المدرسة العليا للتكنولوجيا الناظور
École Supérieure de Technologie de Nador
ⵜⵉⵎⵙⵙⵓⵔⵉⵜ ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵜⵓⵜⵜⵔⵉⵙⵜⵉⵜ ⵅⵍⵓⵎ
جامعة محمد الأول بوجدة
UNIVERSITÉ MOHAMMED PREMIER OUJDA
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵎⵓⵃⴰⵎⵎⴷ ⴰⵎⴰⵣⵡⴰⵔ ⵏ ⵡⵊⴷⴰ

Projet de Fin de semestre

Diplôme Universitaire de Technologie

Filière : Ingénierie Logicielle et Cybersecurity

(ILCS)

Intitulé du projet :

Une application web pour la gestion d'une clinique vétérinaire

Vetecare 360 « Java »

Réalisée par:

Alae Majjati

&

Oumaima Marzak

Professeur :

Mr. Esbai Redouane

Année universitaire : 2024/2025

1. Introduction

Dans le cadre de l'étude et de la pratique du développement d'applications en Java orienté objet, nous avons développé VetCare 360, une application dédiée à la gestion des cliniques vétérinaires.

Cette application permet d’assurer la gestion complète des propriétaires, de leurs animaux de compagnie, des vétérinaires, et de l'historique des visites médicales associées aux animaux.

Le projet s’inscrit dans un contexte réel de besoins pour les cliniques vétérinaires, où la centralisation des informations et la rapidité d’accès aux données sont essentielles.

Développé en Java, il s’appuie sur :

- Une architecture modulaire,
  Et une interface graphique JavaFX

2. Présentation du Projet

Le projet VetCare 360 est une application de gestion complète dédiée aux cliniques vétérinaires, développée en Java avec une interface JavaFX. Conçue pour répondre aux besoins des professionnels du secteur, cette solution centralise et optimise la gestion des données essentielles :

Propriétaires d’animaux (fiches clients, coordonnées, historique)

Animaux (dossiers médicaux, races, âges, vaccinations)

Vétérinaires (spécialisations, disponibilités)

Visites médicales (motifs, diagnostics, traitements, suivi)

Le projet vise à faciliter la gestion quotidienne d’une clinique grâce à une interface ergonomique et une base de données fiable.

3. Objectifs

- Optimiser la gestion des propriétaires d'animaux et de leurs dossiers.
- Suivre l’historique médical des animaux.
- Faciliter la prise en charge des rendez-vous et des visites.
- Lister les vétérinaires et leur associer les visites réalisées.

4. Fonctionnalités attendues

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.
- Supprimer un animal.
- Voir l'historique des visites d'un animal.

Gestion des vétérinaires :

- Ajouter un vétérinaire.
- Modifier ou supprimer un vétérinaire.

- Lister tous les vétérinaires.

Gestion des visites médicales :

- Ajouter une visite médicale pour un animal.

- Visualiser l'historique des visites.

- Modifier une visite médicale.

5. Technologies utilisées

Langage	Java
Interface utilisateur	JavaFX.
Gestion de versions	Git/GitHub.

Détail Approfondi des Rôles des Technologies

1. Langage Java : Le Cerveau du Système

Rôle Principal :

- Moteur de traitement : Exécute toute la logique métier complexe (calculs, décisions, algorithmes).

- Modélisation des données : Transforme les concepts réels (animaux, propriétaires) en objets programmatiques.

Avantages :

- Portabilité : Fonctionne sur tous les systèmes d'exploitation grâce à la JVM.
- Performance : Compilation en bytecode optimisé.
- Sécurité : Gestion mémoire automatique (pas de fuites mémoire).
- Écosystème : Bibliothèques riches pour toutes les fonctionnalités.

Responsabilités Concrètes :

- Gestion des relations :
  - Un propriétaire *a* des animaux (relation 1-n).
  - Une visite appartient à un animal et un vétérinaire.
- Validation des règles métier :
  - Un animal ne peut avoir qu'un seul propriétaire.
  - Un vétérinaire ne peut avoir deux rendez-vous simultanés.
- Persistance simplifiée :
  - Sérialisation des objets pour un stockage temporaire.

