Compendio Estratégico: E-commerce Híbrido Extremo (Zero-JS)

Estantería: DevSecOps e Infraestructura

Subtema: Gobernanza

mercedev.es — 2026-05-27 | Epic 6 - Fase 1

El Desafío (Síntoma)

Integrar un motor de comercio electrónico pesado como WooCommerce en una arquitectura orientada al rendimiento extremo (100/100 Core Web Vitals) parecía una contradicción. Las versiones modernas de este CMS (Content Management System) inyectan por defecto megabytes de librerías React, bloques Gutenberg, telemetría intrusiva (`sourcebuster.min.js`) y pesados scripts AJAX (`wc-cart-fragments`).

Todo este código JavaScript secuestraba el hilo principal del navegador (Main Thread), empujando el Tiempo de Bloqueo Total (TBT) por encima de los 645ms y destruyendo la experiencia de usuario en dispositivos móviles.

La Maniobra (Lógica)

La Épica 6 abordó el reto mediante la filosofía de "Tienda No Tienda", aplicando dos maniobras de aislamiento extremo:

1. **Catálogo Headless (SSOT Bidireccional):** Se desarrolló el agente `merci-shop.py` . En lugar de utilizar el panel de administración, los productos se redactan en local utilizando Markdown y metadatos YAML. El agente orquesta la inyección masiva de este catálogo directamente hacia la API REST de WooCommerce, resolviendo los IDs dinámicamente mediante *slugs* para evitar Zombis o colisiones de datos.
2. **Carrito Zero-JS y Escudos PHP:** Se desencolaron y desregistraron de forma absoluta todas las dependencias JS de WooCommerce en el *Child Theme*. Para recuperar la funcionalidad de compra, se inyectó una rutina de auto-sanación en la base de datos que convierte los modernos bloques Gutenberg en shortcodes clásicos (`[woocommerce_cart]`). Por último, se apagó el rastreador *Sourcebuster* interceptando su filtro de opciones desde la raíz.

El Aprendizaje / Deuda Técnica

El mejor código es el que no existe. Al despojar a WooCommerce de su capa interactiva de JavaScript (Client-side rendering) y obligarlo a funcionar mediante peticiones de formulario `POST` HTML nativas (Server-side rendering), eliminamos todo el bloqueo de CPU en el navegador del cliente.

El resultado empírico es un **Total Blocking Time (TBT) de 0ms**, demostrando que es posible tener un flujo de carrito y *checkout* 100% funcional y ultrarrápido si se extirpa implacablemente la dependencia de JavaScript y se confía en los estándares nativos de la web.

💡 En resumen (Merci Explica):

WooCommerce actúa como un pesado camión logístico en una pista de carreras de Fórmula 1 (la web pública). Su tamaño destruiría el asfalto y hundiría el rendimiento. La arquitectura Headless aísla el camión en los boxes (backend), utilizando bólidos aerodinámicos (frontend Zero-JS) para transportar únicamente la carga necesaria al cliente. El resultado es un proceso de compra instantáneo que jamás sacrifica la velocidad por la funcionalidad.

🔗 Lecturas Recomendadas

- Integración Headless WP y WooCommerce
- Domando WooCommerce y dependencias AJAX