



المدرسة العليا للتكنولوجيا الناھيہ
École Supérieure de Technologie de Nador
۱۳۹۸ + ۱۹۷۸ | +۰۵۲۴۸۰۰۰۰

Rapport de projet académique

Sous le sujet :

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

360

Encadré par : Esbai
Redoune

Réalisé par : Essadiki
Douae

Année universitaire : 2024/2025

Introduction :

Présentation du projet :

L'objectif du projet ‘VetCare 360’ est de développer une application web pour la gestion des cliniques vétérinaires à l'aide des technologies modernes MERN (MongoDB, Express, React, Node.js). L'application permet de gérer toutes les informations sur les animaux, leurs propriétaires et les visites médicales, leur proposant une interface conviviale et intuitive. Dans le monde actuel, un tel système numérique est conçue pour simplifier considérablement le travail des médecins et du personnel, en assurant un accès plus rapide aux informations essentielles.

Analyse du besoin

Problèmes rencontrés par les propriétaires de cliniques vétérinaires :

Il existe de nombreux défis lors de la gestion d'une clinique vétérinaire qui rendent difficile le fonctionnement quotidien et ont un impact négatif sur la qualité des soins. Voici certains des problèmes les plus courants auxquels les propriétaires sont confrontés:

- Mener des dossiers médicaux traditionnels :

Bien que de nombreuses cliniques vétérinaires fonctionnent avec des dossiers papier ou des feuilles Excel qui contiennent des informations sur des animaux et leurs propriétaires, ce sont les méthodes trompeuses car contraignant.

- Difficulté d'accès aux données :
Dans de nombreuses cliniques, les données des animaux ne sont pas centralisées, ce qui rend l'accès aux informations cruciales lent et inefficace. En cas d'urgence, chaque seconde compte, et l'incapacité d'accéder rapidement aux dossiers médicaux peut nuire à la qualité des soins.
- Gestion des rendez-vous :
L'organisation des rendez-vous médicaux peut être un vrai casse-tête. Sans un système centralisé, il est facile de se retrouver avec des doublons, des rendez-vous manqués ou des créneaux mal coordonnés, créant ainsi du stress et des erreurs dans la prise en charge des animaux.
- Communication avec les propriétaires :
Les cliniques vétérinaires ont souvent des difficultés à communiquer efficacement avec les propriétaires, ce qui entraîne des oublis de rendez-vous, de traitements et de vaccinations, affectant la qualité des soins et la satisfaction des clients.

Objectifs généraux et fonctionnels du projet :

Le projet "VetCare 360" a pour objectif de résoudre ces problèmes en proposant une solution numérique complète qui facilite la gestion des cliniques vétérinaires. Voici les objectifs clés de ce projet :

VetCare 360 améliore également la gestion des données en centralisant les informations sur les animaux, les propriétaires, et les visites pour rendre l'accès rapide et précis. L'application permet donc un accès facile des vétérinaires à leur dossier médical, ce qui facilite la prise de décisions . De plus, il permet une meilleure gestion des rendez-vous en automatisant leur planification. Cela facilite également le processus de recherche des informations du propriétaire et de communication avec lui.

Analyse de base de données

1. Afficher les tables/groupes

Étant donné que le projet est basé sur MongoDB (une base de données NoSQL), au lieu de « tables », nous avons 4 collections, et chaque collection contient des documents au format JSON.