Utilisation dans VetCare 360 :

- Classes POO pour modéliser propriétaires, animaux, vétérinaires et visites.
- Structures de données (Listes, Maps) pour gérer les enregistrements.
- Algorithmes de tri/filtre pour les recherches.

Exemple Réel :

Quand un utilisateur ajoute un animal, Java :

1. Vérifie que le propriétaire existe.
2. Crée un nouvel objet Pet.
3. Met à jour la liste des animaux du propriétaire.

2. JavaFX : Le Visage de l'Application

Rôle Principal :

- Pont humain-machine : Traduit les actions utilisateur en commandes Java, et inversement.

Fonctions Clés :

Composant	Rôle Spécifique	Exemple dans VetCare 360
TableView	Affiche des données tabulaires	Liste des vétérinaires avec colonnes
Property	Lie les données Java à l'UI	Mise à jour auto du nom d'un propriétaire
CSS	Personnalisation visuelle	Boutons verts pour les validations
FXML	Définit la structure des écrans	Layout du formulaire d'ajout d'animal

Processus Typique :

1. L'utilisateur clique sur "Ajouter un animal".
2. JavaFX :
    - Affiche un formulaire pré-rempli si le propriétaire est déjà sélectionné.
    - Applique un style CSS aux champs obligatoires.
3. Après soumission :
    - Transmet les données au backend Java.

3. Git/GitHub : La Mémoire Collective

Rôle Principal :

- Machine à voyager dans le temps : Garde une trace de chaque modification depuis le jour 1.
- suivi des modifications du code.

Avantages :

- Historique : Retour à n'importe quelle version antérieure.
- Branches : Développement parallèle de fonctionnalités.
- Collaboration : Fusion aisée des contributions (pull requests).

Architecture Technique Détaillée

Le projet VetCare 360 est construit sur une architecture technique robuste et moderne :

1. Environnement de Développement :

- Java 21 : Version la plus récente du langage, offrant des performances optimales

- JavaFX 21.0.2 : Framework moderne pour l'interface graphique

- Maven : Gestionnaire de dépendances et outil de build

- JUnit 5.10.2 : Framework de tests unitaires

2. Structure du Code :

- Architecture MVC (Model-View-Controller) stricte

- Package `model` : Classes métier (Owner, Pet, Veterinarian, Visit)

- Package `controller` : Logique métier et interaction UI

- Package `view` : Fichiers FXML pour l'interface

- Classe principale `VetCareApplication.java`

3. Gestion des Ressources :

- Fichiers FXML pour la définition des interfaces

- Feuilles de style CSS pour la personnalisation visuelle

- Configuration Maven pour la gestion des ressources

- Support pour la localisation (fichiers .properties)

4. Configuration de Build :

- Compilation optimisée pour Java 21

- Plugin JavaFX Maven pour la création d'images natives

- Configuration de production avec optimisation des ressources

- Support pour les tests automatisés

Aspects Techniques Avancés

1. Performance et Optimisation :

- Compilation optimisée pour Java 21
- Gestion efficace de la mémoire
- Chargement paresseux des ressources
- Optimisation des requêtes de données

2. Sécurité et Maintenance :

- Architecture modulaire et maintenable
- Gestion des exceptions robuste
- Validation des données côté serveur
- Logs détaillés pour le débogage

3. Évolutivité :

- Structure modulaire facilitant l'ajout de fonctionnalités
- Interface extensible pour de nouveaux modules
- Support pour l'internationalisation
- Architecture prête pour l'ajout d'une base de données

4. Tests et Qualité :

- Tests unitaires avec JUnit 5

- Validation des données en temps réel

- Gestion des erreurs utilisateur

- Documentation du code

6. Contraintes techniques

- Le projet suivre une architecture MVC stricte
- Respecter les bonnes pratiques de codage Java. .
- Gestion des exceptions pour la robustesse.
- Encapsulation des données (getters/setters)
- Interface ergonomique (FXML + CSS).

