

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ecole supériure de technologie de Nador  Filliére : Ingénierie logicielle et cybersécurité | | | | | | | | |
|  | | **Rapport de projet académique** | | | | |  | |
|  | Sous le sujet : | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | **Conception et développement d’une application de bureau en JavaFX pour la gestion complète d’une clinique vétérinaire – VetCare360** |  | | | |
|  | | | |  |  | | | |
|  | | |  | | |  | | |

**Encadré par :** Esabai Redoune

**Realisé par :** Essadiki Douae

**Présenter le :** 19/05/25

*ANNEE UNIVERSITAIRE : 2024/2025*

INTRODUCTION :

Présentation de projet :

VetCare 360 est un système d'information intégré développé en Java utilisant l'interface JavaFX. Il vise à digitaliser la gestion des cliniques vétérinaires et à faciliter les tâches quotidiennes des vétérinaires et des réceptionnistes, en fournissant une interface simple et facile à utiliser pour gérer :

* Données des propriétaires
* Informations sur les animaux
* dossiers de visite médicale

Ce système permet d'enregistrer et de mettre à jour avec précision les informations, de relier chaque animal à son propriétaire et de suivre les antécédents médicaux et les visites vétérinaires, améliorant ainsi la qualité du service au sein de la clinique.

Raisons pour choisir langage Java pour le développement de l’application :

Le langage **Java** a été choisi pour le développement de ce projet pour plusieurs raisons essentielles :

* **Stabilité et fiabilité** : Java est reconnu pour sa capacité à construire des applications robustes, performantes et **multi-plateformes** (Cross-platform), ce qui garantit un bon fonctionnement sur différents systèmes d’exploitation.
* **Support des interfaces graphiques** : Java offre des outils avancés comme **JavaFX**, permettant de concevoir des interfaces utilisateur modernes, conviviales et intuitives.
* **Intégration facile avec les bases de données** : Java dispose de bibliothèques puissantes et de frameworks facilitant l’interaction avec les bases de données de manière efficace et sécurisée.
* **Grande communauté et support actif** : Avec une vaste communauté de développeurs, Java bénéficie d’un **écosystème riche** en ressources, en outils et en solutions, ce qui accélère le processus de développement tout en garantissant une haute qualité.

OBJICTIFS GENERAUX ET SOUS-OBJECTIF :

Objectifs généraux du projet :

* Numériser la gestion des cliniques vétérinaires grâce à un système simple d'utilisation et efficace.
* Simplifiez les opérations quotidiennes telles que l’ajout d’informations sur le propriétaire et l’animal et l’enregistrement des visites.
* Fournir une interface graphique organisée qui aide les utilisateurs à accéder rapidement aux informations.
* Améliorer la qualité des services médicaux en suivant avec précision et rapidité l’historique des animaux.

Sous-objectifs :

* Gestion des données du propriétaire : enregistrez, modifiez, supprimez et consultez les informations de chaque propriétaire.
* Lier les animaux à leurs propriétaires : Chaque animal est lié à son propriétaire au sein de la base de données.
* Enregistrement et suivi des visites vétérinaires : précision de la date, du type d'examen ou de traitement et du médecin responsable.
* Consultez des rapports et des analyses : tels que le nombre de visites par animal, les animaux par espèce, etc.

ANALYSE FONCIONELLE :

Dans cette phase, nous analysons les fonctionnalités de base offertes par l'application Ve conçue tCare 360, pour répondre aux besoins des cliniques vétérinaires grâce à des interfaces graphiques simplifiées utilisant des bases de données JavaFX et SQL.

Fonctions principales de l’application:

**1. Enregistrement d’un nouveau propriétaire :**

Description :

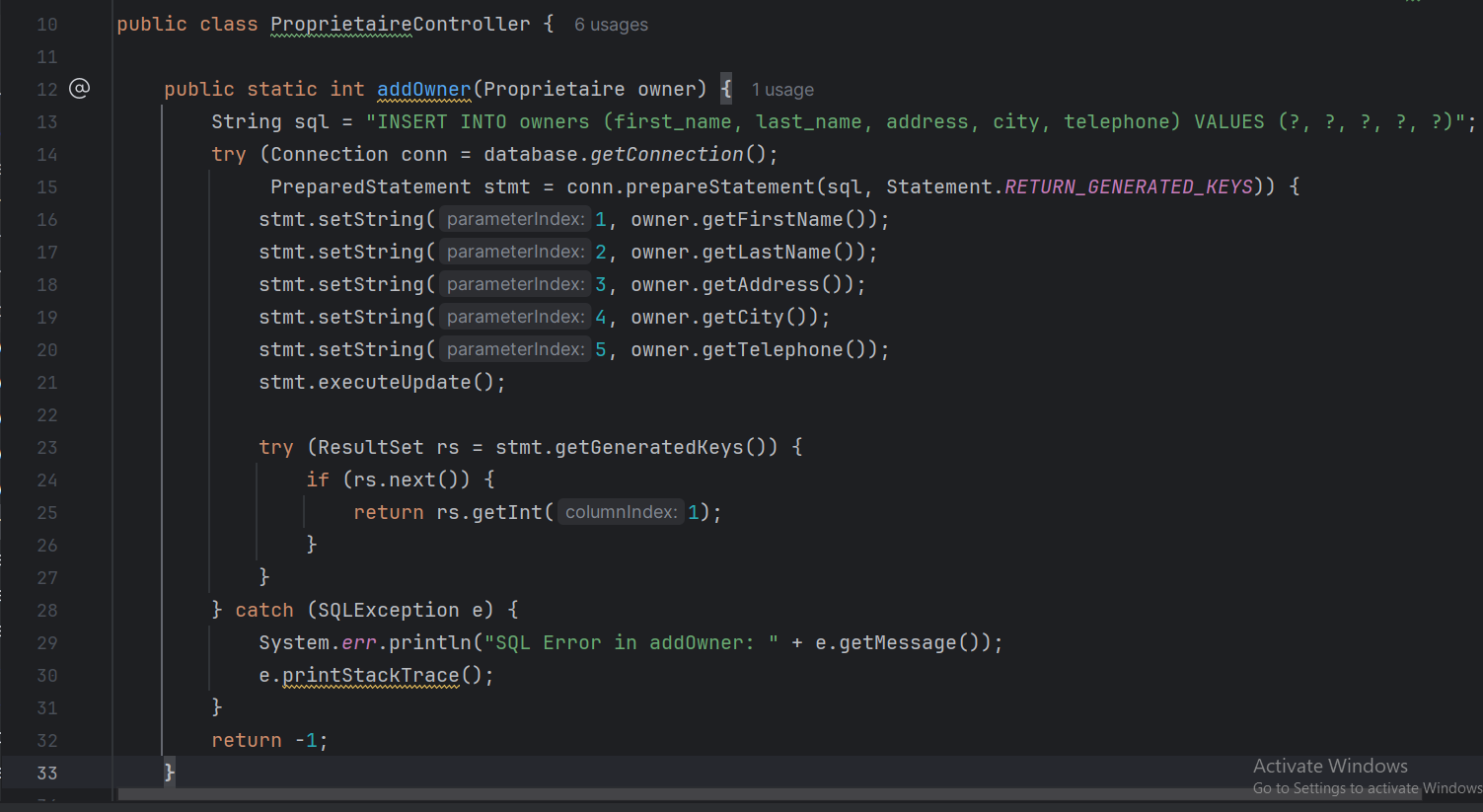
* L’interface permet de saisir le nom, prénom, adresse, ville et numéro de téléphone d’un nouveau propriétaire.

