Ingeniería de Interfaz Estática (SSG): Python Puro

Compendio | Vol. 1

mercedev.es — 2026-05-07 | Fase 7

Ingeniería de Interfaz Estática (SSG): Python Puro

El Desafío (Síntoma)

Mantener un sitio web con contenido extenso editando HTML puro no es escalable. Los frameworks modernos de SSG (Static Site Generation) resuelven esto, pero suponen una violación directa a la filosofía del proyecto: "0 dependencias en entorno anfitrión" y "rendimiento extremo". Necesitábamos un generador que no requiriera ecosistemas Node.js complejos, que garantizara el 100/100 en accesibilidad WAI-ARIA y que gestionara las métricas SEO (Search Engine Optimization) automáticamente.

La Maniobra (Lógica)

Se desarrolló un motor SSG propio en Vanilla Python (`merci-publish.py`) acoplado a un frontend en Vanilla JS (`MerciController.js`):

1. **Extracción Dinámica y SSOT:** En lugar de sistemas de plantillas complejos, el script recorta el `<header>` y `<footer>` directamente del archivo `public/index.html` . La portada es la Única Fuente de Verdad para el diseño estructural.
2. **Shift-Left SEO y JSON-LD:** El orquestador lee el Frontmatter YAML de cada Markdown y autogenera metadatos canónicos, etiquetas de descripción y un bloque estructurado JSON-LD específico para la entidad (ej. `Article` o `CollectionPage`), validando que no falten atributos clave como el `alt_portada` .
3. **Accesibilidad y UX (User Experience):** El script autogenera un índice estructurado ("Mega-Menú") inyectando dinámicamente atributos `aria-label` diferenciados para evitar colisiones de foco WAI-ARIA. Se

implementó *Cache Busting* dinámico (`?v=timestamp`) leyendo el sistema de archivos para evitar que los móviles sirvan CSS/JS obsoleto.
4. **Estado Frontend sin Reactividad:** El asistente Merci se controla mediante una clase ES6 pura. En lugar de mutaciones complejas (`setState`), alterna atributos ARIA (`aria-expanded` , `aria-hidden`) delegando todas las animaciones y transiciones de interfaz a la GPU a través de SASS.

El Aprendizaje / Deuda Técnica

El rendimiento y la accesibilidad no son parches que se añaden al final, sino decisiones de diseño desde la compilación. Un orquestador SSG construido a medida en un único archivo de Python de ~300 líneas puede superar en rendimiento y SEO técnico a ecosistemas de gigabytes, manteniendo el control absoluto sobre cada byte servido al usuario.

Cuadernillos de Referencia (Profundización táctica)

Para estudiar las optimizaciones quirúrgicas aplicadas en el frontend, consulta:

- Semántica HTML y WAI-ARIA: Roles implícitos y acoplamiento BEM
- El CV Anti-ATS: Hablando el idioma nativo de las máquinas
- La tiranía de la caché móvil y el Cache Busting dinámico
- Optimización de copias de seguridad locales y Zero Bloat