



Rapport du Mini-Projet :

VetCare 360

Filière : Ingénierie Logicielle et Cybersécurité .

Année universitaire : 2024-2025.

Réalisé par :

Amadid Soulayman .

Ouazzou Mohamed Amine .

Encadré par :

Mr. Redouane Esbai .

Établissement :

École Supérieure de la Technologie Nador .



Sommaire

• Remerciements	1
• Résumé du projet	2
• Introduction	3
• Cahier des Charges	4
• Analyse et Conception	6
• Réalisation	8
• Résultats	10
• Bilan personnel	12
• Outils utilisés	13

• Perspectives futures	14
• Conclusion	15
• Annexes	16

Remerciements

Nous tenons à remercier chaleureusement notre encadrant Mr Redouane Esbai pour son accompagnement, sa disponibilité et ses précieux conseils tout au long de la réalisation de ce projet.

Nous exprimons également notre gratitude envers notre établissement et l'ensemble du corps enseignant pour la qualité de la formation dispensée, qui nous a permis d'acquérir les compétences nécessaires à la mise en œuvre de ce projet.

Enfin, merci à nos proches pour leur soutien moral tout au long de ce travail. 

Résumé du projet

Le projet **VetCare 360** est une application web développée pour faciliter la gestion d'une clinique vétérinaire.

Son objectif principal est de permettre l'administration efficace des propriétaires d'animaux, des vétérinaires ainsi que des visites médicales, à travers une interface simple .

Pour sa réalisation, nous avons utilisé **MongoDB** pour la base de données, **Express.js** pour la création du backend, **React.js** pour le développement

du frontend , et **Node.js** pour la gestion du serveur. Le style visuel a été renforcé avec **Bootstrap** pour assurer une interface responsive et moderne.

À l'issue du projet, **VetCare 360** propose un système complet permettant d'ajouter, modifier et consulter les informations des propriétaires, animaux, vétérinaires et visites, avec une expérience utilisateur fluide et professionnelle.

L'application offre plusieurs fonctionnalités clés telles que la recherche d'un propriétaire par son nom, l'ajout et la gestion des animaux associés à chaque propriétaire, la visualisation et l'ajout de visites médicales pour chaque animal, ainsi que l'affichage d'une liste complète des vétérinaires avec leurs spécialités.

Introduction

Pourquoi ce projet ?

Dans le domaine vétérinaire, la gestion quotidienne des informations relatives aux propriétaires d'animaux, aux vétérinaires et aux visites médicales est une tâche essentielle mais complexe.

De nombreuses cliniques vétérinaires utilisent encore des méthodes traditionnelles basées sur des registres papier ou des outils bureautiques classiques, ce qui rend l'accès aux informations lent .

Face à ce constat, nous avons choisi de développer **VetCare 360**, une application web moderne , permettant une gestion rapide de toutes les données d'une clinique vétérinaire.

Ce projet nous a également permis de mettre en pratique nos compétences en développement web, en utilisant MongoDB, Express.js, React.js, Node.js , et de créer une solution complète répondant aux besoins réels des professionnels du secteur vétérinaire.

Importance de gérer une clinique vétérinaire avec une application ?

Une application moderne garantit une accessibilité rapide aux informations, que ce soit depuis un ordinateur de bureau ou un appareil mobile, offrant ainsi plus de flexibilité aux vétérinaires et à leur équipe.

L'utilisation d'une application comme **VetCare 360**, permet d'automatiser et de centraliser l'ensemble de ces processus. Cela offre de nombreux avantages, notamment ;

- Un gain de temps dans la recherche et la mise à jour des informations
- Une meilleure organisation interne et un suivi précis des consultations
- Une amélioration de la relation client grâce à la rapidité et la fiabilité des services



Cahier des Charges

1. Objectifs fonctionnels :

Le projet **VetCare 360** vise à développer une application web capable d'assurer la gestion complète d'une clinique vétérinaire. Les principales fonctionnalités attendues sont :

- Gestion des propriétaires :

Permettre l'ajout, la modification et la suppression des informations relatives aux propriétaires d'animaux (nom, prénom, adresse, ville, numéro de téléphone).

- Gestion des animaux :

Permettre l'ajout d'un ou plusieurs animaux associés à chaque propriétaire. Chaque animal possède ses propres informations (nom, date de naissance, type).

- Gestion des vétérinaires :

Afficher la liste des vétérinaires disponibles ainsi que leurs spécialités.