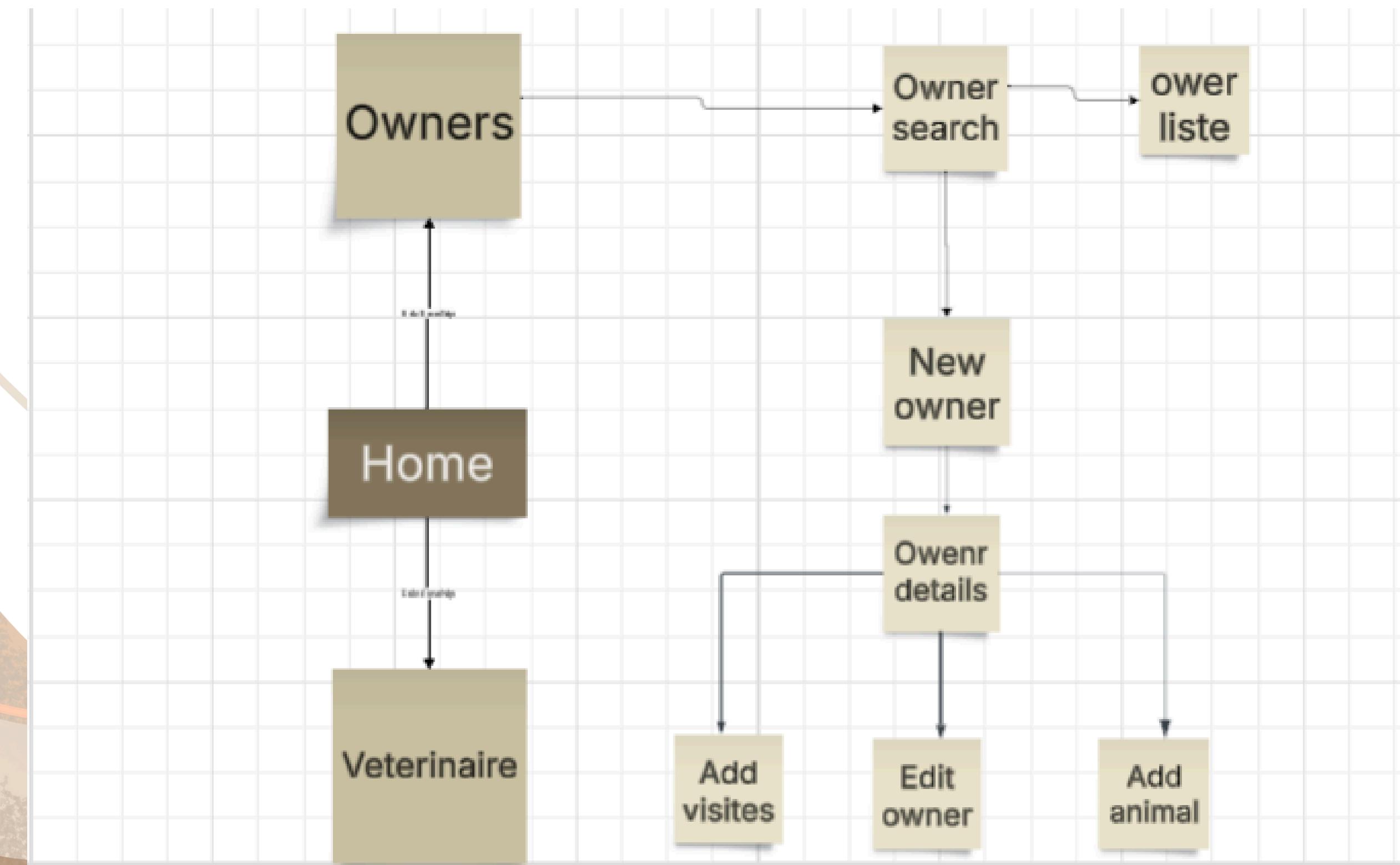
- Collection: Proprietaire
- Collection: Animal
- Collection: Visite
- Collection: Veterinaire

On prendre collection propriétaire comme exemple :

```
{
  _id: ObjectId('6817fbdaee6794a9452d8c2'),
  nom: 'bouzid',
  prenom: 'anware',
  telephone: '0682098919',
  email: 'anwar@gmail.com',
  ville: 'Imzourn',
  adresse: 'rue14',
  animaux: [
    ObjectId('6817fbf3ee6794a9452d8c2b'),
    ObjectId('68180931ee6794a9452d8c43'),
    ObjectId('68181ca8ee6794a9452d8c88')
  ],
  __v: 3
}.
```

Flux de navigation : (Navigation Flow)

Ce diagramme montre comment un utilisateur navigue entre différentes pages d'une application dans une interface utilisateur créée avec React :



Wireframes :

- Figure 1 : Page « Liste des propriétés ».

VetCare 360

Home veterinarians owners

List of owners

Num	Last Name	First Name	Phone Number	Email	City	Address	Animals	Actions
1	Zarioh	chaymae	0682098945	chayzar@gmail.com	Nador	ben tayb	bob jojo	Delete
2	essadiki	Douae	0771220981	medessa@gmail.com	AL HOCIEMA	imzourn	mimi	Delete
3	bouzid	anware	0682098919	anwar@gmail.com	Imzourn	rue14	pop fox jojo	Delete

Wireframes :

- Figure 2 : La page « Infos propriétaire avec historique des visites»

VetCare 360

Home veterinarians owners

Owner Information

Name	Douae essadiki
Address	imzourn
City	AL HOCIEMA
Telephone	0771220981

[Edit Owner](#) [Add New Pet](#)

Pets and Visits

Name: mimi
Birth Date: 5/6/2025
Type: cat

Visit History

Visit Date	Description	Actions
5/11/2025	virus	Delete

[Edit Pet](#) [Add Visit](#) [Delete Pet](#)

Activate Windows
Go to Settings to activate Windows

Wireframes :

- Figure 3 : Page « Ajouter une visite ».

