المدرسة العليا للتكنولوجيا الناظور
École Supérieure de Technologie de Nador
ⵜⵉⵎⵙⵙⵉⵍⵜ ⵜⴰⵏⴰⵡⴰⵜ ⵏ ⵜⴰⵜⵉⴽⵏⵓⵍⵓⵊⵉⵜ ⵅⵍⴰⵟⵓⵔ

جامعة محمد الأول بوجدة
UNIVERSITÉ MOHAMMED PREMIER OUJDA
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵎⵓⵃⴰⵎⵎⴷ ⴰⵎⴰⵣⵡⴰⵔ ⵏ ⵡⵓⵊⴷⴰ

Diplôme Universitaire de Technologie (DUT)

Ingénierie Logiciel et Cybersecurity (ILCS)

Rapport du Projet

VetCare-360 une application web d'une clinique vétérinaire

Réalisé par :

AIT ADDI Tithrite Fatimazahra
Astitou

Encadrement pédagogique : Pr . Redouane Esbai

Année universitaire : *2024 - 2025*


رَبَّنَآ ءَاتِنَا مِن لَّدُنكَ رَحْمَةً
وَهَيِّئْ لَنَا مِنْ أَمْرِنَا رَشَدًا

Remerciements

Nous tenons à exprimer notre profonde gratitude à Monsieur Esbai Redouane, notre professeur de la matière **"Big Data & Bases de Données"**, pour son enseignement précieux et son accompagnement tout au long de ce projet.
Ses explications claires, sa pédagogie exemplaire et sa patience ont été déterminantes pour nous permettre d'assimiler les fondamentaux du Big Data et de les appliquer concrètement dans ce projet. Son expertise et son soutien constant ont joué un rôle essentiel dans l'acquisition des compétences clés en manipulation de bases de données **MongoDB** et en traitement des données massives.
Nous lui sommes sincèrement reconnaissants pour le temps qu'il nous a consacré et pour son engagement à transmettre son savoir avec passion et dévouement. Son mentorat a grandement contribué à la réussite de ce projet et à notre
progression dans ce domaine technologique en plein essor.

Table des matières

I. Introduction générale

Dans un monde où la relation entre les humains et les animaux de compagnie est de plus en plus importante, les cliniques vétérinaires jouent un rôle clé dans le bien-être des animaux. Cependant, la gestion manuelle des dossiers des propriétaires, des animaux et des visites médicales peut rapidement devenir complexe et chronophage. Pour répondre à ce défi, **VetCare 360** a été développé comme une solution moderne et efficace pour automatiser et simplifier la gestion des opérations quotidiennes d'une clinique vétérinaire.

1. Présentation du projet

VetCare 360 est une application web conçue pour simplifier la gestion d'une clinique vétérinaire en automatisant les opérations clés. Développée sur une architecture **MERN** (MongoDB, Express.js, React, Node.js), elle permet de gérer les propriétaires, leurs animaux de compagnie, et les visites médicales de manière centralisée. L'application propose des fonctionnalités telles que l'ajout, la modification et la suppression des propriétaires et des animaux, ainsi que le suivi des historiques médicaux via une interface intuitive développée avec **React** et **Bootstrap** . Le backend, basé sur **Node.js** et **Express.js** , communique avec une base de données **MongoDB** pour stocker les informations des propriétaires, des animaux, des visites et des vétérinaires. Les pages incluent une page d'accueil, une liste des vétérinaires, une recherche de propriétaire, des formulaires pour ajouter ou modifier des informations, et des sections dédiées à l'historique des visites. VetCare 360 offre ainsi une solution pratique et moderne pour rationaliser la gestion des cliniques vétérinaires tout en améliorant l'expérience utilisateur

2. Objectifs du projet

L'objectif du projet **VetCare 360** est de fournir une solution numérique complète et efficace pour la gestion d'une clinique vétérinaire. Cette application web, basée sur une architecture **MERN** (**MongoDB**, **Express.js**, **React**, **Node.js**), vise à simplifier les opérations quotidiennes en automatisant la gestion des propriétaires, des animaux, des visites médicales et des vétérinaires. Elle permet aux utilisateurs d'ajouter, modifier et consulter les informations des propriétaires et de leurs animaux, tout en assurant un suivi précis des historiques médicaux. Avec une interface intuitive et responsive développée en **React** et **Bootstrap**, **VetCare 360** facilite l'accès aux données centralisées dans une base **MongoDB**, offrant ainsi une expérience fluide pour les employés de la

clinique et améliorant leur productivité au quotidien.

