Documentation technique

Dépot git API : https://github.com/Clement-Fourtout/Api-zoo

Contient les addons Jawsdb – Object Rocket – Aws S3 énuméré dans cette Documentation technique

I. Réflexions Initiales Technologiques sur le Sujet

1. *Choix des Technologies :*

- **VSC :** L'IDE par excellence
- **Front-end :** *React.js* pour sa réactivité et ses composants modulaires.
- **Back-end :** *Node.js* avec Express pour sa simplicité et ses performances.
- **Base de Données relationnelle :** *JawsDb MySQL* pour sa robustesse et sa capacité à gérer de grandes quantités de données relationnelles (sur Api).
- **Base de données non-relationnelle :** *Object Rocket MongoDb* est une base de données NoSQL, elle possède une architecture de données évolutive et facile à utiliser même si Atlas MongoDb aurait suffit (sur Api).
- **Gestion de l'état** : Pour la gestion de l'état, nous avons opté pour les hooks *useState* et *useEffect* de React, qui permettent une gestion simple et efficace de l'état local des composants.
- **Authentification :** *JWT (JSON Web Tokens)* pour la gestion sécurisée des sessions.
- **Stockage des Fichiers :** *AWS S3* pour une gestion fiable et évolutive des fichiers (sur Api).

2. *Justification des choix :*

- **IDE :** Virtual Studio Code et l'IDE par excellence, il est open source, utilisable sur différentes plateformes et personnalisable grâce à son nombre d'extensions important.
- **React.js :** Offre une bonne performance et une structure flexible pour construire des interfaces utilisateur dynamiques avec une grand facilité et maniabilité. Cette bibliothèque est également la plus connue et recommandée. Je l'ai choisit pour sa simplicité d'utilisation et sa facilité d'apprentissage.
- **Node.js et Express :** Permet une écriture JavaScript uniforme pour le serveur et le client, simplifiant le développement. Elle permet d'exécuter plusieurs requêtes vers le serveur simultanément et possède des performances inégalées. Cela m'a permis de créer une application Front et Back en JavaScript.

- **MySQL :** Une solution SQL bien établie et fiable, avec une vaste communauté et de la documentation. Grâce à ses grandes performances, elle peut gérer plusieurs connexions simultanées, un grand nombre d'enregistrement. Elle est également plus adaptée à mon projet où une petite base de données suffit.
  JWT : Fournit un moyen sécurisé de gérer les sessions utilisateur sans nécessiter de stockage côté serveur. Associé à un système de hachage Bcrypt pour les authentifications cela assure une sécurité maximale.
- **S3 Bucket :** AWS propose un service d'hébergement cloud pour les fichiers performants et sécurisé. La Console AWS permet une gestion totale du Bucket de façon simple. S3 est également évolutive, disponible et peu couteuse.

II. Configuration de l'environnement de travail

- **IDE :** Visual Studio Code recommandé pour sa flexibilité et ses extensions.
- **Node.js et npm :** Installer à partir de Node.js.
- **MySQL :** Installer en temps qu'addons sur heroku puis configurer avec la CLI mysql et ses variables d'environnements. (sur Api)
- **ObjectRocket :** Installer en temps qu'addons sur heroku puis configurer avec npm mongoose et ses variables d'environnements. (sur Api)
- **AWS CLI :** Installer à partir de AWS CLI et utiliser les variables d'environnements pour les données sensibles AWS. (sur Api)

Installation :

- **Clone du dépôt :**
  - **git clone** https://github.com/Clement-Fourtout/Arcadia-zoo
  - **cd** Arcadia-zoo

- **Installation des dépendances :**
  npm install

- **Configuration de la base de données relationnelle :** (sur Api)
  - Créer une base de données MySQL.
  - Configurer les variables d'environnement pour la connexion à la base de données dans les variables de configurations Heroku.

- **Configuration de la base de données non relationnelle :** (sur Api)
  - Créer une base de données NoSql ObjectRocket MongoDb.
  - Configurer les variables d'environnement pour la connexion à la base de données dans les variables de configurations Heroku.

