Anatomía de Merci Boilerplate v1.18.0 (Obsoleto)

Estantería: Boilerplate: Histórico

Subtema: Versiones

mercedev.es — 2026-06-12 | Epic 8 - 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 original (v1.18.0). **Lee aquí la versión v1.19.0 actualizada**.

Merci Boilerplate v1.18.0 es el antídoto. Lo que comenzó como un proyecto de investigación DevSecOps, ha madurado en un ecosistema operacional extremo y autónomo.

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.

5 Pilares de la Arquitectura actual

1. Rendimiento Extremo (TBT 0ms Real) y E-commerce Zero-JS El núcleo fragmenta tareas asíncronas de Javascript (*Yielding*) y prioriza la red nativa (`fetchpriority`). La tienda WooCommerce opera como un Catálogo Headless inyectado desde Markdown, con un carrito de compra que funciona mediante formularios nativos (Zero-JS Cart). El resultado: un **Total Blocking Time de 0ms** empírico, incluso en pasarelas transaccionales bajo simulaciones 4G.

2. Orquestación Autónoma del Release (Clon Efímero) La distribución de la plantilla está 100% automatizada. El agente `merci-release.py` se encarga de clonar el código en un directorio efímero, aplicar una **Purga de Identidad Agnóstica** (erradicando la telemetría, menú de Art de Coté y metadatos personales) e instanciar el patrón **Gemelo Multimedia** (`tu_logo.webp`). Para evitar el envenenamiento de caché persistente de Nginx, inyecta programáticamente un *Timestamp de Época* como cache-buster dinámico, garantizando que el Showcase siempre despliegue los recursos frescos en la matriz pública sin riesgo de Deriva de Configuración.

3. 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 expone el contexto de extracción al humano para evitar consumir inferencia de IA en falsos positivos.

4. 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.

5. Arquitectura SSG de Compilación Incremental y Robustez del Pipeline (v1.18.0) Las tres capas operan en paralelo sin contaminarse, y el orquestador principal ahora implementa Caché Semántica y *Mark & Sweep*, manteniendo el tiempo total de compilación en la barrera Sub-10s. En la v1.18.0, se refactorizó el Core Pipeline para inyectar anotaciones de tipo estático y control robusto de excepciones (KeyboardInterrupt, fallos de subprocesos), garantizando que anomalías en caliente no muestren volcados de pila y se resuelvan con una salida elegante.

El Ecosistema de Agentes (32 herramientas)

El corazón del Boilerplate son sus 32 agentes programados en Python puro, organizados para gobernar cada aspecto del ciclo de vida:

Auditoría, Seguridad y QA

- `merci-audit.py` : SAST + Prevención de secretos y validación WAI-ARIA.
- `merci-hardening.py` : Inyección automática de políticas CSP y Hardening de Infraestructura.

- `merci-chaos.py` : Ingeniería del caos. Inyecta vulnerabilidades controladas para validar defensas.
- `merci-linkcheck.py` : DAST que rastrea despliegues detectando 404s y enlaces ambiguos (WAI-ARIA).
- `merci-drift.py` : Detector de Deriva Documental temporal y semántica.
- `merci-extract-metrics.py` : Extractor Data-Driven de métricas Core Web Vitals (JSON).
- `merci-sre.py` : Demonio SRE para ingesta de métricas en Prometheus.
- `merci-telemetry.py` : Inyector dinámico de telemetría del proyecto (Commits, Docs).
- `merci-sitemap.py` : Gestión y actualización automatizada de fechas en el mapa del sitio XML.

IA & Gestión del Conocimiento

- `merci-brain.py` : Base de conocimiento estática (Shift-Left IA).
- `merci-ssot.py` : Sincronización documental autónoma (ADR).
- `merci-librarian.py` : Agente técnico formateador (Zero-Hallucination).
- `merci-glosario.py` : Compilador de Glosario Autónomo.
- `merci-blogger.py` : Agente de redacción DevRel y Social.
- `merci-auto-fix.py` : Agente autónomo de reparación en la nube (GitHub Actions).

Publicación, E-commerce y Distribución

- `merci-publish.py` y `merci-promote.py` : Motor SSG y promoción.
- `merci-sync-pages.py` : Sincronizador de estructuras SSOT.
- `merci-wp.py` : Publicador Headless para WordPress vía API REST.
- `merci-shop.py` : Orquestador Headless de catálogo WooCommerce (Tienda No Tienda).
- `merci-linkedin.py` y `merci-queue.py` : Motor OIDC y visor de buffer social.
- `merci-optimizer.py` y `merci-assets-watcher.py` : Compiladores WebP automáticos.
- `merci-styles.py` y `merci-watcher.py` : Compiladores SASS 7-1.

Orquestación y Despliegue

- `merci-total.py` : Orquestador maestro local (Build & QA).
- `merci-commit.py` : Empaquetado atómico impulsado por bitácora.
- `merci-deploy.py` : Agente de despliegue remoto (sincronización SSH y purga Varnish).
- `merci-completo.py` : Orquestador Supremo DevSecOps (QA -> Commit -> Deploy).
- `merci-showcase.py` : Despliegue interactivo de demo pública (Clon Efímero).
- `merci-release.py` : Orquestador de exportación al Boilerplate.
- `merci-init.py` : Inicializador destructivo para nuevos repositorios derivados.
- `merci-backup.py` : Generador instantáneo de instantáneas locales (ZIP).

El Contrato de Desarrollo (Spec-Driven)

La filosofía principal de Merci es el **Spec-Driven Development**. Todo clon del repositorio incluye un archivo `instrucciones.md` que define exactamente qué está permitido hacer (ej. *Vanilla JS*) y qué está prohibido (ej. *Frameworks reactivos o estilos en línea*). Si estás listo para adoptar una infraestructura donde el rendimiento y la seguridad no son una promesa, sino una métrica auditable, **eres bienvenido a clonarlo**.

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