Propuesta 5 — “StoreHub: Sistema de Inventario y Punto de Venta”

1. Descripción General

StoreHub es una aplicación web para la **gestión de inventario, ventas y control básico de clientes** de un pequeño negocio.

El sistema permite registrar productos, administrar existencias, realizar ventas, emitir tickets, aplicar descuentos y consultar reportes de ventas por día, semana o mes.

La aplicación inicia con una **Landing Page** con la identidad visual del negocio (logo, slogan), breve descripción y botones de **Login** y **Registro** para personal autorizado.

Objetivos del Cliente

- Centralizar el control de productos, stock y ventas.
- Disminuir errores de registro y pérdida de inventario.
- Permitir que los administradores revisen estadísticas de ventas en línea.
- Facilitar la atención en mostrador con un flujo de venta rápido.

---

2. Módulos y Entidades Principales

2.1 Usuarios y Autenticación

- **Registro**: nombre, apellidos, correo, contraseña, rol (asignado por Administrador).
- **Login**: correo + contraseña.
- **Perfil**: edición de datos personales y cambio de contraseña.
- **Roles**: Administrador, Cajero/Vendedor.
- **Sesiones**: uso de **JWT** para autenticación.
- **Autorización**: guards en frontend para restringir acceso a inventario y reportes.

2.2 Catálogos

- **Productos**: nombre, descripción, código de barras o SKU, precio de venta, costo de adquisición, cantidad en stock, categoría, estado (activo/inactivo).
- **Categorías**: nombre y descripción.
- **Clientes**: nombre, correo (opcional), teléfono (opcional).

2.3 Ventas

- **Campos obligatorios**: productos seleccionados, cantidad por producto, subtotal, impuestos, total, cliente (opcional).
- **Flujo**:
    - Escanear o seleccionar producto.

    - Calcular total en tiempo real.
    - Confirmar pago y generar ticket.
    - Disminuir stock automáticamente.

2.4 Reportes

- **Ventas diarias**: total vendido, número de transacciones.
- **Ventas por producto**: productos más vendidos, stock crítico.
- **Ingresos por periodo**: filtros por semana, mes, rango de fechas.

2.5 Ajustes de Inventario

- **Entradas y salidas de stock**: registrar compras, mermas o ajustes.
- **Historial de movimientos**: fecha, tipo de movimiento, usuario que lo realizó.

---

3. Flujo de Navegación (Frontend)

3.1 Estructura del Proyecto (Angular 19)

Organización estándar del curso:

- **/screens**
    - landing
    - auth/login, auth/register
    - dashboard (según rol)
    - products/list, products/form
    - categories/list, categories/form
    - clients/list, clients/form
    - sales/pos (pantalla de venta tipo ticket)
    - sales/history
    - inventory/movements
    - reports
    - profile
- **/partials**
    - navbar, sidebar, footer, toast/alerts
- **/modals**
    - confirm-delete-product-modal

    - confirm-sale-modal
    - low-stock-warning-modal
    - adjust-stock-modal
- **/services**
    - auth.service, products.service, categories.service, clients.service, sales.service, inventory.service, reports.service
- **/shared**
    - Interfaces TS: Producto, Categoría, Cliente, Venta, MovimientoInventario.
    - Pipes: formato de moneda, formato de fecha.
    - Guards: AuthGuard, RoleGuard.
    - Interceptors: JWT y manejo de errores.

3.2 Rutas y Acceso

- **Públicas**: /, /auth/login, /auth/register
- **Autenticadas**: /dashboard, /sales/pos, /sales/history, /profile
- **Solo Administrador**: /products, /categories, /inventory/movements, /reports
- **Cajero/Vendedor**: /sales/pos, /sales/history

3.3 Comportamientos UX Clave

- **Pantalla de ventas (POS)** optimizada: búsqueda rápida por código o nombre, suma automática de totales, generación de ticket (vista previa modal).
- **Alertas de stock bajo**: mostrar modal cuando un producto esté por debajo del mínimo.
- **Filtros** en reportes: por fechas, categoría y vendedor.
- **Diseño mobile-first** para poder usarlo en tablets o teléfonos.

