المدرسة العليا للتكنولوجيا الناظور
École Supérieure de Technologie de Nador
ⵜⵉⵎⵙⵙⵓⵔⵉⵜ ⵜⴰⵙⴷⴰⵡⵜ ⵏ ⵜⵓⵜⵉⵔⵉⵙⵜ ⵅⵏⵓⵎⵓⵔ

جامعة محمد الأول بوجدة
UNIVERSITÉ MOHAMMED PREMIER OUJDA
ⵜⴰⵙⴷⴰⵡⵜ ⵏ ⵎⵓⵃⴰⵎⵎⴷ ⴰⵎⴰⵣⵡⴰⵔ ⵏ ⵡⵊⴷⴷⴰ

Diplôme Universitaire de Technologie (DUT)

Ingénierie Logicielle et Cybersécurité (ILCS)

Rapport de projet

Thème :

Développement d'une application web pour la gestion d'une clinique vétérinaire

Réalisé par :

MHARI Ilham

JAAI Malak

Encadré par :

Pr. ESBAI Redouane

Soutenu le 12 Mai 2025, sous la direction de :

Pr. ESBAI Redouane École Supérieure de Technologie de Nador (ESTN)

Année universitaire : 2024 - 2025

Remerciements

Nous tenons à exprimer notre profonde gratitude au Pr. ESBAI Redouane, professeur de la filière Ingénierie Logicielle et Cybersécurité (ILCS) à l'École Supérieure de Technologie de Nador (ESTN). Son soutien constant, ses recommandations avisées et ses orientations précieuses ont été indispensables à la réussite de ce projet. Sa disponibilité et son engagement ont non seulement facilité la surmontée des défis rencontrés, mais ont également enrichi considérablement la qualité du travail accompli. Nous lui sommes sincèrement reconnaissantes pour la confiance qu'il a placée en nous en nous proposant ce sujet, ainsi que pour sa précieuse contribution et son rôle déterminant tout au long de cette expérience académique.

Résumé

Ce rapport présente un projet portant sur le développement de **VetCare 360**, une application web reposant sur l'architecture MERN (MongoDB, Express, React, Node.js) pour la gestion d'une clinique vétérinaire, à travers la gestion des propriétaires d'animaux, des vétérinaires, des animaux eux-mêmes et des visites médicales.

Table des matières

Introduction générale

Dans un monde de plus en plus tourné vers la transformation numérique, la plupart des secteurs d'activité, y compris celui de la médecine vétérinaire, doivent repenser leurs méthodes de gestion en intégrant des outils technologiques performants. Les cliniques vétérinaires ne font pas exception à cette tendance : elles sont aujourd'hui confrontées au défi de moderniser leurs processus administratifs et médicaux afin d'améliorer leur efficacité, renforcer la traçabilité des soins, et offrir un service de meilleure qualité aux propriétaires d'animaux.

C'est dans ce contexte que s'inscrit le projet **VetCare360** , une application web conçue pour répondre aux besoins concrets de gestion quotidienne d'une clinique vétérinaire. En centralisant les informations relatives aux propriétaires, aux animaux, aux visites médicales et au personnel soignant, cette plateforme vise à simplifier les tâches répétitives, à minimiser les risques d'erreurs et à améliorer l'expérience utilisateur, aussi bien pour les employés que pour les clients.

Au-delà de son objectif fonctionnel, ce projet s'inscrit dans une démarche pédagogique ambitieuse. Il a pour but de mettre en pratique les connaissances acquises dans les domaines du développement full-stack, de l'architecture logicielle et de la gestion de projet. À travers cette initiative, nous souhaitons non seulement proposer une solution technique adaptée à un besoin réel, mais aussi approfondir notre compréhension des choix technologiques, des modèles de conception et des architectures modernes utilisées dans le développement d'applications web. Ce travail encourage ainsi une réflexion critique et structurée sur les bonnes pratiques de l'ingénierie logicielle, dans le cadre d'une approche réaliste et professionnelle.

I. Introduction

1. Contexte et motivations

Dans un contexte marqué par l'évolution accélérée des technologies numériques, la digitalisation des processus administratifs s'impose comme une nécessité dans divers secteurs, notamment celui de la médecine vétérinaire.

Aujourd'hui encore, de nombreuses cliniques vétérinaires continuent d'utiliser des méthodes manuelles traditionnelles, telles que les dossiers papier ou les tableurs Excel, pour gérer les dossiers médicaux des animaux, les visites médicales, ainsi que les informations relatives aux vétérinaires et aux propriétaires.

