Mi Tienda

Manual del sistema: cómo funciona, guía técnica y de pruebas

Reescritura del e-commerce en Laravel 12

Generado el 25/07/2026 23:08

1. Qué es este proyecto

Este proyecto es una reescritura completa, desde cero, del e-commerce que antes corría en PHP plano sin framework (carpeta `mi_tienda`, servido por XAMPP). El sistema anterior tenía deuda técnica y de seguridad real: sin llaves foráneas en la base de datos, credenciales de pago en texto plano, contraseñas con salt compartido, y un carrito que vivía solo en el navegador. El proyecto nuevo (`mi_tienda_v2`) corrige todo eso sobre una arquitectura moderna: Laravel 12, con panel de administración y tienda pública completamente funcionales, pensado específicamente para operar un negocio en Perú (facturación electrónica SUNAT, pasarelas de pago locales, WhatsApp) y para poder reutilizarse en clientes nuevos (carga masiva de catálogo vía Excel y una herramienta para dejar la base de datos lista antes de poblarla).

2. Cómo funciona el sistema, de principio a fin

Esta sección explica el funcionamiento completo en lenguaje simple, sin tecnicismos, siguiendo el recorrido real de un cliente y luego el de un administrador. Las secciones siguientes del manual retoman cada punto en detalle técnico.

2.1 El recorrido del cliente

1 **Explorar la tienda.** El visitante entra a la tienda sin necesidad de tener cuenta. Puede navegar por categorías y subcategorías, usar el buscador, filtrar por atributos (por ejemplo, talla o color) y ordenar los resultados. Cada producto muestra su precio, y si tiene una oferta activa, el precio anterior aparece tachado automáticamente.

2 **Elegir un producto.** En la ficha del producto, si tiene variantes (por ejemplo, distintas tallas o colores), el cliente elige la combinación exacta y el sistema le muestra el stock disponible de esa combinación en particular, no del producto en general.

3 **Agregar al carrito.** El carrito no vive solo en el navegador (como en el sistema anterior): queda guardado en el servidor, así que si el cliente cierra la pestaña o cambia de dispositivo después de iniciar sesión, no lo pierde. Si estaba comprando como invitado y luego inicia sesión, su carrito de invitado se fusiona automáticamente con el de su cuenta.

4 **Ir al checkout.** El cliente ingresa (o confirma) su dirección de envío, tipo de comprobante (boleta con DNI o factura con RUC) y, si tiene un cupón, lo aplica ahí mismo. El sistema vuelve a calcular el precio, el envío, el impuesto y el descuento del cupón directamente desde la base de datos — nunca confía en los montos que el navegador le muestra al cliente, precisamente para evitar que alguien manipule el precio desde su propio navegador.

5 **Elegir cómo pagar.** Hay seis formas de pagar, para cubrir distintos tipos de cliente:

- **Tarjeta (Culqi)**: paga con tarjeta de crédito/débito directamente en la tienda.
- **Yape**: paga con su app Yape ingresando su número de celular y el código que le llega por SMS/notificación.
- **Yape / Plin (QR)**: se le muestra un código QR (o un código para ingresar manualmente) que escanea o ingresa desde su app Yape o Plin — a diferencia de la opción anterior, aquí Plin también es una alternativa válida, y el pago se confirma automáticamente apenas el banco lo procesa, sin que el cliente tenga que volver a hacer nada.
- **Izipay**: otra pasarela de tarjeta, como opción alternativa a Culqi.
- **PayPal**: pensado para clientes que compran desde el extranjero.
- **Pago contra entrega**: paga en efectivo cuando el repartidor le entrega el pedido — solo disponible para pedidos de hasta S/ 500, sin restricción de a qué ciudad se envía.

En cualquier caso, el pedido queda registrado en el sistema en estado **"Pendiente"** desde el primer instante, y el stock del producto queda *reservado* (apartado) para que otro cliente no se lo lleve mientras se confirma el pago — pero todavía no se descuenta del inventario ni se cobra nada hasta que el pago se confirme de verdad.

6 **Confirmación del pago.** Cuando el pago se confirma (al instante con tarjeta/Yape/Izipay, apenas Culqi o PayPal avisan que se completó con Yape/Plin QR o PayPal, o cuando el administrador confirma el cobro en efectivo de un pedido contra entrega), el sistema recién ahí descuenta el stock real, emite el comprobante electrónico (boleta o factura) automáticamente ante SUNAT, y le envía un correo de confirmación al cliente.

7 **Después de la compra.** Desde "Mi cuenta → Mis pedidos" el cliente puede ver el historial completo, descargar su comprobante, y — una vez que el administrador registra el envío — ver el courier y el número de seguimiento de su paquete (recibe un correo automático la primera vez que esa información queda disponible). Si compró un producto, puede dejar una reseña. Si necesita contactar a la tienda, tiene un botón flotante de WhatsApp en toda la tienda, y en cada producto un enlace de WhatsApp específico para consultar por ese producto en particular. También puede consultar en cualquier momento la política de devoluciones, la política de privacidad y los términos y condiciones, enlazados desde el pie de página.

8 **Lista de deseos con avisos automáticos.** Si agrega un producto agotado a su lista de deseos, queda suscrito solo a ese producto para recibir un correo automático apenas vuelva a haber stock (y se cancela si lo quita de la lista). Si el producto ya en su lista baja de precio más adelante (nueva oferta, o el admin edita el precio), también recibe un aviso — nunca por el precio que ya vio al agregarlo, solo por bajadas reales desde ese momento.

9 **Si algo sale mal.** Si el cliente no completa el pago a tiempo, la reserva de stock se libera sola pasados 30 minutos (o 3 días si eligió contra entrega, ya que ahí no hay una pasarela que avise de inmediato) y el pedido queda cancelado automáticamente. Si ya pagó y necesita una devolución, el administrador puede reembolsarlo: el dinero se revierte de verdad en la pasarela que corresponda, se repone el stock si el producto se puede revender, y se emite automáticamente la nota de crédito ante SUNAT que corresponde a la boleta/factura original.

