Actividades unidad 6: Asincronía en JavaScript II

Para los siguientes ejercicios utiliza:

- db.json.
- json server.
- Incluye por CDN Axios en el HTML: *<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>*

1. Proyecto base y primera petición GET con Axios

Crea la estructura base del proyecto y realiza la primera petición GET con Axios, que servirá como punto de partida para el resto de ejercicios.

La estructura que debe tener el proyecto es:

```
/axios-jsonserver
├── index.html
├── js/
│   └── app.js
├── css/
│   └── styles.css (opcional)
├── db.json (ya creado y ejecutándose con JSON Server)
```

Crea el archivo index.html, que deberá incluir:

1. Un título visible en la página.
2. Un contenedor ( ) para mostrar mensajes de estado.
3. Un contenedor (<ul>) donde se mostrarán los alumnos.
4. Inclusión de Axios mediante CDN.
5. Enlace al archivo app.js.

En app.js realiza los siguientes apartados:

1. rea una constante con la URL base del servidor: http://localhost:3000/alumnos
2. Realiza una petición GET usando Axios y promesas (.then / .catch).
3. Muestra en consola:
    a. El objeto completo response.
    b. Los datos obtenidos con response.data.

4. Accede a una propiedad concreta de cada alumno (por ejemplo, nombre) y muéstrala por consola.
5. Gestiona los errores mostrando el mensaje de error por consola.
6. Después, comenta la petición anterior y repite la petición utilizando async/await. y try/catch.
7. Muestra por consola los datos obtenidos.
8. Si ocurre un error, muestra el código de estado HTTP.

2. Mostrar datos en el DOM

Partiendo del proyecto creado en el Ejercicio 1. Muestra en la página los alumnos obtenidos desde JSON Server utilizando Axios.

Realiza los siguientes apartados:

1. Utiliza el index.html y app.js existentes.
2. Crea una función mostrarAlumnos(alumnos) que reciba un array de alumnos.
3. Recorre response.data y muestra cada alumno dentro del contenedor ya creado (#lista-alumnos).
4. Muestra, para cada alumno:
    - Nombre.
    - Email.
    - Curso.
5. Muestra un mensaje de "Cargando alumnos…" en el contenedor de mensajes mientras se realiza la petición.
6. Si ocurre un error en la petición, muestra un mensaje de error visible en la página.

3. Crear alumnos

Partiendo del mismo index.html y app.js del Ejercicio 1.Añade nuevos alumnos al servidor usando Axios y actualiza la interfaz.

Realiza los siguientes apartados:

1. Añade al index.html un formulario para crear alumnos con:
    - Nombre.
    - Email.
    - Curso.
2. Captura el evento submit del formulario desde app.js.
3. Envía los datos al servidor usando una petición POST con Axios y async/await.
4. Gestiona la petición con try/catch.
5. Muestra un mensaje de éxito cuando el alumno se cree correctamente.
6. Añade el nuevo alumno a la lista sin recargar la página.

7. Limpia el formulario tras el envío.

4. Editar alumnos

Usando el mismo proyecto base de los ejercicios anteriores.
Modifica la información de un alumno existente.

Realiza los siguientes apartados:

1. Añade un botón “Editar” a cada alumno mostrado en la lista.
2. Al pulsar el botón, carga los datos del alumno en el formulario existente.
3. Guarda el id del alumno que se está editando.
4. Realiza una petición PUT con Axios al alumno correspondiente.
5. Muestra un mensaje de confirmación cuando la actualización sea correcta.
6. Gestiona el error si el alumno no existe o el servidor falla.
7. Actualiza la lista de alumnos sin recargar la página.

5. Eliminar alumnos

Usando el mismo proyecto base de los ejercicios anteriores.
Elimina alumnos del servidor y del DOM.

Realiza los siguientes apartados:

1. Añade un botón “Eliminar” junto a cada alumno.
2. Solicita confirmación antes de realizar el borrado.
3. Realiza una petición DELETE con Axios utilizando async/await.
4. Muestra el código de estado HTTP devuelto por el servidor.
5. Elimina el alumno del DOM si la petición es correcta.
6. Gestiona errores de red o del servidor mostrando un mensaje adecuado.