Proyecto de Veterinaria

BACKEND (Node.js + Express + Sequelize + SQLite)

1. db.js – Conexión con la base de datos

javascript

CopiarEditar

```
const { Sequelize } = require('sequelize');
const sequelize = new Sequelize({
  dialect: 'sqlite',
  storage: './database.sqlite',
});

module.exports = sequelize;
```

¿Qué hace?

- Usa Sequelize para conectar con una base de datos SQLite.
- Define la ubicación del archivo .sqlite que almacena los datos.
- Exporta la instancia de conexión para que otros archivos puedan utilizarla.

2. models/Pet.js – Modelo de Mascotas

javascript

CopiarEditar

```
const { DataTypes } = require('sequelize');
const sequelize = require('../db');

const Pet = sequelize.define('Pet', {
  name: DataTypes.STRING,
  species: DataTypes.STRING,
  age: DataTypes.INTEGER,
  healthStatus: DataTypes.STRING,
```

```
});


module.exports = Pet;
```

¿Qué hace?

- Define un modelo de datos llamado Pet que representa a una mascota.
- Los campos incluyen nombre (name), especie (species), edad (age) y estado de salud (healthStatus).
- Este modelo permite realizar operaciones CRUD directamente sobre la tabla Pets.

3. index.js – Servidor principal y rutas API

javascript

CopiarEditar

```
const express = require('express');

const cors = require('cors');

const sequelize = require('./db');

const Pet = require('./models/Pet');


const app = express();

app.use(cors());

app.use(express.json());


// Rutas CRUD

app.get('/pets', async (req, res) => res.json(await Pet.findAll()));

app.post('/pets', async (req, res) => res.json(await Pet.create(req.body)));

app.put('/pets/:id', async (req, res) => {

  const pet = await Pet.findByPk(req.params.id);

  pet.set(req.body);

  await pet.save();
```

```
  res.json(pet);
});
app.delete('/pets/:id', async (req, res) => {
  const pet = await Pet.findByPk(req.params.id);
  await pet.destroy();
  res.json({ message: 'Mascota eliminada' });
});

// Iniciar servidor
sequelize.sync().then(() => {
  app.listen(3001, () => console.log('Servidor en http://localhost:3001'));
});
```

¿Qué hace?

- Crea un servidor Express con CORS y JSON habilitados.
- Implementa las rutas principales de la API para:
  - GET /pets: obtener todas las mascotas.
  - POST /pets: crear una nueva mascota.
  - PUT /pets/:id: actualizar datos de una mascota por ID.
  - DELETE /pets/:id: eliminar una mascota.
- Conecta con la base de datos y arranca el servidor en el puerto 3001.

FRONTEND (React)

¿Cómo funciona?

El frontend (normalmente ubicado en una carpeta como /frontend o /src) se conecta al backend para mostrar, registrar, editar y eliminar mascotas. Aquí algunos puntos clave:

- **Componentes**: cada componente representa una parte de la interfaz (lista de mascotas, formulario, etc.).
- **useEffect y fetch**: se usan para consumir la API del backend (http://localhost:3001/pets).

- **useState**: mantiene el estado de las mascotas y los formularios.
- **Axios o fetch**: envían solicitudes al backend para obtener o modificar datos.

Funcionalidades comunes del frontend:

- Ver todas las mascotas desde el backend.
- Registrar nuevas mascotas (formulario).
- Editar información existente.
- Eliminar mascotas con confirmación