Ces pratiques, bien qu'encore répandues, présentent de nombreuses limitations et posent plusieurs défis, notamment un risque élevé d'erreurs, des difficultés de traçabilité, une perte potentielle de données et un manque d'efficacité organisationnelle dans la gestion globale du cabinet.

Face à ces défis, nous avons conçu **VetCare360**, une application web moderne et intuitive, destinée à centraliser et automatiser les différentes tâches administratives et médicales au sein d'une clinique vétérinaire.

Au-delà de sa vocation fonctionnelle, ce projet constitue également une opportunité concrète de mettre en œuvre nos compétences acquises dans le développement full-stack, l'architecture logicielle, ainsi que la gestion de projet.

2. Présentation du projet

L'application **VetCare360** est une plateforme web conçue pour la gestion d'une clinique vétérinaire. Son développement s'inspire du projet open source bien connu PetClinic Community , souvent utilisé dans un cadre pédagogique pour illustrer les bonnes pratiques de développement avec Java et Spring Boot . Ce dernier propose des fonctionnalités classiques telles que la gestion des propriétaires d'animaux, le suivi des visites médicales, ainsi que l'organisation des données des vétérinaires.

Cependant, **VetCare360** se distingue principalement par ses choix technologiques et architecturaux. Contrairement à PetClinic, qui repose sur l'écosystème Java/Spring et une architecture serveur classique (Spring MVC), VetCare360 est entièrement développé sur l'architecture MERN (MongoDB, Express.js, React.js, Node.js). Ce choix permet de proposer une solution plus moderne, plus légère et mieux adaptée au développement rapide d'applications web interactives, notamment sous forme de Single Page Application (SPA).

Ainsi, **VetCare360** ne vise pas à introduire de nouvelles fonctionnalités, mais plutôt à réadapter techniquement un concept éprouvé, en le modernisant et en l'adaptant aux standards actuels du développement web.

3. Objectifs du projet

Le projet **VetCare360** vise à atteindre plusieurs objectifs, articulés autour de trois axes principaux : fonctionnel, technique et pédagogique.

Objectifs fonctionnels

L'application vise à simplifier la gestion quotidienne d'une clinique vétérinaire grâce à une interface intuitive et conviviale. Les principales fonctionnalités attendues sont les suivantes :

- Permettre l'enregistrement des propriétaires et de leurs animaux.
- Gérer les visites médicales, avec la possibilité de consulter leur historique.
- Faciliter la recherche rapide d'un propriétaire par son nom de famille.
- Afficher dynamiquement la liste des vétérinaires depuis la base de données.

Objectifs techniques

Sur le plan technologique, **VetCare360** repose sur une architecture moderne et performante :

- Utiliser l'architecture MERN (MongoDB, Express.js, React.js, Node.js) pour le développement full-stack.
- Appliquer le modèle MVC (Modèle-Vue-Contrôleur) pour une meilleure organisation du code.

- Exploiter MongoDB, une base de données NoSQL, pour stocker des données simples mais évolutives.
- Développer un serveur backend avec Node.js/Express pour gérer les routes et la logique métier.
- Concevoir un frontend réactif et interactif à l'aide de React.js, enrichi avec Bootstrap pour une interface à la fois responsive, visuellement confortable et attrayante.
- Utiliser Git et GitHub pour la gestion des versions et la collaboration en équipe.

Objectifs pédagogiques

Enfin, ce projet constitue une opportunité concrète d'apprentissage et de mise en pratique :

- Approfondir nos compétences en développement web full-stack avec des technologies modernes.
- Travailler en équipe sur un projet logiciel réaliste, en respectant les bonnes pratiques.
- Apprendre à structurer, organiser et documenter un projet informatique de manière professionnelle.
- Se familiariser avec les différentes étapes du cycle de développement logiciel : conception, développement, tests et validation.