2.2 El recorrido del administrador

1 **Preparar la tienda.** El administrador carga el catálogo — a mano desde el panel, o de una sola vez subiendo un archivo Excel con todos los productos (con sus variantes) o todas las categorías/subcategorías. Si está reutilizando el sistema para un cliente nuevo, puede usar "Limpiar catálogo" para dejar la tienda vacía y lista para poblarla desde cero, sin tocar administradores, pedidos ni configuración.

2 **Configurar la tienda.** Define el impuesto y las tarifas de envío, el número de WhatsApp de atención, y — si la tienda va a usar analítica — sus IDs de Google Analytics 4 y Meta Pixel. También puede personalizar el texto de la política de devoluciones, la política de privacidad y los términos y condiciones (todos con un contenido de partida ya redactado, pensado como base a revisar, no como asesoría legal definitiva).

3 **Gestionar pedidos día a día.** Ve el listado completo de pedidos, filtra por estado o por comprobantes que fallaron al emitirse, cambia el estado de un pedido (procesando, enviado, entregado, cancelado), y registra el courier y número de seguimiento cuando lo envía (el cliente recibe un correo automático apenas se registra). Para pedidos contra entrega o pagos coordinados manualmente, confirma el cobro con un botón.

4 **Reembolsar cuando haga falta.** Si un cliente necesita una devolución, el administrador la procesa desde el detalle del pedido: escribe el motivo, decide si el producto vuelve al stock, y el sistema revierte el cargo real en la pasarela (Culqi o PayPal — Yape/Plin por QR todavía se reembolsa manualmente desde el panel de Culqi), notifica al cliente por correo, y genera la nota de crédito SUNAT automáticamente.

5 **Medir el negocio.** Desde Reportes puede ver ventas por período, el crecimiento de ventas día a día (con el porcentaje de variación contra el día anterior, pensado para seguimiento comercial), productos más vendidos, stock bajo (se actualiza solo), carritos abandonados (con el valor total en juego), y pedidos pendientes por antigüedad — todo exportable a CSV, Excel o PDF. Si configuró Google Analytics/Meta Pixel, también puede ver el comportamiento de los visitantes fuera del sistema, en esas mismas plataformas.

6 **Trabajar más rápido con varios productos a la vez.** En el listado de productos puede seleccionar varios con casillas y activar, desactivar o eliminar todos de una sola vez, en vez de repetir la acción producto por producto.

7 **Proteger su propia cuenta.** Desde "Mi seguridad" cada administrador puede activar verificación en dos pasos (2FA): además de su contraseña, se le pide un código de su app de autenticación (Google Authenticator, Authy, etc.) al iniciar sesión, con códigos de recuperación de un solo uso por si pierde el teléfono. Es opcional y la gestiona cada uno para su propia cuenta.

8 **Saber quién hizo qué.** La "Bitácora de actividad" deja un registro de las acciones sensibles del panel — reembolsos, cambios de estado de pedido, alta/edición/baja de administradores, y "Limpiar catálogo" — con quién la hizo y cuándo, filtrable por administrador o por tipo de acción.

3. Accesos

3.1 Tienda pública

URL	http://127.0.0.1:8000
Cliente de prueba	test@example.com / password
Registro	Cualquier visitante puede crear su propia cuenta desde "Crear cuenta"

3.2 Panel de administración

URL	http://127.0.0.1:8000/admin/login
Super administrador	admin@mitienda.test / password

El super administrador tiene acceso a todo. Existe también un rol **editor** con acceso restringido: Catálogo (categorías, productos, ofertas, cupones), Marketing (banners), Clientes y Reportes — **sin** acceso a Pedidos, Configuración de tienda, Perfiles de administrador ni a "Limpiar catálogo" (esta última reservada exclusivamente al super administrador por lo irreversible de la acción). Los roles se gestionan desde **Perfiles** en el panel.

La cuenta `admin@mitienda.test` tiene además un permiso individual (`manual.view`) que nunca se asigna a ningún rol y por lo tanto no lo hereda ningún super administrador nuevo que se cree después. Ese permiso es lo único que decide quién ve el enlace "Manual (PDF)" y puede abrir este documento. Por la misma razón, esta cuenta no aparece en el listado de **Perfiles** ni puede editarse o eliminarse desde ahí, ni siquiera por otro super administrador — se administra únicamente a sí misma desde **Mi cuenta → Mi perfil**, donde cualquier administrador (de cualquier rol) puede cambiar su propio nombre, correo y contraseña confirmando la contraseña actual.

3.3 Cómo levantar el servidor

Desde una terminal, dentro de la carpeta del proyecto:

`cd C:\xampp\htdocs\mi_tienda_v2`

`php artisan serve` — levanta la tienda en http://127.0.0.1:8000

`php artisan queue:work` — **en otra terminal, obligatorio**: sin esto, la confirmación de pedido, el aviso de contra entrega, la emisión de comprobantes y notas de crédito SUNAT, el aviso de envío, y el recordatorio de carrito abandonado se quedan encolados sin procesarse nunca (se guardan en la tabla `jobs` , no se pierden, pero tampoco se

envían hasta que corra este proceso).

`npm run dev` — (opcional, en otra terminal) recompila el CSS/JS automáticamente si vas a editar vistas. Para dejar los assets listos de forma definitiva: `npm run build`.

