Compendio Estratégico: Épica 4 - Rendimiento Extremo y Estabilización

Estantería: DevSecOps e Infraestructura

Subtema: Gobernanza

mercedev.es — 2026-05-22 | Epic 4 - Fase 1

Contexto: El proyecto padecía una fuerte inestabilidad en la métrica TBT (Total Blocking Time) bajo emulación móvil estricta (4G CPU Throttling). La fluctuación aleatoria destruía la fiabilidad de las auditorías.

Maniobras Arquitectónicas: 1. **Yielding en JS (Shift-Left Performance):** Fragmentamos la inicialización asíncrona en `main.js` (`requestIdleCallback`) y `MerciController.js` (Promesas `setTimeout`) para ceder el paso al hilo principal, evadiendo los bloqueos del Garbage Collector de V8. 2. **Inversión de Prioridad LCP:** Mediante análisis empírico de red, descubrimos que el verdadero LCP era el titular `<h2>` y no el logotipo. Retiramos `fetchpriority="high"` del logo y aplicamos `decoding="async"` , liberando ancho de banda masivo para el CSS crítico. 3. **Fail-Fast en Orquestación:** Endurecimos el script `merci-commit.py` para devolver códigos de salida fatales (`sys.exit(1)`) en cancelaciones, bloqueando despliegues "fantasma" en el orquestador maestro. 4. **Diagnóstico de Física de Redes (SRE):** Migramos a extracción JSON Data-Driven. Incorporamos lógica para identificar si una caída temporal de puntuación es debida a latencias de red insalvables (Speed of Light) o TTFB alto (>300ms), evadiendo la trampa de la sobreingeniería (Premature Optimization).

Aprendizaje / **Deuda Técnica:** El rendimiento extremo a nivel 100/100 exige abandonar las suposiciones teóricas (como creer que la imagen más grande es el LCP) y abrazar el empirismo de los JSON de PageSpeed. Identificar que una caída a 96/100 puede deberse a un ping de 170ms y no a un código lento protege la moral del equipo. Este chasis blindado a 0ms de TBT constante es el cimiento obligatorio y no-negociable antes de inyectar arquitecturas pesadas.