aux tâches quotidiennes des utilisateurs
* ***Feedback***: Retour d'information visuel pour les actions utilisateur
* ***Accessibilité*** : Interface accessible à tous les utilisateurs, y compris ceux ayant des limitations

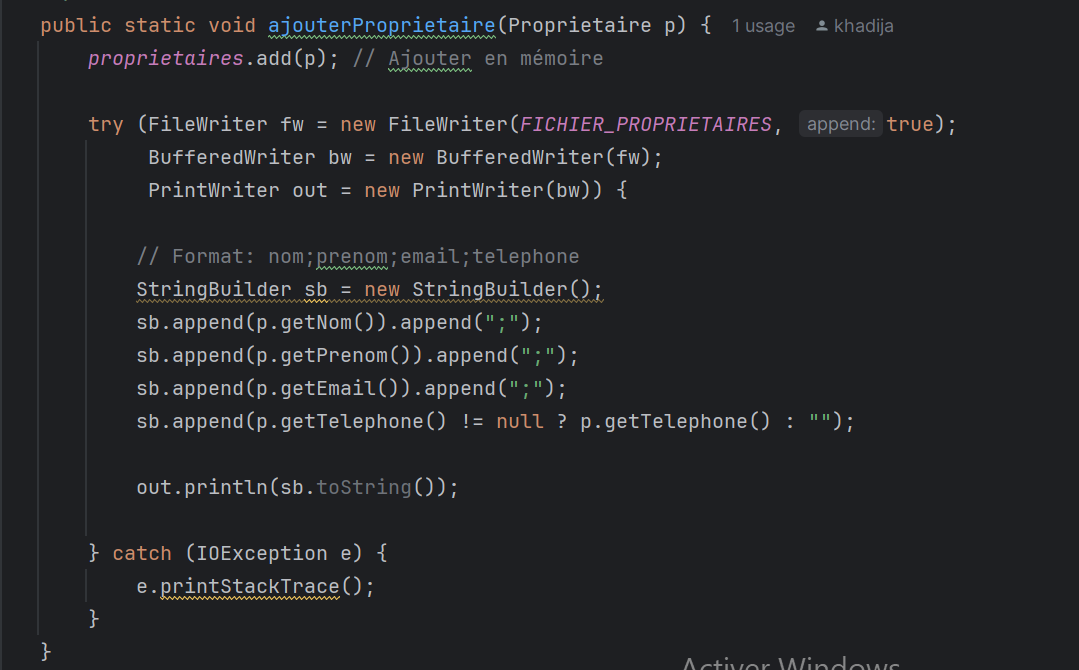
**7. Fonctionnalités développées**

### **7.1 Gestion des propriétaires**

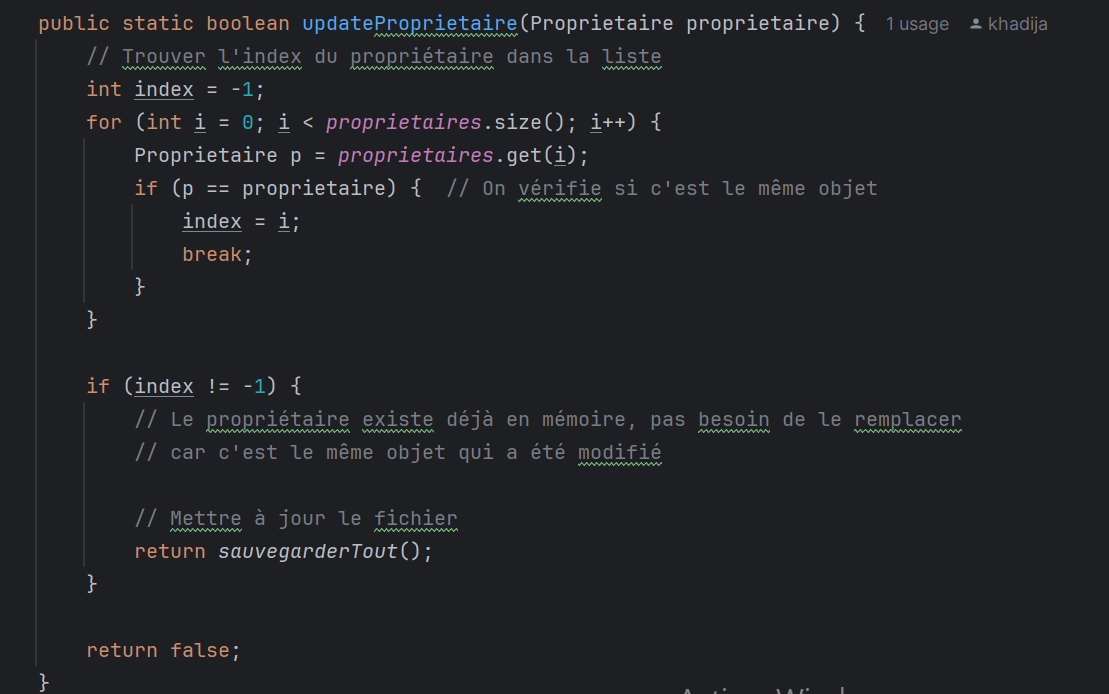
La gestion des propriétaires comprend les fonctionnalités suivantes :

