Trabajo Práctico Integrador Final: Desarrollo de E-commerce Full Stack

Objetivo General

Desarrollar una aplicación web completa de comercio electrónico (E-commerce) que permita a los usuarios navegar por un catálogo de productos, gestionar un carrito de compras, realizar pedidos y procesar pagos (la integración con pasarela de pagos es opcional, puede solo generarse una orden en Base de Datos).

Se valorará que el sistema cuente con un panel de administración para la gestión de contenidos y usuarios, pero esta característica es opcional.

El proyecto debe implementarse utilizando el stack **MERN** (MongoDB, Express, React, Node.js), separando claramente el Backend (API REST) del Frontend (SPA).

Requerimientos Técnicos

1. Backend (API RESTful)

Debe ser desarrollado con **Node.js** y **Express**, utilizando **MongoDB** como base de datos.

Autenticación y Seguridad

- Implementar registro y login de usuarios.
- Uso de **JWT (JSON Web Tokens)** para manejo de sesiones.
- Hashing de contraseñas con **Bcrypt**.
- Middleware de protección de rutas (verificación de token).
- Middleware de autorización por roles (Admin vs User).
- Validación de datos de entrada (ej. express-validator).

Base de Datos (MongoDB + Mongoose) (Puede variar si se explica en readme)

- **Usuarios:** Nombre, email, password, rol (admin/user), estado de verificación.
- **Productos:** Nombre, descripción, precio, stock, categoría, imagen (URL).
- **Órdenes:** Referencia a usuario, array de productos (con precio congelado), total, estado, ID de pago.

🔌 **Endpoints sugeridos (pueden variar si se explica en readme)**

- **Auth:** /register, /login, /me, /forgot-password, /reset-password.
- **Productos:**
  - GET / (Público)

    - GET /:id (Público)
    - POST /, PUT /:id, DELETE /:id (Privado - Solo Admin).
- **Órdenes:**
    - POST / (Crear orden - Usuario autenticado).
    - GET / (Mis órdenes - Usuario autenticado).
    - GET /all (Todas las órdenes - Solo Admin).
- **Usuarios:**
    - GET / (Listar todos - Solo Admin).
    - DELETE /:id (Eliminar - Solo Admin).

💳 **Integraciones**

- **Pasarela de Pagos (opcional):** Integración con **MercadoPago** (o Stripe) para procesar pagos reales o en modo sandbox.
- **Email Service:** Envío de correos transaccionales (Bienvenida, Recuperación de clave) usando **Nodemailer**.

---

2. Frontend (Single Page Application)

Debe ser desarrollado con **React** (preferiblemente usando **Vite**).

Interfaz y UX

- **Diseño Responsivo:** Adaptable a móviles y escritorio.
- **CSS:** A elección del alumno. Puede usarse CSS puro o algun Framework.
- **Modo Oscuro(Opcional):** Implementación de tema Dark/Light (Opcional pero recomendado).
- **Feedback (Opcional):** Uso de notificaciones (Toasts) para acciones (éxito/error).

Gestión de Estado y Lógica

- Uso de **Context API**, **Zustand** para estados globales:
    - **Auth:** Usuario logueado, token.
    - **Carrito:** Agregar/Quitar items, calcular total, persistencia en LocalStorage.

Navegación (React Router)

- **Rutas Públicas:** Home, Catálogo, Detalle de Producto, Login, Registro.
- **Rutas Privadas:** Perfil, Historial de Órdenes, Checkout.
- **Rutas de Admin (Opcional):** Dashboard, CRUD de Productos y Usuarios.
- Manejo de rutas 404 (No encontrado).

Funcionalidades

- **Catálogo:** Visualización de productos en grilla.
- **Detalle:** Vista individual con información completa. **(Opcional)**
- **Carrito:** Drawer o página dedicada para gestionar items antes de comprar.
- **Checkout:** Integración con el flujo de pago del backend.
- **Admin Panel:** Tablas para ver y administrar recursos (CRUD). **(Opcional)**

Repositorios, Documentación y Despliegue

1. **Repositorios:** El código debe estar alojado en **GitHub** (dos repositorios separados).
2. **Documentación:** Cada repositorio debe tener un README.md detallado con:
    - Instrucciones de instalación.
    - Variables de entorno necesarias.
    - Explicación de la arquitectura.
3. **Despliegue (Deploy):**
    - **Frontend:** Vercel, Netlify o similar.
    - **Backend:** Vercel, Render, Railway o similar.
    - La aplicación debe estar 100% funcional en producción.

Criterios de Evaluación

- **Funcionalidad:** Cumplimiento de todos los requerimientos técnicos.
- **Calidad de Código:** Estructura limpia, modularización, nombres de variables descriptivos.
- **UX/UI:** Usabilidad y estética general.
- **Seguridad:** Manejo correcto de información sensible.

Formato de entrega

Deberán entregarse mediante Google drive u otro medio de compartir archivos, un documento en formato txt o md que contenga:

1. Nombre y Apellido del alumno.
2. URL del repositorio backend (publico)
3. URL del repositorio frontend (publico)
4. URL del deploy del backend (funcionando).
5. URL del deploy del frontend (funcionando).
6. Contenido del .env del backend.
7. Contenido del .env del frontend.

Formulario de entrega

https://cgsw.short.gy/integrador1