II. Architecture et conception

1. Modèle MVC

L'architecture MVC (Modèle-Vue-Contrôleur) est un modèle de conception largement utilisé dans le développement d'applications web. Elle permet de séparer clairement les différentes responsabilités au sein d'une application, ce qui facilite la maintenance, la modularité et la collaboration en équipe. Elle se compose de trois couches principales :

- **Modèle (Model)** : représente la logique métier de l'application et gère les interactions avec la base de données. Il est chargé de récupérer, stocker et mettre à jour les données. Dans le cas de **VetCare360**, le modèle est implémenté côté serveur à l'aide de Node.js et de la bibliothèque Mongoose, qui permet de créer des schémas de données structurés pour MongoDB.
- **Vue (View)** : correspond à l'interface utilisateur, c'est-à-dire ce que l'utilisateur voit et avec quoi il interagit. Bien que React.js n'implémente pas directement le concept classique de « Vue », il remplit cette fonction en permettant de construire des interfaces dynamiques à partir de composants réutilisables. Dans **VetCare360**, chaque page est conçue sous forme de composants React indépendants, garantissant une interface modulaire, interactive et facile à maintenir.
- **Contrôleur (Controller)** : joue le rôle d'intermédiaire entre le modèle et la vue. Il reçoit les requêtes HTTP envoyées par l'utilisateur via les routes, traite ces demandes en s'appuyant sur le modèle approprié, puis retourne une réponse. Cette logique est gérée côté serveur grâce à Express.js, qui orchestre les interactions entre les routes et les fonctions du contrôleur.

Intégration du modèle MVC dans l'architecture MERN

Bien que React.js soit une bibliothèque et non un framework complet, il est courant dans l'écosystème MERN de structurer le projet selon le principe MVC :

- **Modèle** : côté backend — fichiers `.js` définissant les schémas avec *Mongoose*.
- **Vue** : côté frontend — composants *React*.
- **Contrôleur** : côté backend — fonctions exécutées après appel de routes *Express*.

Cette organisation rend le projet plus clair, plus facile à maintenir et à faire évoluer.

2. Architecture MERN

L'application **VetCare360** s'appuie sur l'architecture MERN, une pile technologique full-stack basée entièrement sur JavaScript. Cette architecture permet une cohérence du langage utilisé et une grande flexibilité pour le développement rapide d'applications web modernes.

Les quatre technologies principales de MERN

- **MongoDB** : base de données NoSQL orientée documents, utilisée pour stocker les données sous forme de JSON. Idéale pour des structures de données flexibles telles que propriétaires, vétérinaires, animaux ou visites.
- **Express.js** : framework léger pour Node.js, utilisé pour créer des API RESTful. Il gère les requêtes HTTP, les routes et la logique serveur.
- **React.js** : bibliothèque JavaScript utilisée pour construire des interfaces utilisateur dynamiques et interactives, sous forme de Single Page Application (SPA).
- **Node.js** : environnement d'exécution côté serveur permettant de faire fonctionner du code JavaScript en dehors du navigateur.

Grâce à cette architecture, **VetCare360** bénéficie d'un développement homogène, écrit entièrement en JavaScript, ce qui facilite la maintenance et l'intégration des différents modules.

3. Modélisation de la base de données

L'application utilise MongoDB, une base de données NoSQL orientée documents, pour stocker les informations de manière flexible et évolutive. Chaque entité principale est représentée par une collection contenant des documents JSON.

Collections principales :

- `owners` : représente les propriétaires d'animaux. Contient leur nom, prénom, adresse, téléphone, e-mail, et une liste d'animaux associés.
- `pets` : représente les animaux. Stocke des informations comme le nom, l'espèce, la date de naissance, le type, et inclut une référence (ObjectId) vers leur propriétaire.

▷ `visits` : regroupe les visites médicales. Chaque visite est liée à un animal via son `petId`, et contient des informations comme la date et la description.

▷ `veterinarians` : liste des vétérinaires avec leurs noms et spécialités.

Relations entre les entités :

Les relations sont gérées par des références (`ObjectId`) entre collections :

- Un `owner` peut avoir plusieurs `pets`.
- Un `pet` peut avoir plusieurs `visits`.
- Une `visit` est toujours liée à un seul `pet`.
- Les `veterinarians` sont stockés de manière indépendante ; bien qu'il n'y ait pas de lien direct avec les visites dans cette version, cela pourrait être ajouté ultérieurement si nécessaire.

Cette modélisation offre à la fois simplicité, performance et évolutivité, tout en reflétant fidèlement le fonctionnement d'une clinique vétérinaire.

III. Développement et Implémentation

1. Environnement de développement

1.1. Environnement de développement intégré (IDE)

Pour le développement frontend et backend, nous avons choisi Visual Studio Code (VS Code) comme éditeur principal. Cet IDE léger et personnalisable offre une prise en charge optimale des technologies JavaScript/Node.js/React, grâce à de nombreuses extensions utiles, notamment :

- ▷ **Prettier** : pour le formatage automatique du code.
- ▷ **ESLint** : pour l'analyse syntaxique et la détection d'erreurs.
- ▷ **GitLens** : pour une meilleure visualisation et gestion des versions via Git.
- ▷ **MongoDB for VS Code** : pour interagir directement avec la base de données depuis l'éditeur.