4. Arquitectura técnica

4.1 Stack

Componente	Tecnología
Framework	Laravel 12 (PHP 8.2)
Base de datos	MariaDB 10.4 (BD: `mi_tienda_v2`)
Frontend interactivo	Livewire 3 + Volt (componentes reactivos sin escribir una API aparte)
Estilos	Tailwind CSS, compilado con Vite; interactividad ligera puntual con Alpine.js
Autenticación	Laravel Breeze (clientes, en español) + guard separado `admin` a medida (administradores)
Roles y permisos	Spatie Laravel-Permission, granulares por módulo
Imágenes	Intervention Image (recorte manteniendo proporción, sin deformar)
Pagos - tarjeta/Yape (Culqi)	Culqi: cargo directo con tarjeta, Yape por teléfono+OTP, y Yape/Plin por QR (Órdenes de Culqi, confirmación asíncrona vía webhook/polling)
Pagos - Izipay	Pasarela independiente (motor Lyra/"Micuentaweb"), confirmación por firma HMAC-SHA256
Pagos - internacional	PayPal vía `srmklive/paypal` (API Orders v2)
Pagos - contra entrega	Sin pasarela: pedido queda pendiente hasta que el repartidor cobra y un admin lo confirma
Facturación electrónica	Nubefact (OSE) — emite boleta/factura y nota de crédito (en reembolsos), y las declara ante SUNAT
Analítica	Google Analytics 4 + Meta Pixel (opcionales, se activan solo si se configuran): eventos de ver producto, iniciar checkout y compra (una sola vez por pedido)
Contacto	WhatsApp click-to-chat (botón flotante, pie de página, y enlace por producto)
Zona horaria	`America/Lima` (configurable vía `APP_TIMEZONE`) — todas las fechas se guardan en hora local de Perú
Carga masiva / Reportes	`maatwebsite/excel` (import de catálogo y export de reportes), CSV nativo, PDF vía `barryvdh/laravel-dompdf`
Verificación en dos pasos	`pragmarx/google2fa` + `bacon/bacon-qr-code` (TOTP, QR generado localmente, sin llamar a ningún servicio externo)

Backups automáticos	`spatie/laravel-backup`: dump de la base de datos + `storage/app/public` (no todo el código, ya vive en git), programado a diario vía `routes/console.php`, con limpieza y monitoreo de salud; avisa por correo solo si algo falla
Integración continua	GitHub Actions: corre Pest, análisis estático (Larastan) y formato de código (Pint) en cada `push`/PR a `main`
Testing	Pest (sobre PHPUnit) — 382 tests automatizados en verde a la fecha de este manual, más 9 pruebas E2E con Playwright (navegador real)
Calidad de código	Larastan/PHPStan (nivel 5, con baseline para deuda preexistente) y Laravel Pint, ambos como gate en el CI

4.2 Patrón de la aplicación

Controladores delgados que delegan la lógica de negocio a **servicios** reutilizables, en vez de mezclar todo en el controlador (como hacía el legacy):

- `PricingService` — resuelve el precio efectivo de un producto respetando la prioridad oferta de producto > subcategoría > categoría > precio base.
- `CartService` — agrega/actualiza/quita productos del carrito, valida stock, fusiona el carrito de invitado al hacer login.
- `ShippingCalculator` / `TaxCalculator` — calculan envío e impuesto desde la configuración de tienda.
- `OrderService` — crea el pedido reservando stock (30 min en pagos electrónicos, 3 días en contra entrega), confirma el pago (descuenta stock definitivo, dispara la facturación electrónica), confirma pagos asíncronos de Culqi QR/Izipay de forma idempotente (nunca confía en lo que diga un webhook o el navegador sin re-verificar), procesa reembolsos totales (reversa el cargo real, repone stock, dispara la nota de crédito), y libera la reserva si el pedido expira sin pagarse.
- `NubefactService` — arma y envía el comprobante electrónico (y la nota de crédito, en reembolsos) a Nubefact; si falla, notifica por correo a los administradores con permiso sobre pedidos.
- `CulqiService` / `IzipayService` — encapsulan cada pasarela: cargos con tarjeta/Yape, creación y consulta de Órdenes (Yape/Plin QR), generación de token de pago y verificación de firma (Izipay), y reembolsos.
- `ProductImportService` / `CategoryImportService` — procesan la carga masiva desde Excel, fila por fila, aislando errores por producto/categoría sin tumbar el archivo completo.
- `CatalogResetService` — borra productos/categorías/atributos/ofertas de forma controlada para dejar la tienda lista antes de poblarla en un despliegue nuevo.
- `ImageUploadService` — redimensiona y guarda imágenes de forma consistente.
- `StockAlertService` — gestiona las suscripciones de "avísame cuando haya stock" por variante, y dispara el correo (una sola vez) cuando el stock vuelve a estar disponible.
- `WishlistNotifier` — avisa a quien tiene un producto en su lista de deseos si su precio efectivo bajó desde la última vez que se le avisó (nunca si sube).
- `AdminActivityLogger` — deja constancia en la bitácora de auditoría de las acciones sensibles del panel (quién, cuándo, sobre qué).
- `TwoFactorAuthService` — genera el secreto TOTP y el QR de activación, verifica códigos, y genera los códigos de recuperación de un solo uso.

Las partes interactivas (carrito, selector de variantes, formulario de productos con variantes dinámicas, wishlist, checkout) usan **Livewire** o **Alpine.js** para pequeños estados de UI (deshabilitar un botón al enviar, confirmaciones de texto, polling del estado de un pago QR); el resto son vistas Blade tradicionales con controladores.

Los **webhooks** (avisos automáticos de Culqi e Izipay cuando se confirma un pago) son el único tipo de ruta que no pasa por la protección CSRF habitual de Laravel — por ser peticiones servidor-a-servidor que no pueden llevar el token de la aplicación. Para compensar, el código nunca confía ciegamente en lo que el webhook dice: para Culqi, siempre re-consulta el estado real de la orden con la propia API; para Izipay, valida la firma HMAC-SHA256 antes de procesar cualquier dato.

