

```

import api from "../Utils/api";
import { API_ENDPOINTS } from "../Utils/constant";

const animalService = {
  // Create a new animal
  createAnimal: async (animalData) => {
    try {
      const response = await api.post(API_ENDPOINTS.ANIMAL, animalData);
      return response.data;
    } catch (error) {
      console.error("Error creating animal:", error);
      throw error.response?.data?.message || "Failed to create animal";
    }
  },

  // Get all animals
  getAllAnimals: async () => {
    try {
      const response = await api.get(API_ENDPOINTS.ANIMAL);
      return response;
    } catch (error) {
      console.error("Error fetching animals:", error);
      throw error.response?.data?.message || "Failed to fetch animals";
    }
  },

```

```

  // Get animal by ID
  getAnimalById: async (id) => {
    try {
      const response = await api.get(`${API_ENDPOINTS.ANIMAL}/${id}`);
      console.log('API Response:', response);
      return response;
    } catch (error) {
      console.error("Error fetching animal by ID:", error);
      throw error.response?.data?.message || "Failed to fetch animal";
    }
  },

```

```

  // Update an animal
  updateAnimal: async (id, animalData) => {
    try {
      const response = await api.put(`${API_ENDPOINTS.ANIMAL}/${id}`, animalData);
      return response.data; // Return the updated animal data
    } catch (error) {
      console.error("Error updating animal:", error);
      throw error.response?.data?.message || "Failed to update animal";
    }
  },

```

```
// Delete an animal
deleteAnimal: async (id) => {
  try {
    const response = await api.delete(`${API_ENDPOINTS.ANIMAL}/${id}`);
    return response.data; // Return the response after deletion
  } catch (error) {
    console.error("Error deleting animal:", error);
    throw error.response?.data?.message || "Failed to delete animal";
  }
};

export default animalService;
```

Explication du fichier animalService.js

Ce fichier définit un **service API** pour gérer les **animaux** dans une application. Il utilise **Axios** pour envoyer des requêtes HTTP à une API, permettant ainsi de **créer, récupérer, mettre à jour et supprimer** des animaux.

• Fonctionnement des différentes méthodes

• Création d'un animal (createAnimal)

Cette méthode envoie une **requête POST** pour ajouter un nouvel animal à l'API. Elle prend en paramètre `animalData`, qui contient les informations de l'animal à créer. En cas d'erreur, un message est affiché et une exception est levée.

• Récupération de tous les animaux (getAllAnimals)

Elle effectue une **requête GET** pour récupérer la liste complète des animaux stockés dans l'API. Si une erreur survient, elle est loggée dans la console et un message d'erreur est retourné.

• Récupération d'un animal spécifique (getAnimalById)

Cette méthode utilise une **requête GET** pour récupérer un animal précis en fonction de son id. Elle affiche également la réponse API dans la console pour faciliter le débogage.



• Mise à jour d'un animal (updateAnimal)

Elle envoie une **requête PUT** pour modifier un animal existant. Elle prend en paramètres `id` (l'identifiant de l'animal à mettre à jour) et `animalData` (les nouvelles données à enregistrer).

• Suppression d'un animal (deleteAnimal)

Cette méthode effectue une **requête DELETE** pour supprimer un animal via son id. Elle retourne la réponse API confirmant la suppression.

Résumé des fonctionnalités

 Méthode	 Action
createAnimal	Ajoute un nouvel animal
getAllAnimals	Récupère la liste de tous les animaux
getAnimalById	Récupère un animal par son identifiant
updateAnimal	Met à jour un animal
deleteAnimal	Supprime un animal

```
import {
  Box, Button, FormControl, FormLabel, Input, Textarea, FormErrorMessage,
  IconButton, Modal, ModalOverlay, ModalContent, ModalHeader, ModalFooter,
  ModalBody, ModalCloseButton, useDisclosure, useToast, Grid, Select,
} from '@chakra-ui/react';
import { AddIcon } from '@chakra-ui/icons';
import { Formik, Field, Form, ErrorMessage } from 'formik';
import * as Yup from 'yup';
import { useMutation, useQueryClient, useQuery } from '@tanstack/react-query';
import animalService from '../../Services/animalService';
import habitatService from '../../Services/habitatService'; // Supposons
que vous avez ce service pour récupérer les habitats
import { useState } from 'react';

const CreateAnimal = () => {
  const { isOpen, onOpen, onClose } = useDisclosure();
  const toast = useToast();
  const queryClient = useQueryClient();
  const [file, setFile] = useState(null); // État pour le fichier image

  // Récupérer les habitats lorsque le composant est monté
  const { data: habitats = [], isLoading, isError } = useQuery({
    queryKey: ['habitats'],
    queryFn: habitatService.getAllHabitats,
    onError: (error) => {
      toast({
        title: 'Erreur lors de la récupération des habitats',
        description: error.message,
        status: 'error',
        duration: 5000,
        isClosable: true,
      });
    },
  });
};
```

```

const createAnimalMutation = useMutation({
  mutationFn: animalService.createAnimal,
  onSuccess: () => {
    toast({
      title: 'Animal créé avec succès !',
      status: 'success',
      duration: 5000,
      isClosable: true,
    });
    queryClient.invalidateQueries({ queryKey: ['animals'] });
    onClose();
  },
  onError: (error) => {
    toast({
      title: 'Erreur lors de la création de l\'animal.',
      description: error.response?.data?.message || error.message,
      status: 'error',
      duration: 5000,
      isClosable: true,
    });
    console.error('Erreur lors de la création de l\'animal :',
error.response?.data || error.message);
  },
});

