المدرسة العليا للتكنولوجيا الناظور
École Supérieure de Technologie de Nador
ⵜⵉⵎⵙⵙⵉⵏ ⵜⵓⵣⵣⵉⵡⵜ ⵏ ⵜⵓⵜⵙⵔⵉⵡⵉⵏⵜ ⵅⵏⵓⵎ

جامعة محمد الأول بوجدة
UNIVERSITÉ MOHAMMED PREMIER OUJDA
ⵜⵓⵙⵏⴰⵡⵜ ⵜⵉⵎⵣⵡⴰⵔⵉⵏ ⵏ ⵡⵓⵊⴷⴰ

Diplôme Universitaire de Technologie (DUT)

Ingénierie Logiciel et Cybersecurité (ILCS)

VetCare360 – Application Web de Gestion d'une Clinique Vétérinaire

Présenter Par :
JAAI Malak & MHARI Ilham

Encadrer par:
Pr. ESBAI Redouane

Année universitaire : 2024-2025

Le Plan Du Présentation

Introduction

Introduction générale :

Le secteur vétérinaire, à l'image des autres domaines de la santé, fait face à une transition numérique de plus en plus marquée. Les cliniques vétérinaires traitent chaque jour une quantité considérable de données : informations sur les propriétaires, dossiers médicaux des animaux, rendez-vous, et interventions.

L'absence d'un système centralisé engendre des risques d'erreurs, de pertes de données et une surcharge administrative, ce qui impacte directement l'efficacité et la qualité des services. Pour remédier à ces enjeux, le développement d'applications spécifiques est devenu incontournable

Dans ce contexte, le projet **VetCare360** propose une solution web moderne et performante, pensée pour répondre aux besoins particuliers des cliniques vétérinaires. S'inspirant de modèles existants, il adopte une architecture technique moderne, tirant parti des technologies web actuelles pour offrir une solution à la fois intuitive et efficace.

Introduction

Contexte et Motivations :

Dans un contexte de transformation numérique rapide, de nombreuses cliniques vétérinaires utilisent encore des méthodes traditionnelles, comme les dossiers papier ou Excel, pour gérer leurs activités. Ces pratiques entraînent des erreurs, un manque de traçabilité et une faible efficacité. Pour répondre à ces enjeux, nous avons développé **VetCare360**, une application web moderne qui centralise et automatise la gestion administrative et médicale d'un cabinet vétérinaire. Ce projet nous a également permis de mobiliser nos compétences en développement full-stack, architecture logicielle et gestion de projet.

Introduction

Présentation du Projet

VetCare360 est une application web de gestion de clinique vétérinaire, inspirée du projet open source **PetClinic Community**. Contrairement à ce dernier, basé sur Java et Spring Boot, VetCare360 adopte une architecture **MERN** (MongoDB, Express.js, React.js, Node.js), plus moderne et orientée vers les applications web interactives. L'objectif n'est pas d'ajouter de nouvelles fonctionnalités, mais de **réinterpréter une solution existante** avec des technologies actuelles, plus légères et adaptées aux besoins du web d'aujourd'hui.

Architecture et conception

Présentation de l’architecture MERN :

L'application VetCare360 s'appuie sur l'architecture MERN, un ensemble de technologies JavaScript modernes et complémentaires :

- **MongoDB** stocke les données sous forme de documents JSON, idéal pour modéliser les informations des vétérinaires, animaux, et visites.
- **Express.js**, couplé à Node.js, gère les requêtes serveur et les routes de l’application.
- **React.js** permet de construire une interface utilisateur fluide et interactive en mode Single Page Application.

- **Node.js** assure l’exécution du code côté serveur et la communication avec la base de données.

L’un des grands atouts de cette architecture est la **cohérence technique**, car tout le projet est développé en JavaScript, ce qui facilite l'intégration des modules, la maintenance et la collaboration entre les développeurs.