7. Structure du projet – VetCare 360 (Java)

Voici **l'arborescence complète** de du projet , avec tous les fichiers et dossiers présents :

```
| vetcareJava/
├── .mvn/                    # Configuration Maven
├── src/
|   ├── main/
|   |   ├── java/
|   |   |   └── com/example/
|   |   |       ├── fxproject/    # Modules principaux
```

```
│   │   │       │   ├── HelloApplication.java

│   │   │       │   ├── HelloController.java
│   │   │       │   ├── VetCareApplication.java
│   │   │       │   └── model/
│   │   │       │       ├── Owner.java
│   │   │       │       ├── Pet.java
│   │   │       │       ├── Veterinarian.java
│   │   │       │       ├── Visit.java
│   │   │       │       └── DataManager.java
│   │   │
│   │   │           └── controller/
│   │   │       │   ├── OwnerFormController.java
│   │   │       │   ├── PetFormController.java
│   │   │       │   ├── VeterinarianFormController.java
│   │   │       │   ├── VetFormController.java
│   │   │       │   └── VisitFormController.java
│   │   └── resources/
│   │       └── com/example/fxproject/
│   │   │           └── styles/
│   │   │               └── main.css
│   │   ├── owner-form.fxml
│   │   ├── pet-form.fxml
│   │     ├── veterinarian-form.fxml
│   │     ├── vet-form.fxml
│   │     └── visit-form.fxml
├── target/                 # Fichiers compilés
├── .gitignore
```

```
├── mvnw
├── mvnw.cmd (Windows)
└── pom.xml                  # Dépendances Maven
```

Détails :

- **controller/** : Contient les classes qui gèrent la logique métier et éventuellement l'interface utilisateur.
- **model**/ : Contient les classes métier représentant les entités du système (Animal, Propriétaire, Vétérinaire, Visite).
- **App.java** : Point d'entrée principal du projet (probablement pour lancer le programme ou tester).

Par exemple:

Owner.java

Modèle qui stocke les informations des propriétaires d'animaux (nom, adresse, téléphone, email).

Pet.java

Modèle qui gère les données des animaux (nom, espèce, race, date de naissance) et leur lien avec un propriétaire.

Veterinarian.java

Modèle qui représente les vétérinaires (nom, spécialisation, coordonnées).

Visit.java

Modèle qui enregistre les détails des consultations (date, raison, diagnostic, traitement).

DataManager.java

"Base de données" centrale qui stocke et relie toutes les entités (owners, pets, vets, visits) en mémoire.

OwnerFormController.java

Contrôleur qui gère les actions de l'interface propriétaire (ajout, modification, suppression).

PetFormController.java

Contrôleur qui traite les interactions avec les formulaires animaux.

VisitFormController.java

Contrôleur qui pilote la création et la gestion des visites médicales.

FXML files

Fichiers XML qui définissent le design des interfaces (formulaires propriétaires, animaux, visites).

main.css

Feuille de style qui standardise l'apparence des boutons, tableaux et autres éléments de l'interface utilisateur.

pom.xml

Fichier de configuration Maven qui liste les bibliothèques nécessaires (comme JavaFX) et décrit comment compiler et construire le projet.

mvnw

Script Maven qui permet de lancer et de construire le projet sans avoir besoin d’installer Maven manuellement sur la machine.

Exemple d’interaction typique dans l’application :

Saisie utilisateur :

L’utilisateur remplit les champs du formulaire (via pet-form.fxml).

Contrôleur :

Le PetFormController :

