Rapport du Mini-Projet :

Vet Haven

Conception et réalisation D'une application Web De Gestion d'un Clinique viterinare

Filière : Ingénierie Logicielle et Cybersécurité

Réalisé par :

- Abdelkarim Wakrim
- Ahmed Oukhchine

Encadré par :

Mr. Redouane Esbai

Année universitaire : 2024-2025

المدرسة العليا للتكنولوجيا الناظور
ⵜⵉⵎⵣⴳⵉⵜ ⵏ ⵜⵓⵙⵙⵏⵉⵜ ⴰⵏⴰⵡⴰⵔ
École Supérieure de Technologie de Nador

📂 Sommaire

1. 📜 Introduction
2. 📚 Résumé du projet
3. 📄 Cahier des Charges
4. 🧠 Analyse et Conception
5. 💻 Réalisation
6. 📊 Résultats
7. 🛠️ Outils utilisés
8. 🔮 Perspectives futures
9. 👨‍🎓 Bilan personnel
10. 📞 Conclusion
11. 📌 Annexes
12. 🙏 Remerciements

Année universitaire : 2024-2025

📜 Introduction

La gestion des données dans les cliniques vétérinaires, telles que les informations sur les propriétaires d'animaux, les vétérinaires et les visites médicales, est une tâche incontournable mais souvent complexe. Beaucoup de cliniques continuent d'utiliser des systèmes traditionnels tels que des registres papier ou des outils bureautiques classiques, ce qui rend l'accès aux informations lent et inefficace.

Face à ces limitations, nous avons décidé de développer **Vet Haven**, une application web innovante qui permet de centraliser et d'automatiser la gestion des données d'une clinique vétérinaire. Cette solution moderne offre une interface facile à utiliser et permet de gérer efficacement toutes les informations nécessaires, qu'il s'agisse des propriétaires, des animaux ou des consultations médicales.

Le projet nous a permis de mettre en pratique nos compétences en développement web, en utilisant la stack **MERN** (MongoDB, Express.js, React.js, Node.js) pour construire une application complète répondant aux besoins des professionnels du secteur vétérinaire.

Une application comme **Vet Haven** garantit une gestion rapide et fluide des informations, accessible à la fois sur ordinateurs et appareils mobiles. Elle offre ainsi une plus grande flexibilité pour les vétérinaires et leurs équipes, facilitant leur quotidien.

L'intégration d'une telle solution numérique permet également de réaliser de nombreux bénéfices, tels que :

- ✓ Un gain de temps considérable lors de la recherche et de la mise à jour des informations.

- ✓ Une organisation interne plus optimisée et un suivi précis des consultations.
- ✓ Une amélioration notable de la relation avec les clients grâce à la rapidité et la fiabilité des services fournis.

Résumé du projet

Vet Haven est une application web conçue pour simplifier la gestion des cliniques vétérinaires. Son objectif est de fournir une solution intuitive et efficace pour gérer les informations relatives aux propriétaires d'animaux, aux vétérinaires et aux visites médicales.

Pour réaliser ce projet, nous avons adopté la stack **MERN** (MongoDB, Express.js, React.js, Node.js) pour la gestion de la base de données, le backend, le frontend et le serveur. De plus, nous avons intégré **Bootstrap** pour offrir une interface responsive et moderne, garantissant une expérience utilisateur fluide, quel que soit l'appareil utilisé.

Vet Haven permet aux utilisateurs d'ajouter, modifier et consulter les informations concernant les propriétaires, les animaux, les vétérinaires (juste l'affichage) et les visites médicales.
L'application offre des fonctionnalités essentielles, telles que :

- La recherche d'un propriétaire par son nom
- L'ajout et la gestion des animaux associés
- La gestion des visites médicales pour chaque animal
- Consulter la liste des vétérinaires avec leurs spécialités

Ce système intégré répond à la fois aux besoins pratiques des professionnels vétérinaires et à ceux des utilisateurs, en simplifiant et en automatisant la gestion quotidienne des cliniques.

Année universitaire : 2024-2025

📄 Cahier des Charges

❖ Objectifs :

Fonctionnels Le projet **Vet Haven** a pour objectif de concevoir une application web qui facilite la gestion intégrale d'une clinique vétérinaire. Les principales fonctionnalités incluent :

Gestion des propriétaires :

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

Gestion des animaux :

Offrir la possibilité d'ajouter un ou plusieurs animaux pour chaque propriétaire, avec des informations spécifiques telles que le nom, la date de naissance et le type.

Gestion des vétérinaires :

Afficher une liste de vétérinaires avec leurs spécialités.