The wireframe shows a user interface for adding a visit to an animal. At the top, there is a dark header bar with the text "VetCare 360" on the left and navigation links for "Home", "veterinarians", and "owners" on the right. Below the header, the main content area has a title "Add a visit for the animal". It contains two input fields: one for "mm/dd/yyyy" and another for "Description", both with placeholder text. A blue button labeled "Add the visit" is positioned below these fields. In the bottom left corner of the main area, there is a section titled "Previous visits" containing two entries: "5/11/2025 — virus" and "5/22/2025 — maladie".

VetCare 360

Home veterinarians owners

Add a visit for the animal

mm/dd/yyyy

Description

Add the visit

Previous visits

5/11/2025 — virus

5/22/2025 — maladie

Technologies utilisées :

Pourquoi on utilise MERN Stack ?

On utilise MERN Stack (MongoDB, Express.js, React.js, Node.js) pour développer l'application VetCare 360 car il fournit un environnement de développement intégré utilisant un seul langage, JavaScript, sur le front-end et le back-end. Cette approche simplifie le processus de développement, rend le code plus standardisé et réduit la complexité de l'utilisation de différents langages.

Principales raisons d'utiliser MERN Stack :

- JavaScript : Écrire le front-end et le back-end dans un seul langage.
- Haute performance : grâce à Node.js et au moteur V8.
- Évolutivité : Idéal pour les applications à croissance rapide comme VetCare 360.
- Grande communauté et large soutien : des milliers de bibliothèques et de références disponibles.
- Intégration naturelle : tous les composants fonctionnent de manière transparente les uns avec les autres.

Avantages des technologies MERN et des outils auxiliaires

◆ MongoDB

- Une base de données NoSQL flexible qui stocke les données sous forme de documents JSON.
- Idéal pour stocker des données complexes telles que des informations sur l'animal, le propriétaire et la visite.
- Prend en charge la mise à l'échelle horizontale et les opérations de lecture/écriture rapides.

◆ Express.js

- Un framework léger pour créer des API.
- S'intègre facilement avec MongoDB et Node.js.
- Aide à organiser le code et à contrôler le routage du middleware.

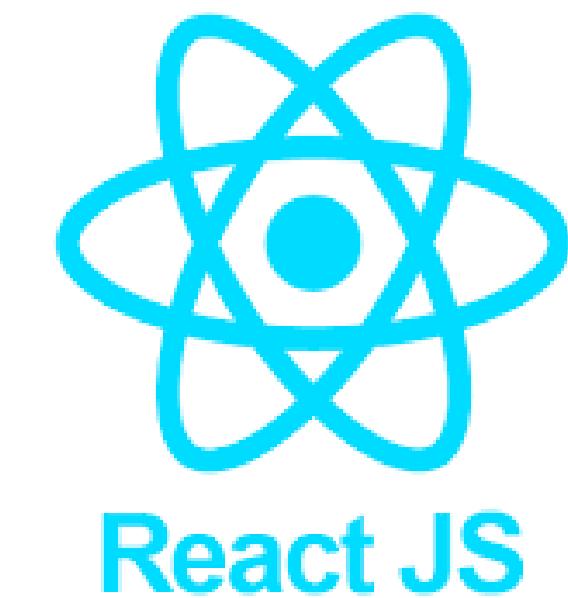
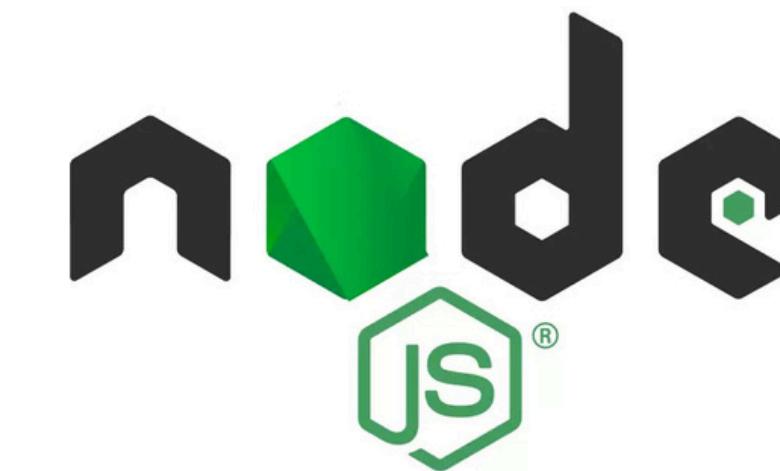


◆ React.js

- Une bibliothèque d'interface utilisateur qui permet de créer des composants interactifs.
- S'appuie sur Virtual DOM pour une mise à jour rapide et fluide des éléments.
- Parfaitement adapté aux applications avec une interface dynamique telle que VetCare 360.