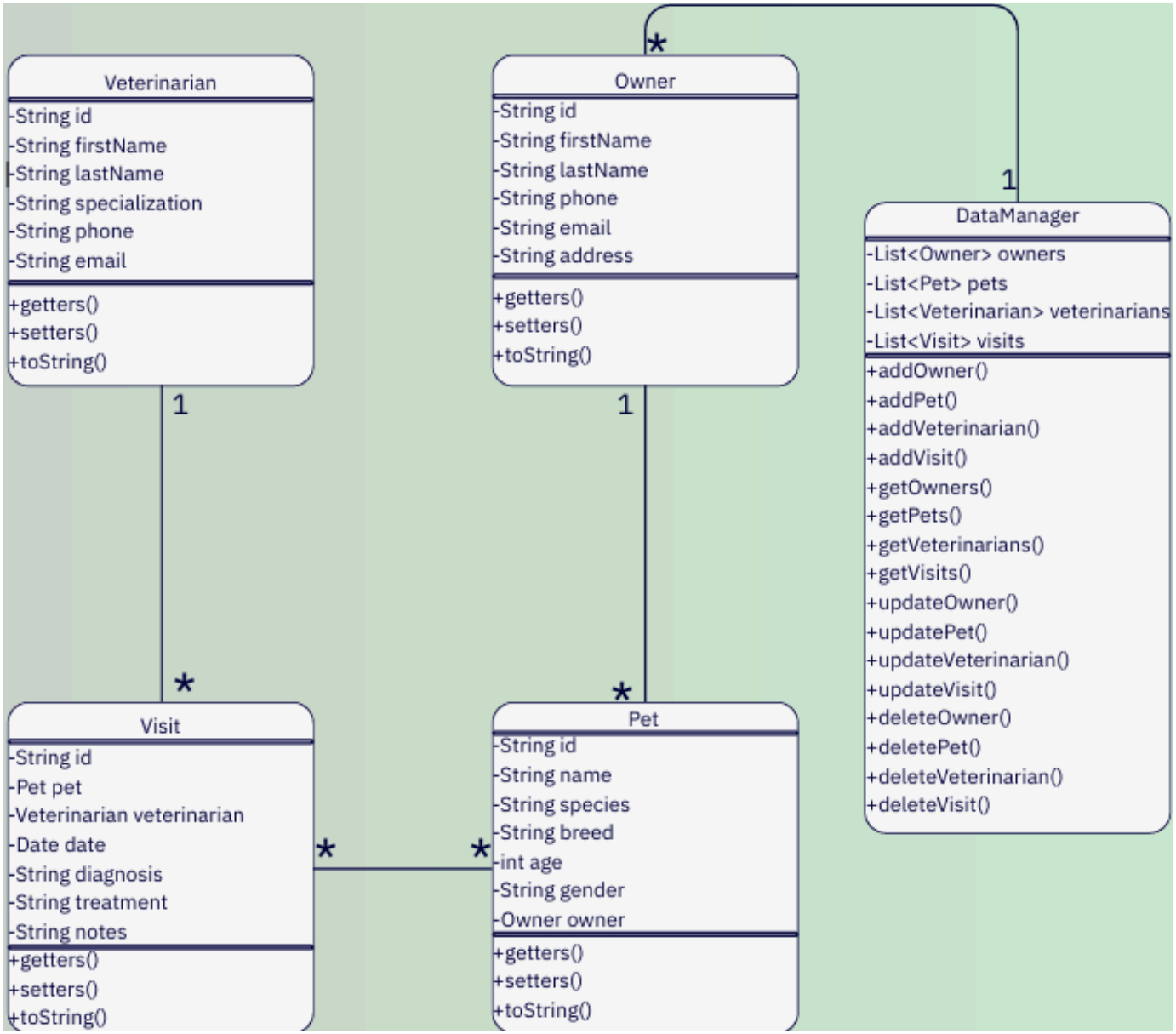
Récupère les données saisies dans les champs.

Crée un nouvel objet Pet basé sur ces données.

Gestion des données :
Le PetFormController fait appel à DataManager pour enregistrer ce nouvel objet Pet (par exemple dans une base de données ou une liste en mémoire).

Mise à jour de l'affichage :
Grâce à l'utilisation d'une ObservableList, toute modification des données (ajout, suppression, mise à jour) entraîne automatiquement la mise à jour de l'interface sans intervention supplémentaire.

8. Diagramme de classes UML



Explications:

1. Veterinarian, Owner, Pet, et Visit sont des classes avec leurs attributs et méthodes (getters, setters, toString).

2.DataManager est une classe qui gère des listes de ces entités et fournit des méthodes pour les manipuler (ajout, suppression, mise à jour, etc.).