4.3 Base de datos

38 tablas (incluyendo autenticación, roles/permisos y cola de trabajos de Laravel), todas con llaves foráneas reales (el legacy no tenía ninguna). Las más relevantes:

Tabla	Contenido
`categories`, `subcategories`, `products`	Taxonomía del catálogo
`attributes`, `attribute_values`, `product_variants`	Variantes de producto (Talla/Color) con stock real por combinación
`offers`	Ofertas polimórficas: aplican a categoría, subcategoría o producto
`carts`, `cart_items`	Carrito persistente en servidor (invitado o cliente)
`orders`, `order_items`, `payments`, `order_status_histories`	Pedidos con líneas (con snapshot de precio/título), pagos (incluye intentos fallidos y reembolsos) y auditoría de cambios de estado; los pedidos también guardan courier y número/enlace de seguimiento del envío
`invoices`	Comprobantes SUNAT por pedido: boleta, factura, o nota de crédito (en un reembolso) — un pedido puede tener como máximo un comprobante original y una nota de crédito
`coupons`	Cupones de descuento (porcentaje o monto fijo, con tope de usos y vigencia)
`reviews`, `wishlists`	Reseñas (solo compradores) y lista de deseos — cada fila de `wishlists` guarda además el último precio efectivo que se le avisó al cliente
`stock_alerts`	Suscripciones de "avísame cuando haya stock" por variante y correo (invitado o cliente registrado)
`banners`	Slides del carrusel de la página de inicio
`users` / `admins`	Clientes y administradores en tablas y guards de autenticación separados; `admins` también guarda el secreto y los códigos de recuperación de la verificación en dos pasos, si el admin la activó
`roles`, `permissions`	Roles granulares del panel (Spatie)
`admin_activity_logs`	Bitácora de auditoría: quién hizo qué acción sensible del panel y cuándo
`store_settings`	Impuesto, tarifas de envío, número de WhatsApp, textos de política de devoluciones/privacidad/términos, e IDs de Google Analytics 4/Meta Pixel (configuración única de la tienda)

5. Módulos construidos

5.1 Tienda pública

- Catálogo: home, categoría, subcategoría, ficha de producto, buscador, filtros y orden en el listado
- Selector de variantes (Talla/Color) con stock disponible en tiempo real
- Ofertas con precio tachado automático en toda la tienda
- Carrito persistente (sobrevive entre sesiones, se fusiona al iniciar sesión)
- Checkout con **seis métodos de pago**: tarjeta y Yape por OTP (Culqi), Yape/Plin por QR (Culqi, confirmación automática por webhook/polling), Izipay (tarjeta, pasarela independiente), PayPal (internacional), y pago contra entrega (tope de S/ 500, sin restricción geográfica); botón protegido contra doble clic
- Checkout de invitado: crea la cuenta de forma transparente con el nombre/correo del formulario
- Cupones de descuento aplicables en el checkout
- Facturación electrónica: boleta (DNI) o factura (RUC) emitida automáticamente al confirmarse el pago
- Reseñas (solo para quien compró el producto, y editable después: reenviar el formulario actualiza la reseña existente en vez de duplicarla) y lista de deseos — que además avisa por correo si el producto baja de precio, y suscribe automáticamente al aviso de restock si estaba agotado al agregarlo
- Aviso de "avísame cuando haya stock": en un producto/variante agotado, cualquier visitante deja su correo y recibe un aviso automático (una sola vez) apenas vuelve a haber stock
- Productos relacionados en la ficha de producto, priorizando primero los que tienen oferta activa y luego los más vendidos
- Imágenes con carga progresiva (efecto *skeleton* mientras cargan) y un ícono propio de "Sin imagen" (cámara, en el indigo de marca) cuando el producto no tiene foto, en vez de un ícono de error o de un placeholder externo
- El carrito y la ficha de producto limpian solos el aviso de stock insuficiente en cuanto se corrige la cantidad o la variante elegida, sin tener que reintentar la acción
- Link "Volver a la tienda" en las páginas de inicio de sesión y registro, e ícono de pestaña (favicon) con la marca propia
- Mi cuenta: historial de pedidos, detalle de cada uno, comprobante descargable, y seguimiento de envío (courier + número de rastreo) una vez que el administrador lo registra
- Botón flotante de WhatsApp, enlace en el pie de página, y enlace de consulta específico en cada producto (todo se oculta solo si la tienda no configuró un número)
- Páginas públicas de política de devoluciones, política de privacidad y términos y condiciones, enlazadas desde el pie de página y editables desde el panel
- Analítica de comportamiento (Google Analytics 4 y Meta Pixel), si la tienda las configura: registra ver producto, iniciar checkout y compra (esta última solo una vez por pedido, aunque el cliente recargue la página)
- Recordatorio de carrito abandonado por correo
- Toda la interfaz de cliente (login, registro, checkout, mensajes de validación) está en español — incluyendo los correos: la plantilla usa el indigo de marca en vez del genérico de Laravel, y ya no mezcla textos en inglés del framework ("Regards,", etc.)

5.2 Panel de administración

- Login propio con roles (super-admin / editor) y permisos granulares por sección; límite de intentos de inicio de sesión (5 por correo+IP) y cierre de sesión inmediato si el administrador es desactivado mientras tenía la sesión abierta
- **Verificación en dos pasos (2FA)**, opcional y gestionada por cada admin desde "Mi seguridad": código TOTP de una app de autenticación al iniciar sesión, códigos de recuperación de un solo uso, y exige la contraseña actual para desactivarla
- Dashboard con métricas (pedidos pendientes/pagados, ingresos del mes, stock bajo), cada tarjeta visible solo si el administrador tiene el permiso correspondiente
- CRUD de categorías y subcategorías (con imagen de cabecera)
- CRUD de productos: imagen principal, galería, y variantes dinámicas (stock/SKU/precio especial por combinación); si se sube foto solo a la galería y se deja la imagen principal vacía, se usa la primera de la galería automáticamente como

