UTN - Facultad Regional Córdoba (FRC)

Desarrollo de Software (DDS)

Docente: Ing. Alejandro Rey Corvalán

EJERCITACIÓN TIPO PARCIAL: Sistema de Gestión de Pacientes de Veterinaria

Objetivo: Desarrollar una aplicación full stack para gestionar pacientes de una veterinaria, implementando operaciones CRUD básicas (Crear, Leer, Actualizar, Eliminar) sobre los datos de los pacientes, utilizando la misma stack tecnológica vista en clase (Node.js, Express, Sequelize, SQLite, HTML, CSS, Bootstrap, JavaScript, Fetch).

Tecnologías a Utilizar:

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

Contexto: Se requiere un sistema básico para llevar un registro de los pacientes (mascotas) que atiende una veterinaria. La aplicación debe permitir listar todos los pacientes, buscar pacientes por el nombre de su propietario, eliminar un paciente existente y modificar los datos de un paciente.

Prerrequisitos:

- Tener Node.js y npm instalados.
- Tener Visual Studio Code instalado.
- Tener una conexión a internet.

REQUISITOS FUNCIONALES:

1. **Listado de Pacientes:** Al cargar la página, se debe mostrar un listado de todos los pacientes registrados en la base de datos.
2. **Búsqueda por Propietario:** Debe existir un campo de texto que permita filtrar el listado de pacientes. Al escribir y/o confirmar la búsqueda, la lista debe actualizarse mostrando solo los pacientes cuyo nombre de propietario coincida (total o parcialmente) con el texto ingresado.

3. **Eliminación de Paciente:** Cada paciente en el listado debe tener un botón "Eliminar". Al hacer clic en este botón, se debe eliminar el paciente correspondiente de la base de datos y actualizar el listado en la interfaz. Se recomienda solicitar confirmación antes de eliminar.
4. **Edición de Paciente:** Cada paciente en el listado debe tener un botón "Modificar" o "Editar". Al hacer clic en este botón, los datos del paciente seleccionado deben cargarse en un formulario en la misma página o en una modal.
5. **Actualización de Datos:** Una vez que los datos de un paciente están cargados en el formulario de edición, el usuario debe poder modificar los campos (`NombreMascota`, `Propietario`, `Telefono`). Al guardar los cambios, estos deben persistirse en la base de datos a través de la API del backend, y el listado de pacientes debe actualizarse.
6. **(Bonus/Extensión): Creación de Nuevo Paciente:** Aunque el enunciado se centra en modificar/eliminar, sería una extensión natural tener también un botón para "Agregar Paciente" que utilice el mismo formulario de edición/creación para dar de alta un nuevo paciente.

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.
    - Utilizar el modelo llamado `Paciente` con los siguientes atributos:
        - `IdPaciente`: INTEGER, clave primaria, auto-incremental, no nulo.
        - `NombreMascota`: STRING, no nulo.
        - `Propietario`: STRING, no nulo.
        - `Telefono`: STRING (o INTEGER, decidir según se necesite guardar con formatos o solo números), permitir nulo (opcional, si no siempre se tiene teléfono).
    - Verificar la configuración de Sequelize para sincronizar el modelo y, al iniciar el servidor (en desarrollo), precargar algunos datos iniciales de ejemplo (seeding) si la tabla está vacía. Usar `force: true` en `sync` durante el desarrollo para facilitar las pruebas (¡Cuidado en producción!).
    - Implementar las siguientes rutas (API RESTful):
        - `GET /api/pacientes`: Obtener todos los pacientes.
        - `GET /api/pacientes?propietario=[nombre]`: Obtener pacientes filtrados por el nombre del propietario.
        - `GET /api/pacientes/:id`: Obtener un paciente específico por su ID.
        - `PUT /api/pacientes/:id`: Actualizar un paciente específico por su ID. Recibe los datos actualizados en el cuerpo de la petición (JSON).
        - `DELETE /api/pacientes/:id`: Eliminar un paciente específico por su ID.

        - `(Bonus/Extensión) POST /api/pacientes`: Crear un nuevo paciente. Recibe los datos del nuevo paciente en el cuerpo de la petición (JSON).
    - Configurar `nodemon` para reiniciar el servidor automáticamente durante el desarrollo.
3. **Frontend:**
    - Utilizar HTML para la estructura básica de la página.
    - Utilizar Bootstrap 5 (CDN) para el diseño y layout (grilla para el listado, estilos para botones, formulario, etc.). No es necesario usar componentes interactivos de Bootstrap que requieran su JS (como modales), pueden implementar la lógica de mostrar/ocultar el formulario con CSS y JavaScript si lo desean, o usar el bundle completo de Bootstrap.
    - Utilizar JavaScript puro (ES6+) y la Fetch API para interactuar con el backend.
    - Referenciar los elementos HTML necesarios (contenedor del listado, campos del formulario, botones, campo de búsqueda) utilizando sus **IDs** (`getElementById`, etc.).
    - Manejar dinámicamente el DOM para mostrar el listado de pacientes, el formulario de edición/creación, mensajes de carga o error.
    - Implementar la lógica de los botones "Eliminar", "Modificar" y "Guardar" (del formulario), utilizando las rutas de la API correspondientes con `fetch`.
    - Validar (básicamente) en el frontend que los campos obligatorios no estén vacíos antes de enviar al backend.

PUNTOS DE PARTIDA SUGERIDOS:

Se proporciona una estructura base para el `index.js` del backend y el `index.html`/`script.js` del frontend. Deben completar la lógica necesaria para cumplir con 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.