Architecture et conception

Présentation de l'architecture MVC :

L'architecture MVC (Modèle - Vue - Contrôleur) est utilisée dans VetCare360 pour structurer l'application de manière claire et modulaire.

- Le **modèle**, géré avec **Mongoose** côté serveur, traite les données et la logique métier.
- La **vue**, construite avec **React.js**, propose une interface interactive sous forme de composants réutilisables.
- Le **contrôleur**, via **Express.js**, fait le lien entre les requêtes utilisateur, les modèles et les réponses envoyées.

Même si React n'est pas un framework MVC à part entière, **le projet suit ce principe** en séparant les responsabilités entre le backend (modèle + contrôleur) et le frontend (vue), ce qui facilite la maintenance et l'évolution de l'application.

Architecture et conception

Modélisation de la base de données :

La base de données de VetCare360 repose sur MongoDB, une solution NoSQL qui stocke les données au format JSON, offrant ainsi une grande flexibilité et une gestion optimisée des évolutions du modèle de données. Grâce à l'utilisation de Mongoose, nous gérons efficacement les collections et les relations entre les entités.

Les principales collections sont :

- **owners** : Contient les informations des propriétaires d'animaux, avec leurs coordonnées et une liste des animaux associés.
- **pets** : Regroupe les informations des animaux, dont le nom, la date de naissance, l'espèce, la race, et une référence au propriétaire via l'ObjectId.
- **visits** : Enregistre les visites médicales des animaux, avec des détails tels que la date, la description, et les traitements associés.
- **veterinarians** : Stocke les informations des vétérinaires, incluant leur nom et spécialité.

Architecture et conception

Les relations entre les entités sont établies par des références ObjectId :

- Un propriétaire peut avoir plusieurs animaux.
- Un animal peut avoir plusieurs visites médicales.
- Chaque visite est associée à un seul animal.
- Les vétérinaires sont listés indépendamment, sans lien direct avec les visites,
- mais cette fonctionnalité pourra être ajoutée ultérieurement.

⇨ **Cette architecture assure une gestion efficace et souple des données, tout en restant facilement extensible pour intégrer de nouvelles fonctionnalités à l'avenir.**

Développement et Implémentation

Environnement de d´ eveloppement :

Éditeur utilisé :

Pour le développement de VetCare360, nous avons opté pour **Visual Studio Code**, un éditeur léger et performant. Il a été enrichi par plusieurs extensions utiles comme *Prettier* pour le formatage automatique du code, *ESLint* pour l'analyse syntaxique, *GitLens* pour le suivi des versions, et *MongoDB for VS Code* pour la gestion directe de la base de données.

Gestion de versions :

Nous avons utilisé **Git** comme système de gestion de versions local, ce qui nous a permis de suivre efficacement l'évolution du code. Pour le travail collaboratif et l'hébergement du projet, nous avons utilisé la plateforme **GitHub**, facilitant ainsi les échanges et la gestion du code en équipe.

Navigateurs de test :