```

composant CreateAnimal

Ce composant React permet d'**ajouter un nouvel animal** via un **formulaire modal** utilisant Chakra UI et Formik.

Récupération des habitats → Utilisation de useQuery pour charger les habitats depuis l'API.

Création d'un animal → Envoi des données (nom, race, habitat et image) via useMutation et FormData.

```

import React, { useState, useEffect } from 'react';
import {
  Box,
  Heading,
  Text,
  IconButton,
  Button,
  useDisclosure,
  Modal,
  ModalOverlay,
  ModalContent,
  ModalHeader,
  ModalBody,
  ModalFooter,

```

```

FormControl,
FormLabel,
Input,
useToast,
Spinner,
HStack,
Flex,
Avatar,
Grid,
Table,
Thead,
Tr,
Th,
Tbody,
Td,
} from '@chakra-ui/react';
import { EditIcon, DeleteIcon, InfoIcon } from '@chakra-ui/icons';
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import animalService from '../../Services/animalService';
import CreateAnimal from './CreateAnimal';

const AnimalComponent = () => {
  const [selectedAnimal, setSelectedAnimal] = useState(null);
  const [updatedAnimalData, setUpdatedAnimalData] = useState({});
  const { isOpen: isEditModalOpen, onOpen: onEditOpen, onClose: onEditClose } = useDisclosure();
  const { isOpen: isDetailsModalOpen, onOpen: onDetailsOpen, onClose: onDetailsClose } = useDisclosure();
  const toast = useToast();
  const queryClient = useQueryClient();

  const { data: animals = [], isLoading, isError } = useQuery({
    queryKey: ['animals'],
    queryFn: animalService.getAllAnimals,
    onError: (error) => {
      toast({
        title: 'Error fetching animals',
        description: error.message,
        status: 'error',
        duration: 5000,
        isClosable: true,
      });
    },
  });

  const deleteAnimalMutation = useMutation({
    mutationFn: animalService.deleteAnimal,
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ['animals'] });
    },
  });

```

```

    toast({
      title: 'Animal deleted',
      description: 'The animal was successfully deleted.',
      status: 'success',
      duration: 3000,
      isClosable: true,
    });
  },
  onError: (error) => {
    toast({
      title: 'Error deleting animal',
      description: error.message,
      status: 'error',
      duration: 5000,
      isClosable: true,
    });
  },
});

const updateAnimalMutation = useMutation({
  mutationFn: ({ id, updatedData }) => animalService.updateAnimal(id,
updatedData),
  onSuccess: () => {
    queryClient.invalidateQueries({ queryKey: ['animals'] });
    toast({
      title: 'Animal updated',
      description: 'The animal was successfully updated.',
      status: 'success',
      duration: 3000,
      isClosable: true,
    });
    onEditClose();
  },
  onError: (error) => {
    toast({
      title: 'Error updating animal',
      description: error.message,
      status: 'error',
      duration: 5000,
      isClosable: true,
    });
  },
});

const handleDelete = (id) => {
  if (window.confirm('Are you sure you want to delete this animal?')) {
    deleteAnimalMutation.mutate(id);
  }
};

```

```
const handleEdit = (animal) => {
  setSelectedAnimal(animal);
  setUpdatedAnimalData(animal);
  onEditOpen();
};

const handleUpdate = () => {
  updateAnimalMutation.mutate({ id: selectedAnimal._id, updatedData:
updatedAnimalData });
};
```

Ce composant **AnimalComponent** gère l'affichage, la modification et la suppression des animaux.

- **Récupération des animaux** : Utilise `useQuery` pour récupérer la liste des animaux via `animalService.getAllAnimals()`.
- **Création d'un animal** : Gérée par le composant `CreateAnimal`.
- **Suppression d'un animal** : Utilise `useMutation` avec `animalService.deleteAnimal()`, puis met à jour la liste des animaux.
- **Modification d'un animal** : Ouvre un modal de modification et utilise `useMutation` avec `animalService.updateAnimal()`.