Fonction technique :

* Validation des champs saisis.
* Enregistrement des données dans la table owners de la base de données.
* Génération automatique d’un identifiant unique (ID).

Objectif :

* Créer une fiche propriétaire complète pour associer facilement les animaux à venir.



**2. Ajout d’un animal :**

Description :

* Une fois le propriétaire sélectionné, l’utilisateur peut ajouter un animal (nom, espèce/type, date de naissance).

Fonction technique :

* L’animal est lié au propriétaire via une clé étrangère (foreign key).
* Enregistrement des données dans la table animals.
* Contrôle de la saisie obligatoire de tous les champs.

Objectif :

* Suivre chaque animal et l’associer à son propriétaire.



**3. Enregistrement d’une visite vétérinaire :**

Description :

* Formulaire contenant la date de visite, le vétérinaire concerné et la description du cas.

Fonction technique :

* Chaque visite est liée à un animal via une relation SQL.
* Insertion dans la table visites.
* Affichage chronologique des visites.

Objectif :

* Assurer un suivi médical structuré et historisé de chaque animal.



**4. Affichage de l’historique des visites :**

Description :

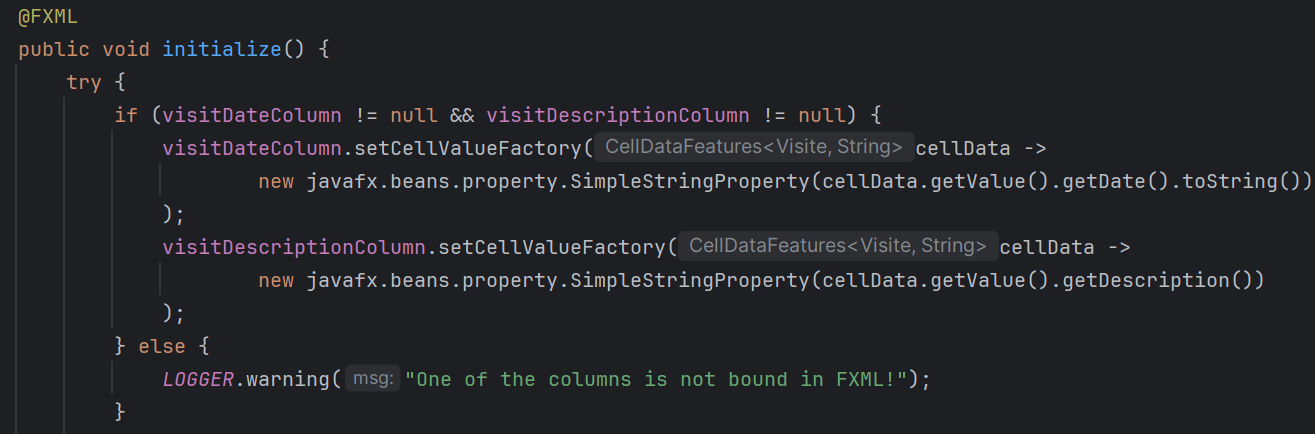
* Liste des visites associées à un animal donné, présentée sous forme de tableau.

Fonction technique :

* Utilisation du composant TableView de JavaFX.
* Récupération des données via requêtes SQL filtrées.
* Affichage dynamique selon l’animal sélectionné.

Objectif :

* Accès rapide à l’historique médical complet pour une meilleure prise de décision.



**5. Recherche et filtrage :**

Description :