1. **Configuration AWS S3 :** (sur Api)
   - Configurer les clés d'accès AWS.
   - Créer un bucket S3 pour le stockage des fichiers.
   - Configurer les variables d'environnement pour la connexion à la base de données dans les variables de configurations Heroku.

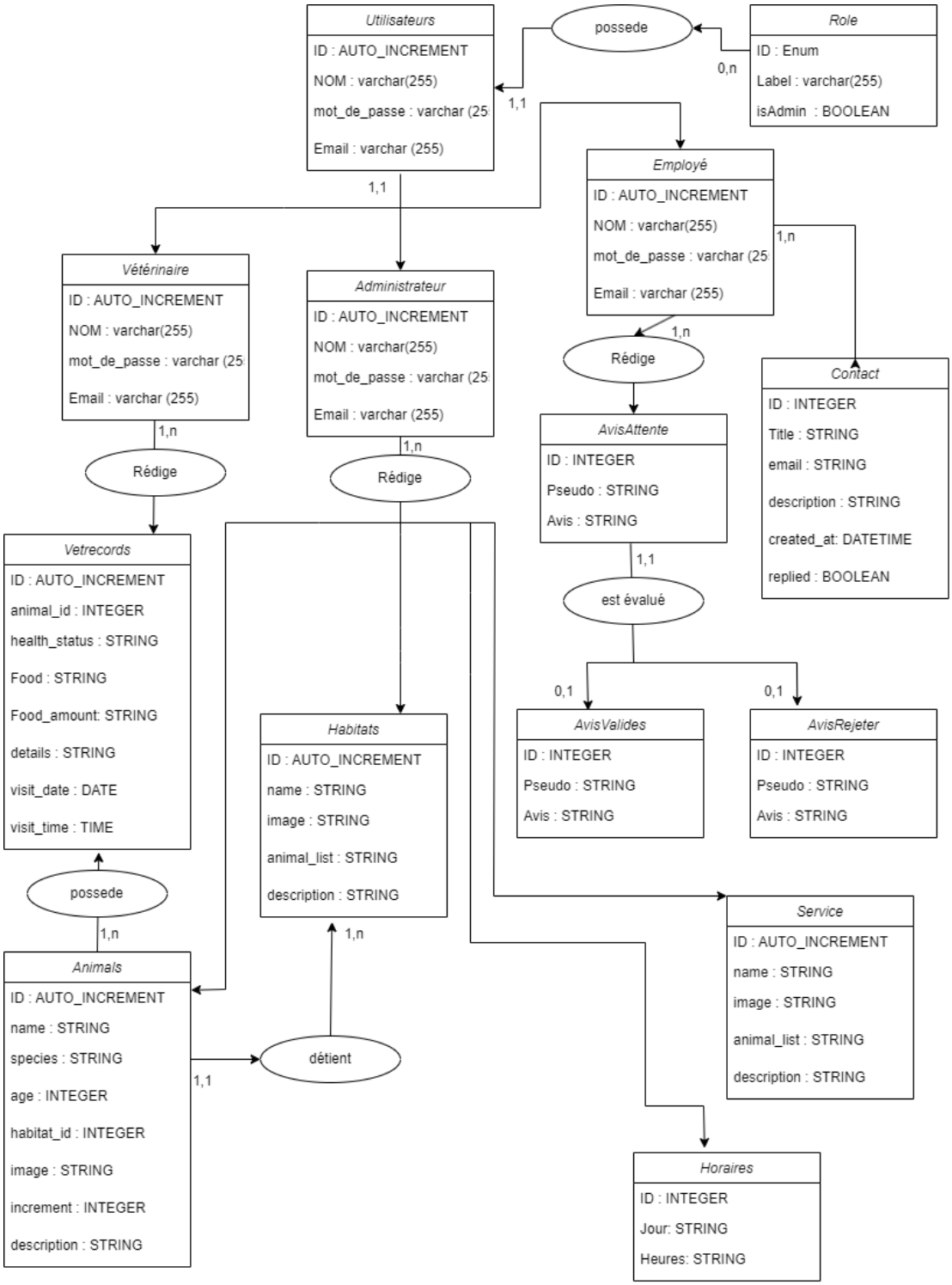
2. **Lancement de l'application :**

   - **Local :**
     npm start

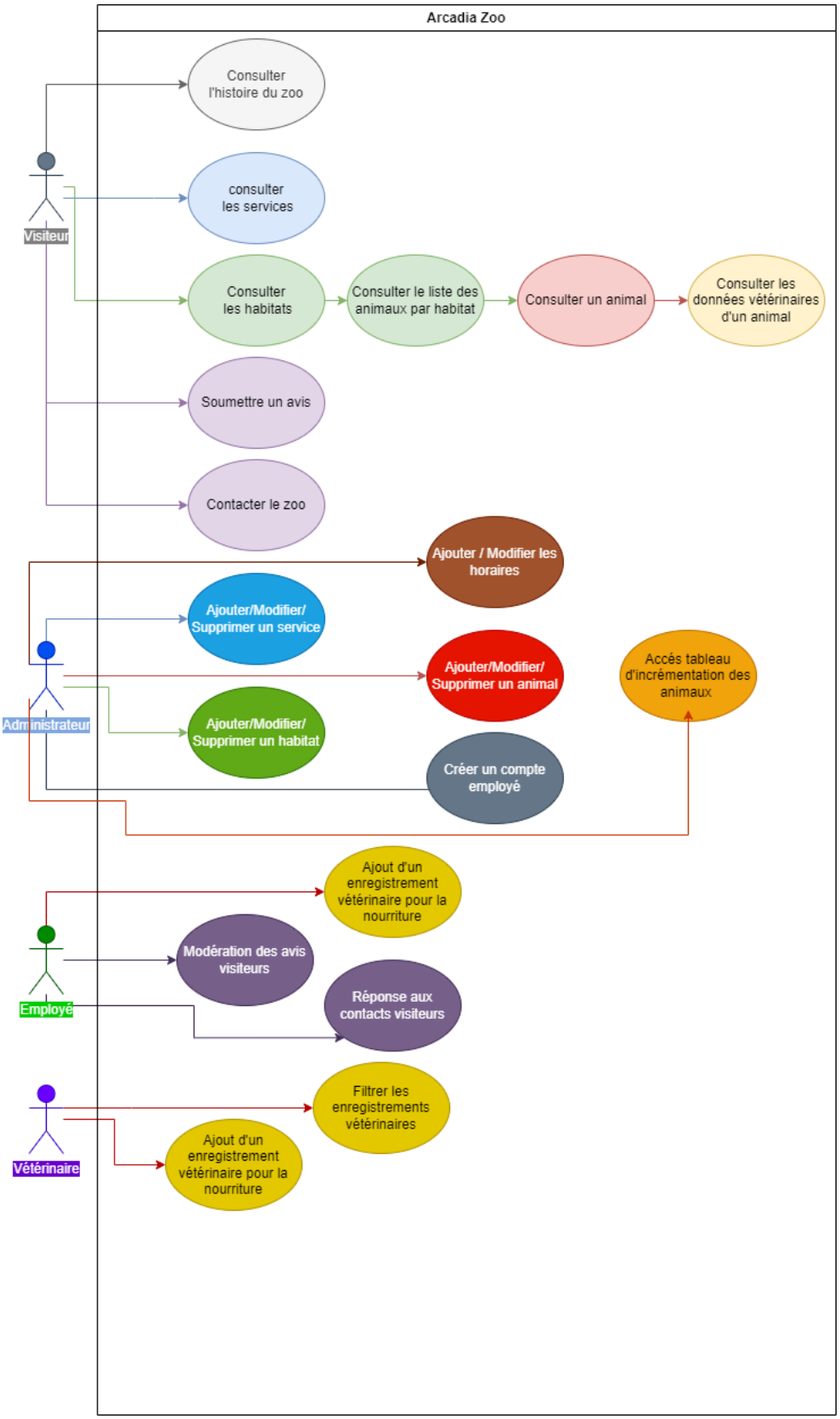
   - **Déploiement :**
     - **git add .**
     - **git commit –m “**message de repère de push**”**
     - **git push** heroku main