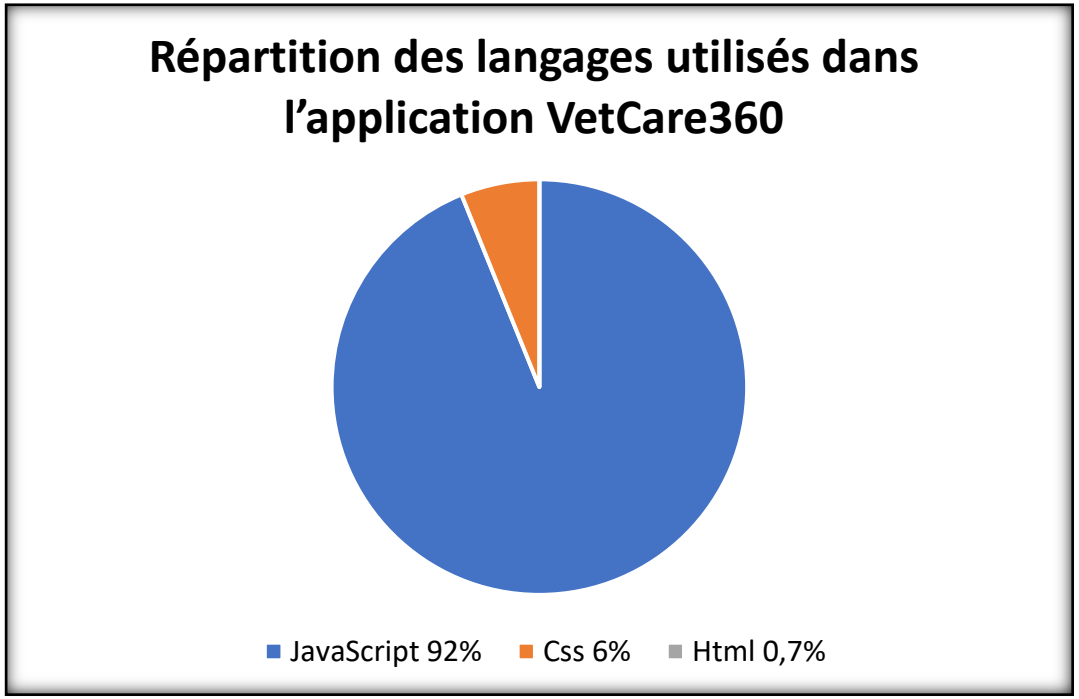
Afin de garantir une bonne compatibilité et une expérience utilisateur fluide, l'application a été testée sur les navigateurs **Google Chrome**, **Mozilla Firefox** et **Microsoft Edge**, qui représentent une large part des usages actuels et couvrent les principaux standards du web moderne.

Développement et Implémentation

Technologies et outils de d´eveloppement

Langages de balisage et de style :
L'application utilise **HTML5** pour structurer les composants React, tandis que **CSS3** et **Bootstrap** sont employés pour styliser l'interface. Cela permet d'assurer une mise en page responsive et une expérience utilisateur agréable.

Langage de programmation :
Le langage principal utilisé est **JavaScript (ES6+)**, à la fois côté client avec React et côté serveur avec Node.js, assurant ainsi une cohérence dans tout le projet.



Développement et Implémentation

Frameworks et bibliothèques :

Nous avons utilisé **React.js** pour construire une interface utilisateur dynamique sous forme de composants réutilisables. **Express.js** assure la gestion des routes et de la logique backend. Pour les échanges entre le frontend et l'API, **Axios** est utilisé, tandis que **Mongoose** facilite l'interaction avec MongoDB grâce à un mapping objet-document clair.

Système de gestion de base de données :

Pour le stockage des données, l'application utilise **MongoDB**, une base de données NoSQL orientée documents. Cette solution est idéale pour gérer des données dynamiques et non structurées, permettant ainsi une grande flexibilité et évolutivité tout au long du développement

Développement et Implémentation

Présentation de l'application :

Pages et Fonctionnalités :

- **Page d'accueil :** point d'entrée principal de l'application.
- **Liste des vétérinaires:** affichage de tous les vétérinaires enregistrés dans la base MongoDB.
- **Recherche de propriétaire** : permet de retrouver un propriétaire par son nom.
- **Ajout de propriétaire** : formulaire pour enregistrer un nouveau propriétaire.
- **Modification de propriétaire** : mise à jour des informations personnelles d'un propriétaire.
- **Ajout d'un animal :** formulaire lié à un propriétaire pour enregistrer un nouvel animal.
- **Liste des propriétaires :** résultats de la recherche, affichant tous les propriétaires correspondants.