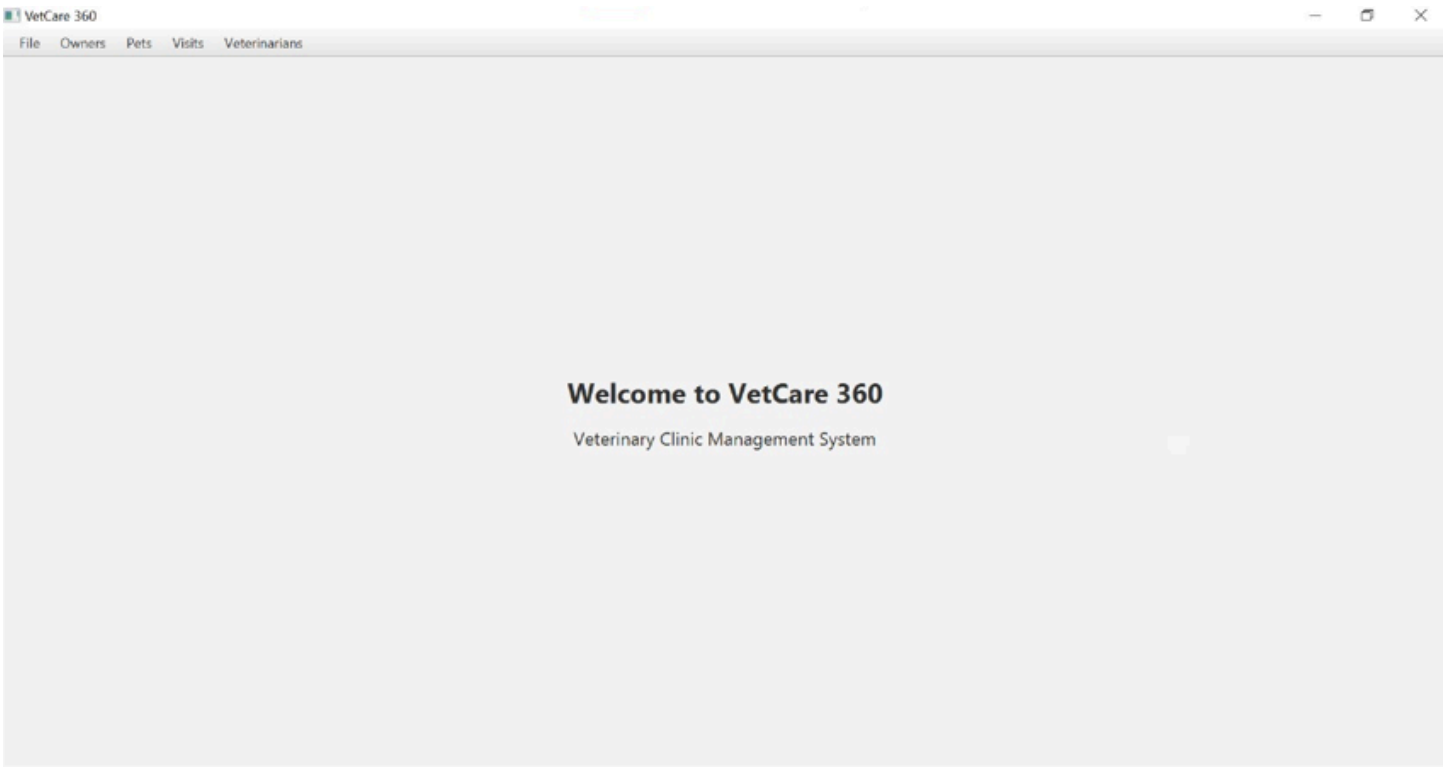
3.Les relations entre les classes :

- Un Visit est associé à un Pet et un Veterinarian.
- Un Pet est associé à un Owner.

