Documentation Technique : Application Arcadia

1. Introduction

L'application Arcadia permet de gérer différents aspects liés à la gestion du zoo, y compris les animaux, les employés, les vétérinaires. Il existe trois rôles dans l'application :

- **Administrateur** : Responsable de la gestion globale de l'application.
- **Vétérinaire** : Gère les animaux, les soins et les suivis médicaux.
- **Employé** : Gère les tâches quotidiennes liées aux animaux et à l'entretien.

L'application est développée avec **React**, **Vite**, **Chakra UI** pour le frontend, et **Node.js**, **Express** et **MongoDB** pour le backend et la base de donnée. Le frontend et le backend sont séparés en deux répertoires différents, chacun étant une entité GitHub distincte.

2. Réflexion Initiale Technologique

2.1 Objectifs du Projet

L'objectif est de créer une application web moderne, réactive et évolutive pour la gestion d'un zoo. L'application doit permettre une gestion fluide des animaux, du personnel, et des soins, tout en permettant aux utilisateurs d'interagir facilement avec l'application.

2.2 Choix Technologiques

- **Frontend** :

    1. **React** : Bibliothèque JavaScript populaire pour créer des interfaces utilisateur dynamiques et réactives.
    2. **Vite** : Outil de développement moderne qui offre une compilation plus rapide et une meilleure expérience de développement.
    3. **Chakra UI** : Bibliothèque de composants pour une interface utilisateur moderne et accessible avec des éléments prêts à l'emploi.

- **Backend** :

    1. **Node.js** : Plateforme JavaScript permettant de créer des applications backend performantes.
    2. **Express.js** : Framework minimaliste pour Node.js permettant de faciliter le développement de l'API.
    3. **MongoDB** : Base de données NoSQL flexible et scalable, adaptée pour stocker les informations sur les animaux, les employés et les vétérinaires.

2.3 Architecture de l'Application

L'architecture de l'application est divisée en deux parties :

1. **Frontend** : Gère l'interface utilisateur (UI), les interactions avec l'utilisateur et consomme les API du backend.
2. **Backend** : Gère la logique métier, l'authentification, et la gestion des données dans MongoDB.

Les deux parties sont séparées en répertoires distincts avec des repos GitHub séparés pour faciliter le déploiement, la gestion du code source.

3. Configuration de l'Environnement de Travail

3.1 Prérequis

- **Node.js** et **npm** doivent être installés sur le système.
- **MongoDB** doit être installé localement ou un service de MongoDB cloud (comme MongoDB Atlas) peut être utilisé.

3.2 Structure des Dossiers

La structure des dossiers est organisée de la manière suivante :

```
/zoo-frontend
  ├── /src
  ├── /public
  ├── package.json
  ├── vite.config.js
  └── .gitignore
/zoo-api
  ├── /models
  ├── /routes
  ├── /controllers
  ├── /middleware
  ├── server.js
  └── package.json
```

- **/zoo-frontend** : Le dossier contenant le code source du frontend (React).
- **/zoo-api** : Le dossier contenant le code source du backend (Node.js + Express).

3.3 Configuration du Backend (Node.js + Express)

Dans le backend, le fichier principal est **server.js**. Ce fichier lance le serveur Express, se connecte à la base de données MongoDB et définit les routes API pour gérer les différentes ressources (animaux, employés, vétérinaires). Le backend est également responsable de la gestion des rôles des utilisateurs.

3.4 Configuration du Frontend (React + Vite)

Le frontend est créé avec React et Vite pour une expérience de développement rapide. Le projet est configuré avec Chakra UI pour faciliter la création d'une interface utilisateur moderne et accessible. L'application frontend consomme les API du backend pour afficher et manipuler les données.

Chaque page ou fonctionnalité (comme la gestion des animaux, des employés ou des soins vétérinaires) est divisée en composants React. Ces composants sont ensuite assemblés dans des pages ou des sections.

3.5 Configuration des Repos GitHub

- **Frontend** : Un dépôt GitHub est dédié au code source du frontend. Ce dépôt contient tout le nécessaire pour déployer et gérer l'interface utilisateur.
- **Backend** : Un autre dépôt GitHub est dédié au code source du backend. Ce dépôt contient les fichiers pour la logique métier, l'API, et la gestion des données.

Chaque projet est géré séparément sur GitHub, avec une branche principale pour chaque dépôt. Cela permet un développement parallèle et un contrôle de version optimal.

4. Gestion des Rôles

L'application gère trois rôles distincts, chacun ayant des permissions et un accès différent :

- **Administrateur** : Accès complet à toutes les fonctionnalités du backend, y compris la gestion des employés, des vétérinaires et des animaux.
- **Vétérinaire** : Accès limité à la gestion des soins des animaux et de leur historique médical.
- **Employé** : Accès aux tâches quotidiennes liées aux animaux, comme l'alimentation.

Les rôles sont définis dans le backend et appliqués via un système de permissions. Chaque utilisateur est associé à un rôle spécifique, et l'application vérifie ces rôles pour déterminer quelles actions un utilisateur peut effectuer.