Idempotencia en Orquestadores Sociales (Escudo Anti-Duplicidad)

Estantería: DevSecOps e Infraestructura

Subtema: Gobernanza

mercedev.es — 2026-05-16 | Epic 3 - Fase 1

El Desafío (Síntoma)

Al operar orquestadores de publicación masiva (como `merci-wp.py` o `merci-linkedin.py`), cualquier error humano o ejecución programada duplicada (Cronjob) corría el riesgo de enviar el mismo artículo repetidas veces a la API (Application Programming Interface - Interfaz de Programación de Aplicaciones) destino. Esto genera spam en redes sociales y sobrescritura innecesaria en bases de datos Headless CMS.

La Maniobra (Lógica)

Se construyó un "Escudo Anti-Duplicidad" en la capa local delegando la validación del estado al propio documento Markdown, estableciéndolo como la Única Fuente de Verdad (SSOT).

El orquestador en Python lee el YAML Frontmatter antes de iniciar cualquier conexión de red. Si detecta el campo `estado_social: "publicado_linkedin"` o valida que un post en WordPress ya existe físicamente en el directorio `blog/`, el script interrumpe el flujo con un retorno silencioso (`return False`) para ese archivo, pasando al siguiente elemento de la cola.

El Aprendizaje / Deuda Técnica

Todo ecosistema automatizado (CI/CD o CLI) debe ser estrictamente **idempotente**; ejecutar el pipeline cien veces debe tener exactamente el mismo resultado en producción que ejecutarlo una sola vez. Confiar en los metadatos de los archivos locales para determinar el estado de propagación elimina la necesidad de realizar costosas peticiones GET de comprobación a las APIs externas (reduciendo el consumo de red y mitigando bloqueos por límites de cuota).