- Gestion des visites médicales :

Permettre l'enregistrement des visites effectuées pour chaque animal (date de visite, description) afin d'assurer un suivi médical précis.

2. Contraintes techniques :

Afin de garantir la qualité, la fiabilité et la modernité de l'application, plusieurs contraintes techniques ont été imposées :

- Utilisation de la stack MERN :

Le projet est basé sur la stack MERN (MongoDB, Express.js, React.js, Node.js), qui permet de réaliser une application web complète, rapide et évolutive.

- Interface simple et responsive avec Bootstrap :

L'interface utilisateur doit être intuitive, ergonomique et s'adapter à tous les types d'écrans , en utilisant le framework **Bootstrap**.



Analyse et Conception

1. Présentation des modèles :

Nous avons défini quatre modèles principaux pour représenter les différentes entités de la clinique vétérinaire :

- ✓ Owner :

Avec les attributs suivants :

- firstName
- LastName
- Address
- City
- Phone

- ✓ Pet :

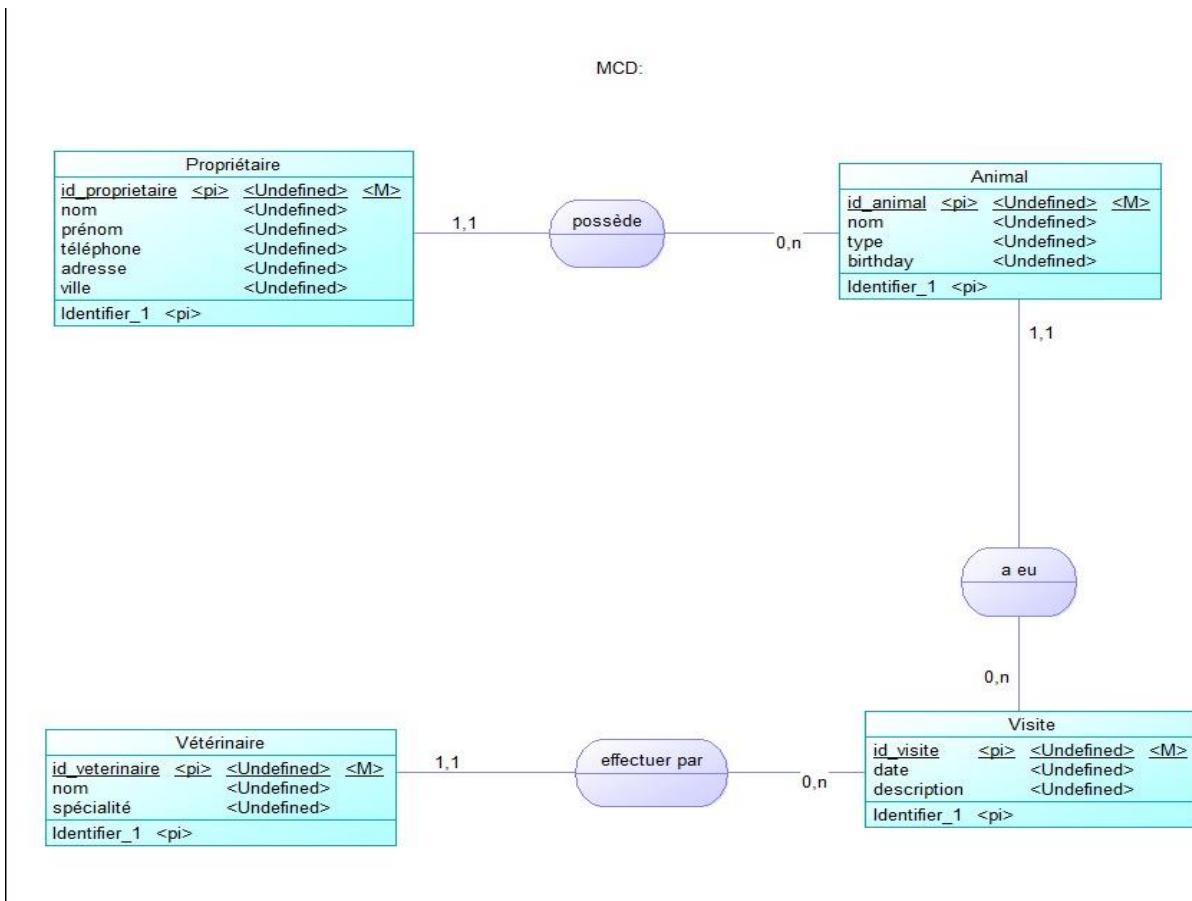
Avec les attributs suivants :

- name
- BirthDate
- Type

- ✓ Vétérinaire :
- Name
- Specialty

- ✓ Visit :
- Date
- Description

👉 Le Modèle Conceptuel de Données (MCD) :



2. Architecture MERN :

L'application **VetCare 360** suit l'architecture **MERN** :

- **MongoDB** : base de données NoSQL qui stocke les informations sous forme de documents .