Gestion des visites médicales :

Enregistrer les visites médicales de chaque animal, comprenant la date et la description de la consultation, pour assurer un suivi médical détaillé.

❖ Contraintes techniques :

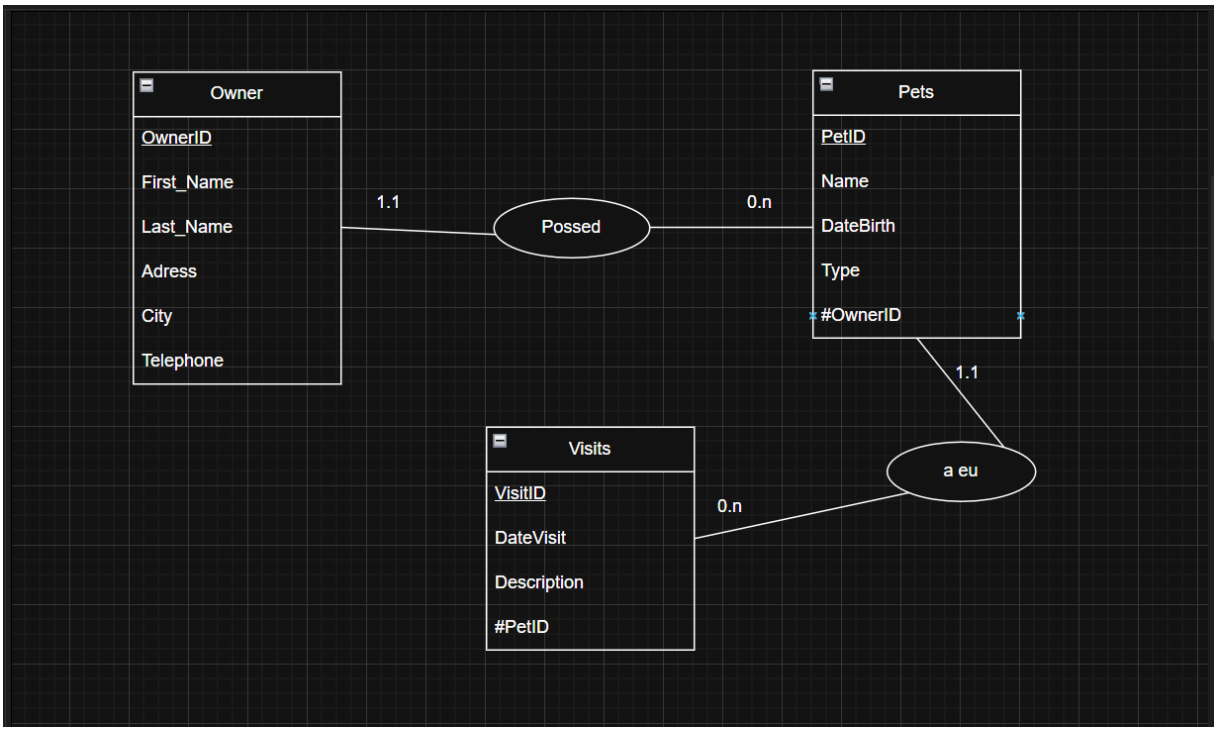
Pour garantir la performance, la fiabilité et la pérennité de l'application, plusieurs contraintes techniques ont été définies : Stack MERN : Le projet est construit sur la stack MERN

Année universitaire : 2024-2025

(MongoDB, Express.js, React.js, Node.js), assurant ainsi une solution web rapide, modulaire et facile à étendre. Interface responsive avec Bootstrap : L'interface utilisateur doit être simple et facile à utiliser, tout en étant responsive, c'est-à-dire s'adaptant à toutes les tailles d'écrans, grâce à l'utilisation de Bootstrap.

🧠 Analyse et Conception

❖ Le Modèle Conceptuel de Données (MCD) :



❖ Présentation des modèles :

Pour représenter les entités clés d'une clinique vétérinaire, nous avons défini quatre modèles principaux dans notre application **Vet Haven**. Ces modèles sont conçus pour faciliter la gestion des informations essentielles :

- **Propriétaire (Owner)** :
  Ce modèle contient les informations des propriétaires d'animaux. Il inclut les attributs suivants :
  - **OwnerId :** l'id de propriétaire
  - **First_Name** : Prénom

- **Last_Name** : Nom de famille
- **address** : Adresse
- **city** : Ville
- **Telephone** : Numéro de téléphone

- **Animal (Pet)** :
  Ce modèle est utilisé pour gérer les informations des animaux associés à chaque propriétaire. Il comprend :
  - **PetId :** l'id de animal
  - **name** : Nom de l'animal
  - **birthDate** : Date de naissance
  - **type** : Type (par exemple : chien, chat, etc.)
  - **OwnerID :** Id de propriétaire de l'animal.

