1. Objetivo del proyecto

Construir un sistema POS profesional, modular y escalable para una cadena de tiendas de tenis con múltiples sucursales físicas. Debe soportar ventas en tienda, gestión de inventario y transferencias entre sucursales, control de cajas, reportes comerciales, gestión de clientes y productos, integraciones de pago y administración centralizada.

2. Alcance (MVP)

Funcionalidades mínimas necesarias para arrancar en producción en sucursales:

- Login / autenticación y control de roles (Cajero, Supervisor, Administrador).
- Pantalla de venta (POS) con búsqueda por SKU, código de barras y nombre.
- Apertura/cierre de caja por usuario por sucursal.
- Gestión de productos: SKU, tallas, colores, stock por sucursal, precio, costo, imágenes.
- Gestión de clientes (básica): nombre, contacto, historial de compras.
- Inventario por sucursal + ajustes manuales y recepción de compras.
- Facturación / recibos (impresión/ticket) y exportación (PDF).
- Reportes básicos: ventas por día/sucursal, top productos, control de caja.
- Registro de usuarios y auditoría de acciones (quién hizo qué).
- Soporte para varias sucursales y vista centralizada.
- Exportes CSV / Excel para contabilidad.

3. Requisitos funcionales (detallados)

Usuarios y roles

- **Administrador**: acceso total, configuración global, crear sucursales, usuarios.
- **Supervisor / Gerente de Sucursal**: reportes extendidos de su sucursal, ajuste de stock, devoluciones aprobadas.
- **Cajero**: operar POS, abrir/cerrar caja, registrar ventas y devoluciones.
- **Contabilidad (opcional)**: acceso a exportes y reportes contables.

POS (flujo de venta)

- Selección de sucursal.
- Crear venta: agregar productos por código, escanear, ajustar cantidad, aplicar descuento global o por línea.
- Manejar tallas/variantes: cuando un producto tenga variantes (talla, color), selección clara en la UI.
- Métodos de pago: efectivo, tarjeta (pos/pasarela), transferencia, cupón/regalo.
- Cierre de venta: generar recibo, disminuir stock en la sucursal, registrar en caja.
- Notas en venta (ej. reserva, pedido a otra sucursal).

Caja

- Apertura de caja: monto inicial.
- Movimientos de caja: ingreso/egreso (gastos), cierre de caja diario con conciliación.
- Historial y auditoría de cierres.

Inventario

- Stock por sucursal y stock total consolidado.
- Recepciones (entradas): compras a proveedor o transferencias entre sucursales.
- Salidas: ventas, ajustes por pérdida/rotura, devoluciones.
- Alertas por stock mínimo.
- Lote/serie (opcional) — no crítico para tienda de tenis, salvo que lo necesites.

Productos y variantes

- Producto padre + variantes por talla/color.
- SKU único por variante (recomendado).
- Gestión de múltiples precios (precio público, precio por promoción, precio mayorista).
- Imágenes y descripción.

Clientes y fidelización

- Registro sencillo: nombre, teléfono, email, notas.
- Historial de compra por cliente.
- Programa de puntos/cupones (extra, para fase posterior).

Reportes

- Ventas por día/semana/mes por sucursal y consolidado.
- Top 10 productos vendidos.
- Margen bruto / costos.
- Movimientos de caja e historial de cierres.
- Inventario valorizado por sucursal.

Integraciones

- Impresora de tickets (local) — idealmente usar impresión a navegador o mediante un pequeño servicio local.
- Pasarelas de pago (integrar POS o gateway según país).
- ERP/contabilidad (exportes o API).
- Barcode scanner (entrada de teclado HID).

4. Requisitos no funcionales

- Autenticación segura (Laravel Fortify ya en composer).
- Multi-tenant lógico por sucursal (misma base, separación por sucursal en datos).
- Alta disponibilidad y tolerancia a fallos mínima: backups diarios, snapshots.
- Escalabilidad horizontal en la capa web (Docker / Kubernetes posible).
- Latencia baja en POS: interfaz rápida; minimizar round-trips a servidor.
- Logs y auditoría (quien hizo qué y cuándo).
- Seguridad: OWASP, cifrado de datos sensibles, protección CSRF, rate limiting.
- Localización: español (Colombia) por defecto con posibilidad de más idiomas.
- Accesibilidad y UI responsive (tablets/PCs de caja).