***Recherche de propriétaire :***

***Ajout d'un propriétaire :***



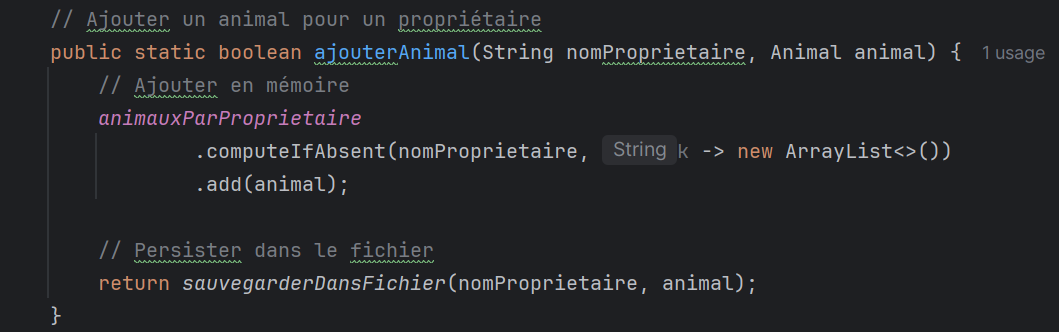
***Modification d'un propriétaire :***



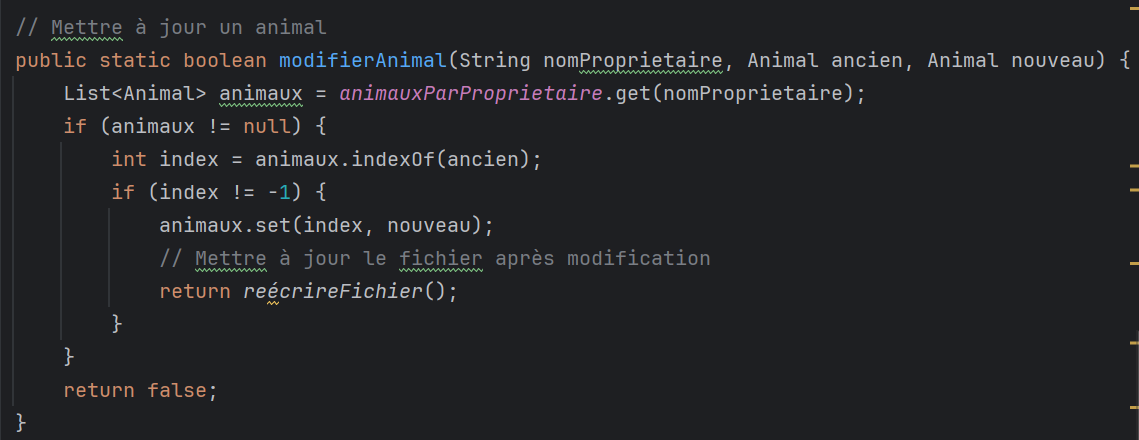
**7.2 Gestion des animaux**

La gestion des animaux comprend les fonctionnalités suivantes :