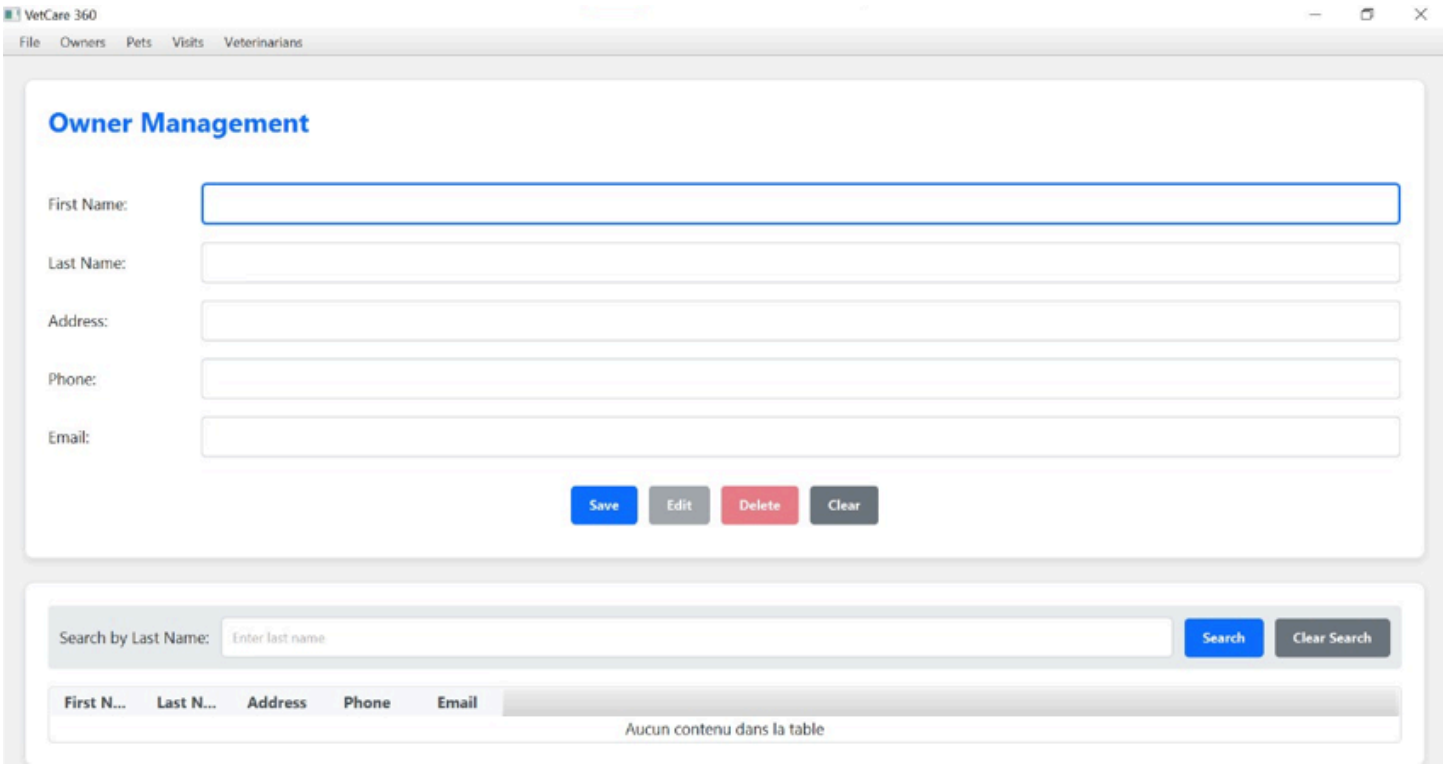
4.Encapsulation : Tous les attributs sont privés (- en UML), accessibles via getters/setters.