portada

- **Acciones masivas** en el listado de productos: activar, desactivar o eliminar varios productos seleccionados a la vez
- CRUD de ofertas (categoría, subcategoría o producto; porcentaje o precio fijo) y de cupones
- **Carga masiva por Excel** de productos (con variantes) y de categorías/subcategorías, con plantilla descargable, ejemplos ya llenos, y reporte de errores fila por fila sin tumbar el archivo completo
- **Limpiar catálogo** ("zona de peligro", solo super-admin): borra productos, variantes, categorías, subcategorías, atributos y ofertas para dejar la tienda lista antes de poblarla para un cliente nuevo, con confirmación escribiendo una frase exacta (no un simple diálogo del navegador)
- Gestión de pedidos: cambio de estado, confirmación manual de pago / cobro en efectivo (contra entrega), cancelación con liberación de stock, registro de courier y número/enlace de seguimiento (con aviso automático al cliente la primera vez), columna de método de pago y de estado del comprobante SUNAT
- Emisión y reintento manual del comprobante SUNAT desde el detalle del pedido; si la emisión automática falla, se notifica por correo a los administradores con acceso a pedidos
- **Reembolso real de pedidos**: revierte el cargo en la pasarela (Culqi o PayPal), repone stock si corresponde (casilla marcada por defecto), notifica al cliente, y emite automáticamente la nota de crédito SUNAT referenciando la boleta/factura original (Yape/Plin por QR se reembolsa manualmente desde el panel de Culqi por ahora)
- Configuración de tienda, organizada en tarjetas por tema para no tener que hacer scroll excesivo (General, Contacto y analítica, Políticas legales en acordeones): impuesto, tarifas de envío, número de WhatsApp, textos de las políticas (devoluciones/privacidad/términos), e IDs de Google Analytics 4/Meta Pixel
- Gestión de administradores y roles (Perfiles); la cuenta del desarrollador (permiso `manual.view`, nunca asignado por rol) no aparece en este listado ni puede editarse o eliminarse desde aquí
- **Mi perfil** (en "Mi cuenta"): cualquier administrador, sin importar su rol, cambia su propio nombre, correo y contraseña ahí, confirmando la contraseña actual — es la única forma de editar la cuenta del desarrollador
- **Bitácora de actividad**: quién hizo qué y cuándo en las acciones sensibles del panel (reembolsos, cambios de estado de pedido, confirmaciones manuales, alta/edición/baja de administradores, "Limpiar catálogo"), filtrable por administrador o por tipo de acción, y exportable a CSV o Excel
- Reportes: ventas por período (CSV/Excel/PDF), **crecimiento diario** (líneas de producto vendidas y % de crecimiento vs. el día calendario anterior, con export), productos más vendidos y **stock bajo** (se auto-actualiza solo cada 15 segundos y filtra en vivo por umbral, sin recargar la página), **carritos abandonados** (clientes registrados sin actividad hace más de N horas, con el valor total en juego y si ya se envió el recordatorio), y pedidos pendientes por antigüedad

6. Seguridad

6.1 Correcciones frente al sistema anterior

Legacy	Ahora
Salt de contraseña compartido y hardcodeado	Bcrypt con salt único por usuario (Laravel `Hash`)
Recuperar contraseña la reenviaba en texto plano por correo	Link de reseteo firmado y con expiración
Token de verificación de email = `md5(email)`, sin expirar	Link de verificación firmado y expirable
Sin llaves foráneas en la base de datos	FKs reales en las 38 tablas

Credenciales de PayPal/PayU en la base de datos, en texto plano	Solo en `.env`, nunca en BD (igual para Culqi, Izipay y Nubefact)
Roles como texto libre (`if ($perfil == "administrador")`)	Roles y permisos granulares (Spatie), incluyendo un permiso aparte para acciones irreversibles ("Limpiar catálogo")
Carrito solo en `localStorage`	Carrito persistente en servidor, con validación de stock real
Precio/stock/cupón confiados al navegador en el checkout	Todo se recalcula server-side antes de crear el pedido
No existía el concepto de webhook	Los avisos de Culqi/Izipay nunca se procesan "a ciegas": siempre se re-verifican (re-consulta a la API o validación de firma HMAC) antes de tocar un pedido

6.2 Auditoría de seguridad exhaustiva

Además de la comparación con el legacy, el proyecto ya en marcha pasó por una auditoría propia, archivo por archivo, agrupada por dimensión (autenticación, IDOR, inyección, pagos/webhooks, archivos, XSS, CSRF/carreras, datos sensibles, panel admin), donde cada hallazgo se verificó de forma adversarial antes de aplicarlo, para descartar falsos positivos. Esa auditoría corrigió 20 hallazgos; junto con una corrección aparte de la misma naturaleza (formularios sin `method="POST"`), estos son los más relevantes:

Severidad	Hallazgo	Corrección
Crítico	Un pedido caro propio podía marcarse como pagado reutilizando (replay) la confirmación firmada de un pedido barato propio en Izipay	Se compara el `order_number` y el monto de la respuesta firmada contra el pedido real de la ruta antes de aceptarla
Alto	Condición de carrera permitía ejecutar `markAsPaid()` / `refund()` dos veces sobre el mismo pedido (doble descuento de stock, doble comprobante/nota de crédito)	Se bloquea la fila del pedido y se re-valida su estado dentro de la misma transacción
Alto	XSS en el checkout: un valor dinámico se interpolaba sin escapar dentro de un atributo `x-data` de Alpine.js	Se usa `@js()` para cualquier valor dentro de contexto JS de Alpine
Alto	Sin límite de intentos en `/admin/login`	Bloqueo de 5 intentos por correo+IP, igual que ya tenía el login de clientes
Alto	Desactivar un administrador no revocaba su sesión ni su cookie "recordarme" ya activa	Middleware que revisa el estado `activo` en cada petición, no solo al iniciar sesión
Alto	Ningún formulario Livewire declaraba `method`: si el JS de Livewire no llegaba a cargar, el navegador caía a un submit nativo por GET, exponiendo la contraseña en la URL	`method="POST"` explícito como respaldo en los 10 formularios Livewire
Medio	Enumeración de usuarios en "olvidé mi contraseña" (la respuesta variaba según si el correo existía)	Mismo mensaje de éxito exista o no la cuenta, salvo por límite de intentos
Medio	Sin límite de peticiones en webhooks/checkout/buscador; el webhook de Culqi logueaba el payload completo	Rate limiting agregado; el log ahora solo registra metadatos, no el cuerpo crudo
Medio	Condición de carrera en el tope de usos de un cupón	Incremento atómico condicionado al tope, en vez de leer-y-luego-escribir

