UTN - Facultad Regional Córdoba (FRC)
Desarrollo de Software (DDS)
Docente: Ing. Alejandro Rey Corvalán

5 de mayo de 2025

INSTANCIA DE EVALUACIÓN PARCIAL: Sistema Básico de Gestión de Libros

Objetivo: Implementar una aplicación full stack básica para gestionar un listado de libros, enfocándose en la lectura (listar todos y filtrar por título) y eliminación de registros, utilizando la stack tecnológica vista en clase.

Contexto: Se requiere un sistema simple para mantener un registro de libros. La aplicación debe permitir visualizar un listado completo de libros, buscar libros *exclusivamente* por título, y eliminar libros individuales del registro.

Tecnologías a Utilizar:

- **Backend:** Node.js, Express.js, Sequelize (con SQLite como base de datos simple), Nodemon (para desarrollo).
- **Frontend:** HTML5, CSS3 (básico), Bootstrap 5 (CDN), JavaScript (Fetch API).
- **Entorno de desarrollo:** Visual Studio Code (VS Code).

Prerrequisitos:

- Descargar **Parcial3k4Grupo1_05_05_2025.zip** y descomprimir en su carpeta de trabajo

---

REQUISITOS FUNCIONALES:

1. **Listado de Libros:** Al cargar la página, se debe mostrar un listado de todos los libros registrados en la base de datos en una tabla.
2. **Búsqueda por Título:** Debe existir un campo de texto que permita filtrar el listado de libros *únicamente por el Título del libro*. Al escribir y/o confirmar la búsqueda (ej: presionando Enter o un botón "Buscar"), la lista debe actualizarse mostrando solo los libros cuyo Título coincida (total o parcialmente, sin distinguir mayúsculas/minúsculas) con el texto ingresado. Si el campo de búsqueda está vacío, se debe mostrar el listado completo.

3. **Eliminación de Libro:** Cada fila en el listado debe tener un botón "Eliminar". Al hacer clic en este botón, se debe solicitar una confirmación al usuario (ej: `confirm()`). Si el usuario confirma, se debe eliminar el libro correspondiente de la base de datos y actualizar el listado en la interfaz.

REQUISITOS TÉCNICOS:

1. **Estructura de Proyecto:** El proyecto debe estar organizado en dos carpetas principales: `backend` y `frontend`.
2. **Backend:**
    - Utilizar Express para crear el servidor HTTP.
    - Utilizar Sequelize como ORM y SQLite como base de datos.
    - Se proporciona el modelo `Libro` con los atributos `IdLibro` (PK, AI), `Titulo`, `Autor`, `AnioPublicacion`.
    - Se proporciona la configuración inicial de Sequelize, la sincronización (`force: true` en desarrollo) y la función de seeding para insertar **exactamente 40 datos iniciales diferentes** si la tabla está vacía.
    - Deben implementar las siguientes rutas (API RESTful):
        - `GET /api/libros`: Obtener todos los libros.
        - `GET /api/libros?search=[term]`: Obtener libros filtrados *únicamente* por el término de búsqueda en el `Titulo`.
        - `DELETE /api/libros/:id`: Eliminar un libro específico por su `IdLibro`.
    - Configurar `nodemon` para reiniciar el servidor automáticamente durante el desarrollo.
3. **Frontend:**
    - Utilizar HTML para la estructura básica de la página. **Se proporciona el HTML base, pero deben agregar los IDs necesarios** a los elementos (input de búsqueda, botón de búsqueda, contenedor de la tabla, cuerpo de la tabla, etc.) para poder referenciarlos desde JavaScript.
    - Utilizar Bootstrap 5 (CDN) para el diseño y layout (tabla, estilos básicos de botones y formulario de búsqueda).
    - Utilizar JavaScript puro (ES6+) y la Fetch API para interactuar con el backend.
    - Referenciar los elementos HTML necesarios (que ustedes deberán identificar y agregar IDs) utilizando métodos como `getElementById`.
    - Manejar dinámicamente el DOM para mostrar el listado de libros en la tabla, y manejar mensajes de carga o error (opcional, pero valorado), así como el mensaje de "No se encontraron libros".
    - Implementar la lógica del botón "Eliminar" utilizando `fetch` para enviar la petición `DELETE` a la API, incluyendo la confirmación.

- Implementar la lógica del campo y botón de búsqueda utilizando `fetch` para enviar la petición `GET` con el parámetro `search`, actualizando la tabla con los resultados.
- Actualizar la tabla dinámicamente después de una búsqueda o una eliminación exitosa.

