1) Objetivo del sistema

Construir un **e-commerce local** para una pyme (venta de artículos de computación) que permita:

- **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.

2) Tecnologías decididas

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`

3) Módulos principales (de momento)

- **Auth & Usuarios**: registro/login, hash de contraseñas (bcrypt), 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**: orden `pending → paid` con **pago simulado**, registro en `payments`, decremento de stock atómico (transacción).
- **Reportes**: agregaciones SQL (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.

4) Flujo de interacción

1. **Cliente no autenticado**
    - Navega catálogo → busca/filtra → ve detalle producto → agrega al carrito (local o user si ya autenticado).
2. **Autenticación**
    - Login → API emite JWT → Front guarda sesión (cookie HttpOnly o memoria segura) → rutas privadas habilitadas.
3. **Checkout y pago simulado**
    - Carrito → confirma → API valida stock → crea **order** + **order_items** (transacción) → **payment** simulado → `status=paid` → decrementa stock.
4. **Backoffice (admin/vendedor)**
    - CRUD productos/categorías → subida de imágenes → reportes (ventas/mes, top, stock bajo) → gestión de tickets de cambio.