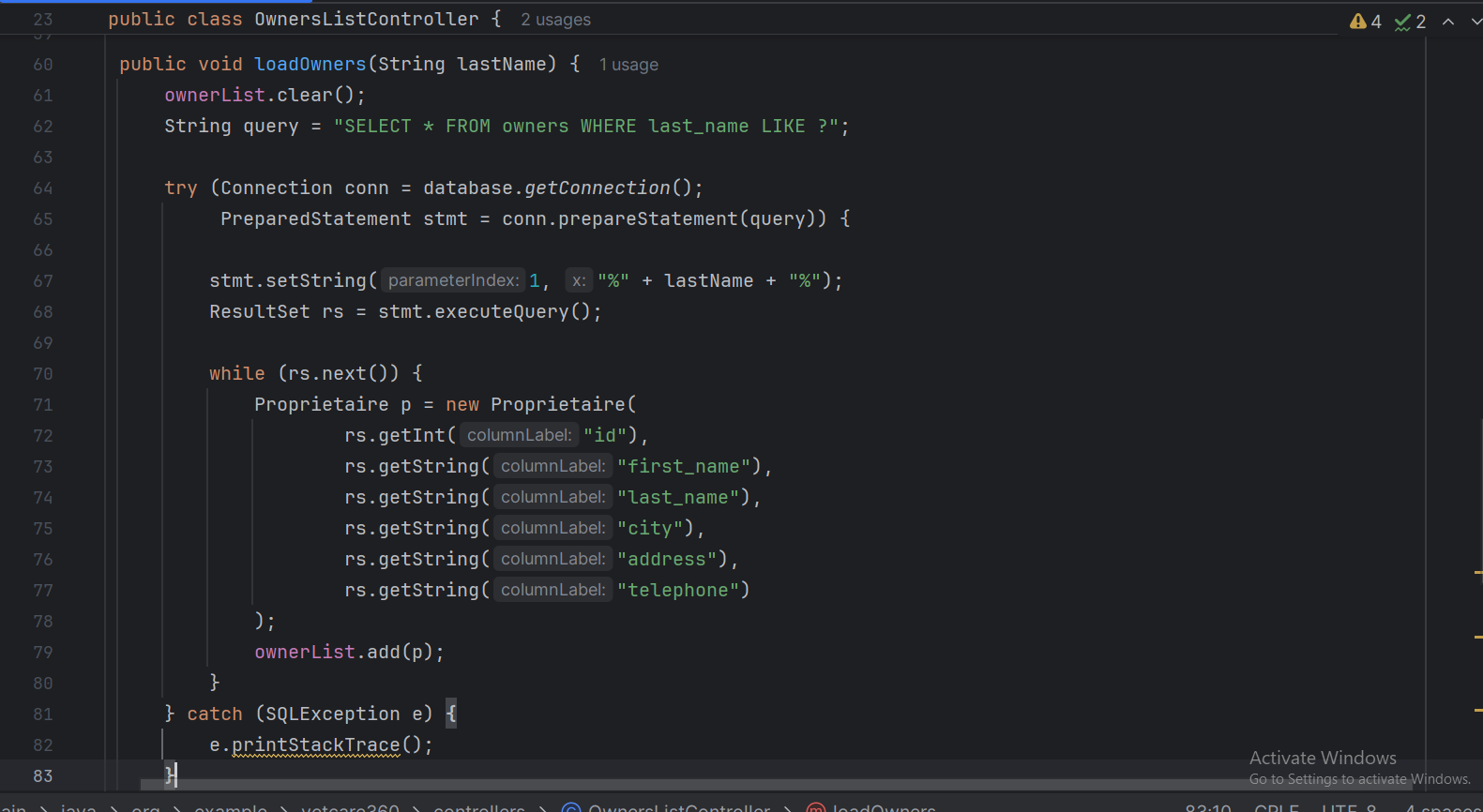
* Recherche par nom du propriétaire.

Fonction technique :

* Champs de recherche liés à des requêtes SQL avec clauses LIKE.
* Mise à jour automatique de l’affichage.

Objectif :

* Faciliter la navigation dans les données et améliorer l’expérience utilisateur.



**6. Mise à jour des informations (Modification):**

Description :

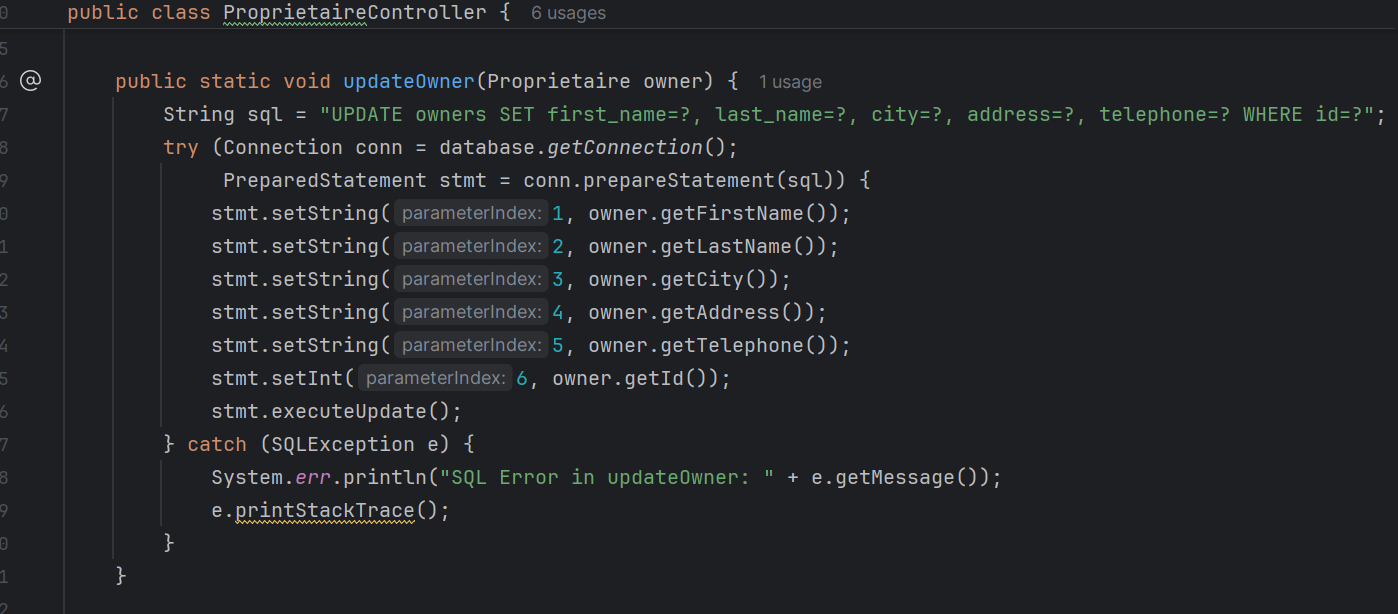
* L’utilisateur peut modifier les informations d’un propriétaire ou d’un animal via un formulaire interactif.