Développement et Implémentation

- **Détails du propriétaire :** présentation complète du propriétaire, incluant ses animaux et l'historique des visites médicales associées.
- **Modification de l'animal :** mise à jour des données concernant un animal spécifique
- **Ajout d'une visite médicale :** saisie d'une nouvelle visite, avec affichage immédiat dans l'historique.
- **Page de synthèse :** vue consolidée affichant toutes les informations relatives à un propriétaire, ses animaux et leurs soins.

Conclusion

Récapitulation des objectifs atteints :

Le projet **VetCare360** a permis d'atteindre plusieurs objectifs clés : développer une application web complète pour la gestion d'une clinique vétérinaire, en utilisant l'architecture **MERN** (MongoDB, Express, React, Node.js) pour une solution full-stack moderne et évolutive. Nous avons adopté le modèle **MVC** afin d'organiser le code de manière claire et structurée, en séparant la gestion des données, la présentation et la logique métier. En parallèle, nous avons modélisé une base de données flexible avec **MongoDB**, permettant d'adapter facilement le système aux besoins futurs. Enfin, la collaboration en équipe a été facilitée par l'utilisation de **Git/GitHub**, optimisant la gestion des versions et le suivi des modifications.

Conclusion

Perspectives d'améliorations futures:

Même si **VetCare360** remplit déjà son objectif initial de gestion d'une clinique vétérinaire, plusieurs améliorations fonctionnelles et techniques peuvent encore être envisagées pour enrichir l'expérience utilisateur et élargir les possibilités offertes par l'application :

- **Ajout d'un système d'authentification sécurisé** : permettre aux utilisateurs d'avoir des rôles différents (administrateur, vétérinaire, assistant) et de sécuriser l'accès aux données
- **Amélioration du module de visites médicales:** Ajouter la possibilité de **modifier** ou **supprimer** une visite médicale, pour corriger ou annuler une erreur.
- **Gestion avancée des vétérinaires** : Ajouter un formulaire d'**ajout de vétérinaire**, Permettre la **modification** ou la **suppression** de vétérinaires depuis une interface dédiée, Possibilité d'**assigner un vétérinaire à une visite**, afin de mieux tracer l'intervention.
- **Ajout d'un tableau de bord/statistiques** : Nombre de visites par mois, espèces animales les plus traitées, répartition des cas par vétérinaire

Conclusion

Conclusion générale :

Le projet **VetCare360** a été une expérience enrichissante, tant sur le plan technique que personnel. En répondant à un besoin réel de gestion numérique pour les cliniques vétérinaires et en partant de l'existant **PetClinic Community**, nous avons conçu une application moderne et fonctionnelle, respectant les standards du développement web. Ce projet a été l'occasion de mettre en œuvre nos compétences en développement full-stack, modélisation des données, et gestion de versions, tout en travaillant en équipe. L'utilisation de la pile **MERN** nous a permis de nous familiariser avec des technologies courantes dans le monde professionnel. Au-delà d'un projet académique, **VetCare360** a consolidé nos compétences pratiques et a ouvert des perspectives d'évolutions futures, voire de mise en production dans un cadre réel.

Références bibliographiques

- ✓ **Spring Team**. (2020). *Spring PetClinic Community Edition*. GitHub Repository. Disponible sur : https://github.com/spring-projects/spring-petclinic.
- ✓ **MongoDB Inc.** (2024). *MongoDB Documentation*. Disponible sur : https://www.mongodb.com/docs.
- ✓ **Meta (React)**. (2024). *React.js Official Documentation*. Disponible sur : https://reactjs.org/.
- ✓ **Node.js Foundation**. (2024). *Node.js Documentation*. Disponible sur : https://nodejs.org/en/docs.
- ✓ **Express.js**. (2024). *Express.js Guide*. Disponible sur : https://expressjs.com/.
- ✓ **Bootstrap**. (2024). *Bootstrap Documentation*. Disponible sur : https://getbootstrap.com.
- ✓ **W3Schools**. (2024). *W3Schools Online Web Tutorials*. Disponible sur : https://www.w3schools.com/.

Merci pour votre attention !