



Projet de Fin de semestre

Diplôme Universitaire de Technologie

Filière : Ingénierie Logicielle et Cybersecurity

*Une application pour
la gestion d'une clinique vétérinaire
vetcare 360*

Réalisé par :

- Mohamed Aymane Jlida
- Norhane Ramzi

Encadré par :

Mr. Esbai Redouane

Table des matières

1. Introduction

2. Présentation Générale

3. Objectifs du Projet

4. Fonctionnalités Principales

5. Diagramme de Classe (UML)

6. Architecture Technique Détailée

7. Interface Utilisateur

8. Difficultés Rencontrées et Solutions Apportées

9. Perspectives d'Amélioration

10. Synthèse

1. Introduction

Dans le cadre de notre apprentissage du développement d'applications orientées objet en Java, nous avons conçu **VetCare 360**, une application dédiée à la gestion des cliniques vétérinaires. Cette solution permet une prise en charge complète des propriétaires d'animaux, de leurs compagnons, des vétérinaires, ainsi que du suivi médical des visites. Le projet répond à un besoin concret de centralisation des données et d'accès rapide à l'information au sein des établissements vétérinaires. Développé en Java, VetCare 360 repose sur une **architecture modulaire** et une **interface graphique moderne réalisée avec JavaFX**.

2. Présentation Générale

VetCare 360 est une application de gestion complète conçue spécifiquement pour les cliniques vétérinaires. Développée en Java avec une interface graphique JavaFX, elle répond aux besoins opérationnels des professionnels du secteur en centralisant l'ensemble des données essentielles. Elle permet notamment la gestion des fiches clients pour les propriétaires d'animaux, des dossiers médicaux des animaux (incluant race, âge, et historique vaccinal), des profils vétérinaires (spécialisation), ainsi que des visites médicales (motifs, diagnostics, traitements et suivi). L'objectif de VetCare 360 est de simplifier les tâches quotidiennes au sein d'une clinique grâce à une interface ergonomique et une organisation structurée des informations.

3. Objectifs du Projet

- Gérer les propriétaires et leurs informations.
- Associer les animaux à leurs propriétaires.
- Planifier et suivre les visites médicales des animaux.
- Lister les vétérinaires et leurs interventions.
- Proposer une interface moderne .

4. Fonctionnalités Principales

Gestion des propriétaires :

- Ajouter un nouveau propriétaire.
- Modifier les informations d'un propriétaire.
- Supprimer un propriétaire.
- Rechercher un propriétaire par son nom.

Gestion des animaux :

- Associer des animaux à un propriétaire.
- Modifier les informations d'un animal.
- Voir l'historique des visites d'un animal.

Gestion des visites :