- **Visite (Visit)** :
  Ce modèle enregistre les informations relatives aux consultations vétérinaires. Il comprend :
  - **VisitId :** l'id de la visite
  - **date** : Date de la visite
  - **description** : Description des soins ou traitements effectués.
  - **PetID :** Id de l'animal de Visit.

❖ Architecture MERN :

L'application **Vet Haven** repose sur l'architecture **MERN** (MongoDB, Express.js, React.js, Node.js), qui permet de concevoir une application web complète et évolutive.

- **MongoDB** : Base de données NoSQL utilisée pour stocker les informations sous forme de documents. Elle permet une gestion flexible des données, en particulier celles liées aux propriétaires, animaux, et visites.
- **Express.js** : Framework Node.js pour la gestion du serveur et la création d'API REST, facilitant ainsi les échanges de données entre le backend et le frontend.
- **React.js** : Framework JavaScript utilisé pour construire l'interface utilisateur..
- **Node.js** : Environnement d'exécution JavaScript permettant de gérer le backend et de traiter les demandes des utilisateurs efficacement.

💻 Réalisation

❖ Backend

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

Express.js est un framework léger pour Node.js, idéal pour construire des applications web et des API. Il simplifie la gestion des routes, les requêtes HTTP (GET, POST, PUT, DELETE), et offre une structure claire au backend.

Dans le cadre du projet **Vet Haven**, Express.js a été utilisé pour :

- Créer un serveur HTTP.
- Gérer les routes de l'API (/api/routes).
- Traiter les requêtes envoyées par le frontend.
- Interagir avec **MongoDB** via **Mongoose**.
  ➢ Les requêtes GET, POST, PUT, DELETE gèrent les opérations CRUD (Create, Read, Update, Delete).
-

Pour la communication avec **MongoDB**, nous avons utilisé **Mongoose**.
Mongoose est une bibliothèque JavaScript pour Node.js facilitant les interactions avec MongoDB. Elle permet de définir des schémas pour structurer les documents en base de données.
Mongoose a été installé avec la commande :

```
npm install mongoose
```

Structure des routes :

- **/api/routes :**
  - **(Owners):**
    Ajouter un propriétaire.
    Modifier les données d'un propriétaire.
    Supprimer un propriétaire.
    Rechercher un propriétaire par son prenom.

  - **(Pets):**
    Ajouter un animal à un propriétaire.
    Modifier les informations d'un animal.
    Supprimer un animal à un propriétaire.

  - **(visits) :**
    Ajouter une visite médicale pour un animal.
    Consulter toutes les visites d'un animal.
    Supprimer une visite à un animal.

❖ Frontend

Le frontend de **Vet Haven** a été développé en **React.js**.
React.js est une bibliothèque JavaScript développée par Facebook, utilisée pour créer des interfaces utilisateur interactives et dynamiques. Elle facilite le développement grâce à des composants réutilisables.

Dans ce projet, **React.js** a permis de :

- Développer différentes pages de l'application (exemple : Home, Owners, Vétérinaires).
- Gérer la navigation avec **React-Router-Dom**.

- Dynamiser l'interface utilisateur pour créer une application en **SPA** (Single Page Application).

Pages développées :

- **Home :**
- À propos de la clinique
- Services offerts
- L'équipe vétérinaire
- Localisation
- Coordonnées

- **Owners :**
  Une barre de recherche pour trouver un propriétaire par prénom.
  Ajouter, voir, et supprimer les informations des propriétaires.
  On a fait un lien sur le nom et prénom de propriétaire qui permet de Nous amène à une autre interface qui gère la modification et l'ajoute d'un animal avec la modification, et suppression, et aussi la création de visite animal avec la modification, et suppression .
- **Veterinaires :**
  Liste des vétérinaires avec leurs spécialités.

Chaque page utilise des composants React pour afficher des données dynamiques récupérées via l'API REST.

3. Base de données

La base de données choisie est **MongoDB**, qui stocke les données sous forme de documents JSON.
MongoDB est une base NoSQL orientée documents.

Contrairement aux bases relationnelles comme MySQL, elle utilise un format JSON (appelé BSON en interne) pour les documents.

Pour **Vet Haven**, MongoDB a été utilisée pour :

- Stocker les données des propriétaires, animaux, et visites médicales.
- Gérer les relations entre collections (exemple : un animal est lié à un propriétaire).
- Effectuer des lectures et écritures facilement avec **Mongoose**.