***Ajout d'un animal :***



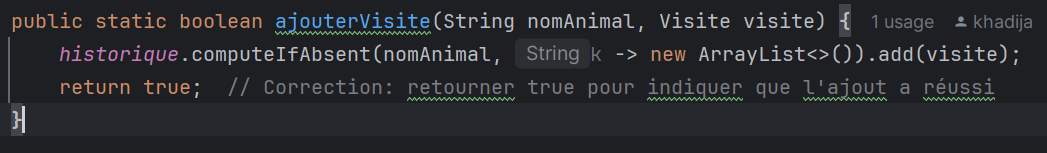
***Modification d'un animal :***

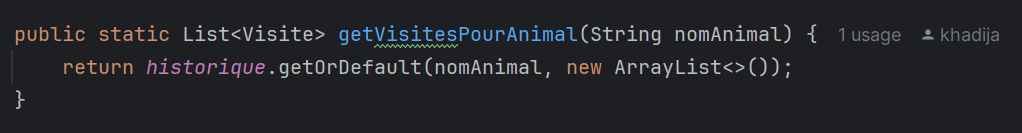


### **7.3 Gestion des visites**

La gestion des visites comprend les fonctionnalités suivantes :

***Ajout d'une visite :***

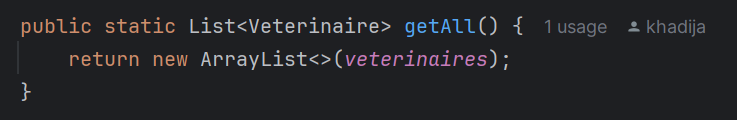
***Récupération des visites d'un animal :***



### **7.4 Gestion des vétérinaires**

La gestion des vétérinaires comprend les fonctionnalités suivantes :

***Liste des vétérinaires :***



## **8. Déploiement**

### **8.1 Prérequis système**

Pour exécuter l'application Pawly, les prérequis suivants sont nécessaires :

* Oracle OpenJDK 23.0.2
* Minimum 4 Go de RAM
* 50 Mo d'espace disque libre pour l'application et les fichiers de données
* Résolution d'écran minimale de 1280x720

### **8.2 Procédure d'installation**

1. **Installation de Java**
   1. Télécharger et installer le Oracle OpenJDK 23.0.2
   2. Vérifier l'installation avec la commande java –version
2. **Installation de l'application**
   1. Télécharger l'archive de l'application
   2. Extraire le contenu dans le répertoire souhaité
   3. S'assurer que l'utilisateur à les droits d'écriture dans ce répertoire
3. **Lancement de l'application**
   1. Au premier lancement, les fichiers de données seront créés automatiquement

### **8.3 Configuration**

L'application ne nécessite pas de configuration particulière, car elle stocke directement ses données dans des fichiers texte à la racine de l'application.

Si nécessaire, les chemins des fichiers de données peuvent être modifiés en éditant les constantes dans les classes DAO correspondantes.

## **9. Conclusion et perspectives**

### **9.1 Bilan du projet**

Le projet Pawly a permis de développer une application complète de gestion de clinique vétérinaire, répondant aux besoins exprimés. L'utilisation de Java et JavaFX a permis de créer une application desktop performante et ergonomique.

Les principales réalisations du projet sont :

* Une architecture robuste basée sur MVC
* Une interface utilisateur intuitive et conviviale
* Une gestion complète des propriétaires, animaux, visites et vétérinaires

### **9.2 Difficultés rencontrées**

Plusieurs difficultés ont été rencontrées lors du développement :

* ***Gestion des relations entre entités***: La gestion des relations entre les différentes entités (propriétaire-animal-visite) a nécessité une conception *soignée du modèle de données.*
* ***Interface utilisateur responsive :*** La création d'une interface qui s'adapte à différentes tailles d'écran a été un défi.
* ***Performance avec de grandes quantités de données***: Des optimisations ont été nécessaires pour garantir de bonnes performances avec un grand nombre d'enregistrements.

### **9.3 Améliorations futures:**

Plusieurs pistes d'amélioration sont envisagées pour les futures versions de Pawly:

* ***Module de gestion des rendez-vous*** :

Ajout d'un calendrier pour la planification des rendez-vous.

* **Module de facturation** :

Intégration d'un système de facturation pour les visites et traitements.

* **Application mobile:**

Développement d'une version mobile de l'application pour les vétérinaires

en déplacement.

* **Synchronisation cloud:**

Mise en place d'une synchronisation des données avec un serveur cloud.

* **Rapports et statistiques** :

Ajout de fonctionnalités d'analyse et de reporting.