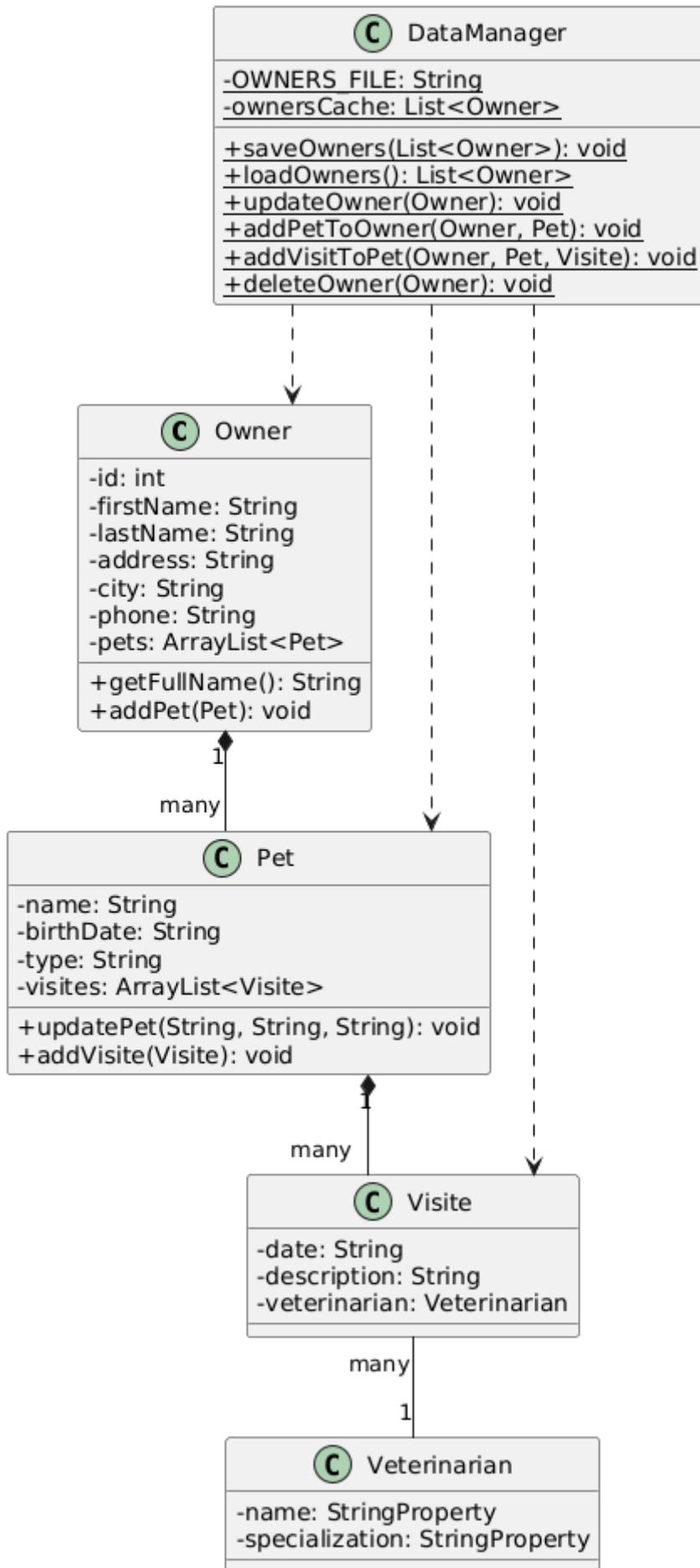
- Ajouter une visite médicale pour un animal.

- Visualiser l'historique des visites.

Affichage des vétérinaires :

- liste récupérée depuis la base de données.

5. Le diagramme de classe (UML)



Explication du Diagramme UML

DataManager

Cette classe centralise toutes les opérations de lecture et d'écriture des données. Elle contient

- Une constante `OWNERS_FILE` désignant le chemin du fichier de sauvegarde,
- Une liste `ownersCache` qui contient tous les propriétaires actuellement chargés en mémoire,
- Des méthodes pour sauvegarder, charger, mettre à jour et supprimer des entités `Owner`, ainsi que pour ajouter des `Pet` ou des `Visite`.

Owner

Représente un propriétaire d'animaux. Il contient :

- Des informations personnelles (nom, prénom, adresse, etc.),
- Une liste d'animaux (`ArrayList<Pet>`),
- Des méthodes pour récupérer le nom complet et ajouter un animal.

Pet

Chaque animal est lié à un propriétaire, et peut avoir :

- Un nom, une date de naissance, un type (ex. : chien, chat),
- Une liste de visites médicales (`ArrayList<Visite>`),
- Des méthodes pour modifier ses informations ou ajouter une visite.

Visite

Une visite représente une consultation vétérinaire. Elle contient :

- Une date, un motif ou une description, et un vétérinaire associé.

Veterinarian

Le vétérinaire est représenté par deux propriétés JavaFX :

- Un nom (StringProperty)
- Une spécialité (StringProperty)

Associations :

- Un Owner possède plusieurs Pet
- Un Pet peut avoir plusieurs Visite
- Une Visite est liée à un seul Veterinarian
- DataManager dépend des autres classes pour effectuer les opérations de gestion

6. Architecture Technique Détaillée

Le projet **VetCare 360**, développé en Java avec JavaFX, repose sur une architecture logicielle modulaire organisée selon le modèle **MVC (Modèle - Vue - Contrôleur)**. Cette architecture permet une séparation claire des responsabilités, une maintenance facilitée, et une bonne évolutivité du système.

Environnement de Développement :

- **Java 21** : utilisé comme langage principal orienté objet.
- **JavaFX 17** : pour la construction de l'interface utilisateur graphique.
- **FXML** : langage déclaratif utilisé pour structurer les interfaces.
- **Maven** : gestionnaire de dépendances et outil de build.
- **Git / GitHub** : pour la gestion de version et le travail collaboratif.
- **IDE IntelliJ IDEA** : environnement de développement intégré utilisé pour coder, compiler et tester le projet.

