Anatomía de Merci Boilerplate v1.15.1 (Obsoleto)

Estantería: Boilerplate: Histórico

Subtema: Versiones

mercedev.es — 2026-05-24 | Epic 4 - Fase 1

La mayoría de plantillas web (boilerplates) actuales entregan un sitio funcional a costa de inyectar megabytes de librerías en el navegador, oscurecer la infraestructura y acumular deuda técnica desde el *commit* cero.

> ⚠️ **Aviso de obsolescencia:** Este documento relata la arquitectura intermedia (v1.15.1). **Lee aquí la versión v1.16.1 actualizada**.

Merci Boilerplate v1.15.1 es el antídoto. Lo que comenzó como un proyecto de investigación DevSecOps, ha madurado en un ecosistema operacional extremo.

El sistema surgió del rigor académico y evolucionó bajo el escrutinio de **Chaos Engineering**: una arquitectura segura, 0 dependencias externas en tiempo de ejecución (Zero-Bloat), integrando CMS Headless y aplicando metodologías SRE (Site Reliability Engineering) para garantizar telemetría constante.

4 Pilares de esta Release

1. Rendimiento Extremo (TBT 0ms Real) El núcleo ha sido refactorizado para fragmentar las tareas asíncronas de Javascript (*Yielding* mediante `requestIdleCallback` y `setTimeout`) y priorizar la red nativa (`fetchpriority`). El resultado: un **Total Blocking Time de 0ms** empírico, incluso en simulaciones extremas (Móvil 4G, CPU throttled 4x).

2. Hybrid Stack de IA (Agent Chaining) La Inteligencia Artificial está orquestada en local y en la nube creando auténticas cadenas de montaje de contenido: * **Agente Bibliotecario (Zero-Hallucination):** Transforma notas de terminal en Markdown puro (Ollama). * **Agente Blogger (DevRel):** Toma el relevo, redacta la promoción para el blog y encola el post para redes sociales, inyectando URLs canónicas automáticamente. * **Triage Interactivo:** El Agente Glosario ahora expone el contexto de extracción al humano para evitar consumir inferencia de IA en falsos positivos.

3. Observabilidad SRE & Chaos Engineering La infraestructura incluye un demonio `merci-sre.py` que ingiere telemetría continua hacia Prometheus y Grafana (IaC). Además, un "Mono del Caos" local inyecta vulnerabilidades XSS de

forma autónoma para validar empíricamente que el linter `merci-audit.py` es infranqueable, registrando cada ataque en bitácoras privadas encriptadas.

4. Arquitectura SSG de Compilación Incremental Las tres capas operan en paralelo sin contaminarse, y el orquestador principal ahora implementa Caché Semántica y *Mark & Sweep*. Al evitar recomponer PDFs y HTMLs que no han cambiado, el tiempo total de compilación ha bajado a la asombrosa cifra de **9.39 segundos**.

La Filosofía "Standalone Compliance"

Toda esta orquestación se logra **sin acoplamientos ni dependencias ocultas**. Los 25+ agentes de Python obedecen a un "Presupuesto de Dependencias" estricto y utilizan exclusivamente binarios asilados y librerías nativas (`sqlite3` , `pathlib`). Ningún agente contamina la máquina anfitriona.

Si estás listo para adoptar una infraestructura donde el rendimiento y la seguridad no son una promesa, sino una métrica auditable y matemáticamente demostrable, **eres bienvenido a clonarlo**.

🔗 **Repositorio Oficial de Merci Boilerplate en GitHub** https://github.com/MercedesDF/merci-boilerplate