Proyecto Práctico: Sistema Web CRUD con Flask y SQLite3

1. Descripción General

El objetivo de este proyecto es diseñar, desarrollar e implementar una aplicación web funcional utilizando el micro-framework **Flask** (Python) para el backend y una base de datos relacional local administrada con **SQLite3**. La interfaz de usuario debe ser interactiva, adaptativa y construida utilizando estándares modernos de **HTML5, CSS3 y JavaScript (JS)**. El sistema debe resolver un problema de gestión de información mediante la implementación completa de un ciclo **CRUD** (Crear, Leer, Actualizar y Eliminar registros).

2. Requisitos Técnicos y Arquitectura

El proyecto debe seguir una estructura de archivos limpia y organizada. Se sugiere la siguiente arquitectura de directorios:

```
mi_proyecto/
│
├── app.py                # Servidor principal de Flask (Rutas y lógica de negocio)
├── database.db           # Archivo de la base de datos SQLite3
├── schema.sql            # Script SQL para la creación de tablas
│
├── static/               # Archivos estáticos
│   ├── css/
│   │   └── estilos.css   # Hojas de estilo personalizadas
│   └── js/
│       └── main.js       # Lógica de interactividad y peticiones AJAX/Fetch
│
└── templates/            # Plantillas HTML procesadas por Jinja2
    ├── base.html         # Plantilla base (Estructura global/Navbar)
    ├── index.html        # Vista principal (Lectura y visualización de datos)
    ├── crear.html        # Formulario de inserción
    └── editar.html       # Formulario de modificación
```

A. Backend (Python + Flask + SQLite3)

- **Servidor Web:** Configurar un servidor activo con Flask que gestione las peticiones HTTP (GET, POST).
- **Control de Rutas:** Definir rutas limpias para la navegación y el procesamiento de datos (por ejemplo: /, /crear, /editar/<id>, /eliminar/<id>).
- **Gestión de Base de Datos:**
  - Uso directo de la librería nativa sqlite3 de Python.

- Implementación de consultas parametrizadas (para prevenir vulnerabilidades de Inyección SQL).
- Manejo adecuado de conexiones (abrir y cerrar la conexión en cada petición).

B. Frontend (HTML5, CSS3 y JavaScript)

- **HTML5:** Estructura semántica, formularios validados en el lado del cliente (atributos required, tipos de datos correctos, etc.).
- **CSS3:** Diseño responsivo (adaptable a dispositivos móviles y escritorio), interfaz de usuario limpia e intuitiva.
- **JavaScript (Vanilla JS):**
  - Manipulación dinámica del DOM.
  - Confirmaciones de seguridad antes de realizar acciones críticas (como eliminar un registro).
  - *Opcional/Recomendado:* Uso de la API fetch para consumir rutas de Flask de forma asíncrona sin necesidad de recargar toda la página.

2. Requisitos Funcionales (El CRUD)

La aplicación debe permitir gestionar una entidad del mundo real (por ejemplo: *Proyectos Comunitarios, Inventario de Equipos, Registro de Estudiantes, o Tareas*). Se deben garantizar las siguientes operaciones:

1. **Create (Crear):** Un formulario que recopile la información de la entidad, valide los campos requeridos y guarde de manera persistente el nuevo registro en la base de datos.
2. **Read (Leer / Visualizar):** Una vista principal donde se listen todos los registros existentes en una tabla o panel de tarjetas (cards). Debe incluir un mensaje amigable si la base de datos está vacía.
3. **Update (Actualizar / Editar):** Capacidad de seleccionar un registro específico, cargar sus datos actuales en un formulario de edición, modificarlos y guardar los cambios.
4. **Delete (Eliminar):** Un botón o acción que permita remover un registro de la base de datos. **Requisito obligatorio:** Mostrar un cuadro de diálogo de confirmación mediante JavaScript antes de proceder con la eliminación.

5. Entregables del Estudiante

El proyecto debe ser entregado en un archivo comprimido (.zip o .tar.gz) o mediante un enlace a un repositorio de control de versiones (GitHub/GitLab) que contenga:

1. **Código Fuente Completo:** Estructurado según la arquitectura especificada.
2. **Script de Base de Datos (schema.sql):** Archivo con las sentencias SQL necesarias para crear las tablas e insertar al menos 3 registros de prueba.
3. **Archivo README.md:** Un documento de texto explicativo que contenga:
   - Instrucciones paso a paso para instalar las dependencias (ej. pip install flask).
   - Instrucciones para inicializar la base de datos y ejecutar el servidor local.
   - Breve descripción del caso de estudio o problema que resuelve la aplicación web.

4. **Criterios de Evaluación (Rúbrica Sugerida)**

Criterio	Descripción	Puntaje / Porcentaje
Funcionamiento del CRUD	Las operaciones de creación, lectura, actualización y eliminación funcionan correctamente sin generar errores de servidor (Errores 500).	35%
Modelado y Conexión de BD	Estructura lógica de la tabla en SQLite3, uso de tipos de datos adecuados y manejo de conexiones seguras (consultas parametrizadas).	20%
Interfaz y UX (CSS/HTML)	Diseño visual ordenado, uso de HTML semántico, legibilidad de las fuentes y adaptabilidad móvil (Responsive Design).	15%
Interactividad (JavaScript)	Validaciones del lado del cliente, ventanas de confirmación para eliminar y manipulación	15%

	interactiva del DOM.	
Estructura y Buenas Prácticas	Código fuente organizado, modular, limpio y documentado con comentarios pertinentes en Python, JS y CSS.	15%