Structure du Code :

```
vetcare-javafx/
├── .idea/          # Fichiers de configuration de l'IDE IntelliJ
├── .mvn/           # Scripts Maven
└── src/
    └── main/
        ├── java/
        │   └── org/
        │       └── vetjavafx/
        │           ├── model/          # Modèles de données
        │           │   ├── Owner.java    # Entité représentant un propriétaire
        │           │   ├── Pet.java      # Entité représentant un animal
        │           │   ├── Veterinarian.java # Entité représentant un vétérinaire
        │           │   ├── Visite.java     # Entité pour les visites médicales
        │           │   └── DataManager.java # Gestion centralisée des données
        │           └── Controller/     # Contrôleurs de l'application
        │               └── AccueilController.java
```

```

|   |   |   |   └── AddOwnerController.java
|   |   |   |   └── ModifyOwnerController.java
|   |   |   |   └── OwnerDetailsController.java
|   |   |   |   └── AddPetController.java
|   |   |   |   └── ModifyPetController.java
|   |   |   |   └── AddVisitController.java
|   |   |   |   └── PetVisitsController.java
|   |   |   |   └── VeterinarianController.java
|   |   |   └── HelloController.java
|   |   └── HelloApplication.java    # Classe principale (main)
|
|   └── resources/
|
|       └── org/vetjavafx/view/      # Fichiers FXML et ressources
|
|           ├── *.fxml            # Interfaces utilisateurs
|
|           ├── logo.png          # Logo utilisé dans l'interface
|
|           └── animaux.png        # Image illustrant les animaux
|
└── pom.xml          # Fichier de configuration Maven
|
└── module-info.java # Configuration modulaire Java
|
└── mvnw             # Script Maven Linux/macOS
|
└── mvnw.cmd         # Script Maven Windows
|
└── .gitignore        # Fichiers ignorés par Git
|
└── owners.dat        # Fichier de sauvegarde des données en mémoire
|
└── target/           # Dossier généré à la compilation

```

Explication des fichiers du projet VetCare 360

 **.idea/**

- Contient les fichiers de configuration de l'IDE (IntelliJ IDEA). Ce dossier est spécifique à l'environnement de développement et n'est pas directement lié au code source.

.mvn/

- Répertoire interne utilisé par Maven pour gérer l'exécution des commandes (mvnw, mvnw.cmd) sans nécessiter d'installation préalable.

src/main/java/org/vetjavafx/

Contient tout le code source de l'application, structuré en trois parties principales :

model/

- Owner.java : classe représentant un propriétaire d'animaux.
- Pet.java : classe représentant un animal avec ses caractéristiques (nom, race, date de naissance...).
- Veterinarian.java : modèle représentant un vétérinaire (nom, spécialité, etc.).
- Visite.java : classe qui décrit une visite médicale (date, diagnostic, traitement).
- DataManager.java : gestionnaire central des données. Il conserve les listes d'objets (propriétaires, animaux, vétérinaires, visites) en mémoire.

Controller/

Contient tous les contrôleurs FXML qui relient l'interface utilisateur à la logique applicative :

- `AddOwnerController`, `ModifyOwnerController` : pour ajouter ou modifier les informations d'un propriétaire.
- `AddPetController`, `ModifyPetController` : pour gérer les fiches des animaux.
- `AddVisitController`, `PetVisitsController` : pour gérer les visites médicales.
- `OwnerController`, `OwnerDetailsController` : pour la gestion globale des propriétaires.
- `VeterinarianController` : pour l'affichage et la modification des vétérinaires.
- `AccueilController`, `HelloController` : pour la page d'accueil ou les écrans généraux.

`HelloApplication.java`

- Classe principale (`main`) de l'application. Elle initialise JavaFX, charge la vue d'accueil et démarre l'interface graphique.

`src/main/resources/org.vetjavafx.view/`

Contient les ressources non-code du projet :

- Fichiers `.fxml` : définissent les interfaces graphiques de l'application.
- `logo.png`, `animaux.png` : images utilisées dans l'interface utilisateur.

`pom.xml`

- Fichier de configuration Maven. Il liste les dépendances nécessaires (JavaFX, etc.) et configure la compilation et l'exécution du projet.

module-info.java

- Spécifie les modules utilisés par l'application (JavaFX, java.base, etc.) dans une approche modulaire introduite à partir de Java 9.

target/

- Dossier généré automatiquement par Maven. Il contient les fichiers compilés (.class) et les packages .jar.

