UTN - Facultad Regional Córdoba (FRC)
Desarrollo de Software (DDS)
1 Parcial 2025

Sistema Básico de Gestión de Muebles

Objetivo: Implementar una aplicación full stack básica para gestionar un listado de muebles, enfocándose en la lectura: listar todos y filtrar por nombre los registros, utilizando el stack tecnológico visto en clase. Se requiere un sistema simple para mantener un registro de muebles. La aplicación debe permitir visualizar un listado completo de muebles, y buscarlos exclusivamente por nombre.

Tecnologías a utilizar: HTML, CSS, Bootstrap, JavaScript, Node.js, SQLite, Sequelize, Express

Herramientas a utilizar: Visual Studio Code

Prerrequisitos:

- Descargar Parcial_1_DDS_3k1.zip y descomprimir en su carpeta de trabajo

REQUISITOS FUNCIONALES:

1. **Listado de Muebles**: Al cargar la página, se debe mostrar un listado de todos los muebles registrados en la base de datos en una tabla. Se deben mostrar los campos: IdMueble, Nombre, Material, Fecha de Fabricación y Precio Estimado.

2. Búsqueda por Nombre: Debe existir un campo de texto que permita filtrar el listado de muebles únicamente por el **Nombre** del mueble. Al escribir y confirmar la búsqueda con un botón "Buscar", la lista debe actualizarse mostrando solo los muebles cuyo **Nombre** 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.

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 `Mueble` con los atributos `IdMueble` (PK, Autonumerico), `Nombre` (STRING), `Material` (STRING), `FechaFabricacion` (DATEONLY), `PrecioEstimado` (DECIMAL).

- 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 a la tabla.

- Deben implementar las siguientes rutas (API RESTful):

  - `GET /api/muebles`: Obtener todos los muebles.

  - `GET /api/muebles?buscar=parteDelNombre`: Obtener muebles filtrados únicamente por el término de búsqueda en el `Nombre`.

- 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. La tabla debe mostrar las columnas: IdMueble, Nombre, Material, Fecha Fabricación, Precio Estimado y Acciones.

- 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 muebles en la tabla, y manejar mensajes de carga o error (opcional, pero valorado), así como el mensaje de "No se encontraron muebles".

- 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.

• El formato de la fecha debe ser legible (ej: DD/MM/YYYY) y el precio debe mostrarse con dos decimales y el símbolo de moneda si es posible (ej: $120.50). (Esto último es un extra valorado para el manejo de datos en JS).

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_1_DDS_3k1_<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 y se les hará preguntas que comprueben el conocimiento del funcionamiento del mismo.

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

1. **Backend: GET `/api/muebles` (Todos)**: Implementación correcta en el backend para retornar todos los muebles y fundamentación oral. (2 puntos)

2. **Frontend: Listado Inicial**: Uso de `fetch` para `GET /api/muebles` y renderizado inicial de la lista en la tabla (incluyendo Fecha Fabricación y Precio Estimado con formato) al cargar la página y fundamentación oral. (2 puntos)

3. **Backend: GET `/api/muebles?buscar=parteDelNombre`** (Filtrado por Nombre): Implementación correcta en el backend para filtrar muebles únicamente por `Nombre` usando el parámetro `search`. (2 puntos)

4. **Frontend: Búsqueda (Fetch y Render)**: Uso de `fetch` para `GET /api/muebles?buscar=parteDelNombre` y actualización de la tabla con los resultados de la búsqueda y fundamentación oral. (2 puntos)

5. **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 con nuevas columnas, formulario de búsqueda, botones) y fundamentación oral. (2 puntos)

CONDICIÓN DE APROBACIÓN:

Para aprobar la evaluación, deben alcanzar un mínimo de **6 puntos (60%)**.

Es decir que se alcanza el 60% con la implementación funcionando y habiendo fundamentado en la evaluación oral las preguntas que realice el docente.