Proyecto: Tienda de Ropa Online
Fecha: 2 de julio de 2025
Equipo de Trabajo: Diego Alejandro Parra Diaz, Valentina Parra Ordóñez, Esteban Villalba y Jonathan Felipe Ochoa Silva

Introducción

El presente documento describe la arquitectura lógica y física del sistema desarrollado como parte del proyecto de la asignatura, una tienda en línea funcional orientada al modelo cliente-servidor. Esta arquitectura fue diseñada para garantizar modularidad, facilidad de mantenimiento y claridad en las responsabilidades de cada componente.

El proyecto fue desarrollado en Python y HTML/CSS puro, con separación entre frontend y backend, siguiendo una estructura MVC simplificada.

Componentes Principales del Sistema

1. Frontend (Interfaz de Usuario)

- Implementado en HTML, CSS y recursos estáticos (imágenes).
- Páginas principales: index.html, login.html, register.html, carrito.html.
- Estilos definidos en style.css.
- Accede al backend mediante formularios HTML estándar (sin JavaScript dinámico).

2. Backend (Lógica de Negocio y Controlador)

- Lenguaje: Python
- Framework: Flask (o base mínima de Python para servidor web)
- Archivos principales:
    - app.py: controlador principal que recibe las peticiones HTTP y direcciona al service correspondiente.
    - db.py: establece la conexión con la base de datos.
    - CRUD/: contiene los scripts para manejar la lógica de Crear, Leer, Actualizar y Eliminar información de usuarios, carrito y pedidos.
    - services/: encapsula la lógica de negocio por separado del acceso a datos, haciendo el sistema más limpio y escalable.

3. Base de Datos

- Se asume el uso de SQLite o base de datos simulada en memoria.
- Las tablas clave son: usuarios, productos, carrito, pedidos.
- La conexión está centralizada en db.py, garantizando un punto único de acceso y consistencia.

Diagrama de Arquitectura Lógica (describir o ilustrar)

[Cliente (navegador web)]
↓ HTML request
[index.html, login.html, etc.]
↓
[app.py]
↓
[services/usuario_service.py]
↓
[CRUD/user.py]
↓
[db.py] → [Base de Datos (SQLite)]

Este diagrama ilustra el flujo desde la solicitud del cliente hasta la persistencia en la base de datos.

Flujo General del Sistema

1. El usuario accede al index.html y puede navegar los productos.
2. Si no está registrado, puede ir a register.html y enviar un formulario con sus datos.
3. Esta información llega a app.py, que la valida y envía al user_service.
4. Luego, user_service se comunica con user.py (CRUD) para guardar al usuario.
5. El mismo flujo se repite para agregar productos al carrito, generar pedidos y consultar información.
6. Todo está diseñado con una arquitectura simple pero separada en capas para simular el enfoque de microservicios a pequeña escala.

Seguridad y Validaciones

- Validaciones básicas están presentes en el lado del backend (verificación de datos).
- A falta de JS o tokens, el sistema funciona como un prototipo de flujo funcional (MVP).

Entorno de Desarrollo

- Sistema operativo: Windows/Linux (equipo mixto)
- Editor de código: Visual Studio Code
- Gestor de versiones: GitHub
- Motor de base de datos: SQLite (por simplicidad y portabilidad)
- Reuniones de planeación y revisión: Google Meet y WhatsApp

Conclusión

La arquitectura del sistema refleja un enfoque modular, escalable y comprensible para un proyecto universitario. Se priorizó el correcto flujo entre cliente, servidor y base de datos, dividiendo responsabilidades entre vistas, lógica y persistencia. Este diseño permitió una colaboración efectiva entre los integrantes y facilitó la localización de errores durante las fases de integración.

Para futuros desarrollos, se podría migrar a una arquitectura con API REST, validaciones más robustas y una interfaz más interactiva usando JavaScript o frameworks modernos.