Les fonctionnalités principales de l'application VetCare 360 sont les suivantes :

II. Fonctionnalités Principales

1. Gestion des propriétaires d'animaux

- Ajouter, modifier et supprimer des propriétaires
- Rechercher des propriétaires par nom de famille.
- Afficher les informations détaillées d'un propriétaire, y compris ses animaux associés.

2. Gestion des animaux

- Ajouter des animaux et les associer à leurs propriétaires.
- Modifier ou supprimer les informations d'un animal.
- Suivre l'historique des visites médicales pour chaque animal.

3. Gestion des visites médicale

- Ajouter des visites médicales pour un animal spécifique.
- Consulter l'historique des visites précédentes d'un animal.
- Afficher une synthèse des visites après chaque ajout.

4. Liste des vétérinaires

- Afficher une liste complète des vétérinaires disponibles dans la clinique.
- Inclure des détails sur leurs spécialités (le cas échéant).

Les technologies utilisées pour le développement de l'application VetCare 360 sont les suivantes :

III. Technologies Utilisées

1. Frontend :

- React : Framework JavaScript utilisé pour créer une interface utilisateur dynamique et interactive.
- Bootstrap : Bibliothèque CSS intégrée pour un design responsive et des composants prêts à l'emploi.

2. Backend :

- Node.js : Environnement d'exécution JavaScript côté serveur.
- Express.js : Framework minimaliste pour Node.js, utilisé pour gérer les routes et les requêtes API.

3. Base de données :

- MongoDB : Base de données NoSQL qui stocke les données sous forme de documents JSON, idéale pour la flexibilité et la scalabilité.

4. Gestion des versions :

- Git : Outil de gestion des versions pour suivre les modifications du code.
- GitHub : Plateforme de collaboration pour héberger et partager le code source.

Ces technologies forment une architecture MERN (MongoDB, Express.js, React, Node.js), qui permet de développer une application web moderne, performante et facile à maintenir.

IV. Architecture de l'Application

1. Architecture MERN :

L'architecture MERN (MongoDB, Express, React, Node.js) est une architecture complète de développement full-stack JavaScript.

- MongoDB : Base de données NoSQL utilisée pour stocker les données de l'application (propriétaires, animaux, visites, vétérinaires).
- Express.js : Framework backend qui simplifie la création des API et la gestion des routes.
- React.js : Framework frontend pour créer une interface utilisateur dynamique et réactive.
- Node.js : Environnement d'exécution JavaScript côté serveur.

2. Flux de Données :

- L'utilisateur interagit avec l'interface (React).
- Les requêtes sont envoyées au serveur via Axio(HTTP).
- Le serveur (Express) traite les requêtes et interagit avec MongoDB.
- Les réponses sont envoyées au frontend pour affichage.

3. **Architecture MVC** :

L’architecture MVC , pour Modèle-Vue-Contrôleur , est un modèle largement utilisé dans le développement d’applications logicielles, notamment pour la création d’interfaces utilisateur. Elle permet de séparer les différentes responsabilités au sein d’une application en trois composants principaux : le Modèle, la Vue et le Contrôleur. Cette séparation facilite la gestion du code, rend l’application plus modulaire, et permet à plusieurs développeurs de travailler efficacement sur différentes parties du projet.

Le Modèle est chargé de gérer les données et la logique métier de l’application. Il représente la source principale d’informations, comme une base de données ou des services externes, et contient également les règles métiers qui définissent comment ces données peuvent être manipulées. Lorsque les données changent, le Modèle peut notifier les autres composants, notamment la Vue, afin qu’elle puisse se mettre à jour si nécessaire.