Medio	El dashboard y el manual del panel admin no respetaban permisos granulares (el rol editor veía datos fuera de su alcance documentado)	Cada sección del dashboard y la ruta del manual ahora exigen el permiso correspondiente
Bajo	Se podía reasignar una variante de otro producto manipulando su id (IDOR)	La búsqueda de la variante se restringe siempre al producto actual
Bajo	Importación de Excel sin límite de tamaño de archivo	Límite de tamaño agregado a la validación de subida
Bajo	Cookie de sesión sin `Secure` forzado en producción	`Secure` activado automáticamente cuando `APP_ENV=production`

Aparte de la auditoría, dos correcciones más de esta misma naturaleza: los enlaces de navegación y los assets (CSS/JS/imágenes) usaban rutas absolutas armadas con el host interno, rompiéndose al compartir la tienda por un túnel HTTPS (devtunnels/ngrok) — se corrige generando rutas relativas y configurando `trustProxies` para reconocer el dominio público real vía cabeceras `X-Forwarded-*`.

6.3 Verificación en dos pasos para administradores

Cada admin puede activar 2FA (TOTP) por su cuenta desde "Mi seguridad" — no es obligatoria para todos, cada uno decide si la activa. El QR se genera de forma local (`pragmarx/google2fa` + `bacon/bacon-qr-code`), sin llamar a ningún servicio externo que expondría el secreto por la red. Al iniciar sesión con 2FA activado, la contraseña correcta **no** deja entrar directamente: el admin queda en un estado intermedio (sin sesión de guard todavía) hasta que confirma un código de su app o uno de sus códigos de recuperación de un solo uso, con el mismo límite de 5 intentos por admin+IP que ya protege el login. Desactivarla exige volver a confirmar la contraseña actual.

6.4 Trazabilidad de acciones sensibles

La bitácora de actividad (`admin_activity_logs`) registra automáticamente quién hizo qué en las acciones más sensibles del panel: reembolsos (con el motivo y si repuso stock), cambios de estado de pedido, confirmaciones manuales de pago, alta/edición/baja de administradores (incluyendo cambios de rol y activar/desactivar), acciones masivas sobre productos, y "Limpiar catálogo". Antes de esto, esas acciones no dejaban ningún rastro de quién las había hecho.

6.5 Segunda auditoría de seguridad

Con el proyecto ya más avanzado (autoservicio de "Mi perfil", exportación de la bitácora, protección de la cuenta del desarrollador), una segunda auditoría — 6 dimensiones en paralelo (autenticación, inyección, pagos/webhooks, archivos, CSRF/configuración, datos sensibles) — confirmó que las correcciones de la auditoría anterior seguían vigentes y buscó superficie nueva. Detalle completo, incluyendo lo evaluado y descartado por no aplicar (un índice único en `payments` que habría roto un flujo legítimo de Culqi QR), en `docs/auditoria-seguridad.md` . Hallazgos corregidos:

Severidad	Hallazgo	Corrección
Alto	Inyección de fórmulas (CSV/Excel) en la exportación de la bitácora: cualquier admin podía sembrar el payload cambiando su propio nombre en "Mi perfil", sin permisos especiales	Se neutraliza cualquier celda que empiece con `=+-@` antes de exportar

Alto	Host Header Injection: sin `trustHosts()`, un Host falso podía envenenar el enlace de "olvidé mi contraseña"	`trustHosts()` (Laravel ya lo desactiva solo en entorno local y en tests, sin afectar el desarrollo)
Alto	Sin límite de intentos sobre `current_password` en Mi perfil, 2FA, cambio de contraseña y borrado de cuenta: una sesión comprometida podía adivinarla por fuerza bruta y tomar la cuenta de forma permanente	Bloqueo de 5 intentos por cuenta, mismo patrón que el login
Medio-Alto	Sin límite de intentos en el registro de cuentas (creación masiva + spam de verificación de email hacia terceros)	Límite de 5 registros por minuto por IP
Medio-Alto	Sin límite de pedidos contra entrega pendientes por cliente (congelamiento de inventario / fraude de "no-show")	Máximo 2 pedidos contra entrega pendientes por cliente
Medio	Livewire no revalidaba el permiso ni el estado activo en las peticiones AJAX del formulario de productos: revocarle el permiso o desactivar a un admin a mitad de sesión no le impedía seguir guardando cambios	Chequeo explícito en cada acción que modifica datos, no solo al cargar la página
Medio	Sin headers de seguridad HTTP (`X-Frame-Options`, etc.)	Middleware global; el `Content-Security-Policy` queda pendiente de calibrar contra los widgets de Culqi/Izipay/GA4/Meta Pixel antes de activarlo
Medio	Pagos síncronos (Culqi tarjeta/Yape) sin timeout ni manejo de fallo de conexión: una llamada colgada terminaba en un error sin capturar, sin saber si el cargo sí se había ejecutado	Timeout explícito y mensaje cauteloso ("si se realizó un cargo, contáctanos") en vez de un error genérico
Medio	El backup podía correr sin cifrar si faltaba `BACKUP_ARCHIVE_PASSWORD` en producción	Se salta el backup en vez de correrlo inseguro; `backup:monitor` avisa por correo que falta uno reciente
Bajo	Un super-admin podía cambiar su propio rol/estado desde "Perfiles"	Bloqueado ahí; solo puede hacerlo desde "Mi perfil"