MongoDB est adaptée aux projets web modernes grâce à sa flexibilité, rapidité et évolutivité.

MongoDB Compass, l'interface graphique officielle de MongoDB, a été utilisée pour :

- Créer les collections (owners, pets, visits).
- Insérer et modifier des documents directement.
- Vérifier l'enregistrement des données après les opérations front-end.
- Déboguer grâce à la visualisation des données et de leur structure.

Schémas des collections :

- **Owner :** First_Name, Last_Name, Adress, City, Telephone.
- **Pet :** Name, Type, DateBirth, OwnerID.
- **Visit :** DateVisite, Description, PetID.

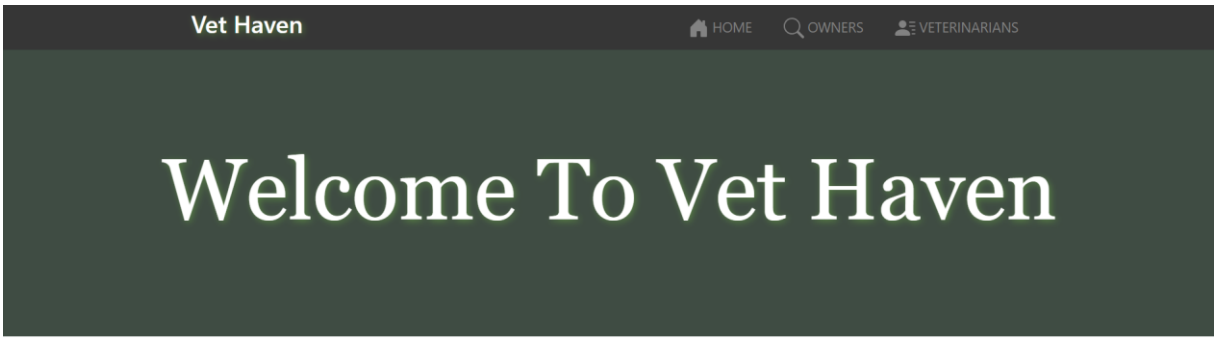
MongoDB Compass a été un outil précieux pour simplifier et accélérer le développement des données.

📊 Résultats

Dans cette section, nous présentons des captures d'écran de l'application **Vet Haven** pour illustrer les principales fonctionnalités développées.

Page d'accueil :

La page d'accueil sert de point d'entrée à l'application. Elle comprend un menu de navigation permettant d'accéder aux différentes sections du site.



You can now schedule your pet's vet appointement with just one click

Formulaire d'ajout d'un propriétaire :

Le formulaire permet de créer un nouveau propriétaire en renseignant ses informations personnelles.

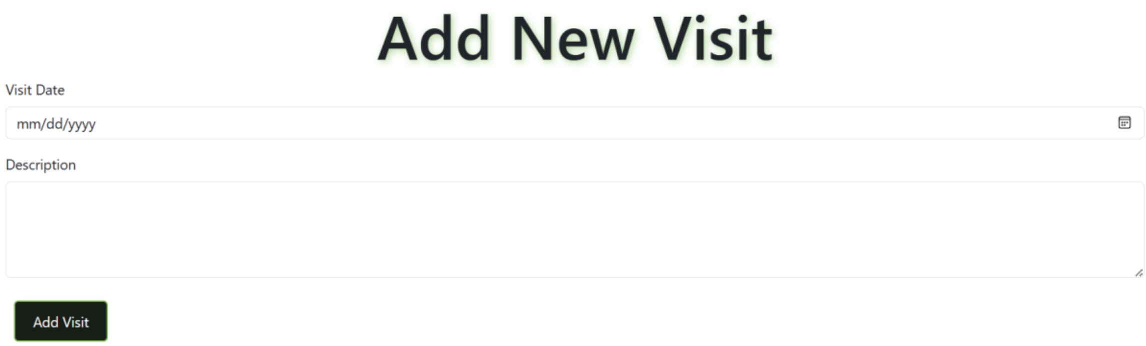
Ajout des animaux :

Après la création d'un propriétaire, il est possible d'ajouter ses animaux, en précisant leurs caractéristiques.

Année universitaire : 2024-2025

Gestion des visites :

Cette fonctionnalité offre la possibilité de consulter et d'ajouter des visites médicales pour chaque animal, avec un choix de date, et une description.



Liste des vétérinaires (Affichage sans base de données)

Dans cette section, nous simulons l'affichage des vétérinaires enregistrés sans interaction directe avec la base de données. Les données sont statiques, codées en dur, et utilisées uniquement pour illustrer l'interface utilisateur.