- **Express.js** : framework backend léger qui permet de créer les APIs REST pour gérer les échanges entre la base de données et le frontend.
- **React.js** : bibliothèque JavaScript utilisée pour construire l'interface utilisateur .
- **Node.js** : environnement serveur qui exécute le code JavaScript côté serveur, et gère la communication entre Express et MongoDB.

Réalisation

1. Backend :

Le serveur backend a été développé en utilisant **Node.js** avec le framework **Express.js**.

Express.js est un framework web minimaliste pour Node.js, utilisé pour construire des applications web et des API. Il facilite la gestion des routes, des requêtes HTTP (GET, POST, PUT, DELETE), et permet une organisation claire du backend.

Dans notre projet **VetCare 360**, Express.js a été utilisé pour :

- Créer un serveur HTTP.
- Gérer les routes de l'API (ex : /api/owners, /api/vets).
- Traiter les requêtes venant du frontend.
- Communiquer avec la base de données MongoDB via **Mongoose**.

- **GET, POST, PUT , DELETE** sont utilisés pour gérer les différentes opérations CRUD (Create, Read, Update, Delete).
- La communication avec la base de données MongoDB via **Mongoose** .

Dans le développement du backend de VetCare 360, nous avons utilisé **Mongoose**, une bibliothèque JavaScript pour Node.js qui facilite l'interaction avec **MongoDB**.

Mongoose permet de créer des **modèles de données** (schemas) pour définir la structure des documents stockés dans la base de données.

Nous avons installé Mongoose à l'aide de la commande suivante :

```
npm install mongoose
```

Passant maintenant Structure des routes :

- /api/owners :
 - Ajouter un propriétaire.
 - Modifier un propriétaire.
 - Chercher un propriétaire par nom.
- /api/pets :
 - Ajouter un animal pour un propriétaire.
 - Modifier les informations d'un animal.
- /api/veterinaire :
 - Afficher la liste des vétérinaires.
- /api/visits :
 - Ajouter une visite médicale pour un animal.
 - Afficher toutes les visites d'un animal.

2. Frontend :

Le frontend de VetCare 360 a été développé avec **React.js**.

React.js est une bibliothèque JavaScript développée par Facebook, utilisée pour construire des interfaces utilisateur dynamiques et réactives. Elle permet de diviser l'interface en **composants** réutilisables, ce qui facilite le développement et la maintenance.

Dans le projet **VetCare 360**, React.js a été utilisé pour :

- Créer les différentes pages de l'application (accueil, recherche de propriétaires, liste des vétérinaires...).
- Gérer la navigation avec **React Router**.
- Dynamiser l'interface utilisateur sans recharger toute la page (SPA - Single Page Application).

React nous a permis de créer une application fluide, moderne et facile à utiliser.

Pages créées :

- **Home :**

Page d'accueil , un menu de navigation et des photos d'animaux.

- **Find Owners :**

Barre de recherche pour trouver un propriétaire par son nom.

Ajout et modification des propriétaires.

Ajout d'animaux associés a un propriétaires.

- **Veterinaires :**

Liste des vétérinaires avec leurs spécialités.

Chaque page utilise des composants React pour afficher dynamiquement les données récupérées depuis l'API REST.

3. Base de Données :

La base de données utilisée est **MongoDB**, qui stocke les données sous forme de documents JSON.

MongoDB est une base de données **NoSQL** orientée documents.

Contrairement aux bases relationnelles (comme MySQL), elle stocke les données sous forme de documents **JSON** (appelés BSON en interne).

Dans le projet **VetCare 360**, MongoDB a été utilisée pour :

- Stocker les données des propriétaires, animaux, vétérinaires et visites médicales.
- Gérer les relations entre les collections (par exemple, un animal appartient à un propriétaire).
- Faciliter les opérations de lecture et d'écriture avec **Mongoose**, une bibliothèque qui permet de manipuler MongoDB facilement en Node.js.

