INSTITUTO POLITÉCTICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA Y CIENCIAS SOCIALES Y ADMINISTRATIVAS

6NM60 Ingeniería de Pruebas

Proyecto, desarrollo de una página de e-commerce.

REPORTE

Alumnos:

García Méndez Juan Carlos

Conde Basilio Leonardo

Ramos Velázquez Felipe

Villaseñor Trejo Javier Enrique

Docente:

Cruz Martínez Ramón

Fecha de entrega: 09 de mayo del 2025

Contenido

Misión

Desarrollar una plataforma de e-commerce intuitiva y funcional que integre un sistema de gestión de carrito de compras (CRUD), simulación de cobro seguro y experiencia personalizada para usuarios registrados e invitados. Nuestra misión es ofrecer un sitio web robusto que permita explorar productos con filtros de búsqueda eficientes, visualizar detalles clave (clave, nombre, precio, descripción e imágenes) y garantizar una navegación fluida, incluso en modo invitado (sin permisos de compra). Nos comprometemos a facilitar la gestión de pedidos mediante historiales actualizados, generación automatizada de tickets de compra y envío de estos por correo, priorizando la seguridad, la escalabilidad y la satisfacción de un mercado segmentado estratégicamente.

Visión

Posicionarnos como una solución de e-commerce de referencia en nuestro nicho de mercado, destacando por la integración de funcionalidades esenciales y una experiencia de usuario diferenciadora. Aspiramos a evolucionar continuamente, incorporando tecnologías innovadoras que mejoren la simulación de transacciones, la personalización de búsquedas y la gestión de datos en tiempo real. Buscamos ser reconocidos por nuestra capacidad para combinar simplicidad operativa con herramientas avanzadas (como historiales de pedidos y tickets automatizados), consolidando un modelo escalable que inspire confianza en clientes y adaptarse a las demandas emergentes del comercio electrónico

Marco Teórico

Antecedentes

El comercio electrónico ha transformado la industria retail desde finales de los años 1990, evolucionando desde catálogos básicos hasta plataformas complejas con funcionalidades como carritos dinámicos y pasarelas de pago integradas. Sin embargo, muchas soluciones actuales carecen de flexibilidad para usuarios no registrados o no priorizan la simulación de cobros seguros, limitando la experiencia de clientes en etapas exploratorias. Además, la gestión eficiente de historiales de pedidos y la generación automatizada de tickets siguen siendo desafíos técnicos recurrentes en proyectos emergentes.

Funcionalidades y Base Técnica

- **CRUD del carrito**:
    - Implementación de operaciones *Create* (agregar productos como "Abrigo Clásico", S1000), *Read* (visualizar items en el carrito), *Update* (modificar cantidades) y *Delete* (eliminar productos) mediante lógica de estado en el frontend (ej: React, Vue) y persistencia en bases de datos (ej: MySQL, Firebase).
    - Manejo de eventos (ej: clic en *"AGREGAR"*) para actualizar el carrito en tiempo real.
- **Simulación de cobro**:
    - Integración de APIs de pago ficticias (ej: Stripe en modo sandbox) para validar tarjetas, montos y generar transacciones simuladas sin riesgos financieros.
- **Modo invitado**:
    - Restricción de permisos mediante autenticación (ej: JWT): usuarios no registrados pueden navegar en secciones como *"Abrigos"* o usar el buscador, pero al intentar finalizar la compra, se redirigen a *"Registrarse"*.
- **Filtro de búsqueda**:
    - Consultas simples usando parámetros como *nombre* o *categoría* (ej: "Abrigos") mediante operadores LIKE en SQL o métodos de filtrado en arrays (JavaScript).

    - Posible escalabilidad a consultas avanzadas (precio, tallas) con Elasticsearch o Algolia.

- **Atributos de producto**:

    - Estructura de datos con campos obligatorios: clave única (ej: S1000), nombre, descripción detallada, precio e imagen (almacenada en servicios como AWS S3 o Cloudinary).

- **Historial de pedidos**:

    - Almacenamiento en bases de datos relacionales, vinculando usuarios registrados con sus compras (tablas *users* y *orders*). Visualización en la sección *"Mis pedidos"* con detalles como fecha, productos y total.