1.2. Outils de gestion de projet

- **Git** : système de gestion de versions distribué permettant aux développeurs de suivre les modifications et de collaborer efficacement.
- **GitHub** : plateforme d'hébergement et de gestion collaborative des projets basée sur Git.

1.3. Navigateurs web pour tests

Les tests fonctionnels de l'interface utilisateur ont été réalisés principalement sur Google Chrome, Mozilla Firefox, et Microsift Edge afin de garantir une compatibilité optimale entre les fonctionnalités de l'application et les navigateurs couramment utilisés.

2. Technologies et outils de développement

2.1. Langages de balisage et de style

- **HTML5** : utilisé pour la structure de base des composants React.
- **CSS3 / Bootstrap** : utilisés pour styliser l'interface, assurer une mise en page responsive et offrir une expérience utilisateur agréable.

2.2. Langages de programmation

- **JavaScript (ES6+)** : utilisé uniformément côté client (React) et côté serveur (Node.js).

2.3. Frameworks & Bibliothèques

- **React.js** : bibliothèque JavaScript utilisée pour construire une interface utilisateur dynamique sous forme de composants réutilisables.
- **Express.js** : framework backend léger pour gérer les routes, les requêtes HTTP et la logique métier.
- **Axios** : bibliothèque pour effectuer des requêtes HTTP depuis le frontend vers l'API backend.
- **Mongoose** : ODM (Object Data Modeling) facilitant la modélisation et l'interaction avec la base MongoDB.

2.4. Système de gestion de base de données (SGBD)

- **MongoDB** : base de données NoSQL orientée documents utilisée pour stocker les données sous forme de documents JSON.

3. Présentation de l'application

L'application **VetCare360** est constituée de plusieurs pages accessibles via React Router, assurant une navigation fluide sans rechargement complet de la page (SPA). Les principales fonctionnalités sont regroupées dans les pages suivantes :

- ▷ **Page d'accueil** : point d'entrée principal de l'application.
- ▷ **Liste des vétérinaires** : affichage de tous les vétérinaires enregistrés.
- ▷ **Recherche de propriétaire** : permet de retrouver rapidement un propriétaire par son nom.
- ▷ **Ajout de propriétaire** : formulaire pour enregistrer un nouveau propriétaire.
- ▷ **Modification de propriétaire** : mise à jour des informations d'un propriétaire existant.
- ▷ **Ajout d'un animal** : formulaire lié à un propriétaire pour enregistrer un nouvel animal.
- ▷ **Liste des propriétaires** : résultats de la recherche, affichant les propriétaires correspondants.
- ▷ **Détails du propriétaire** : présentation complète incluant ses animaux et l'historique des visites médicales.
- ▷ **Modification de l'animal** : mise à jour des données d'un animal spécifique.
- ▷ **Ajout d'une visite médicale** : saisie d'une nouvelle visite avec affichage immédiat.
- ▷ **Page de synthèse** : vue consolidée de toutes les informations relatives au propriétaire, ses animaux et leurs soins.

IV. Conclusion

1. Récapitulation des objectifs atteints

Le projet **VetCare360** a permis de concrétiser avec succès un ensemble d'objectifs techniques, fonctionnels et pédagogiques. Voici les principales réalisations du projet :

- **Développement d'une application web complète** : Une plateforme entièrement fonctionnelle permettant de gérer efficacement une clinique vétérinaire.
- **Implémentation de l'architecture MERN (MongoDB, Express, React, Node.js)** : Démonstration pratique de la capacité à concevoir une application full-stack moderne, intégrant des technologies largement utilisées dans le domaine du développement web.
- **Utilisation du modèle MVC (Modèle-Vue-Contrôleur)** : Organisation claire et structurée du code, assurant une séparation logique entre la logique métier, la présentation et la gestion des données.
- **Modélisation d'une base de données flexible et évolutive** : Utilisation de MongoDB pour représenter les différentes entités du système (propriétaires, animaux, visites, vétérinaires), avec la possibilité d'évolution vers des relations plus complexes.
- **Collaboration en équipe via Git/GitHub** : Maîtrise des bonnes pratiques de gestion de versions, facilitant le travail collaboratif, le suivi des modifications et la résolution des conflits.