5. Arquitectura propuesta (alto nivel)

- **Frontend**: Blade + Livewire (Flux/Volt) — UI reactiva sin SPA completa; componentes para POS, formularios y reportes.
- **Backend**: Laravel 12 (PHP ^8.2) con Fortify para autenticación.
- **Base de datos**: MySQL / MariaDB (transaccional). Diseño relacional normalizado con tablas: sucursales, users, productos, variantes, inventarios, ventas, lineas_venta, cajas, movimientos_caja, clientes, proveedores, transferencias, logs.
- **Capa de servicios**: servicios PHP (Domain Services) para lógica de stock, cierre de caja, cálculo de precio/margen — separar lógica de controladores.
- **Repositorio**: repository pattern opcional para desacoplar Queries complejas.
- **Colas**: Redis + Laravel Queue para tareas asíncronas (envíos de correo, generación de reportes pesados, sincronización).
- **Cache**: Redis para caches de catálogo y sesiones.
- **File Storage**: S3 compatible o almacenamiento local para imágenes de productos.
- **Contenedor**: Docker para desarrollo y despliegue reproducible.
- **CI/CD**: GitHub Actions / GitLab CI para tests y despliegue automatizado.
- **Monitoreo**: Sentry (errores), Prometheus/Grafana u otros para métricas.

6. Modelo de dominio (tablas principales — sin SQL)

- **sucursales**: id, nombre, dirección, contacto, config_impresora…
- **users**: id, nombre, email, password, rol, sucursal_id (default).
- **productos**: id, nombre, descripción, categoría, marca.
- **variantes**: id, producto_id, sku, talla, color, precio, costo, imagen_url.
- **inventarios**: id, variante_id, sucursal_id, cantidad, stock_minimo.
- **ventas**: id, sucursal_id, user_id, cliente_id (opcional), total, impuestos, estado, metodo_pago, created_at.
- **lineas_venta**: id, venta_id, variante_id, cantidad, precio_unitario, descuento.
- **cajas**: id, sucursal_id, user_id, fecha_apertura, monto_inicial, fecha_cierre, monto_cierre, estado.

- **movimientos_caja**: id, caja_id, tipo (ingreso/egreso/venta), monto, descripcion, referencia.
- **transferencias**: id, from_sucursal_id, to_sucursal_id, estado.
- **clientes**: id, nombre, telefono, email, notas.
- **auditoría/logs**: id, user_id, accion, modelo, referencia_id, payload, timestamp.

7. Flujos clave (resumidos)

Venta en POS (flujo simple)

1. Cajero inicia sesión y selecciona sucursal (si aplica).
2. Abre caja (si no está abierta).
3. Escanea o busca producto -> agrega variante (talla) -> qty.
4. Selecciona método de pago y finaliza.
5. Sistema registra venta, decrementa inventario en la sucursal, crea movimiento de caja.
6. Genera ticket/recibo y actualiza reportes en background (cola).

Transferencia entre sucursales

1. Supervisor solicita transferencia desde sucursal A a B.
2. Se reserva stock en A (estado “en tránsito”), y al confirmar recepción en B se incrementa stock B.
3. Auditoría de la operación.

8. Historias de usuario (ejemplos con criterios de aceptación)

HU-01: Como cajero quiero registrar una venta para entregar un recibo.

Criterios:

- Puedo buscar/escaneo por SKU.
- Puedo seleccionar talla/variante.
- Puedo aplicar descuento por línea o total (si tengo permiso).
- Al finalizar, se imprime o genera PDF del comprobante.

- Inventario de la sucursal se decrementa correctamente.

HU-02: Como supervisor quiero cerrar la caja y obtener conciliación.

Criterios:

- Se muestra monto inicial, ventas registradas, ingresos/egresos.
- Puedo confirmar montos y generar reporte de cierre.
- Registro de quién cerró y a qué hora.

HU-03: Como administrador quiero ver ventas consolidadas por sucursal.

Criterios:

- Puedo filtrar por rango de fechas y sucursal.
- Puedo exportar a CSV/Excel.

9. Consideraciones técnicas importantes / decisiones

- **Stock por variante**: obligatorio usar SKU por variante para evitar ambigüedades con tallas.
- **Consistencia transaccional**: operaciones de venta e inventario deben estar dentro de transacción DB para evitar inconsistencias.
- **Offline / conectividad inestable**: si las sucursales tienen conexión inestable, considerar un modo offline (app local o caché + sincronización). Esto añade complejidad — sugerido para fase 2.
- **Impresión**: la impresión local en caja suele requerir integración con impresoras térmicas; mejor usar impresión por navegador o un pequeño agente local que reciba jobs desde el servidor.
- **Pagos con tarjeta**: integración con terminales POS locales o pasarela (con tokenización). Evitar almacenar datos de tarjeta.
- **Separación de responsabilidades**: UI (Livewire) para interacción, Domain Services para reglas de negocio, Repositories para acceso a datos, Jobs para procesos en background.