La Vue quant à elle est responsable de l’affichage. Elle présente les données fournies par le Modèle à l’utilisateur sous une forme visuelle (comme une page web, une fenêtre d’application, etc.). Elle n’est pas seulement passive : elle capte aussi les actions de l’utilisateur (comme un clic ou une saisie) et les transmet au Contrôleur pour traitement.

Enfin, le Contrôleur agit comme un intermédiaire entre la Vue et le Modèle. Il reçoit les requêtes de l’utilisateur (par exemple via des formulaires ou des liens), traite ces informations (souvent en interagissant avec le Modèle), et décide quelle Vue afficher en retour. Ainsi, il orchestre le flux d’informations entre les différents éléments de l’application.

Grâce à cette organisation claire, l’architecture MVC favorise la maintenabilité , la réutilisabilité du code et une meilleure gestion des évolutions futures de l’application

V. Installation et Configuration

1. Prérequis techniques

- Node.js : Pour exécuter le serveur backend.
- npm (Node Package Manager) : Pour installer les dépendances.
- MongoDB : Pour stocker les données de la clinique.

2. Installation de Node.js et npm :

- Téléchargez et installez Node.js depuis le site officiel.
- npm est installé automatiquement avec Node.js.

3. Installation de MongoDB :

- Téléchargez MongoDB Community Server.
- Démarrez le service MongoDB sur votre machine.

4. Configuration du Backend :

- Initialisez un projet Node.js avec npm init.
- Installez Express et Mongoose (ORM pour MongoDB).
- Configurez un fichier .env pour les variables d'environnement (URL MongoDB).

5. **Configuration du Frontend :**
    - Utilisez npx create-react-app pour créer un projet React.
    - Installez Axios pour les requêtes HTTP.
    - Configurez Bootstrap pour le design.

6. **Lancement de l'Application** :

    - Démarrez le serveur backend avec npm start.
    - Démarrez le frontend avec npm start.

VI. Base de Données

1. **Structure des Collections :**

    - owners : Informations sur les propriétaires (nom, adresse, téléphone).
    - pets : Informations sur les animaux (nom, espèce, âge) et leur propriétaire.
    - visits : Détails des visites médicales (date, motif, vétérinaire).
    - veterinarians : Liste des vétérinaires (nom, spécialisation)

2. **Relations entre les Collections :**

    - Un propriétaire peut avoir plusieurs animaux (relation One-to-Many).
    - Un animal peut avoir plusieurs visites (relation One-to-Many).

3. **Exemple de Données Initiales :**

    - Un propriétaire nommé John Doe avec deux animaux : Max (Chien) et Mimi (Chat).

- Des visites associées à chaque animal

4. **Requêtes MongoDB pour Remplir la Base de Données**
    - db.owners.insertOne({...}) pour ajouter un propriétaire.
    - db.pets.insertOne({...}) pour ajouter un animal.
    - db.visits.insertOne({...}) pour ajouter une visite.

VII. Développement du Backend

1. Création des Modèles (Mongoose) :

    - Un modèle Owner pour les propriétaires.
    - Un modèle viterinarian pour les veterinaire.
    - Un modèle Pet pour les animaux.
    - Un modèle Visit pour les visites.

2. **Implémentation des Contrôleurs :**

    - Un contrôleur pour gérer les opérations CRUD (Create, Read, Update, Delete) pour chaque modèle.
    - Exemple : getOwners() pour récupérer tous les propriétaires.

3. **Définition des Routes API :**

    - GET /owners : Récupérer tous les propriétaires.
    - POST /owners : Ajouter un propriétaire.
    - PUT /owners/:id : Modifier un propriétaire.
    - DELETE /owners/:id : Supprimer un

propriétaire.

4. **Tests des API avec Postman :**

- Test des différentes routes pour vérifier leur bon fonctionnement.