III. Modèle Conceptuel de Données (ou Diagramme de Classe)

I. Diagramme de classe

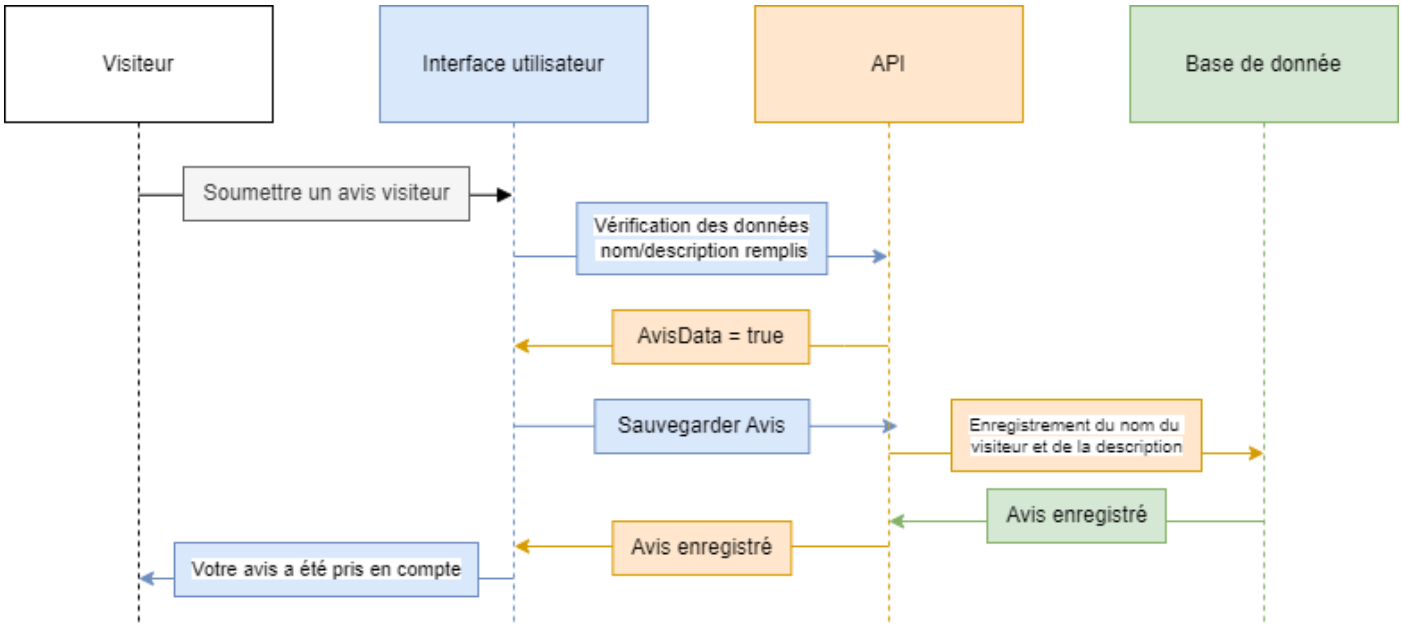


IV. Diagramme de cas d'utilisation

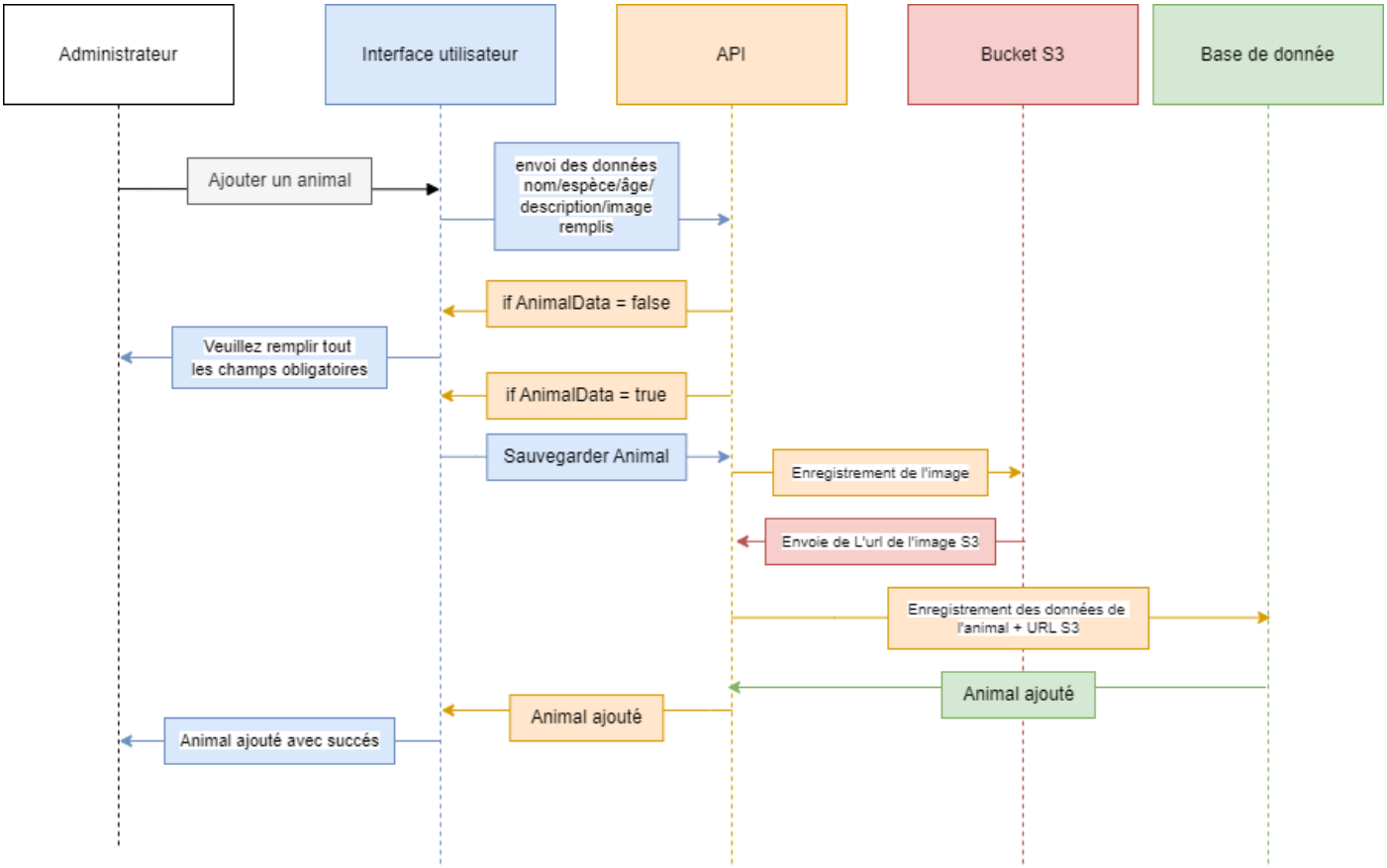


IV. Diagramme de séquence

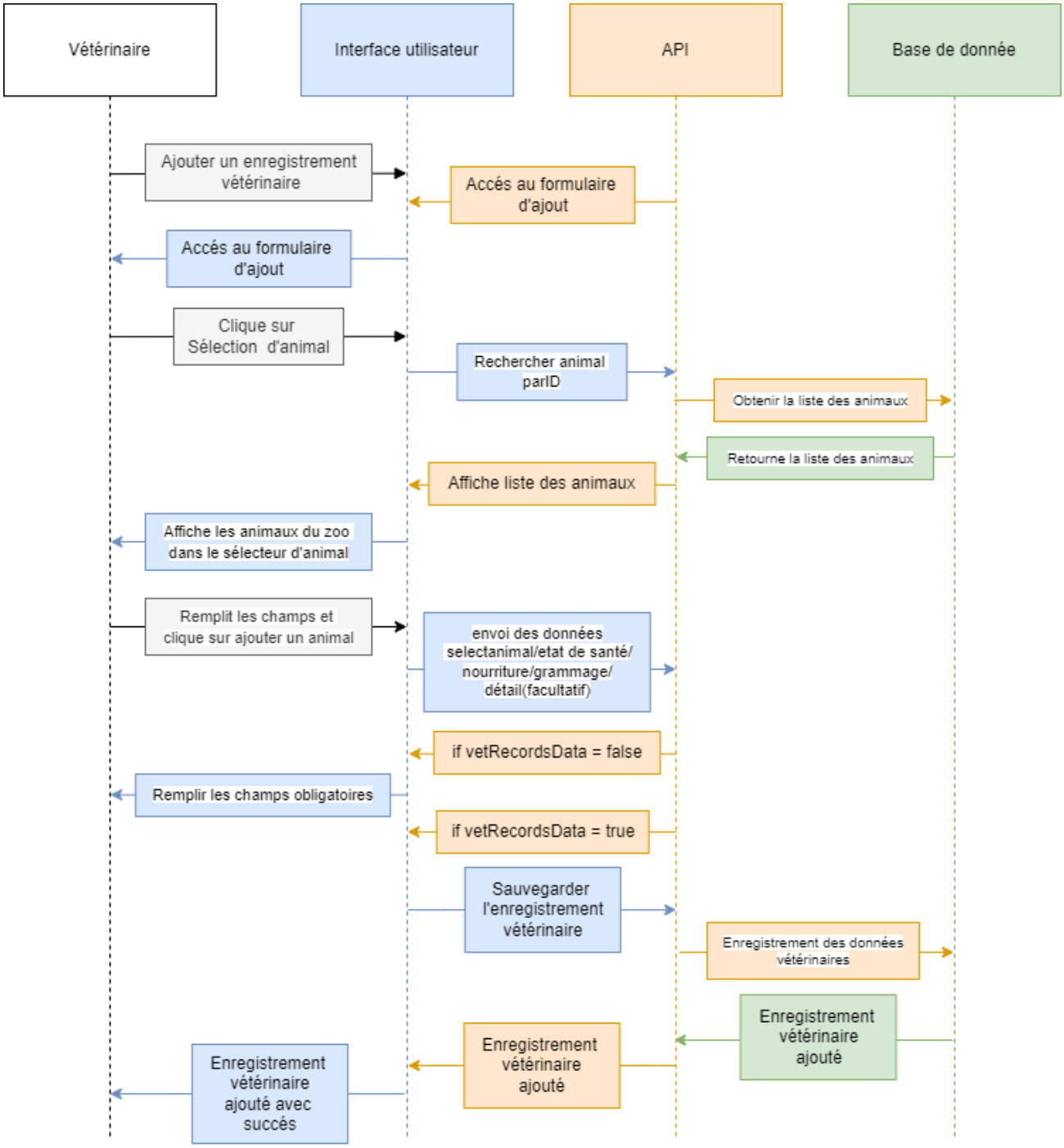
- Visiteur souhaite soumettre un avis



- Administrateur ajoute un Animal au zoo



- Vétérinaire ajoute un Enregistrement vétérinaire



V. ***Documentation du déploiement de l'application***

Démarche et Étapes de Déploiement :

1. Préparation :

- S'assurer que l'ensemble du code est stable et fonctionnel en local
- Merge toutes les branches pertinentes dans la branche principale

2. Configuration du serveur :

- Choisir un service d'hébergement (Heroku pour ce projet)
- Création d'un compte Heroku
- Installation de Heroku CLI
- Heroku login (ligne de commande)

3. Déploiement : (actions effectuées en ligne de commande)

- Heroku :

  - Git init
  - Heroku create Arcadia-zoo
  - Git add .
  - Git commit –m "Déploiement heroku"
  - Git push heroku main

  Dès lors notre application est déployée et visitable en ligne via son URL de déploiement Heroku.

4. Test post-déploiement :

- Vérification de la bonne mise en ligne de l'application à l'URL indiquée.
- Tester l'ensemble des fonctionnalités et pages de l'application pour assurer qu'il n'y ait aucune régression.

5. Surveillance et maintenance :

- Analyser les logs pour suivre les performances et les erreurs.