- **Generación y envío de tickets**:

    - Creación de PDFs automatizados con bibliotecas como **PDFKit** (Node.js) o **ReportLab** (Python), incluyendo logos de *ClothesShop*, datos del cliente y resumen de compra.

    - Integración de servicios de correo (ej: SendGrid, Nodemailer) para enviar tickets al email registrado, usando plantillas HTML personalizadas.

Relevancia

La implementación de un CRUD para el carrito y la simulación de cobro responde a la necesidad de ofrecer una experiencia realista sin exponer datos sensibles, crucial para ganar confianza en etapas iniciales del negocio. Además, el modo invitado reduce la fricción en la navegación, mientras que el historial de pedidos y los tickets automatizados optimizan la gestión postventa, un diferenciador clave en un mercado altamente competitivo como la moda. La estructura técnica planteada asegura escalabilidad para incorporar futuras funcionalidades (ej: tallas personalizadas, recomendaciones basadas en IA), alineándose con la visión de posicionar *ClothesShop* como referente en su nicho.

Planteamiento del problema:

Las plataformas de e-commerce en el sector de la moda enfrentan desafíos recurrentes que limitan su eficiencia y experiencia de usuario:

- **Fragmentación en la experiencia de compra**:
  • Los carritos de compra estáticos no permiten modificar cantidades o eliminar productos fácilmente, generando frustración.
  • Los usuarios invitados pueden explorar productos (ej: "Abrigos"), pero no finalizar compras sin registrarse, aumentando la tasa de abandono.

- **Falta de simulaciones realistas**:
  • Muchas plataformas no integran pasarelas de pago simuladas, lo que dificulta la validación segura de transacciones en etapas tempranas del negocio.

- **Gestión ineficiente de procesos postventa**:
  • Historiales de pedidos no accesibles o tickets de compra manuales, retrasando la confirmación de compras y generando desconfianza.
  • Búsquedas básicas que no filtran por atributos clave (ej: precio, categoría como *"Camisetas"*), dificultando la exploración intuitiva.

- **Dependencia excesiva de la autenticación**:
  • Interfaces que obligan a registrarse antes de interactuar con funcionalidades esenciales (ej: agregar al carrito), alienando a clientes en fase exploratoria.

Esta falta de integración entre funcionalidades críticas (como el CRUD del carrito, la simulación de cobro y el acceso para invitados) genera una experiencia discontinua para usuarios finales, especialmente en un mercado tan competitivo como la venta de ropa. Por ejemplo, clientes interesados en productos como el *"Abrigo Clásico"* (S1000) pueden abandonar el sitio al no poder ajustar su carrito o visualizar un resumen de compra realista. Además, la ausencia de tickets automatizados y el manejo manual de pedidos incrementan errores operativos, afectando la escalabilidad del negocio.

ClothesShop surge para resolver estas brechas, priorizando una navegación fluida, transacciones simuladas confiables y herramientas de autogestión (historial, tickets), asegurando que tanto usuarios registrados como invitados encuentren valor en cada interacción, desde la búsqueda hasta la postcompra.

Objetivo:

Desarrollar una plataforma de e-commerce para *ClothesShop* que:

- Implemente un **CRUD del carrito de compras** para gestionar productos (ej: agregar/eliminar "Abrigo Clásico", S1000) con actualización en tiempo real.
- Simule cobros seguros mediante APIs ficticias, replicando el flujo de una transacción real sin riesgo financiero.
- Permita a usuarios invitados explorar productos (en categorías como *"Abrigos"* o mediante el buscador) sin completar compras, incentivando el registro.
- Defina un **nicho de mercado claro** (ej: ropa casual premium) y estructure atributos clave de productos (clave, nombre, precio, imágenes en alta resolución).
- Integre un **filtro de búsqueda eficiente** basado en categorías, precios o palabras clave (ej: "Abrigo" en el campo *"Buscar productos..."*).
- Genere y envíe **tickets de compra automatizados** por correo, con detalles como productos adquiridos y total pagado.
- Ofrezca un **historial de pedidos accesible** (en *"Mis pedidos"*) para usuarios registrados, facilitando el seguimiento postventa.
- Garantice una **experiencia intuitiva** mediante una interfaz alineada con estándares de e-commerce (ej: carrito visible, navegación por categorías).