Fonction technique :

* Enregistrement des modifications avec une requête SQL UPDATE.
* Affichage en temps réel des nouvelles données.

Objectif :

* Maintenir la base de données à jour et exacte.



**7. Suppression de données (Suppression):**

Description :

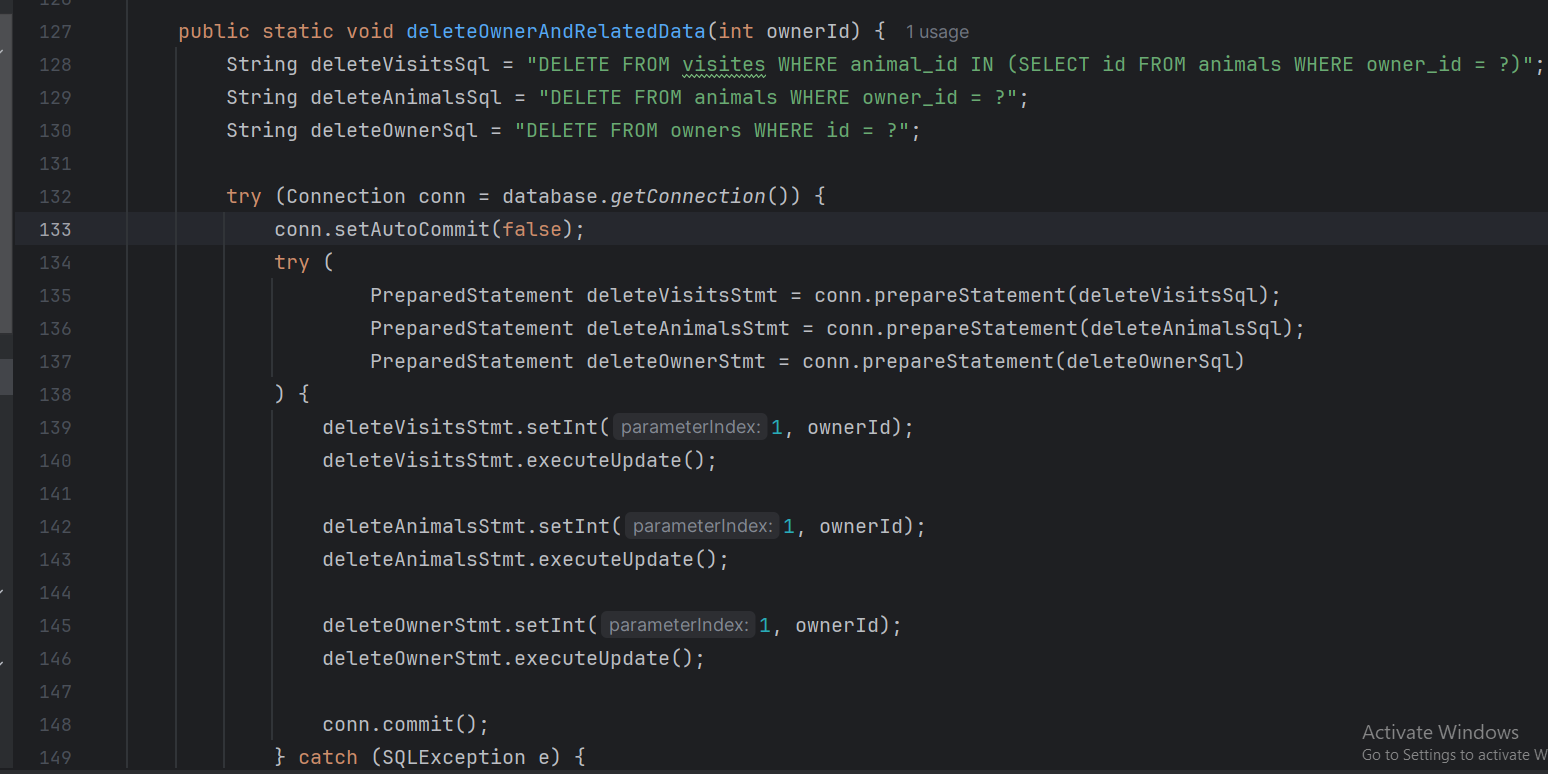
* Possibilité de supprimer un propriétaire, un animal ou une visite.

Fonction technique :

* Exécution de requêtes DELETE sécurisées.
* Demande de confirmation avant suppression pour éviter les erreurs.

Objectif :

* Nettoyer la base de données et supprimer les informations obsolètes ou incorrectes.



ANALYSE DE LA BASE DE DONNEES:

1. Type de base de données utilisée :

Le projet VetCare360 utilise une base de données relationnelle : MySQL.

2. Les principales tables :

Table : owners

id INT (PK, AI)

first\_name VARCHAR(50)

last\_name VARCHAR(50)

address VARCHAR(100)

city VARCHAR(50)

telephone VARCHAR(20)

Table : animals

id INT (PK, AI)

name VARCHAR(50)

birth\_date DATE

type VARCHAR(30)

owner\_id INT (FK)

Table : visites

id INT (PK, AI)

animal\_id INT (FK)

VisitDate DATE

description TEXT

Table : veterinarian

Id INT (PK , AI)

first\_name VARCHAR(50)

last\_name VARCHAR(50)

