Análisis técnico de Mi Tienda v2

Arquitectura, dependencias, entradas, rutas y riesgos · 27 de julio de 2026

Resumen ejecutivo. Mi Tienda v2 es una aplicación monolítica de comercio electrónico construida con Laravel 12, PHP 8.2, Blade, Livewire y Volt. Incluye catálogo, carrito, checkout para invitados, pagos con Culqi, Izipay y PayPal, facturación mediante Nubefact, panel administrativo, permisos, reportes, respaldos, autenticación de dos factores y pruebas automatizadas.

1. Alcance de la revisión

La revisión original fue de solo lectura. No se inspeccionaron valores privados de `.env`, no se ejecutaron migraciones, comandos Artisan, pruebas funcionales ni operaciones sobre la base de datos.

Se validó el manifiesto Composer y la sintaxis de los archivos PHP. El manifiesto es válido y los archivos examinados no presentaron errores de sintaxis.

2. Tecnologías principales

Capa	Tecnología	Versión observada
Backend	PHP	Requisito ^8.2; entorno 8.2.12
Framework	Laravel	12.64.0
Interfaz dinámica	Livewire / Volt	3.8.2 / 1.11.1
Plantillas	Blade	Incluido con Laravel
CSS	Tailwind CSS	3.4.19
Construcción frontend	Vite	6.4.3
Pruebas PHP	Pest / PHPUnit	3.8.7 / 11.5.56
Pruebas E2E	Playwright	1.62.0
Análisis estático	Larastan	3.10.0, nivel 5

3. Estructura del proyecto

Directorio	Responsabilidad
`app/Http/Controllers`	44 controladores para tienda, cuenta, checkout, webhooks y administración.
`app/Models`	28 modelos Eloquent: productos, variantes, usuarios, pedidos, pagos, comprobantes y otros.
`app/Services`	20 servicios para carrito, pedidos, precios, envíos, importaciones, pagos y facturación.
`app/Livewire`	Carrito, wishlist, formularios de producto y componentes administrativos.

`app/Jobs`	Facturación, notas de crédito, reservas, recordatorios y alertas de stock.
`resources/views`	Vistas Blade de tienda, checkout, cuenta, autenticación y panel administrativo.
`database`	49 migraciones, factories, seeders y datos de ubigeo peruano.
`tests`	76 archivos de pruebas unitarias y funcionales.
`e2e`	Pruebas de navegador que utilizan una aplicación y base local reales.
`docs`	Documentación de seguridad, lanzamiento, pagos y diferenciadores locales.

4. Módulos funcionales

- **Catálogo:** categorías, subcategorías, productos, variantes, atributos, imágenes, ofertas y búsqueda.
- **Compra:** carrito, cupones, cálculo de precios, checkout invitado, stock y reserva de pedidos.
- **Pagos:** contraentrega, Culqi tarjeta/Yape/QR, Izipay y PayPal.
- **Postventa:** seguimiento, cancelaciones, reembolsos, comprobantes y notas de crédito.
- **Cliente:** registro, verificación, pedidos, wishlist, reseñas y alertas de reposición.
- **Administración:** catálogo, clientes, pedidos, marketing, reportes, configuración y bitácora.
- **Perú:** DNI/RUC, ubigeo, envío por zona, Libro de Reclamaciones y Nubefact.
- **Seguridad:** guard administrativo, roles/permisos, 2FA, límites de tasa y encabezados HTTP.

5. Dependencias relevantes

Paquete	Versión	Uso
spatie/laravel-permission	6.25.0	Roles y permisos administrativos.
spatie/laravel-backup	9.3.6	Respaldo de base de datos y archivos.
barryvdh/laravel-dompdf	3.1.2	Generación de documentos PDF.
maatwebsite/excel	3.1.69	Importaciones y exportaciones Excel.
srmklive/paypal	3.1.1	Integración con PayPal.
intervention/image-laravel	1.5.9	Procesamiento de imágenes.
pragmarx/google2fa	9.0.0	Autenticación administrativa en dos pasos.
bacon/bacon-qr-code	3.1.1	Generación de códigos QR.
laravel/socialite	5.29.0	Base para autenticación OAuth.

6. Configuración y entradas

Puntos de entrada

- `public/index.php`: entrada de todas las solicitudes HTTP.
- `artisan`: entrada de consola y tareas operativas.
- `bootstrap/app.php`: registro de rutas, middleware, proxies, CSRF y endpoint de salud `/up`.
- `resources/js/app.js` y `resources/css/app.css`: entradas del frontend compiladas con Vite.

Configuración principal

`composer.json`, `package.json`, `vite.config.js`, `tailwind.config.js`, `phpunit.xml`, `phpstan.neon`, `playwright.config.js`, `.env.example` y el directorio `config/`.

7. Mapa de rutas

Archivo	Áreas cubiertas
`routes/web.php`	Catálogo, carrito, checkout, cuenta, políticas, reclamos, APIs locales, pagos y webhooks.
`routes/auth.php`	Registro, acceso, recuperación de contraseña y verificación de correo.
`routes/admin.php`	Panel administrativo protegido por guard, estado de cuenta y permisos.
`routes/console.php`	Backups, monitorización y limpieza de la bitácora.

8. Riesgos y observaciones

1. **Alto — Confianza global en proxies.** `TRUSTED_PROXIES` admite `*` por defecto. En producción debe restringirse y configurarse de forma compatible con la caché de configuración.

2. **Alto — Pruebas E2E sobre datos locales reales.** Playwright usa la aplicación y base de desarrollo. Puede registrar usuarios, crear pedidos o cambiar datos; debe ejecutarse en una base desechable.
3. **Medio — Sin Content Security Policy.** Existen encabezados de seguridad básicos, pero no CSP debido a widgets de pagos y analítica.
4. **Medio — Deuda de análisis estático.** El baseline de PHPStan contiene alrededor de 90 marcadores ignorados que deberían auditarse periódicamente.
5. **Medio — Manual versionado.** El PDF del manual incluye credenciales de ejemplo y detalles internos; debe verificarse que no contenga secretos reales.
6. **Medio — Dependencias Tailwind mezcladas.** Se usa Tailwind 3 mediante PostCSS, pero está instalado el plugin Vite de Tailwind 4 sin registrarlo.
7. **Medio — Estabilidad Composer.** `minimum-stability` permite paquetes `dev`, aunque `prefer-stable` reduce el riesgo.
8. **Medio — Valores de ejemplo para desarrollo.** Antes de producción deben comprobarse modo, depuración, correo, pagos, cookies, backups y proxies.
9. **Operativo — Workers y cron.** Colas, facturación, avisos y respaldos dependen de procesos externos correctamente supervisados.
10. **Operativo — Sesiones.** El ejemplo configura sesiones en base de datos sin cifrado del contenido almacenado.

> **Recomendación prioritaria:** antes de desplegar, utilizar una base exclusiva para pruebas, restringir proxies, revisar secretos y valores de producción, confirmar workers/cron y ejecutar la suite automatizada junto con Larastan en un entorno aislado.

9. Estado general

La solución tiene una cobertura funcional amplia y una organización coherente por controladores, servicios, modelos y tareas. Las integraciones sensibles muestran controles específicos: autorización de pedidos, verificación de firma de Izipay y confirmación de Culqi consultando su API. Los principales riesgos están relacionados con configuración de producción, aislamiento de pruebas, deuda estática y operación continua.
