Compilación en Frío: La Superioridad del SSG frente al PDF Dinámico

Estantería: Desarrollo y Arquitectura

Subtema: Rendimiento

mercedev.es — 2026-05-20 | Epic 3 - Fase 2

El Desafío (Síntoma)

A medida que el ecosistema documental crece, surge la necesidad de ofrecer descargables en formato PDF de las páginas web y manuales técnicos. La intuición y muchas prácticas heredadas sugieren dos posibles rutas de desarrollo, ambas plagadas de compromisos arquitectónicos:

1. **Generación en Cliente (Client-Side Rendering):** Utilizar librerías de JavaScript (`jsPDF` , `html2canvas`) para que el navegador del usuario "dibuje" el PDF al pulsar un botón.
    - *Síntoma:* Obliga al usuario a descargar cientos de kilobytes de código JS bloqueante. Rompe el paradigma **Zero-JS** y genera resultados inconsistentes dependiendo del navegador y el dispositivo (Safari, Firefox, Móvil).
2. **Generación en Servidor (Server-Side Rendering):** Delegar la creación del PDF a un backend dinámico (PHP, Node.js) que se ejecuta en tiempo real cada vez que un usuario lo solicita.
    - *Síntoma:* Rompe el paradigma de **Sitio Estático (SSG)**. Pintar un PDF es una tarea gráfica pesada (requiere calcular CSS, fuentes y cajas tipográficas). Exponer un endpoint dinámico de este tipo crea un vector masivo para ataques de Denegación de Servicio (DoS), permitiendo que un bot sature la CPU del servidor con múltiples peticiones simultáneas.

La Maniobra (Lógica)

La arquitectura de *Mercedev* apuesta por el sacrificio local en favor de la seguridad global mediante la **Compilación en Frío (Build-Time Generation)**.

Se implementó el script `merci-publish.py` como parte integral del orquestador local (`merci-total`). Este *pipeline* asume toda la carga computacional de

renderizado gráfico de forma síncrona en la máquina del desarrollador *antes* de realizar el despliegue.

Aunque esto incremente el tiempo de compilación local en varios segundos (ej. ~8 segundos por lote), el resultado es un artefacto (`.pdf`) completamente inerte y cristalizado.

El Aprendizaje / Deuda Técnica

- **Seguridad de Servidor Estático:** Al servir los PDFs precompilados mediante Nginx o un CDN estático, el coste de CPU en producción es estrictamente cero (`0.0%`). El servidor es invulnerable a ataques DoS de renderizado y no se requiere mantenimiento de software backend complejo.
- **Fidelidad *Pixel-Perfect*:** Puesto que el PDF se pinta en un entorno local controlado (con acceso garantizado a las fuentes y sin injerencias de motores JS de terceros navegadores), el documento descargado es matemáticamente idéntico para todos los usuarios.
- **Transparencia SRE:** Las métricas de tiempo (Profiler) demuestran que el coste se asume en *Dev*, protegiendo el entorno de *Prod*.
- **Posible Deuda Técnica:** Si el volumen de PDFs crece hasta hacer insufrible el tiempo del `merci-total` en local, la mitigación no será cambiar el modelo arquitectónico a dinámico, sino implementar **cachés basadas en Hash**. El script solo regeneraría un PDF si el *hash SHA* del Markdown original de la bitácora ha sido modificado.

En resumen

En lugar de hacer que tu teléfono móvil se esfuerce fabricando un PDF cuando pulsas "Descargar", o de saturar nuestro servidor web creándolo en el momento de la petición, hemos decidido generar todos los PDFs de antemano en nuestro propio ordenador antes de publicar la página web. Esto hace que la web sea ultrarrápida, completamente inmune a ataques de saturación por descargas

masivas y asegura que el documento tenga un diseño perfecto e idéntico para todos los usuarios.