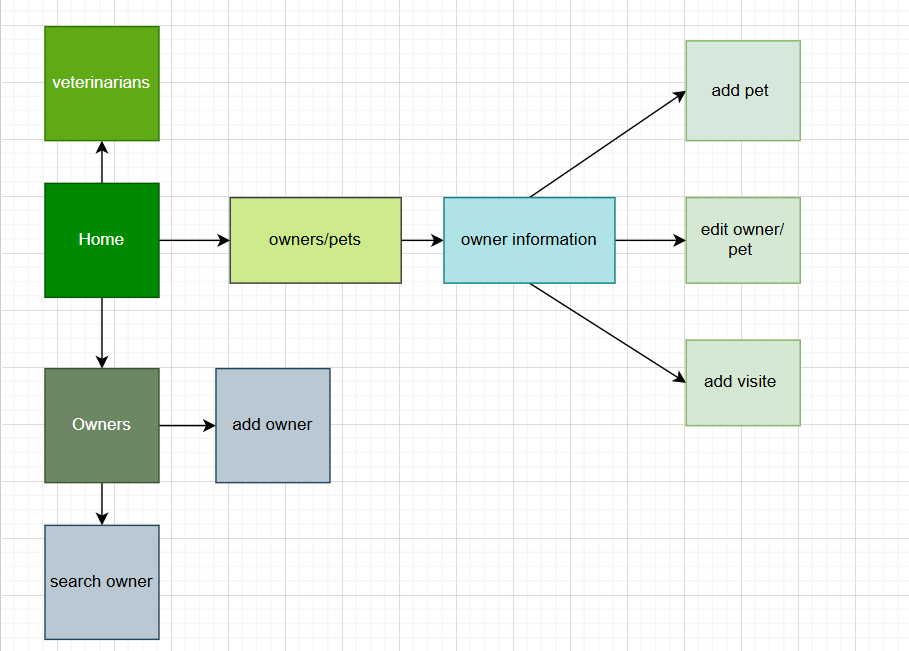
Specialties VARCHAR(50)

3. Relations entre les entités :

* Un propriétaire peut avoir plusieurs animaux (relation 1→n)
* Un animal peut avoir plusieurs visites (relation 1→n)

FLUX DE NAVIGATION :

Ce diagramme montre comment un utilisateur navigue entre les différentes interfaces de l’application VetCare360, développée avec JavaFX. L’utilisateur commence par l’écran d’accueil (Home), qui permet d’accéder à plusieurs sections : la gestion des vétérinaires, la liste des propriétaires (Owners), ou directement à la vue combinée des propriétaires et de leurs animaux (owners/pets). Depuis l’écran des propriétaires, il peut soit rechercher un propriétaire existant (search owner), soit en ajouter un nouveau (add owner). Une fois qu’un propriétaire est sélectionné, l’utilisateur accède à la page de ses informations détaillées (owner information). Cette page permet ensuite plusieurs actions : ajouter un animal (add pet), modifier les informations du propriétaire ou de l’animal (edit owner/pet), ou bien enregistrer une nouvelle visite vétérinaire (add visite). Chaque flèche du diagramme représente une transition d’interface déclenchée par une interaction utilisateur, comme un clic sur un bouton ou un choix dans une liste.



INTERFACES UTILISATEUR :

Figure 1 – Interface de recherche des propriétaires :

Cette interface permet de rechercher un propriétaire à l’aide de son nom de famille. L'utilisateur saisit le nom puis lance la recherche.

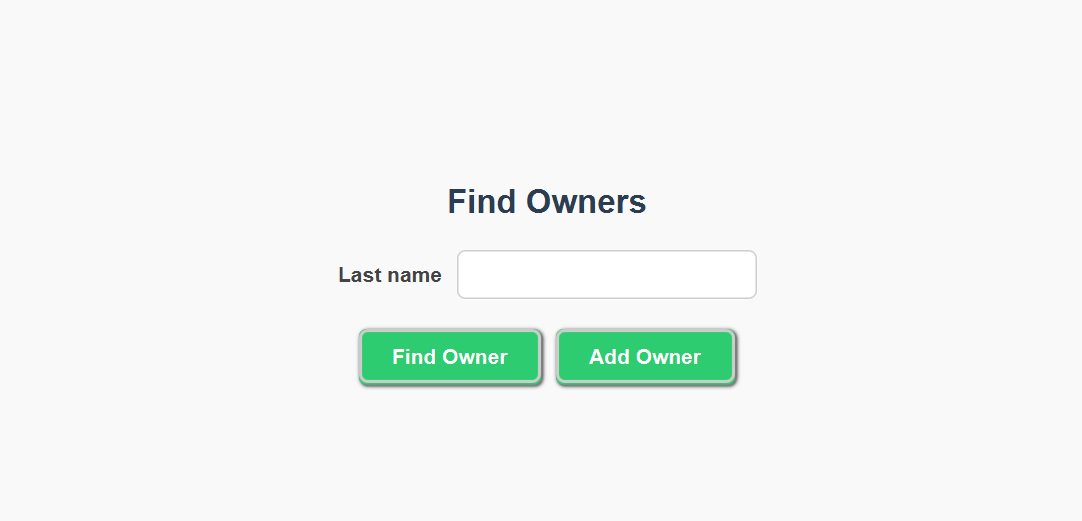
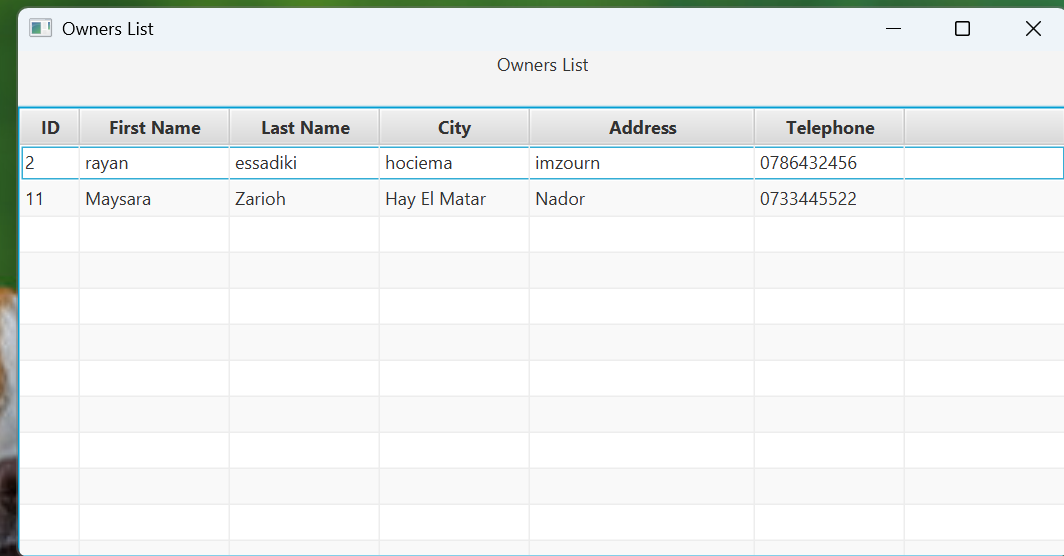


Figure 2 – Liste des propriétaires :

Cette interface affiche les résultats de recherche sous forme de tableau contenant les informations principales du propriétaire : prénom, nom, ville, adresse et téléphone.

