Automatización del Dashboard: Extracción de métricas Core Web Vitals desde PDFs

Cuadernillo | Vol. 1

mercedev.es — 2026-05-06 | Fase 1

El Desafío (Síntoma)

Mantener actualizado el "Engineering Dashboard" de la portada con las métricas reales de Core Web Vitals (LCP, INP, CLS, TBT) requería intervención manual tras cada auditoría en Google PageSpeed Insights. Copiar y pegar números desde un reporte a un archivo HTML genera fricción y es propenso a olvidos, rompiendo el principio de automatización DevSecOps.

La Maniobra (Lógica)

Desarrollamos el script `merci-extract-metrics.py` utilizando `pypdf`, una librería de Python puro (0 dependencias de binarios de sistema operativo).

El flujo es el siguiente: 1. Buscamos el reporte PDF más reciente en la carpeta designada (`auditorias-pagespeed.web.dev`). 2. Extraemos el texto bruto de las páginas del PDF. 3. Utilizamos Expresiones Regulares (Regex) avanzadas y tolerantes a saltos de línea (`re.DOTALL`) para cazar las métricas, ignorando las traducciones variables de Google (ej. *"Cambio acumulativo de diseño"* o *"Cumulative Layout Shift"*). 4. Leemos el archivo `public/index.html` e inyectamos los nuevos valores reemplazando el contenido de las etiquetas `<span class="hero__metric-value">` mediante Regex, preservando el HTML original.

```
 # Ejemplo de la robustez de las expresiones regulares aplicadas:
patrones = {
    "LCP": r'(?:Largest Contentful Paint|Despliegue.*?extenso|LCP)
[^\d<]*?([<>\d]+[.,]?\d*\s*(?:m?s)?)',
    "CLS": r'(?:Cumulative Layout Shift|Cambio.*?dise[ñn]o|CLS)[^\d<]*?
([<>\d]+[.,]?\d*)'
}
```

El Aprendizaje / Deuda Técnica

Extraer texto de PDFs generados por navegadores es impredecible. Descubrimos que extensiones como *ScreenCapture* no generan un PDF real, sino una imagen gigante encapsulada, volviendo ciego al script. Además, los saltos de línea que insertan los motores de PDF a mitad de las frases nos obligaron a usar `.*?` y `re.DOTALL` para que la búsqueda fluyera entre distintas líneas.

Waste Not (Cero Desperdicio): Este andamiaje ha transformado un reporte visual de Google en una fuente de datos estructurada para nuestro ecosistema estático, demostrando que con Python y expresiones regulares robustas podemos conectar sistemas cerrados a nuestro *pipeline*.