7. Pendientes para un uso en producción real

Checklist completa de lanzamiento, paso a paso, en `docs/checklist-lanzamiento.md`. Resumen de lo más importante:

- **Credenciales reales de las pasarelas/facturación**: hoy `PAYPAL_SANDBOX_CLIENT_ID`, `CULQI_SECRET_KEY`, `IZIPAY_USERNAME` / `IZIPAY_PASSWORD` y `NUBEFACT_TOKEN` están vacíos (o ni existen) en `.env`. Sin esto, el pago con PayPal/Culqi/Izipay y la emisión real de comprobantes no se pueden completar de punta a punta (se puede probar todo el flujo hasta ese paso, o usar "Confirmar pago manual"/"Confirmar cobro en efectivo" desde el panel para simularlo). El checklist detallado de lo que falta específicamente para Yape/Plin por QR e Izipay está en `docs/produccion-pagos-pendiente.md`.
- **Cola de trabajos**: para que el pedido pendiente se cancele automáticamente pasados 30 minutos (o 3 días en contra entrega) sin pago, y para que se envíen correos y se emitan comprobantes, debe correr `php artisan queue:work`

en segundo plano de forma permanente (con Supervisor o similar en el servidor real).

- **Correo real**: actualmente los correos (confirmación de pedido, verificación de email, aviso de comprobante fallido, aviso de envío) se registran en log/array, no se envían de verdad. Falta configurar un proveedor SMTP real en `.env`.
- **Webhooks reales**: Culqi e Izipay necesitan una URL pública HTTPS para avisar los pagos (no pueden avisar a `localhost`). Para probar antes de desplegar, se puede usar un túnel como `ngrok`.
- **`APP_DEBUG=false`** en el `.env` del servidor real — hoy está en `true` para desarrollo, y expone detalles internos si algo falla.
- **Dominio y HTTPS** para el despliegue final (hoy corre en `127.0.0.1:8000`).
- **`TRUSTED_PROXIES`**: hoy vale `*` (confía en cualquier proxy), razonable para desarrollo detrás de un túnel (devtunnels/ngrok) pero no para producción — ahí hay que restringirlo a la IP real del balanceador/reverse proxy delante de la aplicación.
- **`BACKUP_ARCHIVE_PASSWORD`**: obligatoria en producción — sin ella, `backup:run` se salta solo (no corre sin cifrar) y `backup:monitor` termina avisando por correo que no hay un backup reciente.

8. Calidad de código y automatización

Además de la suite de tests, el CI corre dos herramientas de calidad en cada `push`/PR a `main`:

- **Larastan/PHPStan (nivel 5)**: detecta tipos incorrectos, propiedades/métodos inexistentes y otros errores que los tests no necesariamente cubren. Usa `laravel-ide-helper` (vía `@mixin`, sin tocar la lógica real de los modelos) para poder resolver relaciones y columnas de Eloquent. Los ~68 hallazgos preexistentes a la fecha de este setup —casi todos la misma limitación conocida de PHPStan al perder el tipo genérico en relaciones de Eloquent encadenadas con `->with()`/`->get()`, no bugs reales— quedaron congelados en `phpstan-baseline.neon`: el código nuevo no puede sumar hallazgos nuevos sin arreglar de golpe esa deuda preexistente no relacionada.
- **Laravel Pint**: formato de código automático (estaba como dependencia desde el inicio, pero nunca se había corrido ni exigido en el CI).

Aparte del CI, hay una **suite E2E con Playwright** (`e2e/`, se corre con `npm run test:e2e`) que maneja un navegador real contra la app corriendo de verdad — a diferencia de Pest, que solo prueba a nivel HTTP, esta sí ejecuta JS/Livewire/Alpine tal como lo haría un cliente real. Cubre: catálogo y buscador, login de cliente (válido e inválido) a través del formulario Livewire real, agregar al carrito con la actualización reactiva del contador del header (o el aviso de stock si el producto está agotado), selección masiva de productos en el panel admin, y el flujo completo de 2FA —activar, cerrar sesión, volver a entrar pasando por el desafío con un código TOTP real, y desactivarlo de nuevo—. No corre en GitHub Actions (para no alentar cada push); usa la base de datos local y cuentas ya sembradas, con acciones pensadas para no ensuciar el catálogo de desarrollo.

9. Checklist de pruebas

9.1 Tienda pública

- ☐ El home carga y muestra categorías y productos destacados
- ☐ Navegar categoría → subcategoría → producto
- ☐ El buscador devuelve resultados relevantes
- ☐ Registro de una cuenta nueva (en español)
- ☐ Iniciar y cerrar sesión
- ☐ Agregar un producto con variantes (elegir Talla/Color) al carrito
- ☐ Ver el carrito, cambiar cantidad, quitar un producto
- ☐ Aplicar un cupón de descuento válido en el checkout
- ☐ Completar el checkout con tarjeta/Yape (Culqi), Izipay o PayPal: totales correctos (subtotal + envío + impuesto)
- ☐ Elegir Yape/Plin (QR) en un pedido dentro de S/ 6–500, y confirmar que la opción se oculta fuera de ese rango
- ☐ Completar el checkout con **pago contra entrega** en un carrito ≤ S/ 500, y confirmar que no aparece esa opción si el carrito supera S/ 500
- ☐ Confirmar que el pedido queda "Pendiente" y reserva el stock (no lo descuenta aún)
- ☐ Ver el pedido en "Mi cuenta → Mis pedidos" y descargar el comprobante una vez pagado
- ☐ Una vez que el admin registra courier/tracking, verlo en el detalle del pedido
- ☐ Dejar una reseña en un producto ya comprado
- ☐ Agregar y quitar un producto de la lista de deseos
- ☐ Agregar a la lista de deseos un producto agotado y confirmar que queda suscrito al aviso de restock; quitarlo y confirmar que se cancela
- ☐ En un producto agotado, dejar el correo en "avísame cuando haya stock" y confirmar que se registra la suscripción
- ☐ Ver "Productos relacionados" al final de una ficha de producto
- ☐ Ver el botón de WhatsApp flotante, el enlace del pie de página, y el enlace específico en una ficha de producto
- ☐ Abrir las páginas de política de devoluciones, privacidad y términos desde el pie de página