2. Perspectives d'améliorations futures

Bien que **VetCare360** réponde déjà aux besoins fonctionnels attendus, plusieurs pistes d'évolution sont envisageables afin d'enrichir ses fonctionnalités, d'améliorer l'expérience utilisateur, ainsi que de renforcer son utilité, sa sécurité et son ergonomie.

- **Ajout d'un système d'authentification sécurisé** : Implémenter un système permettant de gérer les utilisateurs avec des rôles différents (administrateur, vétérinaire, assistant), afin de contrôler les accès et sécuriser les données sensibles.

- **Gestion complète des rendez-vous** : Intégrer un module dédié à la prise de rendez-vous, comprenant un calendrier interactif pour organiser les visites et planifier les disponibilités.
- **Notification par email ou SMS avant les visites** : Permettre l'envoi automatique de rappels aux propriétaires quelques heures ou jours avant leur rendez-vous, afin de réduire les absences non justifiées.
- **Amélioration du module des visites médicales** :
  - ▷ Modifier une visite médicale pour corriger une erreur.
  - ▷ Supprimer une visite en cas d'annulation ou de saisie incorrecte.
- **Gestion avancée des vétérinaires** :
  - ▷ L'ajout d'un nouveau vétérinaire via un formulaire dédié.
  - ▷ La modification ou la suppression des informations d'un vétérinaire existant.
  - ▷ L'attribution d'un vétérinaire spécifique lors de l'enregistrement d'une visite médicale, pour assurer une meilleure traçabilité.
- **Développement d'une version mobile de l'application** : Créer une application mobile ou une Progressive Web App (PWA) basée sur React Native ou React.js, permettant d'accéder à **VetCare360** depuis un smartphone ou une tablette.
- **Ajout d'un tableau de bord statistique** :
  - ▷ Le nombre de visites par mois ou par saison.
  - ▷ Les espèces animales les plus fréquentes.
  - ▷ La répartition des interventions entre les différents vétérinaires.
- **Génération de documents PDF** : Permettre aux utilisateurs de générer des rapports médicaux, des ordonnances ou des historiques de soins au format PDF, téléchargeables et imprimables.

Avec ces perspectives, **VetCare360** pourrait évoluer d'un projet pédagogique à une solution professionnelle opérationnelle, répondant aux exigences modernes de digitalisation dans le secteur vétérinaire.

Conclusion générale

La réalisation du projet **VetCare360** a été pour nous une expérience à la fois enrichissante sur le plan technique et humain. En partant d'un besoin concret — la gestion numérique d'une clinique vétérinaire — et en nous inspirant d'une solution existante telle que PetClinic Community, nous avons réussi à concevoir une application complète, moderne et conforme aux standards actuels du développement web.

Ce projet nous a permis de mettre en pratique de manière concrète les connaissances acquises durant notre formation : développement full-stack, architecture logicielle, modélisation des données, mais aussi collaboration en équipe, gestion de versions avec Git et résolution de problèmes techniques.

Le choix de l'architecture MERN (MongoDB, Express, React, Node.js) nous a offert l'opportunité de travailler avec des outils largement répandus dans le milieu professionnel, tout en maîtrisant les différentes étapes de développement d'une application web dynamique et fonctionnelle.

Au-delà d'un simple exercice ou problème académique, **VetCare360** nous a permis de mieux comprendre les exigences du métier de développeur, de renforcer nos compétences pratiques et de poser les bases d'éventuelles améliorations futures.

Références bibliographiques

[1] Spring Team. (2020). *Spring PetClinic Community Edition*. GitHub Repository. https://github.com/spring-projects/spring-petclinic

[2] MongoDB Inc. (2024). *MongoDB Documentation*. https://www.mongodb.com/docs

[3] Meta. (2024). *React.js – Official Documentation*. https://reactjs.org/

[4] Node.js Foundation. (2024). *Node.js Documentation*. https://nodejs.org/en/docs

[5] Express.js. (2024). *Express.js Guide*. https://expressjs.com/

[6] Bootstrap Team. (2024). *Bootstrap 5 Documentation*. https://getbootstrap.com/docs/5.3/

[7] Git Team. (2024). *Pro Git Book – Git Documentation*. https://git-scm.com/book/fr/v2

[8] GitHub, Inc. (2024). *Git Documentation*. https://git-scm.com/doc