10. Seguridad y cumplimiento

- HTTPS obligatorio.
- Laravel Fortify para autenticación, con 2FA opcional.

- Logs de auditoría y registro de acciones.
- Backups diarios y retención (7-30 días según políticas).
- Políticas de contraseñas fuertes y manejo de roles/permissions.
- No almacenar datos de tarjeta; usar tokenización del proveedor de pagos.

11. Tests y calidad

- **Unit tests** para servicios de negocio (cálculo de precios, decremento de stock).
- **Feature tests** para flujos críticos (venta, apertura/cierre de caja).
- **E2E** para la UI/Pos con Playwright o Cypress (fase posterior).
- Integrar cobertura mínima en CI (ej. >60% inicialmente).

12. Infraestructura y despliegue (sugerido)

- Entorno: Docker Compose para desarrollo, Docker + Kubernetes (o Docker Swarm) para producción.
- Hospedaje: proveedor cloud (DigitalOcean, AWS, GCP) — recomendar empezar con VPS o Managed App para simplificar.
- BD: Managed MySQL (RDS / Cloud SQL / Managed DB).
- Storage: S3 compatible para imágenes.
- Redis para cache y colas.
- Certificados SSL automáticos (Let's Encrypt).
- CI/CD: GitHub Actions para tests y despliegue a staging/production.

13. Plan de trabajo inicial (fases y entregables — alto nivel)

Fase 0 — Levantamiento y diseño (TU petición actual)

- Entrevistas con stakeholders (operaciones, caja, inventario, gerencia).
- Recolección de requerimientos específicos por sucursal.
- Mapear procesos físicos (cómo funciona la caja hoy).
- Documento de especificación funcional y diagrama de dominio.

Fase 1 — MVP (core POS)

- Infraestructura y scaffolding (repo, CI).
- Modelo de datos y CRUD para productos/variantes.
- POS (UI Livewire) + apertura/cierre de caja.
- Inventario por sucursal y reportes básicos.
- Test básicos y despliegue a staging.

Fase 2 — Integraciones y mejoras

- Pasarela de pago / integración TPV.
- Gestión avanzada: transferencias, devoluciones, promociones.
- Impresión tickets y personalización.
- Mejoras de rendimiento y caché.

Fase 3 — Escala y extras

- Offline sync, fidelización, notificaciones, analítica avanzada.

Nota: no doy tiempos concretos aquí por tu instrucción de “levantamiento” y porque los tiempos dependen de detalles como equipo, infraestructura y alcance por sucursal.

14. Riesgos y mitigaciones

- **Inconsistencia de stock**: usar transacciones DB y auditoría; reconciliaciones periódicas.
- **Conectividad en sucursales**: definir plan offline o reglas claras para fallas.
- **Integraciones de pago**: validar requisitos del proveedor antes de diseñar.
- **Adopción por usuarios**: invertir en UX simple y entrenamiento a cajeros.

15. Entregables que puedo preparar ahora (sin código)

- Documento de especificación funcional completo (word/markdown) por módulo.
- Diagramas: ERD (texto / descripción), diagramas de flujo de venta y caja.
- Historias de usuario completas y backlog inicial priorizado (para sprint 0/1).
- Plantilla de pruebas de aceptación (checklist por HU).

- Checklist de infra/DevOps (dockerfile, pipelines, backups, monitorización).
- Propuesta de tablero Kanban/Sprints y definición de “ready”/“done”.

16. Qué necesito de ti para avanzar con el levantamiento (datos a recopilar)

(No es una pregunta directa, solo un listado de inputs útiles; elige los que puedas darme)

- Número de sucursales y ubicación (si hay diferencias operativas).
- Volumen esperado: tickets/día por sucursal (apróx.).
- ¿Se factura con facturación electrónica obligatoria? ¿Requiere integración con proveedor tributario?
- ¿Usarán impresoras térmicas específicas o POSEn línea?
- ¿Deseas soporte offline desde el inicio?
- ¿Tienes proveedor de pagos o terminales ya contratados?
- ¿Preferencia de hosting (cloud local / proveedor internacional)?
- Nivel de integración con contabilidad / ERP existente.

17. Recomendaciones finales y buenas prácticas

- Empezar por modelar procesos reales de tienda (observación de 1-2 días en sucursal).
- SKU por variante: evita confusiones entre tallas.
- Registrar siempre usuario y sucursal en todas las operaciones para auditoría.
- Automáticamente versionar y realizar migraciones con cuidado en producción (backups antes).
- Priorizar UX del POS: la velocidad en el flujo de caja es crítica.