Figure 3 – Détails du propriétaire et de ses animaux :

Affiche les informations complètes d’un propriétaire sélectionné, ainsi que la liste de ses animaux .

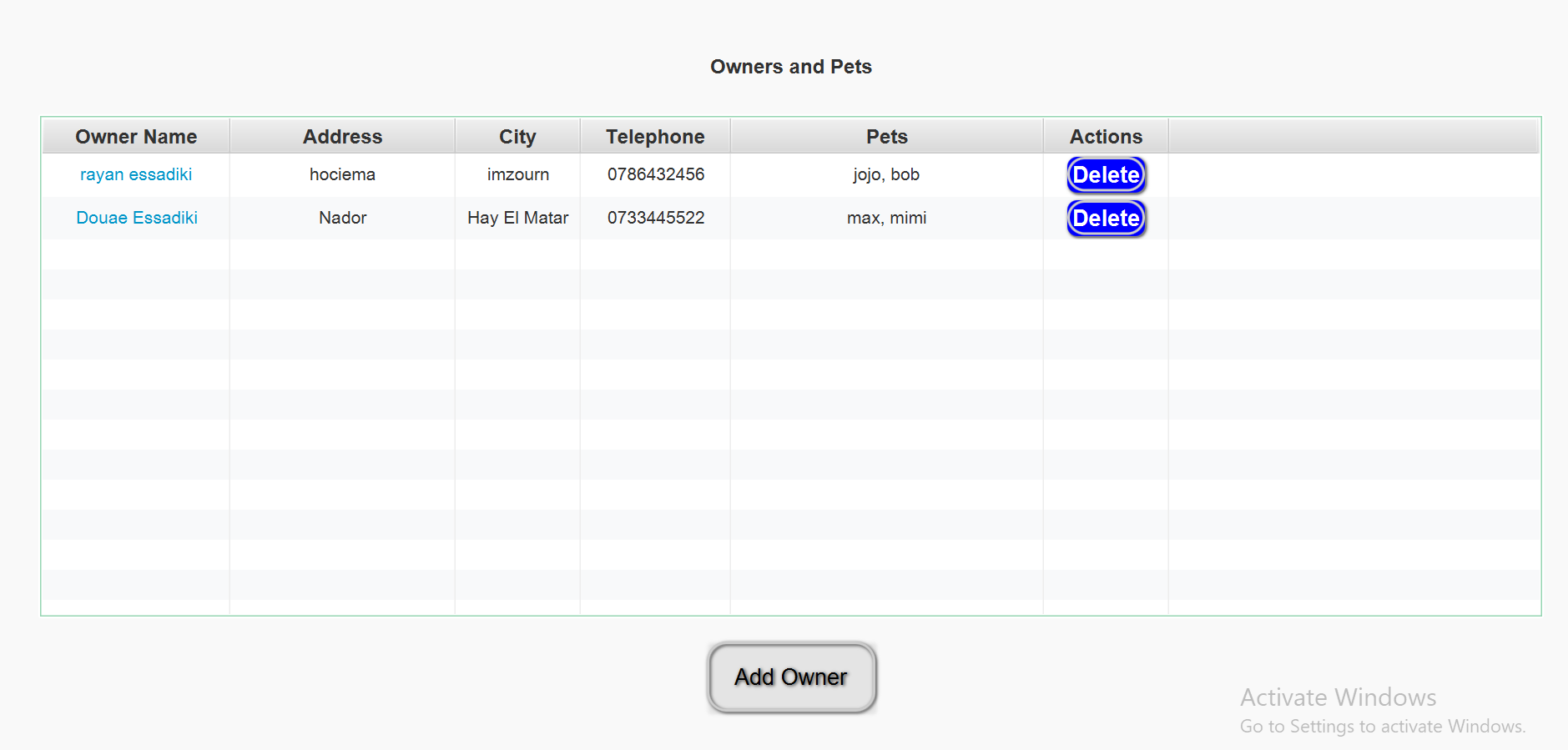
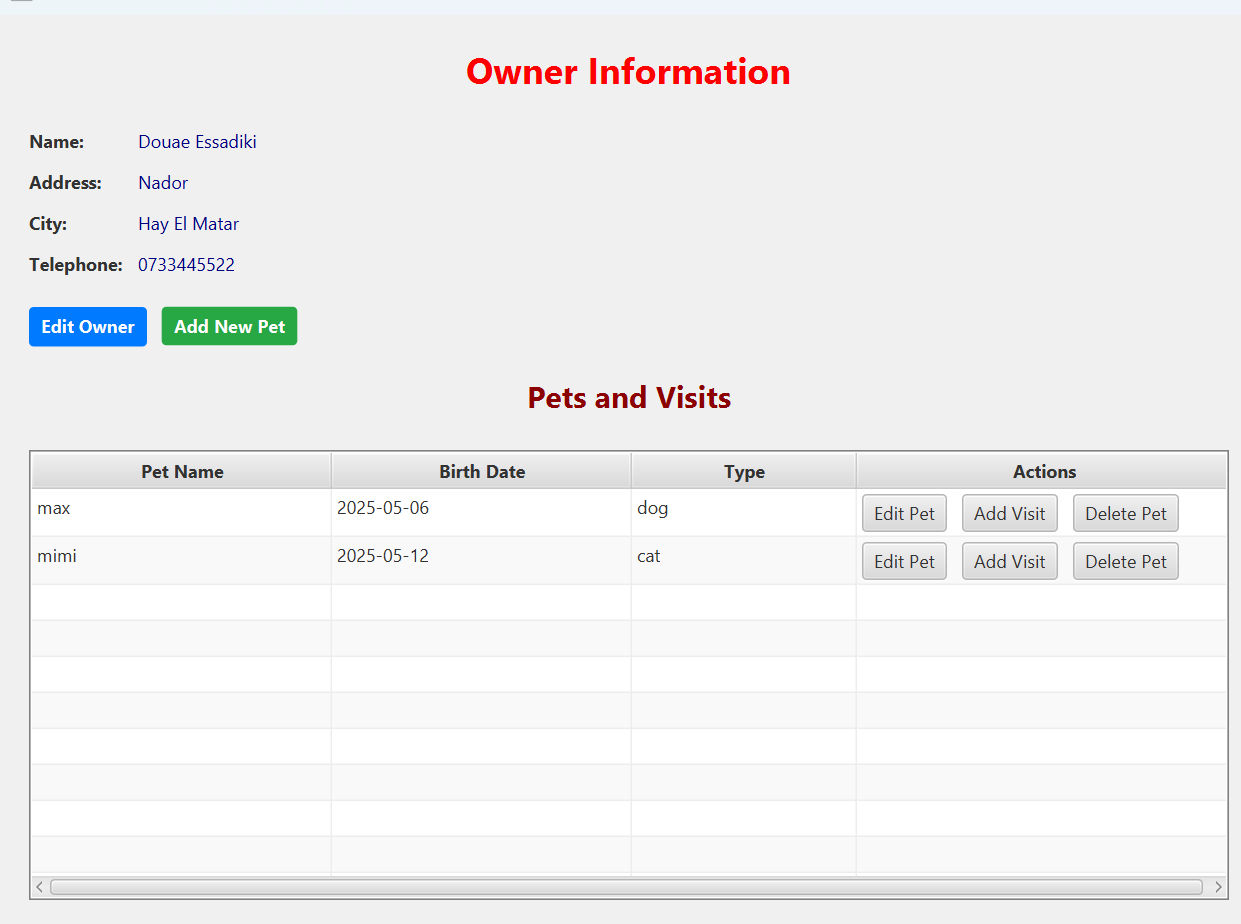


