Informe de Análisis de Aplicación Web: Tienda de Milanesas

1. Objetivo General y Propósito de la Aplicación

El objetivo principal de esta aplicación web es funcionar como un **Sistema de Comercio Electrónico (E-commerce) simplificado** para la venta de productos alimenticios, específicamente milanesas, con la opción de personalización mediante la adición de una bebida (refresco).

Propósito:

1. **Mostrar un catálogo** atractivo y navegable de milanesas.
2. Permitir a los usuarios **personalizar y agregar productos** a un carrito de compras.
3. Gestionar el carrito, permitiendo la **visualización, eliminación y el cálculo** del total de la compra.
4. Simular un **proceso de pago** y confirmación de la orden.
5. Ofrecer una **experiencia de usuario moderna** mediante el soporte de modo claro y oscuro (Dark Mode).

2. Público General

Característica	Descripción
Tipo de Usuario	Consumidores de comida para llevar o entrega a domicilio.
Edad	Amplio, desde jóvenes adultos (18+) hasta personas mayores, que utilizan servicios de comida online.
Conocimiento Digital	Intermedio/Básico. La interfaz es intuitiva y sigue patrones de E-commerce conocidos.
Ubicación	Potencialmente local o regional, dependiendo del servicio de entrega.
Intereses	Búsqueda de un proceso de pedido de comida rápido, sencillo y visualmente agradable.

3. Estructura de la Aplicación (General)

La aplicación sigue una arquitectura **Cliente-Servidor (Full Stack)** utilizando Flask (Python) como *backend* y una combinación de HTML, CSS y JavaScript como *frontend*.

Categoría	Archivos	Función
Backend (Python)	app.py	Lógica del servidor: manejo de rutas, sesión, carrito, lógica de negocio (productos/precios) y simulación de API.
Frontend (HTML)	index.html, carrito.html	Estructura de las dos páginas principales. Utiliza el motor de plantillas Jinja2 (Flask).
Frontend (CSS)	style.css, index.css, carrito.css	Estilos y presentación. style.css es la base, y los otros dos son específicos de cada página.
Frontend (JS)	index.js, carrito.js	Comportamiento dinámico: interactividad de productos, gestión de tema (Dark Mode), manejo de carrito y simulación de pago.

4. Análisis Detallado de Funcionalidad por Archivo

A. Archivos de Backend (Lógica del Servidor)

Archivo	Funcionalidad Principal	Detalles Clave
app.py	**Núcleo de la Aplicación Flask.** Define la lógica de enrutamiento y la gestión de la sesión del usuario.	- Define los **productos (milanesas)** y sus precios base. - Define los **precios de los refrescos**. - **Rutas**

		principales: / (tienda), /carrito (ver carrito). - **Rutas de acción:** /agregar/<id> (añade ítem, calcula precio con refresco), /vaciar, /eliminar/<item_key>, /api/procesar_pago (simulación de transacción). - Utiliza la **sesión de Flask** para almacenar el estado del carrito (session["carrito"]).

B. Archivos de Frontend (Estructura y Presentación)

Archivo	Funcionalidad Principal	Detalles Clave
index.html	**Página principal (Catálogo).** Muestra la lista de productos disponibles.	- Usa Jinja2 para iterar sobre la lista de productos y generar las tarjetas (producto-card). - Incluye el botón de **cambio de tema** (theme-toggle) y el **contador del carrito** (cart-count). - Cada tarjeta incluye opciones para elegir el refresco antes de agregar.
carrito.html	**Página del Carrito de Compras.** Muestra el resumen del pedido.	- Muestra la lista detallada de ítems en el carrito (incluyendo refresco). - Permite **eliminar ítems** y **vaciar el carrito**. - Muestra el **total** a pagar. - Incluye el **Modal de Pago** y un **Diálogo de Compra Realizada** flotante para una experiencia de éxito más moderna.

style.css	**Estilos Globales y Variables.** Define el look & feel de toda la aplicación.	- Establece las **variables CSS modernas** (:root) para el modo claro y oscuro. - Configura la fuente (**Inter**), el fondo, el estilo del encabezado y la utilería de botones genéricos (.btn-accion). - **Base fundamental** para la estética moderna y responsiva de la tienda.
index.css	**Estilos Específicos del Catálogo.** Define la presentación de las tarjetas de producto.	- Configura el layout de la cuadrícula (grid) de productos para ser responsivo (auto-fit). - Estiliza las tarjetas (producto-card), imágenes y los **botones de selección de refresco** (incluyendo tooltips). - Define el aspecto del botón **"Agregar al Carrito"** (.btn-agregar).
carrito.css	**Estilos Específicos del Carrito.** Define la presentación de la lista de ítems y el modal de pago.	- Estiliza los ítems del carrito (.item-detalle), el resumen total (.carrito-total) y el **Modal de Pago**. - Incluye estilos para mensajes de error/éxito (en el modal) y para el **diálogo flotante de éxito** (.floating-success-dialog).

C. Archivos de Frontend (Comportamiento Dinámico)

Archivo	Funcionalidad Principal	Detalles Clave
index.js	**Lógica de la Tienda**	- **Gestión de Tema:**

	(Página principal). Gestiona las interacciones de los productos.	Persiste la selección del modo claro/oscuro en localStorage. - **Lógica de Refrescos:** Maneja la selección de refrescos por tarjeta, actualizando el precio total en el botón "Agregar al Carrito" y la propiedad data-refresco-id del botón. - **Envío Asíncrono:** Utiliza fetch para agregar el producto al carrito sin recargar la página (ruta /agregar/<id>) y actualiza el contador del carrito.
carrito.js	**Lógica del Carrito y Pago.** Gestiona el comportamiento de la página carrito.html.	- **Gestión de Tema:** Igual que index.js, asegura la consistencia del tema. - **Modal de Pago:** Muestra y oculta el modal de pago al hacer clic en "Finalizar Compra". - **Proceso de Pago:** Escucha la sumisión del formulario de pago. Realiza una llamada asíncrona (fetch) a la ruta /api/procesar_pago. - **Respuesta de Pago:** En caso de éxito, muestra el **Diálogo Flotante de Compra Realizada** y redirige a la página principal tras un breve retraso. En caso de fallo, muestra el error en el modal.