Hipótesis:

Si se implementa un carrito de compras dinámico (CRUD), una simulación de cobro realista y un modo invitado que permite explorar productos sin registro, entonces los usuarios de *ClothesShop* experimentarán una **reducción en la tasa de abandono del carrito**, un **aumento en las conversiones** (al facilitar el registro post-exploración) y una **mejora en la satisfacción general**, gracias a la transparencia en transacciones (tickets automatizados) y la autonomía para gestionar su proceso de compra (historial de pedidos). Además, la segmentación estratégica del mercado y los filtros de búsqueda eficaces posicionarán la plataforma como una solución especializada, atrayendo a un público objetivo definido y fomentando la lealtad a largo plazo.

Justificación:

Técnica:

• La implementación del **CRUD del carrito** garantiza una gestión dinámica de productos (ej: agregar/eliminar el *"Abrigo Clásico"*, S1000), optimizando la interacción del usuario y reduciendo errores en el proceso de compra.
• La **simulación de cobro** mediante APIs ficticias (ej: Stripe sandbox) asegura un entorno seguro para validar transacciones, replicando flujos reales sin exponer datos financieros sensibles.
• El uso de **bases de datos relacionales** (MySQL, Firebase) y servicios de almacenamiento (AWS S3) garantiza escalabilidad y precisión en la gestión de productos, pedidos y usuarios.

Operativa:

• Los **filtros de búsqueda** (por categoría, precio o palabra clave) agilizan la navegación, mejorando la eficiencia en la toma de decisiones de compra.
• La **generación automatizada de tickets** (mediante PDFKit) y su envío por correo electrónico (con SendGrid) eliminan procesos manuales, minimizando errores y acelerando la confirmación de compras.

Social/Económica:

• La plataforma democratiza el acceso a la moda, ofreciendo una experiencia de compra en línea intuitiva, incluso para usuarios con menor familiaridad tecnológica.
• Al segmentar un **nicho de mercado específico** (ej: ropa casual premium), *ClothesShop* atiende demandas insatisfechas, fomentando la especialización y reduciendo la saturación del mercado generalista.
• El **historial de pedidos** y la transparencia en transacciones (tickets detallados) fortalecen la confianza del cliente, un factor crítico en el crecimiento de marcas emergentes.

Calidad:

• La **documentación exhaustiva** (casos de uso, diagramas de flujo, manuales técnicos) asegura la mantenibilidad del código y facilita futuras actualizaciones, alineándose con estándares de ingeniería de software.
• Las **pruebas rigurosas** (unitarias, de integración y UX) validan la robustez del sistema, desde la estabilidad del carrito hasta la coherencia en el envío de correos.

ClothesShop no solo resuelve problemas técnicos y operativos, sino que también se posiciona como una solución socialmente relevante, al combinar accesibilidad, seguridad y eficiencia en un mercado donde la experiencia del usuario es el principal diferenciador.

Marco Metodológico:

La metodología se basó en un enfoque lineal con fases desarticuladas y documentación limitada, lo que generó retrasos y funcionalidades críticas sin implementar.

- **Fases**:
    1. **Análisis superficial**: Recolección de requerimientos sin priorización clara (ej: no se definió el flujo completo para el modo invitado).
    2. **Diseño no modular**: Desarrollo monolítico sin separación clara entre frontend y backend, dificultando la escalabilidad.
    3. **Desarrollo parcial**:
        - Implementación básica del CRUD del carrito (solo agregar productos, sin eliminar o modificar cantidades).
        - Integración de un filtro de búsqueda estático (solo por nombre, sin categorías o precios).
    4. **Pruebas insuficientes**: Validación manual sin casos de prueba estructurados (ej: no se probó el envío de tickets).
    5. **Retroalimentación omitida**: No se realizaron tests con usuarios reales.
- **Herramientas**:
    - HTML/CSS/JavaScript vanilla (sin frameworks).
    - Firebase (solo para almacenar productos, sin historial de pedidos).
    - Sin control de versiones (Git no implementado).

Brechas identificadas:

- Falta de simulación de cobro y historial de pedidos funcional.
- Modo invitado incompleto: los usuarios podían agregar al carrito pero no visualizarlo sin registrarse.
- Documentación técnica ausente, complicando el mantenimiento.

