Informe del Proyecto: Sistema de Gestión de Productos

Estudiante: Elias Efrain Manchego Navarro

Curso:ngeniería de Software I

Universidad:Universidad La Salle

Repositorio:
https://github.com/ELMANCHE/practica_final_ingsoft1

1. Introducción

El presente informe detalla el desarrollo de un sistema de gestión de productos, diseñado como proyecto final del curso de Ingeniería de Software I. El objetivo fue construir una aplicación funcional y mantenible aplicando los principios de arquitectura por capas y tecnologías web modernas. La solución permite gestionar productos de forma eficiente mediante operaciones CRUD (Crear, Leer, Actualizar y Eliminar).

2. Explicación del Proyecto

2.1 Arquitectura por Capas

El proyecto está dividido en tres capas bien definidas:

Capa de Presentación (Frontend)

HTML: Define la estructura visual de la aplicación.

CSS (style.css): Se utilizó diseño neumórfico para lograr una estética moderna con sombras suaves y efectos 3D, y se adaptó a distintos tamaños de pantalla (responsive).

JavaScript (script.js): Maneja la interactividad mediante la API Fetch para operaciones asincrónicas. También se encarga de la validación de formularios y actualización dinámica del DOM.

Capa de Lógica de Negocio (backend.py)

Desarrollo de una API REST usando Flask.

Manejo de rutas HTTP para gestionar operaciones CRUD.

Validaciones y lógica empresarial.

Capa de Datos

Base de datos SQLite3.

Definición de la tabla productos.

Ejecución de sentencias SQL para insertar, actualizar, eliminar y consultar datos.

Gestión de la conexión a la base de datos y manejo de errores.

2.2 Funcionalidades Principales

Crear Productos
Validación de campos, inserción en la base de datos y actualización en tiempo real.

Leer Productos
Visualización dinámica y ordenada en tabla responsiva.

Actualizar Productos
Edición inline con validación previa a la persistencia.

Eliminar Productos
Confirmación previa a eliminación con actualización visual.

2.3 Aspectos Técnicos

Lenguajes y tecnologías: HTML5, CSS3, JavaScript, Python, SQLite3

Frameworks/librerías: Flask, Fetch API

Estilo visual: Neumorfismo, diseño responsive

Organización del código: Separación clara entre frontend, backend y datos

Frontend

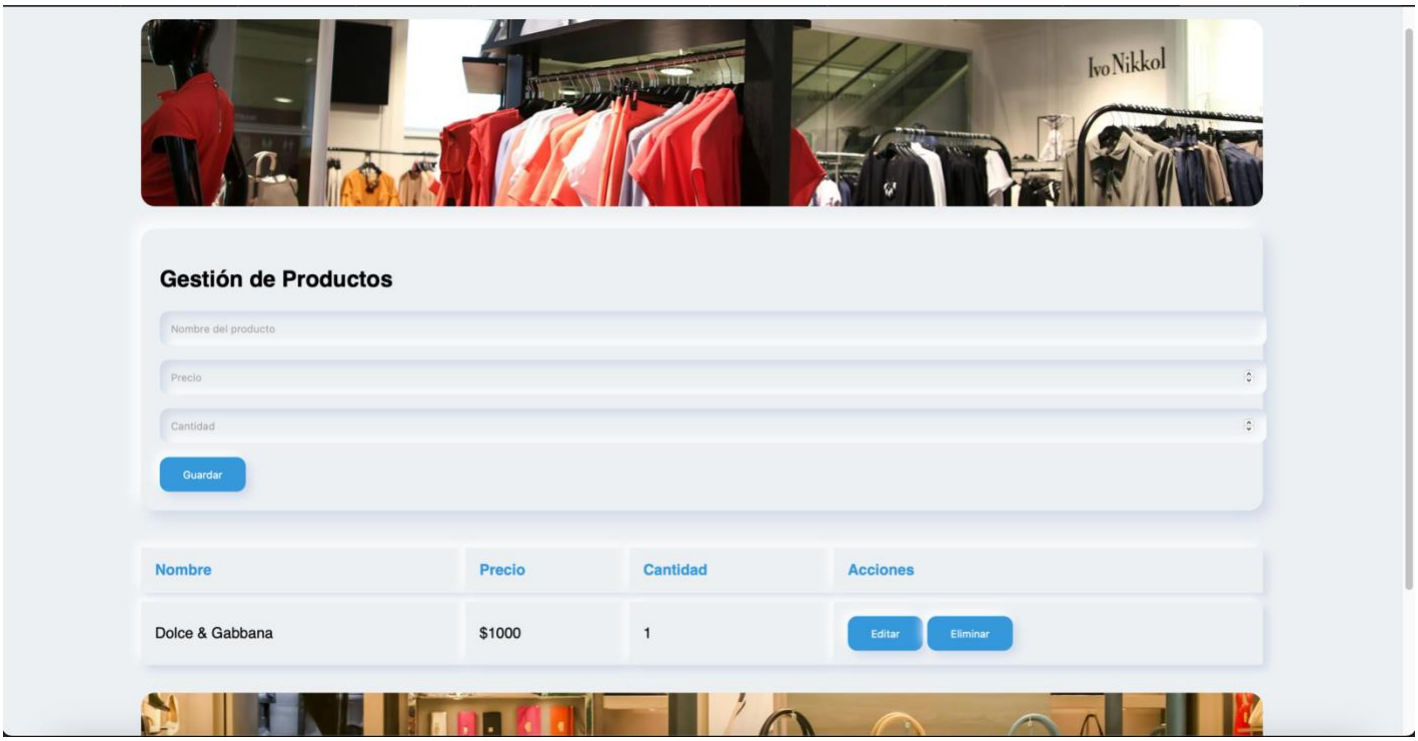
```
async function editarProducto(id) {
    const response = await fetch(`/productos?id=${id}`);
    const productos = await response.json();
    const producto = productos.find(p => p.id === id);

    document.getElementById('productoId').value = producto.id;
    document.getElementById('nombre').value = producto.nombre;
    document.getElementById('precio').value = producto.precio;
    document.getElementById('cantidad').value = producto.cantidad;
}
```

Backend

```
# Ejemplo de endpoint Flask
@app.route('/productos/<int:id>', methods=['PUT'])
def actualizar_producto(id):
    datos = request.get_json()
    # Lógica de actualización
    return jsonify({'mensaje': 'Producto actualizado
```

3. Mockup Visual



4. Conclusiones

Logros Alcanzados

- Implementación exitosa de arquitectura por capas
- Interfaz de usuario moderna y responsiva
- Sistema CRUD funcional y eficiente

Mejoras Futuras

- Implementación de autenticación
- Reportes y estadísticas

- Gestión de categorías

- Mejoras en la interfaz de usuario

5. Referencias

- Flask. (n.d.). *Welcome to Flask — Flask Documentation (2.3.x)*. Flask Documentation. https://flask.palletsprojects.com/en/2.3.x/

- Mozilla Developer Network. (n.d.). *MDN Web Docs*. Mozilla. https://developer.mozilla.org/

- SQLite. (n.d.). *SQLite Documentation*. SQLite.org. https://www.sqlite.org/docs.html