9.Exemples de résultats de tests

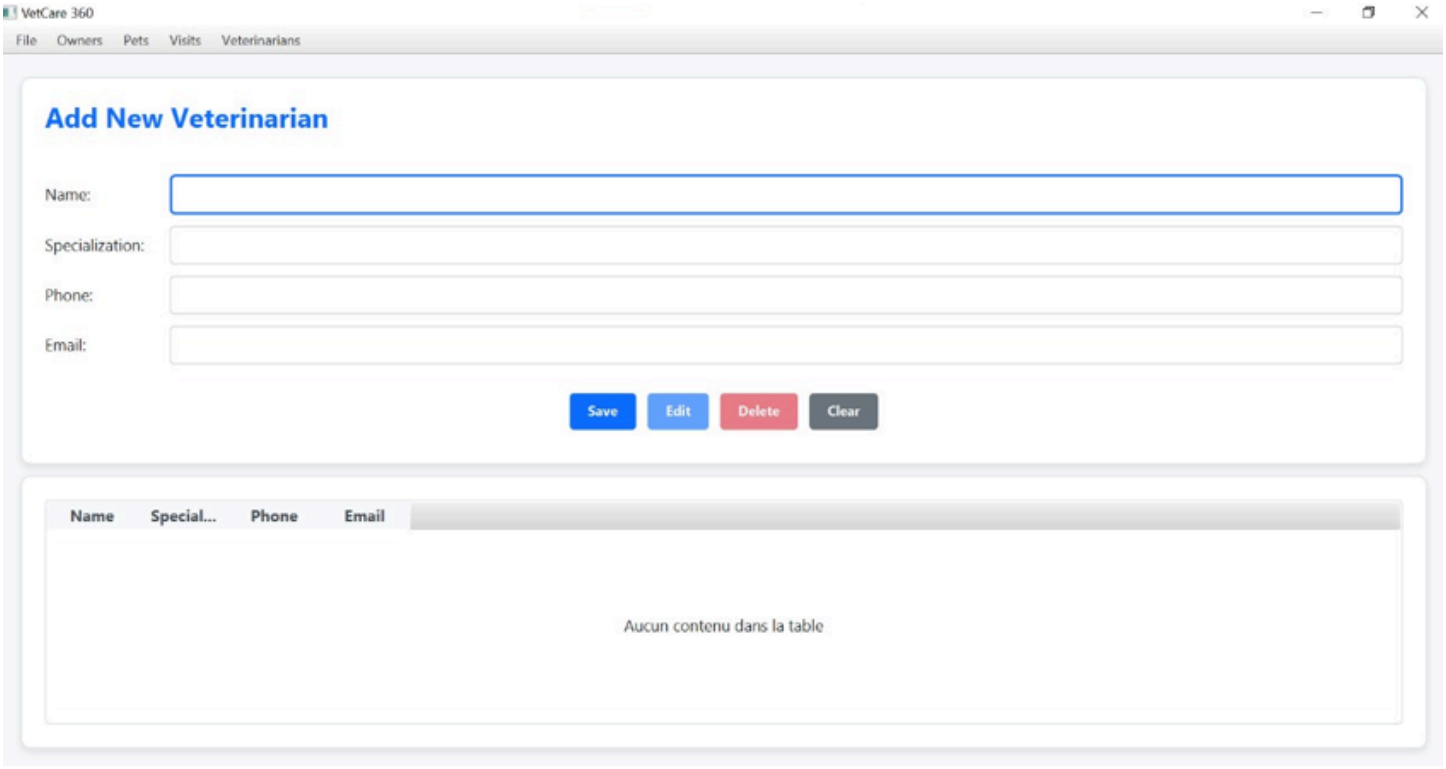
1.Page d’accueil



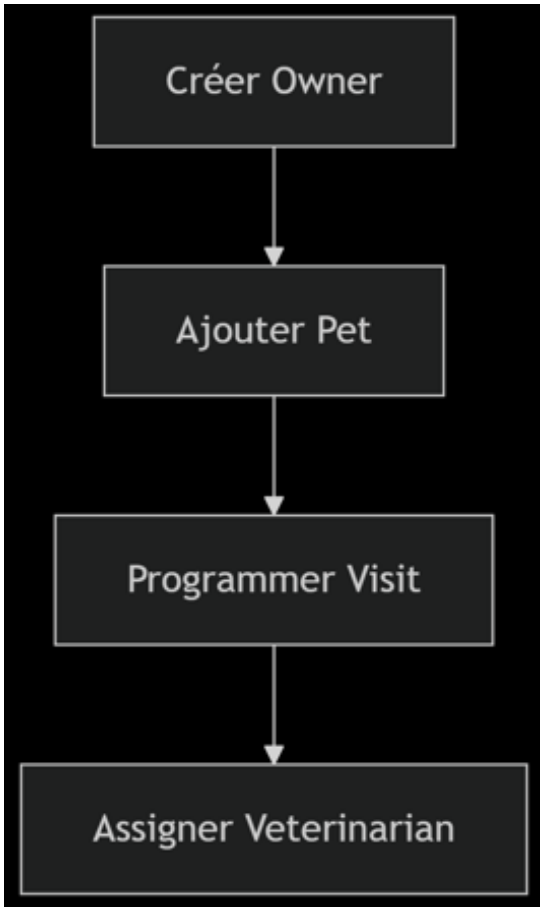
2.Gestion des propriétaires



3.Gestion des vétérinaires



10. Flux de Données



#Explications Détaillées des Fonctionnalités de VetCare 360 (Version Java/JavaFX)

1. Créer un Propriétaire

Cette fonctionnalité permet d'enregistrer les informations des propriétaires d'animaux dans le système.

- Processus :

- L'utilisateur remplit un formulaire avec les détails du propriétaire (nom, prénom, adresse, téléphone, email).

- Les données sont validées pour s'assurer qu'elles sont complètes et correctement formatées (par exemple, vérification du numéro de téléphone ou de l'email).

- Une fois validées, les informations sont enregistrées dans le système et associées à un identifiant unique.

- Utilité :

- Centralise les informations des clients pour un accès rapide.

- Permet de lier ultérieurement les animaux à leurs propriétaires.

---

2. Ajouter un Animal

Cette fonctionnalité permet d'ajouter un animal de compagnie au système et de l'associer à son propriétaire.

- Processus :

- L'utilisateur sélectionne d'abord un propriétaire existant dans une liste.

- Il renseigne ensuite les détails de l'animal (nom, espèce, race, âge, sexe).

- L'animal est enregistré et lié automatiquement au propriétaire choisi.

- Utilité :

- Facilite le suivi médical de chaque animal.

- Permet d'organiser les visites et les traitements en fonction des besoins spécifiques de l'animal.

---

3. Programmer une Visite

Cette fonctionnalité permet de planifier un rendez-vous médical pour un animal.

- Processus :

- L'utilisateur sélectionne un animal dans la liste des animaux enregistrés.

- Il choisit une date et une heure pour la visite, ainsi qu'un vétérinaire disponible.

- Il peut également ajouter des notes ou des symptômes pour préparer la consultation.