7. Interface Utilisateur : Description et Justification

L'interface utilisateur de VetCare 360 a été conçue avec JavaFX dans un souci d'ergonomie, de simplicité d'utilisation et d'élégance visuelle. L'ensemble de l'interface est cohérent, fluide et facilement navigable, même pour un personnel non technique.

Page d'accueil

La page d'accueil affiche un visuel accueillant avec une image d'animaux (chat, chien et lapin) représentant la diversité des espèces prises en charge. Elle propose deux accès directs :

- Un bouton **Vétérinaires** permettant d'afficher la liste des praticiens disponibles
- Un bouton **Propriétaires** pour accéder aux dossiers clients

La barre de navigation supérieure donne également un accès rapide aux différentes sections de l'application.

Liste des vétérinaires

Cette page présente un tableau listant les vétérinaires enregistrés avec leurs noms et spécialités respectives. L'interface offre une visualisation claire de la répartition des compétences dans la clinique, facilitant l'affectation des visites.

Gestion des propriétaires

La page des propriétaires permet de :

- Rechercher un propriétaire par nom (champ de recherche dynamique)
- Ajouter un nouveau propriétaire via le bouton « Add Owner »
- Accéder aux détails ou supprimer un propriétaire via des boutons associés à chaque ligne

Le tableau affiche les informations principales telles que le nom complet, l'adresse, la ville et le numéro de téléphone.

Détails du propriétaire

Une fois un propriétaire sélectionné, une fiche détaillée s'affiche avec :

- Nom, prénom, adresse, ville, téléphone
- Liste des animaux associés avec nom, type, date de naissance et nombre de visites
- Des actions permettent d'ajouter un animal, de modifier le propriétaire, ou de gérer les visites pour chaque animal (ajout/modification/consultation)

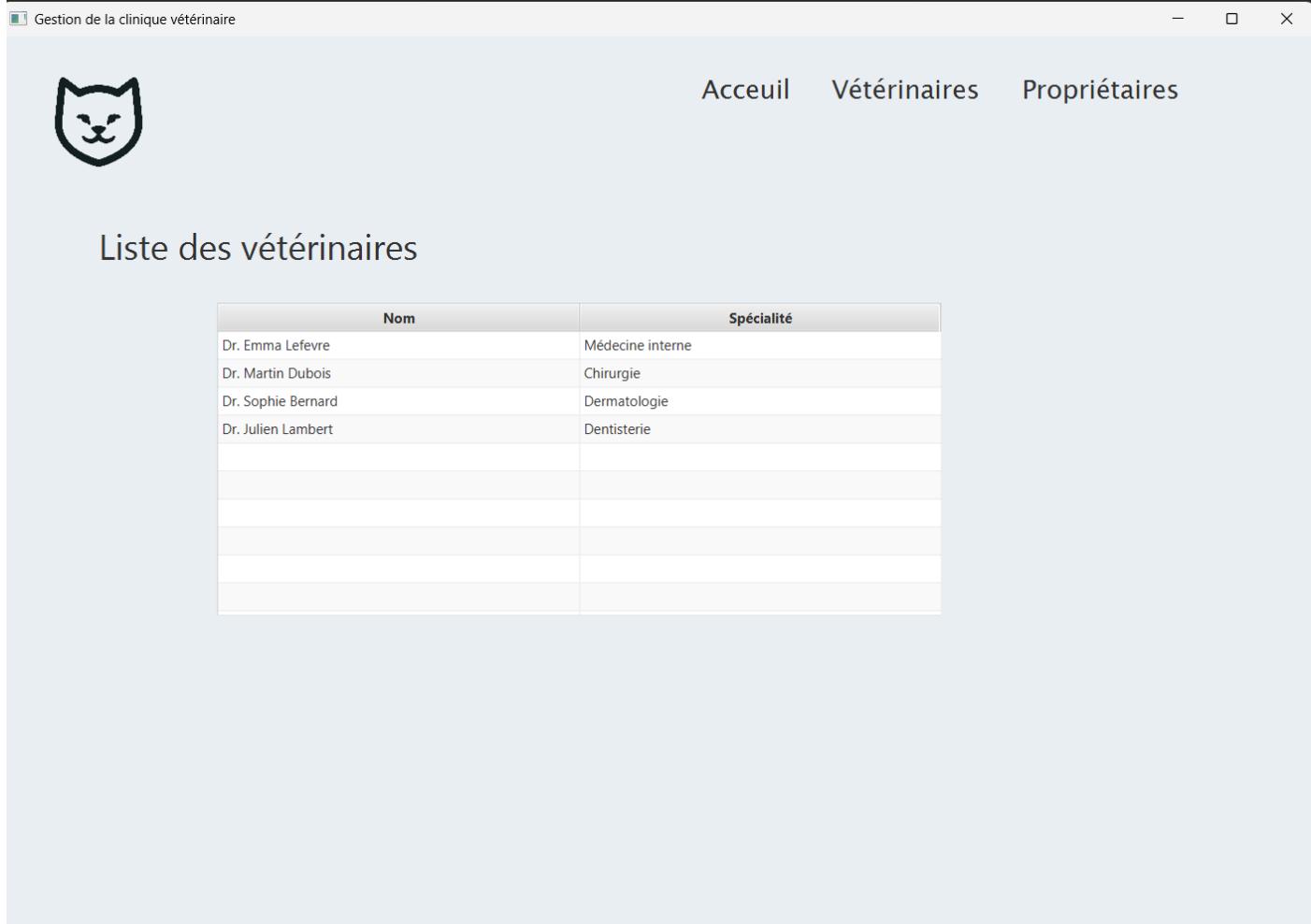
Cette fiche centralise toutes les informations relatives à un client et ses animaux, tout en restant intuitive à utiliser grâce à la disposition verticale des éléments.

