DOCUMENTACIÓN DE LA APLICACIÓN: VUELOS

Descripcion del Proyecto

Este proyecto consiste en desarrollar una aplicación web para gestionar vuelos. La aplicación permitirá realizar operaciones de alta (crear), baja (eliminar) y modificación (actualizar) de vuelos. La información de cada vuelo se visualizará en una tabla, que incluirá los siguientes detalles:

- Número de vuelo: Identificador único del vuelo (Texto)
- Horario de llegada: Fecha y hora de llegada del vuelo (Fecha)
- Línea aérea: Nombre de la aerolínea operadora del vuelo (Texto)
- Demorado: Indica si el vuelo está demorado (Booleano)

Requisitos:

Backend:

Tecnologías Utilizadas:

- Express: Framework para construir la API RESTful.
- MongoDB: Base de datos NoSQL para almacenar la información de los vuelos.
- Mongoose: Librería para interactuar con MongoDB usando un modelo basado en esquemas.
- Morgan: Middleware para el registro de solicitudes HTTP.
- Zod: Biblioteca para la validación de datos en el backend.
- body-parser: Middleware para analizar cuerpos de solicitudes entrantes en formato JSON.
- nodemon: Herramienta para reiniciar automáticamente el servidor durante el desarrollo.

Funcionalidades:

1. Alta de vuelos: Endpoint para crear nuevos registros de vuelo.
2. Baja de vuelos: Endpoint para eliminar vuelos existentes.
3. Modificación de vuelos: Endpoint para actualizar detalles de vuelos existentes.

4. Obtención de vuelos: Endpoint para recuperar todos los registros de vuelos y visualizarlos en la tabla del frontend.

Frontend:

Tecnologías Utilizadas:

- ReactJS: Biblioteca para construir la interfaz de usuario de la aplicación.
- React-Router: Manejo de rutas para la navegación entre diferentes vistas de la aplicación.
- React Hook Form: Gestión de formularios y validación en el frontend.
- Axios: Cliente HTTP para realizar solicitudes a la API con interceptores para manejar errores y respuestas.
- useContext: Hook de React para manejar el estado global y compartir información entre componentes.
- TailwindCss: Framework para estilización de entornos visuales html.
- SweetAlert2: Despliega Alertas para mostrar las respuestas del servidor.

Funcionalidades:

1. Formulario de alta: Interfaz para ingresar los datos de un nuevo vuelo.
2. Formulario de modificación: Interfaz para editar los detalles de un vuelo existente.
3. Formulario de baja: Interfaz para seleccionar y eliminar un vuelo.
4. Visualización en tabla: Mostrar todos los vuelos en una tabla, con la capacidad de ordenar y filtrar datos si es necesario.

Planificación del Desarrollo

1. **Diseño de la Base de Datos:**
    - Crear un esquema de Mongoose para el modelo de vuelo que incluya los campos flightNumber, timeArrive, airlineName y isDelayed.
2. **Desarrollo del Backend:**
    - Configurar la conexión a MongoDB.
    - Implementar los endpoints para alta, baja, modificación y obtención de vuelos.

    Se estableció una API para la comunicación entre el entorno del Cliente y el Servidor. Para ello, se crearon cuatro rutas.

    (Ruta raíz: ‘http://localhost:3000/api’)

Post ‘/flight’
Put ‘/flight/:id’
Delete ‘/flight/:id’
Get ‘/flights’
-

- Implementar la validación de datos con Zod y manejar errores apropiadamente.

3. **Desarrollo del Frontend:**
   - Configurar la estructura del proyecto con React y React-Router.
   - Crear los componentes para los formularios de alta, baja y modificación de vuelos.
   - Implementar la tabla para mostrar la lista de vuelos utilizando datos obtenidos de la API.
   - Configurar Axios con interceptores para manejar respuestas y errores de la API.
   - Usar React Hook Form para gestionar la validación de formularios.
4. **Pruebas:**
   - Realizar pruebas unitarias y de integración tanto en el frontend como en el backend.
   - Verificar el correcto funcionamiento de los formularios y la visualización en la tabla.
5. **Configuración:**
   - Configurar el servidor backend y el frontend para que se comuniquen correctamente.

Necesidades

- Entorno de Desarrollo: Configuración de Node.js, MongoDB, herramientas de desarrollo como VSCode y extensiones para la gestión de base de datos.
- Recursos de API: Documentación y pruebas de la API para asegurar la correcta comunicación entre el frontend y el backend.
- Control de Versiones: Uso de Git para el control de versiones y colaboración en el desarrollo del proyecto.

Enlaces externos que se usaron para el apoyo del desarrollador

- Vite: https://vitejs.dev
- Tailwind: https://tailwindcss.com
- React Hook Form: https://react-hook-form.com/get-started
- Axios: https://axios-http.com/docs/intro