9.2 Panel de administración

- ☐ Iniciar sesión como super administrador
- ☐ El dashboard muestra las métricas correctamente
- ☐ Crear, editar y eliminar una categoría (y verificar que bloquea el borrado si tiene subcategorías/productos)
- ☐ Crear un producto con variantes (Talla/Color) e imágenes
- ☐ Descargar la plantilla de importación de productos, llenarla (un producto simple y uno con variantes) y subirla; confirmar que se crean correctamente y que una fila con error no bloquea el resto
- ☐ Descargar la plantilla de importación de categorías/subcategorías y subirla; confirmar que no duplica una categoría que ya existía
- ☐ Crear una oferta y verificar que el precio tachado aparece en la tienda pública

- ☐ Crear un cupón y aplicarlo en un pedido de prueba
- ☐ Ver el listado de pedidos, filtrar por estado y por "solo con comprobante fallido"
- ☐ Confirmar manualmente el pago de un pedido pendiente, y "Confirmar cobro en efectivo" en uno contra entrega
- ☐ Registrar courier y número de seguimiento en un pedido pagado
- ☐ Reembolsar un pedido pagado: confirmar que repone stock (si corresponde), notifica al cliente, y genera la nota de crédito
- ☐ Emitir/reintentar el comprobante SUNAT de un pedido pagado y descargar el PDF/XML
- ☐ Cancelar un pedido pendiente y confirmar que libera el stock
- ☐ Cambiar la configuración de impuesto/envío/WhatsApp/políticas/analítica de la tienda
- ☐ Crear un administrador con rol "editor" y confirmar que NO accede a Pedidos/Configuración/Perfiles/"Limpiar catálogo" (debe dar 403) pero SÍ a Categorías/Productos/Ofertas/Cupones/Banners/Clientes/Reportes
- ☐ Generar el reporte de ventas y el de crecimiento diario, y exportar ambos en CSV, Excel y PDF
- ☐ Revisar el reporte de productos más vendidos y de stock bajo
- ☐ Revisar el reporte de pedidos pendientes por antigüedad y el de carritos abandonados
- ☐ Seleccionar varios productos con las casillas y activar/desactivar/eliminar en masa
- ☐ Activar 2FA desde "Mi seguridad", cerrar sesión, y volver a entrar confirmando el código de la app; probar también un código de recuperación
- ☐ Revisar la "Bitácora de actividad" después de un reembolso o de editar un administrador, confirmar que quedó registrado, y probar la exportación a CSV/Excel
- ☐ Desde "Mi cuenta → Mi perfil", cambiar el propio nombre/correo/contraseña confirmando la contraseña actual
- ☐ Crear un segundo super administrador y confirmar que NO ve la cuenta del desarrollador en "Perfiles" ni puede editarla/eliminarla (ni por URL directa)
- ☐ (Solo super-admin) Entrar a "Limpiar catálogo", confirmar que el botón sigue deshabilitado hasta escribir la frase exacta, y que tras confirmarlo el catálogo queda vacío sin afectar admins/pedidos/configuración
- ☐ Escribir mal la contraseña actual 6 veces seguidas en "Mi perfil" y confirmar que la 6ª ya no deja intentar, ni con la contraseña correcta
- ☐ Crear 3 pedidos contra entrega seguidos con la misma cuenta y confirmar que el 3ro se rechaza (tope de 2 pendientes)

10. Ubicación de archivos clave

Qué	Dónde
Modelos	`app/Models/`
Lógica de negocio (servicios)	`app/Services/` (pasarelas de pago en `app/Services/Payments/`)
Carga masiva desde Excel	`app/Imports/` (lectura) + `app/Exports/` (plantillas y reportes)
Controladores de tienda pública	`app/Http/Controllers/Catalog/`, `CheckoutController.php`, `Checkout/`
Webhooks de pasarelas	`app/Http/Controllers/Webhooks/`

Controladores del panel admin	`app/Http/Controllers/Admin/`
Componentes interactivos (Livewire)	`app/Livewire/`
Notificaciones por correo	`app/Notifications/` (confirmación de pedido, contra entrega, reembolso, envío, falla de comprobante, carrito abandonado, restock, bajada de precio) — plantilla con marca propia en `resources/views/vendor/mail/`
Jobs en cola	`app/Jobs/` (emisión de comprobante/nota de crédito, liberación de reserva, recordatorio de carrito, aviso de restock)
Vistas de la tienda	`resources/views/catalog/`, `checkout/`, `account/`, `policies/`
Vistas del panel admin	`resources/views/admin/` (bitácora en `admin/activity-log/`, seguridad/2FA en `admin/security/`)
Traducciones al español	`lang/es/` y `lang/es.json`
Rutas	`routes/web.php` (tienda, incluye webhooks), `routes/admin.php` (panel)
Migraciones (esquema de BD)	`database/migrations/`
Tests automatizados	`tests/Feature/` (382 tests Pest en verde a la fecha de este manual) + `e2e/` (9 pruebas Playwright)
Integración continua	`.github/workflows/tests.yml` (Pest, Larastan y Pint)
Análisis estático	`phpstan.neon` (config) + `phpstan-baseline.neon` (deuda congelada) + `_ide_helper_models.php` (docblocks de modelos)
Checklist de lanzamiento a producción	`docs/checklist-lanzamiento.md` (credenciales, pasarelas, dominio/HTTPS, correo, backups...)
Checklist de credenciales pendientes (Culqi QR / Izipay)	`docs/produccion-pagos-pendiente.md`
Análisis del sistema legacy	`mi_tienda/_analisis-legacy/` (esquema de BD y reglas de negocio del sistema anterior)