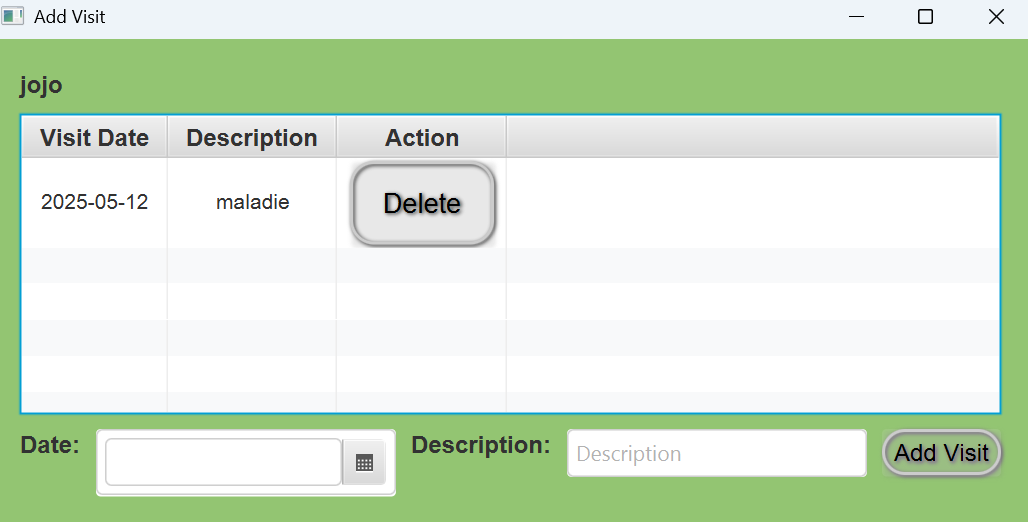
Figure 4 – Formulaire d’ajout ou de modification d’un animal :

Permet d’ajouter un nouvel animal ou de modifier les données d’un animal existant (nom, espèce, date de naissance...).

Figure 5 – Enregistrement d’une visite vétérinaire:

Cette interface permet au vétérinaire d’enregistrer une nouvelle visite pour un animal avec les détails nécessaires (date de visite, description).

Figure 6 – Historique des visites :

Affiche l’ensemble des visites effectuées pour un animal donné, utile pour le suivi médical.

TECHNOLOGIES UTILISEES DANS LE PROJET :

Le projet VetCare 360 a été développé en utilisant les technologies Java pour construire une application de bureau robuste et interactive. Voici les principales technologies et outils utilisés:

Java SE & JavaFX

Le projet est basé sur Java SE pour la logique métier et utilise JavaFX pour la création de l'interface graphique. JavaFX offre une approche moderne et orientée objet pour construire des interfaces utilisateur élégantes et réactives. Ce choix permet d’avoir une application de bureau multi-plateforme avec une expérience utilisateur fluide.

Environnement de développement (IDE)

L’environnement de développement utilisé est IntelliJ IDEA, un IDE puissant pour le développement Java. Il offre une excellente prise en charge de JavaFX, le débogage, l’intégration Maven, et facilite la navigation dans les fichiers FXML.

Outil de gestion de projet : Maven

Le projet utilise Maven pour la gestion des dépendances, le build et l’organisation du projet. Maven permet de structurer le projet proprement et d'intégrer facilement des bibliothèques externes .

Base de données & JDBC

L’application utilise JDBC (Java Database Connectivity) pour communiquer avec une base de données relationnelle (par exemple MySQL). Les classes DAO utilisent JDBC pour exécuter des requêtes SQL et manipuler les données (CRUD pour les propriétaires, animaux, et visites).