MongoDB est particulièrement adaptée pour les projets web modernes car elle est flexible, rapide et évolutive.

MongoDB Compass est l'interface graphique officielle de MongoDB. Elle permet de **visualiser**, **interroger** et **gérer** les bases de données MongoDB de façon simple et intuitive, sans avoir besoin de taper des commandes en ligne.

Dans le cadre du projet **VetCare 360**, MongoDB Compass a été utilisé pour :

- Créer les collections (owners, pets, vets, visits).
- Insérer et modifier des documents directement.
- Vérifier si les données étaient bien enregistrées après les opérations du frontend.
- Déboguer facilement en consultant la structure et le contenu de la base.

MongoDB Compass est un vrai gain de temps pour le développement et le suivi des données.

Schémas de collections :

- **Owner** :

Prénom, nom, adresse, ville, téléphone .

- **Pet** :

Nom, type, date de naissance .

- **Vétérinaire :**

Nom et spécialité

- **Visit :**

Date de visite, description .

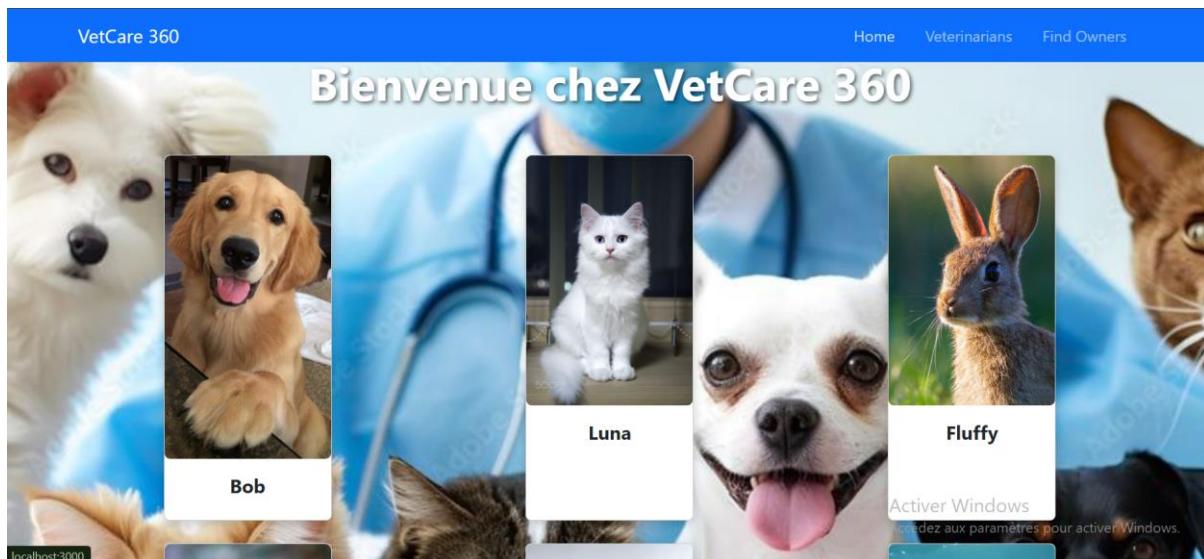
Résultats

Dans cette partie, nous présentons des captures d'écran de l'application

VetCare 360 afin d'illustrer les fonctionnalités principales développées.

Page d'accueil :

Cette page présente l'entrée principale de l'application avec un menu de navigation vers les différentes sections du site.



Liste des vétérinaires :

Affichage dynamique de tous les vétérinaires enregistrés dans la base de données MongoDB.

Nos Vétérinaires

Nom	Spécialité
Dr. Amina El Fassi	Chirurgie générale
Dr. Yassine Amrani	Dermatologie
Dr. Nadia Bennis	Médecine interne
Dr. Karim Lahiou	Urgences
Dr. Salma Kabaj	Cardiologie animale
Dr. Mehdi El Idrissi	Anesthésie
Dr. Imane Raji	Comportement animal
Dr. Hicham El Khalfi	Radiologie
Dr. Sara El Yousfi	Nutrition animale
Dr. Anas Moujahid	Dentisterie vétérinaire
Dr. Kawtar El Alami	Neurologie

✓ Formulaire d'ajout d'un propriétaire :

Ce formulaire permet d'ajouter un nouveau propriétaire d'animal en renseignant ses informations personnelles.