Desarrollo y Solución:

Estrategia Técnica Limitada:

1. **CRUD del Carrito**:
    - Implementación básica usando arrays en JavaScript sin persistencia de datos.
        - *Agregar*: Función agregarAlCarrito(idProducto) sin validar stock o duplicados.
        - *Eliminar*: No implementado (los productos quedaban "atascados" en el carrito).
    - Interfaz estática: Sin actualización en tiempo real al agregar/eliminar.
2. **Simulación de Cobro**:
    - Flujo ficticio sin integración de APIs, solo un mensaje "Compra exitosa" sin validación de datos.
3. **Modo Invitado**:
    - Los usuarios podían agregar al carrito, pero al intentar comprar, el sistema no redirigía a registro, generando errores.
4. **Filtro de Búsqueda**:
    - Búsqueda por nombre usando array.filter() sin soporte para categorías o precios.
5. **Historial de Pedidos**:
    - Almacenamiento en localStorage sin estructura clara (ej: {"pedidos": [{"productos": "Abrigo Clásico"}]}), sin vinculación a usuarios.
6. **Tickets de Compra**:
    - Generación manual (HTML estático) sin automatización ni envío por correo.

Problemas No Resueltos:

- Carrito no reflejaba cambios en tiempo real.

- Sin manejo de errores en transacciones (ej: productos agotados).
- Búsqueda ineficiente en catálogos grandes.

Forma de Trabajo:

Equipo:

- **Roles ambiguos**:
  • Un solo desarrollador asume múltiples roles (frontend, backend, diseño), generando cuellos de botella.
  • Sin tester dedicado: los errores se detectan manualmente por el equipo, sin casos de prueba estructurados.
  • Documentación técnica relegada a "notas informales" en archivos de texto.

Flujo de Trabajo:

1. **Sprints no definidos**:
   • Desarrollo caótico sin plazos claros (ej: el CRUD del carrito se implementó parcialmente en 3 semanas).
2. **Comunicación desorganizada**:
   • Reuniones esporádicas sin agenda; decisiones clave se pierden en chats informales de WhatsApp.
   • Uso de Google Drive sin estructura de carpetas, dificultando el acceso a documentos.
3. **Control de Calidad insuficiente**:
   • Código sin revisión por pares, acumulando deuda técnica (ej: funciones no optimizadas para el filtro de búsqueda).
   • Checklist básico: solo se verifica si "el sitio no se cae", sin validar funcionalidades completas.

Problemas Clave:

- Falta de responsabilidades claras retrasó la integración del modo invitado y el historial de pedidos.
- Ausencia de herramientas profesionales (Slack, Jira) dificultó el seguimiento de tareas.

Levantamiento de Requisitos:

Proceso Deficiente:

1. **Stakeholders no identificados**:
    - Solo se consultó al equipo interno, ignorando a usuarios potenciales o departamentos clave (ej: logística para el envío de tickets).
2. **Técnicas superficiales**:
    - Reuniones informales sin actas.
    - Requisitos recopilados en una lista desorganizada en Excel, sin categorizar.
3. **Requisitos Identificados (Parciales)**:
    - *Funcionales*:
        - "Agregar productos al carrito".
        - "Mostrar imágenes de productos".
    - *No Funcionales*:
        - "El sitio debe cargar rápido".
4. **Falta de Especificaciones**:
    - No se definió el flujo del modo invitado (ej: ¿pueden ver el carrito sin registrarse?).
    - Atributos de productos ambiguos: "descripción detallada" sin formato o extensión definida.
    - Omisión de requisitos clave:
        - Simulación de cobro.
        - Envío automatizado de tickets.
5. **Validación Ausente**:
    - No se realizaron entrevistas con usuarios reales para confirmar necesidades.

- Sin priorización: Todos los requisitos se marcaron como "urgentes".

6. **Documentación Inexistente**:
    - Sin diagramas de casos de uso o historias de usuario.
    - Requisitos técnicos (ej: integración con APIs) no especificados.

Brechas Críticas:

- Requisitos como "filtro de búsqueda" se interpretaron como "búsqueda por nombre", ignorando categorías o precios.
- El "historial de pedidos" no se vinculó a la autenticación de usuarios.