Propuesta de Proyecto Integrado

Título: Tienda Online Multifuncional con Autenticación Segura y Gestión de Pedidos

---

---

1. Temática (Qué problema se va a resolver)

Problema:
Desarrollar una tienda online totalmente funcional que permita a los usuarios registrarse, autenticarse, gestionar perfiles, realizar compras con métodos de pago integrados, y ofrecer a los administradores herramientas para supervisar productos, pedidos y usuarios.

Objetivo Principal:
Crear una plataforma segura y escalable con:

- Sistema de autenticación basado en JWT y cookies HTTP-only para prevenir ataques XSS/CSRF.
- Confirmación de registro vía correo electrónico.
- Roles de usuario (administrador, gestor, cliente) con control de acceso específico.
- Integración de pasarelas de pago (ej: Stripe, PayPal).
- Gestión de pedidos y seguimiento en tiempo real.

---

2. Perfiles de Usuario

Perfil	Descripción	Finalidad
Administrador	Usuario con acceso total al sistema.	Gestionar usuarios, productos, pedidos y configuraciones globales.
Gestor	Usuario con permisos para actualizar inventario y procesar pedidos.	Supervisar stock, resolver consultas y aprobar/devolver pedidos.
Cliente	Usuario registrado que puede navegar, comprar y gestionar su perfil.	Realizar compras, ver historial de pedidos y recibir notificaciones por email.

---

3. Épicas de la Aplicación.

1. Autenticación y Registro:
    - Registro de usuarios con verificación por correo electrónico.
    - Inicio de sesión seguro con JWT y cookies HTTP-only.
    - Restablecimiento de contraseña vía email.
2. Gestión de Productos:
    - CRUD de productos (solo administradores/gestores).
    - Filtros y búsqueda de productos (clientes).
3. Carrito y Pago:
    - Añadir/eliminar productos del carrito.
    - Integración con Stripe/PayPal para pagos.
    - Generación automática de facturas PDF tras la compra.
4. Pedidos y Seguimiento:
    - Historial de pedidos para clientes.
    - Dashboard de pedidos para gestores/administradores.
    - Notificaciones por email al confirmar pedidos o actualizar estados.
5. Panel de Administración:
    - Estadísticas de ventas y usuarios activos.
    - Gestión de roles y permisos.

---

4. Requisitos de la Interfaz de Usuario (UI)

- Diseño Responsive:
    - Adaptable a dispositivos móviles y escritorio.
    - Uso de SCSS para estilos modulares y mantenibles.
- Accesibilidad:
    - Contraste alto, etiquetas semánticas y compatibilidad con lectores de pantalla.

- Feedback Visual:
    - Mensajes de éxito/error tras acciones (ej: compra exitosa, registro confirmado).
    - Carga de estados con spinners o barras de progreso.
- Animaciones:
    - Transiciones suaves en formularios y cambios de página (ej: fade-in).

---

5. Requisitos Técnicos

Componente	Tecnología	Justificación
Backend	Django + Django REST Framework + PostgreSQL	Robustez, seguridad y escalabilidad para gestión de datos y autenticación.
Autenticación	JWT (JSON Web Tokens) con cookies HTTP-only	Seguridad avanzada contra ataques XSS/CSRF.
Frontend	Nuxt 3 (Vue.js) + Axios + SCSS	Renderizado rápido, integración sencilla con el backend y estilos personalizados.
Base de Datos	PostgreSQL en Docker	Persistencia de datos y facilidad de despliegue en contenedores.
Pasarela de Pago	Stripe/PayPal API	Integración de métodos de pago populares y seguros.
Despliegue	- **Opción 1**: con docker en AWS/DigitalOcean. - **Opción 2:** Backend en Heroku + Frontend en Vercel - **Opción 3**: Backend en PythonAnywhere + Frontend en Netlify.	Contenerización para entornos de desarrollo/producción y escalabilidad en la nube.

---

6. Requisitos del Despliegue

- <u>Entorno de Producción:</u>

    - Servidor:
        - **Opción 1**: con docker en AWS/DigitalOcean.
        - **Opción 2:** Backend en Heroku + Frontend en Vercel
        - **Opción 3**: Backend en PythonAnywhere + Frontend en Netlify.
    - Base de Datos: PostgreSQL.
        - **Opción 1**: con docker en AWS/DigitalOcean.
        - **Opción 2:** Backend en Heroku(con BDD) + Frontend en Vercel

- **Opción 3**: Backend en PythonAnywhere(con BDD) + Frontend en Netlify.
    - Seguridad: HTTPS con Let's Encrypt y firewall configurado.
- Despliegue Continuo:
    - CI/CD con GitHub Actions para automatizar pruebas y despliegues.

---

7. Planificación Temporal Inicial

Fase	Duración Estimada	Hitos
Configuración Inicial	1 semana	- Setup de Django/Nuxt 3. - Configuración de PostgreSQL en Docker.
Autenticación	2 semanas	- Registro con confirmación por email. - Login/logout con JWT.
Gestión de Productos	1.5 semanas	- CRUD de productos (admin/gestor). - Filtros y búsqueda (cliente).
Carrito y Pago	2 semanas	- Integración con Stripe/PayPal. - Generación de facturas PDF.
Despliegue	1 semana	- Contenerización con Docker. - **Opción 1**: en AWS/DigitalOcean. - **Opción 2:** Backend en Heroku + Frontend en Vercel Heroku (Backend): Gratis: Plan Hobby (550 horas/mes). Vercel (Frontend): Gratis: Hosting estático ilimitado. - **Opción 3**: Backend en PythonAnywhere + Frontend en Netlify. PythonAnywhere (Backend): Gratis: Plan Beginner (ejecución de Django con PostgreSQL). Netlify (Frontend): Gratis: Hosting estático con HTTPS.
Pruebas y Ajustes	1 semana	- Pruebas de seguridad y usabilidad. - Corrección de errores críticos.

Diagramas Sugeridos

1. Arquitectura del Sistema:
    - Frontend (Nuxt 3) ↔ API (Django) ↔ Base de Datos (PostgreSQL).
2. Flujo de Autenticación:
    - Registro → Email de confirmación → Login → Cookies HTTP-only.
3. Flujo de Compra:
    - Carrito → Pago → Confirmación → Factura.