Capture d'écran :

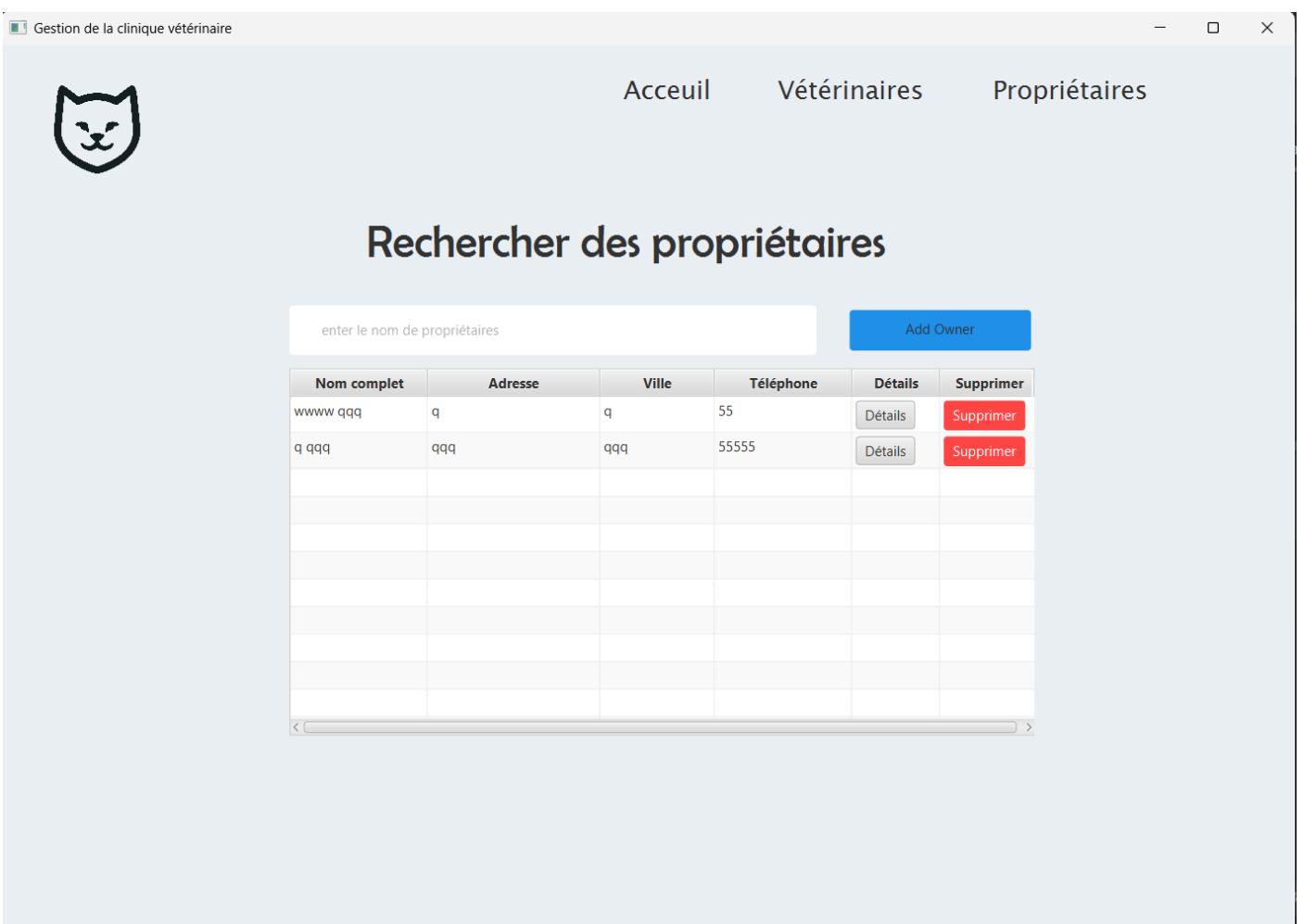
- Page d'accueil avec navigation



- Liste des vétérinaires



- Gestion des propriétaires



- Fiche d'un propriétaire avec ses animaux

Grâce à cette interface utilisateur bien pensée, VetCare 360 assure une gestion fluide, accessible et visuellement agréable du quotidien d'une clinique vétérinaire.

8. Difficultés Rencontrées et Solutions Apportées

Au cours du développement de l'application VetCare 360, nous avons été confrontés à plusieurs obstacles techniques et logiques, que nous avons progressivement surmontés. Voici un aperçu des principales difficultés rencontrées et des solutions mises en œuvre :

- **Intégration JavaFX et FXML** : Au début, il était complexe de faire communiquer les contrôleurs avec les vues FXML. Nous avons structuré notre projet selon le modèle MVC et respecté les conventions de nommage pour permettre une liaison correcte via les annotations @FXML.
- **Mise à jour dynamique de l'interface** : Lorsqu'on modifiait ou ajoutait un élément (propriétaire, animal ou visite), la mise à jour ne s'affichait pas en temps réel. Nous avons utilisé les collections ObservableList pour assurer la réactivité de l'interface.
- **Persistance des données** : En l'absence d'une base de données réelle, nous avons utilisé la sérialisation dans un fichier .dat pour sauvegarder les propriétaires et leurs animaux. Nous avons conçu la classe DataManager pour centraliser les opérations de lecture/écriture.
- **Gestion des erreurs utilisateur** : Nous avons ajouté des vérifications sur les champs obligatoires et mis en place des messages d'erreur clairs pour prévenir les incohérences (ex : ajout d'un animal sans propriétaire).