- La visite est enregistrée dans le calendrier du système.

- Utilité :

- Optimise l'organisation des rendez-vous pour éviter les conflits d'horaire.

- Permet de garder un historique complet des consultations pour chaque animal.

---

4. Assigner un Vétérinaire

Cette fonctionnalité permet de gérer les professionnels disponibles et de les assigner aux visites.

- Processus :

- L'utilisateur ajoute ou modifie les informations d'un vétérinaire (nom, spécialisation, coordonnées).

- Lors de la programmation d'une visite, il peut choisir un vétérinaire dans la liste des professionnels disponibles.

- Le système vérifie la disponibilité du vétérinaire pour éviter les doubles réservations.

- Utilité :

- Assure une répartition équilibrée des tâches entre les vétérinaires.

- Permet aux clients de savoir quel professionnel s'occupera de leur animal.

Leçons Apprises

1. Développement :

- Importance de l'architecture modulaire
- Valeur des tests automatisés
- Nécessité d'une documentation claire
- Avantages du versioning avec Git

2. Gestion de Projet :

- Planification des sprints
- Revue de code systématique
- Documentation continue
- Tests d'intégration réguliers

3. Compétences Acquises :

- Maîtrise de JavaFX
- Bonnes pratiques de développement Java
- Gestion de projet agile
- Développement d'interfaces utilisateur

Synthèse du Projet VetCare 360

Le projet VetCare 360 représente une solution logicielle complète et structurée pour la gestion des cliniques vétérinaires, alliant robustesse technique et ergonomie utilisateur. À travers ce développement, nous avons implémenté un système intégré qui répond aux besoins essentiels des professionnels du secteur vétérinaire tout en respectant les bonnes pratiques de l'ingénierie logicielle.

Si vous voulez découvrir notre application voici le liens vers le compte GitHub pour une meilleure expérience :

https://github.com/Alae06/vetcareJava

Merci.