VIII. Développement du Frontend

1. **Structure du Projet React :**

- Composants pour chaque fonctionnalité : Liste des vétérinaires, recherche de propriétaires, etc.
- Utilisation de React Router pour la navigation entre les pages.

2. **Pages Principales et leurs Fonctionnalités :**

- Page d'Accueil : Présentation de l'application.
- Liste des Vétérinaires : Affichage des vétérinaires depuis la base de données.
- Recherche de Propriétaire : Recherche par nom de famille.
- Ajout de Propriétaire : Formulaire pour créer un nouveau propriétaire.
- Détail des Animaux : Visualisation des animaux associés à un propriétaire.

3. **Intégration avec le Backend via Axios** :

- Utilisation de Axios pour les requêtes HTTP (GET, POST, PUT, DELETE).
- Configuration d'une URL de base pour simplifier les appels.

4. Styles avec Bootstrap :

- Utilisation de Bootstrap pour une mise en page simple et responsive.

IX. Pages de l'Application

Chaque page est décrite :

- Page d'Accueil : Présentation de l'application.
- Liste des Vétérinaires : Affichage de tous les vétérinaires.
- Recherche de Propriétaire : Rechercher un propriétaire par nom.
- Nouveau Propriétaire : Formulaire pour ajouter un nouveau propriétaire.
- Résultat d'Ajout : Confirmation après l'ajout d'un propriétaire.
- Modifier les Informations du Propriétaire : Formulaire de modification.
- Ajouter un Animal de Compagnie : Associer un animal à un propriétaire.
- Liste des Propriétaires : Résultat de recherche des propriétaires.
- Informations sur le Propriétaire et ses Animaux :

Détails sur le propriétaire et ses animaux.
- Modifier les Informations de l'Animal : Éditer les détails d'un animal.
- Ajouter une Visite : Ajouter une nouvelle visite médicale.
- Synthèse après l'Ajout d'une Visite : Résumé des visites de l'animal.

X. Gestion des Versions

1. Utilisation de Git et GitHub

    - Utilisation de Git pour le contrôle des versions.
    - GitHub pour héberger le code et collaborer.

2. Workflow de Développement Collaboratif

    - Branches pour chaque fonctionnalité.
    - Fusion des branches après validation.

3. Bonnes Pratiques pour les Commits et les Branches

    - Messages de commit clairs.
    - Une branche par fonctionnalité.

XI. Améliorations Futures

- Fonctionnalités Supplémentaires : Gestion des rendez-vous en ligne, notifications par email.
- Optimisation des Performances : Améliorer la rapidité de l'interface.
- Sécurité Renforcée : Protéger les données sensibles des utilisateurs.

XII. **Conclusion générale**

VetCare 360 est une solution numérique innovante dédiée à la gestion des cliniques vétérinaires. Grâce à son architecture MERN (MongoDB, Express, React, Node.js), l'application offre une interface utilisateur moderne et intuitive, facilitant la gestion des propriétaires, des animaux et des visites médicales.

L'approche modulaire adoptée dans le développement garantit une évolutivité optimale, permettant à l'application de s'adapter facilement aux futurs besoins de la clinique. Chaque fonctionnalité a été pensée pour offrir une expérience utilisateur fluide, tout en maintenant une organisation claire des données.

L'utilisation de MongoDB pour la base de données assure une flexibilité et une rapidité de gestion des informations, tandis que Node.js et Express assurent une gestion sécurisée des requêtes et des opérations côté serveur. Le frontend en React, associé à Bootstrap, garantit une interface visuellement attrayante et responsive.

Dans un souci d'amélioration continue, plusieurs axes de développement futur ont été identifiés, notamment l'ajout de fonctionnalités supplémentaires (gestion des rendez-vous en ligne, notifications par email), l'optimisation des performances de l'application, et le renforcement de la sécurité des données.

En conclusion, VetCare 360 représente une solution complète, évolutive et sécurisée, parfaitement adaptée aux besoins des cliniques vétérinaires modernes. Son développement basé sur une architecture MERN garantit une évolutivité et une fiabilité à long terme