Contrôle de version : Git

Git est utilisé pour le versionnement du projet. Il permet de sauvegarder l’historique du développement, de collaborer en équipe, et de gérer les versions de manière efficace à travers une plateforme comme GitHub.

CEATION DES VUES AVEC JAVAFX ET FXML :

**Conception des interfaces avec JavaFX**

L’interface graphique de l’application **VetCare360** est réalisée en **JavaFX**, une bibliothèque moderne de Java pour les applications desktop. JavaFX offre une structure claire en suivant le modèle **MVC** (Modèle-Vue-Contrôleur), ce qui facilite la séparation entre la logique métier et la présentation graphique.

**Utilisation de FXML**

Toutes les vues sont définies à l’aide de fichiers **FXML**, ce qui permet une conception déclarative de l’interface. Ces fichiers sont créés à l’aide de **SceneBuilder** ou manuellement, puis liés à des contrôleurs Java spécifiques.

Voici quelques exemples de fichiers FXML utilisés :

* OwnersSearch.fxml : Interface de recherche d’un propriétaire
* OwnersList.fxml : Liste des résultats (propriétaires)
* OwnerDetails.fxml : Informations du propriétaire et ses animaux
* AnimalForm.fxml : Formulaire pour ajouter ou modifier un animal
* VisiteForm.fxml : Interface pour enregistrer une nouvelle visite

Chaque fichier contient des composants JavaFX tels que TextField, Button, TableView, etc., avec un identifiant fx:id permettant de les manipuler dans le contrôleur.

**Liaison des vues avec les données (Controllers)**

Chaque fichier FXML est relié à un contrôleur (Controller) via l’attribut fx:controller. Ce contrôleur est responsable de la logique liée à la vue (chargement des données, gestion des actions, etc.).

Par exemple, dans OwnersSearchController.java 

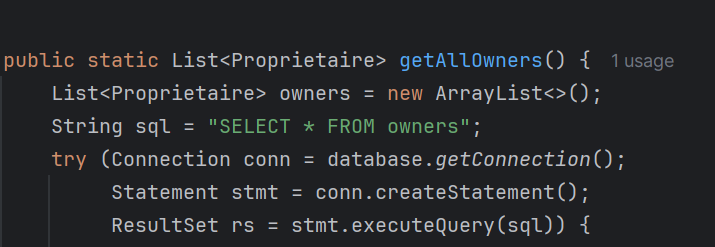
BACKEND :

1. Structure approuvée (modèles, contrôleurs, services)

Le code du projet est organisé à l'aide de la structure MVC (Modèle - Vue - Contrôleur) de sorte que :

* Modèle : contient des objets représentant des données telles que la propriété, l'animal et la visite.
* Contrôleur : gère l'interaction avec l'interface graphique (JavaFX) et appelle des services ou des bases de données lorsque cela est nécessaire.
* Service/DAO : Responsable de l'exécution des opérations liées à la base de données (ajout, modification, suppression, interrogation).

2. Connexion à la base de données (JDBC)

La connexion à la base de données se fait via JDBC (Java Database Connectivity) de manière directe via la classe database.java qui crée la connexion à la base de données MySQL.

3. Gestion des exceptions :

Les blocs try-catch sont utilisés autour des commandes de connexion à la base de données pour garantir la stabilité de l'application et qu'elle ne s'arrête pas lorsqu'une erreur se produit.



TESTS :

Dans le projet VetCare 360, des tests manuels ont été principalement utilisés pour vérifier les différentes fonctionnalités de l'application pendant le développement.

Là où cela a été vérifié :

* Tester manuellement les fonctions
* Essayez de rechercher le propriétaire en utilisant son nom.
* Essayez d'ajouter un nouveau propriétaire et assurez-vous qu'il apparaît dans la liste des propriétaires.
* Saisissez les données de l'animal et liez-les à son propriétaire.
* Enregistrez une nouvelle visite et assurez-vous qu'elle est associée au bon animal.

DEFIS ET SOLUTIONS :

Problèmes d'affichage des données dans l'interface:

Parfois, les données ne se chargent pas correctement dans une TableView parce que les colonnes ne sont pas initialisées correctement ou que vous oubliez de lier l'ObservableList.

Cela a été résolu en garantissant que PropertyValueFactory était utilisé et en liant les données dans initialize().

Gestion des exceptions de base de données

Les exceptions JDBC sont levées lorsqu'une connexion échoue ou qu'une erreur de requête se produit.

Les blocs try-catch sont utilisés avec les erreurs imprimées dans System.err pour faciliter le suivi des erreurs, avec des messages améliorés suggérés à l'utilisateur ultérieurement.

CONCLUSION :

À l'issue de ce projet, l'application VetCare 360 a été développée comme une solution numérique efficace pour la gestion des propriétaires, des animaux de compagnie et des visites vétérinaires, en utilisant les technologies Java SE et JavaFX. Les interfaces utilisateur sont

construites de manière interactive et conviviale, en mettant l’accent sur une navigation facile entre les pages et un accès rapide aux données.

Au cours des étapes de développement, les principes de la programmation orientée objet (POO) ont été appliqués et JDBC a été utilisé pour gérer la base de données, en plus de diviser le projet en couches claires telles que les modèles et les contrôleurs.

Nous avons été confrontés à de nombreux défis pour relier les données entre les tables et garantir que les données étaient affichées correctement dans les interfaces graphiques, mais nous avons pu les surmonter grâce à des tests manuels continus et à des améliorations de l'architecture logicielle.

RESUME :

Ce projet nous a permis d'appliquer nos connaissances en programmation Java, de concevoir des interfaces graphiques interactives à l'aide de JavaFX et de les connecter à une base de données réelle. Cela a également amélioré notre compréhension de la structuration des applications professionnelles et du travail dans une approche structurée basée sur MVC.

***Je tiens à remercier sincèrement mon professeur “Esbai Redoune “ de m’avoir donné l’opportunité de réaliser ce projet, qui a grandement contribué à mon développement personnel .***