◆ Node.js

- Un environnement d'exécution basé sur JavaScript en dehors du navigateur.
- Fournit un serveur rapide et asynchrone qui prend en charge un grand nombre de connexions.
- Vous permet de créer des applications puissantes et flexibles.

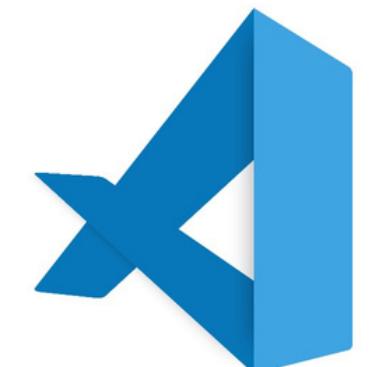


◆ Outils utiles :

- Visual Studio Code (VSCode) : un éditeur de code source utilisé pour le développement front-end et back-end.
- Postman : testez les API et envoyez des requêtes visuellement.
- Git et GitHub : suivez les modifications, enregistrez les versions et collaborez.
- npm (Node Package Manager) : pour installer facilement des bibliothèques et des composants de projet.



POSTMAN



Visual Studio



Pour les composants principaux de React :

- **Component:**

dans React, les composants sont les éléments de base d'une application. Les composants peuvent être des composants fonctionnels ou des composants de classe. En règle générale, les composants fonctionnels sont préférés car ils sont plus simples et plus efficaces.

- **State:**

utilisé pour stocker des informations dans le composant. Le statut peut changer en fonction de l'interaction de l'utilisateur.

- **Props:**

un moyen de transmettre des données d'un composant parent à un composant enfant.

- **UseEffect :**

un moyen de gérer les opérations secondaires telles que les appels d'API ou la gestion

Exemple :

Comme exemple pratique d'utilisation des composants et de l'état dans React, prenons le composant AjouterAnimal.js qui affiche un formulaire pour ajouter un nouvel animal. useState est utilisé pour stocker formData et également pour afficher un message d'erreur en cas de problème avec la saisie ou lorsque la demande au serveur échoue.

- formData : pour stocker les données du formulaire telles que le nom, le sexe et la date de naissance.
- erreur : pour stocker les messages d'erreur qui peuvent apparaître lorsqu'un problème survient.

```
const AjouterAnimal = () => {
  const location = useLocation();
  const navigate = useNavigate();
  const owner = location.state?.owner;

  const [formData, setFormData] = useState({
    name: '',
    type: '',
    birthDate: ''
  });
  const [error, setError] = useState('');
```

- **App.js:**

Le fichier App.js fait partie du front-end de l'application VetCare 360. Il gère la navigation entre différentes pages et affiche les composants en fonction de chemins définis. Il contient également des styles CSS intégrés pour personnaliser l'apparence de l'application.

Caractéristiques principales :

- **Routage :**

Vous utiliserez React Router pour gérer le routage entre les différentes pages de l'application.

Différents chemins sont identifiés tels que :

/ : Home ;

/veterinaires : Page des vétérinaires.

/ajouter-animal/:ownerId : Ajouter une page d'animal.

- **Navbar :**

La barre de navigation s'affiche en haut de l'application et contient des liens vers différentes parties de l'application telles que « Accueil », « Vétérinaires » et « Propriétaires ».

Pour les composants principaux de Backend:

- **Configuration express :**

Dans le backend, Express.js a été utilisé pour créer le serveur qui gère les requêtes HTTP et fournit les données de la base de données. Le serveur est configuré sur le port 5000 dans le projet, en utilisant express.json() pour convertir les données JSON envoyées depuis le front-end en données traitables.

- **Configuration d'une connexion à MongoDB :**

Dans cette partie, vous vous connectez à une base de données MongoDB à l'aide de Mongoose. Une base de données spécifique au projet appelée vetcare est définie. Si la connexion réussit, un message sera imprimé sur la console l'indiquant.

```
// MongoDB Connection
console.log('Attempting to connect to MongoDB...');
mongoose.connect('mongodb://127.0.0.1:27017/vetcare', {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => {
  console.log('✅ Connected to MongoDB successfully');
  console.log('Database name:', mongoose.connection.name);
  console.log('Host:', mongoose.connection.host);
})
.catch(err) => {
  console.error('✗ MongoDB connection error:', err);
  process.exit(1); // Exit if cannot connect to database
});
```

