**Script - Calendario de Eventos**

La aplicación "Calendario de Eventos" permite a los usuarios visualizar, crear, actualizar y eliminar eventos en un calendario interactivo utilizando FullCalendar en el frontend y una API REST en el backend.

**Backend**

**Estructura**

backend/

├── controllers/

├── models/

├── routes/

├── config/

├── app.js

└── database.js

**Configuración de Sequelize**

Archivo: config/database.js

const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('basededatos', 'usuario', 'contraseña', {

host: 'localhost',

dialect: 'mysql',

});

module.exports = sequelize;

**Modelo de Evento**

Archivo: models/Event.js

const { DataTypes } = require('sequelize');

const sequelize = require('../config/database');

const Event = sequelize.define('Event', {

title: DataTypes.STRING,

description: DataTypes.STRING,

start: DataTypes.DATE,

end: DataTypes.DATE,

color: DataTypes.STRING,

allDay: DataTypes.BOOLEAN,

});

module.exports = Event;

**Rutas del Backend**

Archivo: routes/events.js

const express = require('express');

const router = express.Router();

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

router.get('/', async (req, res) => {

const events = await Event.findAll();

res.json(events);

});

router.post('/', async (req, res) => {

const event = await Event.create(req.body);

res.json(event);

});

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

await Event.update(req.body, { where: { id: req.params.id } });

res.json({ status: 'Updated' });

});

router.delete('/:id', async (req, res) => {

await Event.destroy({ where: { id: req.params.id } });

res.json({ status: 'Deleted' });

});

module.exports = router;

**Frontend**

**Estructura**

frontend/

├── src/

│ ├── components/

│ │ └── Calendar.vue

│ └── App.vue

├── public/

├── package.json

└── vite.config.js

**Componente Calendar.vue**

<template>

<FullCalendar :events="events" />

</template>

<script setup>

import FullCalendar from '@fullcalendar/vue3';

import dayGridPlugin from '@fullcalendar/daygrid';

import interactionPlugin from '@fullcalendar/interaction';

import { ref, onMounted } from 'vue';

import axios from 'axios';

const events = ref([]);

onMounted(async () => {

const response = await axios.get('http://localhost:3000/events');

events.value = response.data;

});

</script>

**Interacción con el Backend**

El componente Calendar.vue se comunica con el backend a través de peticiones HTTP usando Axios para obtener y actualizar eventos en la base de datos.

**Instalación y Configuración**

**Requisitos**

* Node.js y npm
* MySQL
* Vite (para frontend)

**Pasos para el Backend**

1. Crear base de datos y actualizar credenciales en config/database.js
2. Instalar dependencias:

npm install express sequelize mysql2

1. Sincronizar modelos y arrancar el servidor:

node app.js

**Pasos para el Frontend**

1. Instalar dependencias:

npm install

1. Instalar FullCalendar:

npm install @fullcalendar/vue3 @fullcalendar/daygrid @fullcalendar/interaction axios

1. Ejecutar frontend:

npm run dev

**Uso de la Aplicación**

1. **Visualización de eventos**: Al abrir la aplicación, se cargan todos los eventos registrados en la base de datos.
2. **Agregar un evento**:
   * Haz clic en “Agregar Evento” o selecciona un rango de fechas
   * Completa el formulario con título, descripción, fecha de inicio/fin, color y si es un evento de día completo
   * Guarda el evento
3. **Editar un evento**:
   * Haz clic en un evento existente
   * Modifica los datos y guarda
4. **Eliminar un evento**:
   * Haz clic en un evento y luego en el botón “Eliminar”
5. **Arrastrar un evento**:
   * Puedes mover eventos para cambiar la fecha

**Consideraciones Finales**

* Se recomienda proteger el backend con autenticación en producción.
* FullCalendar ofrece más plugins como timeGrid y list view para mejorar la experiencia.
* La base de datos puede exportarse para backups o migraciones.