Ajouter un Propriétaire

Prénom	AMADID
Nom	Soulayman
Adresse	CARTIER ZAKARIA, CARTIER ZAKARIA
Ville	MARRAKECH
Téléphone	0666214314
Add Owner	

✓ Ajout des animaux :

Une fois le propriétaire créé, il est possible d'ajouter les animaux qui lui appartiennent avec leurs caractéristiques.

VetCare 360

Home Veterinarians Find Owners

Rechercher un Propriétaire

Soulayman

Find Owner Add Owner

AMADID Soulayman

CARTIER ZAKARIA, MARRAKECH
0666214314

Edit Owner

Add Pet

Animaux :

Nom : Bob
Type : Dog
Date de naissance : 12/04/2022

Add Visit

Visites :

Date : 12/06/2025

Gestion des visites :

Consultation et ajout de visites médicales pour chaque animal, avec choix du vétérinaire, date et description.

VetCare 360

Home Veterinarians Find Owners

Ajouter une Visite Médicale

Date de la visite
16/05/2025

Description
test test

Vétérinaire
Dr. Youssef Chraibi (Chirurgie orthopédique)

Add Visit



Bilan personnel

Ce projet a été une vraie opportunité de progresser sur plusieurs aspects :

- **Technique** : développement full-stack (MERN), création d'API REST, gestion de routes, utilisation de Mongoose.
- **Organisation** : structuration du code, séparation frontend/backend, gestion de fichiers.

- **Résolution de problèmes** : correction de bugs, compréhension des erreurs serveur/client, tests fonctionnels.

Outils utilisés

- **Visual Studio Code** : éditeur de code principal.
- **Node.js & npm** : environnement backend.
- **MongoDB & MongoDB Compass** : base de données NoSQL.
- **Postman** : pour tester les routes API.
- **React & Bootstrap** : création du frontend dynamique et responsive.
- **Git & GitHub** : versionnage du projet.



Perspectives futures

- Ajouter un système de **connexion / inscription** (authentification).
- Intégrer un **système de rôles** (admin, vétérinaire, assistant...).
- Déployer l'application sur une plateforme comme **Render**, **Vercel**, ou **Netlify**.
- Ajouter un tableau de bord statistique avec des graphiques (visites par mois, nombre d'animaux...).



Conclusion

Ce projet de développement de l'application **VetCare 360** m'a permis de renforcer mes compétences en développement web, notamment avec la

stack **MERN** (MongoDB, Express.js, React, Node.js) . J'ai appris à concevoir une application complète, créer une API REST, manipuler des données avec Mongoose, et développer une interface utilisateur claire et responsive grâce à React et Bootstrap .

Difficultés rencontrées :

- Connexion initiale au backend et configuration de MongoDB.
- Problèmes d'affichage ou d'interaction avec les formulaires React.
- Erreurs dans les routes ou la récupération des données.

Ces problèmes ont été résolus à travers des recherches, de la persévérance, et beaucoup de tests .

Ce projet m'a apporté beaucoup en termes de pratique et de compréhension du développement full stack, et représente une base solide pour aller encore plus loin .

Annexes

Exemple de code – Route Express pour afficher les vétérinaires :

Cet extrait montre comment le backend récupère la liste des vétérinaires depuis la base de données MongoDB à l'aide de Mongoose et l'expose via une API REST.

```
1  const express = require('express');
2  const router = express.Router();
3  const Veterinarian = require('../models/Veterinarian');
4
5  router.get('/', async (req, res) => {
6    try {
7      const vets = await Veterinarian.find();
8      res.json(vets);
9    } catch (err) {
10      console.error(err);
11      res.status(500).json({ message: 'Erreur serveur' });
12    }
13  });
14
15 module.exports = router;
16
```

Ce fichier définit une **route GET** pour récupérer tous les vétérinaires stockés dans la base de données MongoDB. Il utilise **Express.js** pour gérer les requêtes HTTP et **Mongoose** pour interagir avec la base de données.

La requête `Veterinarian.find()` récupère tous les documents depuis MongoDB.

Lien vers le projet GitHub :

Voici le lien vers le dépôt GitHub du projet VetCare 360 :

Ce projet m'a permis de comprendre l'importance de la structuration du code, et j'ai réalisé qu'il était essentiel de bien documenter chaque étape pour faciliter le débogage.

Fin.

