E-commerce Pyme — Gestión de productos y ventas

Proyecto académico para construir un sistema **e-commerce local** orientado a una **pyme de venta de artículos de computación**.
Incluye **backend, frontend y base de datos** con funcionalidades completas de catálogo, carrito, checkout, reportes y gestión administrativa.

Objetivo del sistema

El sistema busca profesionalizar la gestión de la pyme mediante:

- **Catálogo público** con búsqueda y filtros por categoría.
- **Cuentas y roles**: admin, vendedor (dueño) y cliente.
- **Backoffice (admin/vendedor)**: CRUD de productos/categorías, gestión de stock, fotos, precios y descripciones.
- **Carrito y checkout (cliente)**: agregar/editar/quitar, validar stock, pago simulado y registro de pago.
- **Reportes (vendedor)**: ventas por mes, top productos y stock bajo umbral.
- **Tickets de cambio**: apertura dentro de 10 días con número de pedido, productos y motivo.

Tecnologías a usar

Backend

- **Node.js + Express + TypeScript**
- **Prisma** (ORM) + **PostgreSQL**
- **Auth**: JWT + bcrypt
- **Validación**: Zod
- **Seguridad**: Helmet, CORS, express-rate-limit
- **Imágenes**: multer + sharp (carpeta `uploads/`)
- **Logs**: pino o winston
- **Docs**: Swagger (OpenAPI) (opcional)

Frontend

- **React + Vite + TypeScript**
- **React Router**, **React Query**
- **React Hook Form** + Zod
- **Tailwind CSS**
- **Recharts** (gráficos)
- **Zustand** (estado ligero)

Base de datos / entorno

- **PostgreSQL** (Windows)

- Extensión recomendada: **pg_trgm** (búsqueda por texto)
- Seeds/backups: `pg_dump` / `pg_restore`

Módulos principales de momento

- **Auth & Usuarios**: registro/login, hash de contraseñas, emisión/refresco de JWT, middleware de roles.
- **Catálogo**: categorías, productos, imágenes, búsqueda y filtros; paginación y orden.
- **Carrito & Checkout**: carrito por usuario, validación de stock, creación de orden.
- **Órdenes & Pagos**: flujo `pending → paid` con pago simulado, registro en payments, decremento de stock atómico.
- **Reportes**: ventas por mes, top productos, stock bajo umbral.
- **Tickets de cambio**: creación y gestión con ventana de 10 días desde la fecha de la orden.

Flujo de interacción de momento

Cliente no autenticado

- Navega catálogo → busca/filtra → ve detalle producto → agrega al carrito (local o user si ya autenticado).

Autenticación

- Login → API emite JWT → Front guarda sesión (cookie HttpOnly o memoria segura) → rutas privadas habilitadas.

Cómo probar (demo local)

Requisitos:

- Node.js 18+
- PostgreSQL local

1. Backend

- Ejecutar en una terminal dentro del proyecto:
  cd backend
  npm run dev

2. Frontend

- Abrir otra terminal dentro del proyecto y ejecutar:
  cd frontend
  npm run dev
  (el script copia .env si falta, instala dependencias y levanta Vite)

3. Flujos

- Catálogo público (home): http://localhost:5173/
- Login: http://localhost:5173/login
    - Credenciales demo: cliente@demo.com / secret12
    - Tras iniciar sesión, el header mostrará “Cerrar sesión”
- Área privada: http://localhost:5173/admin (requiere sesión)

Notas:

- CORS está habilitado para http://localhost:5173 en desarrollo.
- Si el carrusel no muestra imágenes, agrega opcionalmente banners en rontend/public/banners/. Los productos mostrados son solo una demo de lo que se mostrara al final

Backoffice (admin/vendedor)

- CRUD productos/categorías → subida de imágenes → reportes (ventas/mes, top productos, stock bajo) → gestión de tickets de cambio.

Ejecución con Docker

Requisitos:

- Docker Desktop (o motor Docker) + Docker Compose v2

Pasos:

```
docker-compose up --build
```

Servicios expuestos:

- Frontend: http://localhost:5173
- Backend API: http://localhost:4000/api
- PostgreSQL: localhost:5432 (usuario `postgres`, password `postgres`, base `pyme`)

El contenedor del backend ejecuta automáticamente las migraciones (`prisma migrate dev`) y el seed con usuarios demo:

- Cliente: `cliente@demo.com / secret12`
- Vendedor: `vendedor@tienda.com / password123`

Para detener todo:

```
docker-compose down
```

Si quieres comenzar desde cero (reiniciar datos):

```
```

```
docker-compose down -v
```

Desarrollo con hot reload (Docker)

Si quieres trabajar desde Docker con recarga en vivo (sin instalar Node/Postgres localmente), usa:

```
docker compose -f docker-compose.dev.yml up --build
```

- Frontend: expuesto en http://localhost:5173 con Vite dev server (volumen sobre `./frontend`).
- Backend: expuesto en http://localhost:4000 corriendo `npm run dev` (volumen sobre `./backend`).
- Base de datos: Postgres en el contenedor `db`.

Los cambios en el codigo se reflejan inmediatamente.

Resumen para el equipo

1. Clonar el repo.
2. Elegir una de estas opciones:
    - **Local**: instalar dependencias (`npm install`) y usar `npm run dev` en backend/frontend con PostgreSQL local.
    - **Docker demo**: `docker compose up --build` para la version compilada lista para demo.
    - **Docker dev**: `docker compose -f docker-compose.dev.yml up --build` para trabajar con hot reload sin dependencias locales.
3. Credenciales demo:
    - Cliente: `cliente@demo.com / secret12`
    - Vendedor: `vendedor@tienda.com / password123`

Con cualquiera de los modos, el backend ejecuta migraciones y seed autom�ticamente al iniciar.