- **Navigation entre les vues** : La gestion des vues en JavaFX nécessite de bien manipuler les FXMLLoader. Une attention particulière a été portée à la hiérarchie des fichiers FXML et à la réutilisation des scènes.

9. Perspectives d'Amélioration

Plusieurs évolutions pourraient être envisagées pour améliorer VetCare 360 :

- Intégration d'une base de données relationnelle (MySQL ou PostgreSQL)
- Ajout d'un système d'authentification utilisateur (admin, vétérinaire, secrétaire)
- Génération automatique de comptes-rendus PDF des visites
- Statistiques sur les animaux/vétérinaires (nombre de visites, type de pathologies)

Ces évolutions permettraient de transformer VetCare 360 en une application encore plus complète et adaptée à un usage réel en clinique.

10. Synthèse

VetCare 360 est une solution logicielle complète et bien structurée, conçue pour répondre efficacement aux besoins de gestion des cliniques vétérinaires. Ce projet allie une architecture technique robuste à une interface utilisateur intuitive. Son développement s'appuie sur les principes fondamentaux de l'ingénierie logicielle, offrant un outil fiable, adapté aux exigences du secteur vétérinaire.

Pour Consulter le code source complet de l'application, Vous Pouvez
Accéder au dépôt GitHub via le lien suivant :

<https://github.com/M-Aymane-404/vetjavafx>

N'hésitez pas à explorer le projet pour une meilleure compréhension de l'architecture, des fonctionnalités et des choix techniques mis en œuvre.

Merci pour votre attention.