PUNTOS DE PARTIDA SUGERIDOS:

Se proporcionan esqueletos de archivos (`backend/index.js`, `backend/package.json`, `frontend/index.html`, `frontend/scripts/script.js`, `frontend/styles/style.css`) con la configuración inicial. Deben completar la lógica restante según los requisitos.

CONSIDERACIONES ADICIONALES:

- La claridad y organización del código serán consideradas.
- El uso correcto de `async/await` en las operaciones que lo requieran es fundamental.
- Asegúrense de que el backend esté corriendo antes de probar el frontend.
- Pueden usar la consola del navegador (F12) y la pestaña "Network" para depurar las peticiones Fetch.

INSTRUCCIONES PARA LA ENTREGA Y EVALUACIÓN:

- Al finalizar la ejercitación, deberán **comprimir** todo el contenido de las carpetas `backend` y `frontend` (incluyendo los subdirectorios `scripts` y `styles`), **EXCLUYENDO EXPRESAMENTE la carpeta `node_modules`** que se encuentra dentro de la carpeta `backend`.
- El archivo comprimido debe ser un archivo ZIP y tener el siguiente formato de nombre: `parcial_3k4_<legajo>.zip`. Deberán **reemplazar `<legajo>`** con su número de legajo.
- El archivo ZIP deberá ser subido a la plataforma **Moodle** de la facultad, en el espacio habilitado para esta evaluación, dentro del tiempo estipulado.
- Una vez entregado el archivo en Moodle y **previo a la evaluación presencial**, deberán **restablecer la carpeta `node_modules`** en el backend restableciendo la carpeta o ejecutando `npm install` desde la terminal dentro de la carpeta `backend` de su proyecto local. Asegúrense de que su `package.json` esté correcto para que este paso funcione.

- Deberán **permanecer en su estación de trabajo** con el proyecto abierto en VS Code y el servidor backend (`npm start` o `npm run dev`) y el frontend (abriendo `frontend/index.html` en el navegador) **funcionando localmente y listo para ser evaluado**. La evaluación se realizará de forma **presencial** por el docente en sus puestos, donde se verificará el funcionamiento del código entregado.

CRITERIOS DE EVALUACIÓN (Rúbrica - 10 puntos en total):

1. **Backend: `GET /api/libros` (Todos):** Implementación correcta en el backend para retornar *todos* los libros y fundamentación oral. (1 punto)
2. **Frontend: Listado Inicial:** Uso de `fetch` para `GET /api/libros` y renderizado inicial de la lista en la tabla al cargar la página y fundamentación oral. (1 punto)
3. **Backend: `GET /api/libros?search=[term]` (Filtrado por Título):** Implementación correcta en el backend para filtrar libros *únicamente* por `Titulo` usando el parámetro `search`. (1 punto)
4. **Frontend: Búsqueda (Fetch y Render):** Uso de `fetch` para `GET /api/libros?search=[term]` y actualización de la tabla con los resultados de la búsqueda y fundamentación oral. (1 punto)
5. **Frontend: Búsqueda (Eventos):** Manejo de eventos en el campo y/o botón de búsqueda para disparar la función de búsqueda y fundamentación oral. (1 punto)
6. **Backend: `DELETE /api/libros/:id`:** Implementación correcta en el backend para eliminar un libro por su ID y fundamentación oral. (1 punto)
7. **Frontend: Eliminación (Fetch):** Uso de `fetch` para enviar la petición `DELETE /api/libros/:id` y fundamentación oral. (1 punto)
8. **Frontend: Eliminación (Eventos y Confirmación):** Manejo de eventos en los botones "Eliminar" (delegación o listeners individuales) y solicitud de confirmación al usuario antes de enviar la petición y fundamentación oral. (1 punto)
9. **Frontend: Manipulación del DOM General:** Correcta limpieza de la tabla, adición de filas dinámicamente, y manejo del mensaje "No se encontraron libros" y fundamentación oral. (1 punto)
10. **Frontend: HTML y Bootstrap:** Estructura HTML base correcta, inclusión de los IDs necesarios y uso básico de clases de Bootstrap para layout y estilos (tabla, formulario de búsqueda, botones) y fundamentación oral. (1 punto)

CONDICIÓN DE APROBACIÓN:

Para aprobar la evaluación, deben alcanzar un mínimo de 6 puntos (60%). Esto se logra si *al menos 2 de las 3 rutas API requeridas (GET todos/filtrado y DELETE)* están implementadas de *extremo a extremo* (es decir, funcionan correctamente en el backend y

su interacción correspondiente con el frontend (Fetch, manejo de respuesta, actualización de la interfaz) también funciona) y responder las preguntas verbales que se realicen en la instancia de evaluación. Es decir que se alcanza el 60% con la implementación funcionando y habiendo fundamentado en la evaluación oral las preguntas que realice el docente.