---

4. Reglas de Negocio y Validaciones

4.1 Productos

- Código de barras/SKU único.
- Precio y stock mayores o iguales a 0.
- No se puede vender producto inactivo.

4.2 Ventas

- Validar que haya stock suficiente antes de confirmar la venta.
- Registrar fecha, hora y usuario que realizó la transacción.

- Generar número de ticket único por venta.

4.3 Ajustes de Inventario

- Registrar tipo de movimiento (entrada/salida/merma).
- Solo administrador puede hacer ajustes manuales.

---

5. Backend (Django 4.2 + DRF + JWT)

5.1 Estructura de Apps

- accounts (usuarios y roles)
- products (CRUD productos y categorías)
- clients (gestión de clientes)
- sales (registro de ventas y tickets)
- inventory (movimientos de stock)
- reports (indicadores de ventas)

5.2 Modelos

- **User**: nombre, correo, rol.
- **Product**: nombre, código, precio, stock, categoría, estado.
- **Category**: nombre, descripción.
- **Client**: nombre, correo, teléfono.
- **Sale**: usuario, lista de productos vendidos, subtotal, impuestos, total, fecha.
- **InventoryMovement**: producto, tipo de movimiento, cantidad, usuario, fecha.

5.3 Vistas/Endpoints

- CRUD de productos, categorías y clientes.
- Endpoints para ventas: crear venta, listar historial, obtener ticket.
- Endpoints para movimientos de inventario.
- Reportes: ventas por rango de fechas, top productos, stock bajo.

5.4 Paginación, Búsqueda y Orden

- Paginación de productos y ventas.
- Búsqueda por nombre, código, categoría.
- Orden por fecha de venta y cantidad vendida.

---

6. Requisitos del Frontend

6.1 Pantallas Mínimas

- Landing informativa.
- Login / Registro.
- Dashboard por rol.
- CRUD de productos y categorías (Admin).
- Pantalla de ventas tipo POS (Cajero).
- Reportes de ventas (Admin).
- Historial de ventas (Cajero/Admin).
- Ajustes de inventario.

6.2 Validaciones de Formularios

- Producto: nombre y código obligatorios, precio ≥ 0, stock ≥ 0.
- Venta: no permitir cantidades negativas, no confirmar si stock insuficiente.
- Cliente: nombre obligatorio si se asocia a la venta.

6.3 Modales

- Confirmación de venta.
- Advertencia de stock bajo.
- Confirmación de eliminación de producto/categoría.
- Ajuste de stock con motivo.

---

7. Requisitos No Funcionales

- **Mobile-first** para uso en tablet en mostrador.
- **Seguridad**: JWT, permisos por rol.
- **Rendimiento**: consultas optimizadas para ventas en tiempo real.
- **Mantenibilidad**: estructura modular.

---

8. Criterios de Aceptación

1. CRUD de productos y categorías funcional con validaciones.
2. Flujo de venta completo: selección → cálculo → confirmación → ticket.
3. Stock disminuye automáticamente al confirmar venta.

4. Reportes de ventas filtrables por fecha.
5. Alertas de stock bajo funcionando.
6. App desplegada en Vercel + Render.

---

9. Despliegue

- **Frontend en Vercel** con API_BASE_URL.
- **Backend en Render** con MySQL y migraciones aplicadas.
- CORS configurado para permitir origen de Vercel.

---

10. Entregables del Equipo

- **Modelo ER** (productos, ventas, movimientos).
- **Manual de usuario** con capturas.
- **Video demostrativo** mostrando venta, generación de ticket y actualización de stock.
- **Repositorios GitHub** separados.
- **URLs de producción** funcionando públicamente.
- **Evidencia de responsividad**.

---

11. Matriz de Permisos (resumen)

Funcionalidad / Recurso	Administrador	Cajero/Vendedor
Ver landing page	✔	✔
Registro/Login	✔	✔
CRUD de productos/categorías	✔	✖
CRUD de clientes	✔	✔ (sólo alta rápida en venta)
Registrar ventas (POS)	✔	✔
Ver historial de ventas	✔	✔
Ajustar inventario	✔	✖
Ver reportes de ventas	✔	✖