- Configuration des modèles dans MongoDB :

Mongoose a été utilisé pour créer des modèles qui représentent les données stockées dans la base de données. Par exemple, créez un modèle Animal pour stocker les informations sur les animaux de compagnie, qui sont liées à un modèle Propriétaire.



```
vetcare-backend > models > JS Animal.js > ...
C:\Users\douae\myappnp\src\pages\Vete
1  const mongoose = require('mongoose');
2
3  const animalSchema = new mongoose.Schema({
4    name: { type: String, required: true },
5    birthDate: { type: Date, required: true },
6    type: { type: String, required: true },
7    owner: { type: mongoose.Schema.Types.ObjectId, ref: 'Proprietaire', required: true },
8    visites: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Visite' }]
9  );
10
11 const Animal = mongoose.model('Animal', animalSchema);
12
13 module.exports = Animal;
14
```

- Configuration des Routes :

Les données sont gérées via des routes Express. Ces chemins incluent les opérations POST et GET, par exemple pour ajouter de nouveaux animaux à la base de données et pour récupérer tous les animaux stockés.

```
// routes/animals.js
const express = require('express');
const router = express.Router();
const Animal = require('../models/Animal');
const Proprietaire = require('../models/Proprietaire');
const { addAnimal, getAnimalById } = require('../controllers/animalController');

// Log all requests to this router
router.use((req, res, next) => {
  console.log(`[Animals Route] ${req.method} ${req.originalUrl}`);
  next();
});

// Utilisation de la fonction addAnimal de controllers/animalController.js
router.post('/', addAnimal);

// Chemin pour récupérer l'animal par ID
router.get('/:id', getAnimalById);
```

- **Expérience de connexion et de test :**

Le serveur fonctionnait sur localhost en utilisant le port 5000, et nous nous sommes assurés que le frontend était capable d'interagir avec ces routes en utilisant des requêtes HTTP comme POST et GET.

Tests

tester l'interface de l'ajouter ou supprimer : les données ont été saisies via le formulaire, puis la base de données a été vérifiée pour garantir que les données ont été ajoutées ou supprimées correctement

Défis et solutions

Exemples de problèmes auxquels j'ai été confronté et comment je les ai résolus :

- Afficher l'historique des visites : Il était difficile d'afficher correctement l'historique des visites dans l'interface utilisateur, ce qui pouvait rendre difficile le suivi de l'historique des visites.

Solustion: La conception de la page SyntheseVisite a été améliorée pour afficher l'historique des visites dans un format plus organisé et plus facile à lire, et chaque visite est désormais correctement liée à l'animal et au propriétaire.

- erreur des routes avec mongodb : J'ai eu quelques problèmes lors de la récupération des données de MongoDB lorsque les requêtes contenaient des erreurs de chemin.

Solution : les requêtes MongoDB ont été optimisées et les chemins corrigés pour s'adapter à la structure de données correcte. Des gestionnaires d'erreurs ont également été ajoutés pour fournir des messages clairs aux utilisateurs en cas de problème.

Les etapes d'instlation les languages :

- L'installation de React s'est faite en suivant la documentation officielle sur le site de [React](#). Pour initialiser l'interface utilisateur, la commande suivante a été utilisée :

`npx create-react-app myappnpx`

- Le serveur backend a été développé avec Express.js. Après avoir créé le projet, les dépendances ont été installées via la commande suivante :

`npm install express mongoose cors dotenv`

- Ensuite, les bibliothèques nécessaires (comme React Router, Bootstrap, etc.) ont été installées avec :

`npm install react-router-dom bootstrap`

Conclusion et ce quii a fait :

Conception et développement de l'application terminés : La conception et le développement de l'application VetCare 360 ont été réalisés avec succès à l'aide des technologies MERN (MongoDB, Express, React, Node.js).

- Mise en œuvre des fonctions de base : Les fonctions de base telles que l'ajout d'animaux, de propriétaires, de visites et l'affichage de rapports historiques sont correctement implémentées.
- Effectuer des tests complets : des tests manuels et automatisés ont été effectués pour vérifier toutes les fonctionnalités de l'application.



En conclusion, le projet VetCare 360 était une expérience aussi technique qu'éducative pour moi. Il m'a offert l'occasion d'exercer mes compétences en développement Web, en réalisant une application complète en utilisant MERN stack. J'ai vraiment appris à concevoir une base de données, créer une interface utilisateur interactive et développer des API hautement et performantes.

Je tiens à remercier sincèrement mon professeur "Esbai Redoune " de m'avoir donné l'opportunité de réaliser ce projet, qui a grandement contribué à mon développement personnel .