# Lighthouse 100/100/100/100 en WooCommerce: Optimización Extrema en Tiendas Híbridas

**Estantería:** Desarrollo y Arquitectura

**Subtema:** Rendimiento

**mercedev.es** — 2026-06-18 | Epic 9 - Fase 4

# Lighthouse 100/100/100/100 en WooCommerce: Optimización Extrema en Tiendas Híbridas

## El Desafío (Síntoma)

Al integrar la tienda en la arquitectura híbrida headless ( `/blog/tienda/` ), los análisis iniciales de auditoría arrojaron un rendimiento subóptimo en varias métricas esenciales: - **SEO (66/100):** WordPress inyectaba directivas `noindex, nofollow` heredadas de las configuraciones de exclusión del núcleo estático. - **Rendimiento (86/100):** Carga inicial de imágenes sobredimensionadas (resoluciones originales de 2048px en lugar de adaptadas) y retraso por recursos de estilo CSS bloqueantes. - **Buenas Prácticas (96/100):** Errores persistentes en la consola de JavaScript (JS) debido al encolado redundante de librerías asociadas a bloques interactivos de WooCommerce, que no se utilizaban en la maquetación clásica de la tienda.

Para afinar de forma quirúrgica estas métricas, se requería una aproximación basada en datos que aislara cada cambio. De ahí nació la exportación local de las respuestas JSON de Lighthouse ( `tienda-audit-*.json` ), utilizadas para comparar de forma precisa el impacto de cada ajuste sin ensuciar el control de versiones de producción.

## La Maniobra (Lógica)

La optimización quirúrgica se estructuró en tres frentes de acción:

### 1. Desencolado Selectivo de Scripts de Bloques (Buenas Prácticas: 96 → 100)

Las versiones modernas de WooCommerce encolan por defecto scripts interactivos de bloques ( `wc-cart-block` , `wc-checkout-block` , `wc-blocks` , `wc-settings` ). En un frontend clásico minimalista, esto provoca errores de consola críticos. Se inyectó un gancho de control (*hook*) en `functions.php` con prioridad 100 para desencolarlos completamente, indicando explícitamente el uso de carrito clásico mediante el filtro `woocommerce_blocks_has_classic_checkout/cart` .

### 2. Priorización de Carga e Imágenes Responsivas (Rendimiento: 86 → 100)

- **Eliminación del bloqueo de renderizado:** Se inyectaron directivas `<link rel="preload">` del archivo principal `main.css` antes de su carga como stylesheet en `woocommerce.php` .
- **Imágenes a escala:** Se modificó el orquestador Python ( `merci-shop.py` ) para que las URLs subidas a la API REST de WooCommerce apunten automáticamente a las versiones reducidas de tamaño y optimizadas en WebP ( `-400w.webp` ) generadas previamente por `merci-optimizer.py` .

### 3. Fuerza en la Directiva de Indexación (SEO: 66 → 100)

Para contrarrestar la limitación del núcleo estático y forzar la visibilidad ante los motores de búsqueda, se amplió el filtro de metadatos `wp_robots` en `functions.php` . Se le asignó una prioridad máxima (9999) para eliminar de forma inequívoca las directivas restrictivas ( `noindex, nofollow` ) e inyectar de forma imperativa `index, follow` en las páginas comerciales de la tienda.

## El Aprendizaje / Deuda Técnica

Lograr puntuaciones perfectas de 100/100 en entornos dinámicos exige erradicar el comportamiento predeterminado de los plugins masivos. El uso de reportes JSON de Lighthouse guardados en local demostró ser un método científico

infalible para auditar y resolver problemas de rendimiento, evitando la tentación de realizar cambios a ciegas.

*Los archivos detallados del diagnóstico (`tienda-audit-*.json`) se conservan localmente en `observabilidad/audits/` para auditoría técnica sin aumentar la deuda de almacenamiento del repositorio en la nube.*