Veterinarians

Name	Specialties
Dr. Emily Carter	Dental Care
Dr. Michael Lee	Nutritional Counseling
Sarah Thompson	Diagnostics (imaging and laboratory)
Sophia Fisher	Preventive Medicine and Vaccination
David Johnson	Internal Medicine (general)
Karim Ahmed	General Surgery

🛠️ Outils utilisés

- ✓ **Visual Studio Code** : l'éditeur de code principal utilisé pour développer et organiser le projet.
- ✓ **React et Bootstrap** : pour concevoir un frontend dynamique et responsive.
- ✓ **Node.js et npm** : l'environnement pour le développement backend et la gestion des dépendances.
- ✓ **Postman** : utilisé pour tester et valider les routes de l'API.
- ✓ **MongoDB et MongoDB Compass** : la base de données NoSQL et son interface graphique pour gérer les données.
- ✓ **Git et GitHub** : pour la gestion de version et le stockage du projet en ligne.

🔮 Perspectives futures

- ✓ **Implémenter un système d'authentification** : permettre aux utilisateurs de se connecter ou de s'inscrire avec des identifiants sécurisés.
- ✓ **Intégrer un système de gestion des rôles** : définir des rôles tels qu'administrateur, vétérinaire ou assistant pour personnaliser les accès et fonctionnalités.
- ✓ **Déployer l'application** : la rendre accessible en ligne via une plateforme comme Netlify.
- ✓ **Ajouter un tableau de bord statistique** : afficher des graphiques illustrant des données comme le nombre de visites mensuelles ou le total des animaux enregistrés.

👨‍🎓 Bilan personnel

Ce projet a été une expérience enrichissante, permettant de progresser dans plusieurs domaines :

- **Aspect technique** : maîtrise du développement full-stack (MERN), création d'API REST, gestion des routes, et manipulation de Mongoose.
- **Organisation** : structuration efficace du code, séparation claire entre le frontend et le backend, et gestion rigoureuse des fichiers.
- **Résolution de problèmes** : identification et correction des bugs, analyse approfondie des erreurs côté serveur et client, et réalisation de tests fonctionnels.

Année universitaire : 2024-2025

📞 Conclusion

Le développement de l'application **Vet Haven** a été une excellente opportunité pour renforcer mes compétences en développement web, en particulier avec la stack **MERN** (MongoDB, Express.js, React, Node.js). J'ai appris à concevoir une application complète, développer une API REST, manipuler des données avec Mongoose, et créer une interface utilisateur claire et responsive grâce à React et Bootstrap.

Difficultés rencontrées :

- Configuration initiale du backend et connexion à MongoDB.
- Problèmes liés à l'affichage ou à l'interaction avec les formulaires React.
- Erreurs dans les routes ou lors de la récupération des données.

Ces défis ont été surmontés grâce à des recherches approfondies, de la persévérance, et de nombreux tests.

Ce projet m'a permis d'acquérir une expérience pratique précieuse et une compréhension approfondie du développement full stack, constituant ainsi une base solide pour aller encore plus loin.

Année universitaire : 2024-2025

📌 Annexes

Exemple de code – Route Express pour afficher les animaux (Pets)

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

```
145
146 > router.delete('/owners/:id', async (req, res) => {···
156   });
157
158
159   router.get('/owners/:id/pets', async (req, res) => {
160     try {
161       const pets = await Pet.find({ owner: req.params.id });
162       res.json(pets);
163     } catch (err) {
164       res.status(500).json({ message: err.message });
165     }
166   });
167
168
169 > router.get('/owners/:id/pets/:petId', async (req, res) => {···
178   });
```

Le fichier définit une route **GET** pour récupérer tous les animaux 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 **Pet.find()** permet de récupérer tous les documents (animaux) depuis MongoDB. **Lien vers le projet GitHub :**

https://github.com/abdelkarim25/Mern_Project_VetCare/tree/master

Ce projet m'a permis de comprendre l'importance de bien structurer le code, et j'ai appris qu'il est crucial de bien documenter chaque étape pour simplifier le débogage.

🙏 Remerciements

Nous souhaitons exprimer notre sincère gratitude à notre encadrant, M. Redouane Esbai, pour son soutien constant, sa disponibilité et ses précieux conseils tout au long de la réalisation de ce projet.

Nous tenons également à remercier chaleureusement notre établissement ainsi que l'ensemble du corps enseignant pour la qualité de la formation reçue, qui nous a permis de développer les compétences nécessaires à la réalisation de ce projet.

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

Année universitaire : 2024-2025