Generador PDF Interactivo para Documentación Interna

Estantería: Art de Coté

Subtema: Herramientas Periféricas

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

El Desafío (Síntoma)

Durante el desarrollo del ecosistema, surgió la necesidad operativa de imprimir físicamente los manuales estructurales (como el ciclo de vida del YAML Frontmatter) para consulta rápida en el escritorio.

Sin embargo, el motor de generación estática (SSG) de `merci-publish.py` está diseñado exclusivamente para renderizar y enrutar HTML optimizado hacia la red pública. Acoplar una librería de generación de PDFs dentro del orquestador principal habría supuesto una violación al principio de Responsabilidad Única (*Single Responsibility Principle*) y habría introducido dependencias pesadas (`weasyprint`) en un entorno que busca el "Zero Bloat".

La Maniobra (Lógica)

Para resolver la fricción sin comprometer la arquitectura, se diseñó un script táctico e independiente (`generar-pdf-docs.py`). Esta utilidad escanea de forma autónoma el directorio `docs/` , despliega un menú interactivo en la terminal y convierte el documento seleccionado a un PDF limpio.

El script inyecta un CSS atómico basado en las variables visuales del proyecto matriz para asegurar consistencia corporativa incluso en papel. Al ser una herramienta puramente colateral, se ejecuta bajo demanda desde `laboratorio/scripts_temporales/` .

```
 #!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
from pathlib import Path

try:
    import markdown
    from weasyprint import HTML
```

```
except ImportError:
    print("❌ Error: Faltan las dependencias. Ejecuta 'pip install
markdown weasyprint'")
    sys.exit(1)

REPO_ROOT = Path(__file__).resolve().parents
DOCS_DIR = REPO_ROOT / "docs"

def main():
    md_files = list(DOCS_DIR.glob("*.md"))
    if not md_files:
        print(f"❌ Error: No se encontraron archivos Markdown en
{DOCS_DIR.relative_to(REPO_ROOT)}")
        sys.exit(1)

    print("📄 Archivos disponibles en docs/:")
    for i, filepath in enumerate(md_files, 1):
        print(f"  {i}. {filepath.name}")

    try:
        opcion = int(input("\n👉 Elige el número del archivo que
quieres imprimir: "))
        if opcion < 1 or opcion > len(md_files):
            raise ValueError
    except ValueError:
        print("❌ Opción no válida. Operación abortada.")
        sys.exit(1)

    md_file = md_files[opcion - 1]
    pdf_file = md_file.with_suffix(".pdf")

    md_content = md_file.read_text(encoding="utf-8")
    html_content = markdown.markdown(md_content,
extensions=['fenced_code', 'tables'])

    html_string = f"""
    <!DOCTYPE html>
    <html lang="es">
    <head>
        <meta charset="UTF-8">
```

```
        <style>
            @page {{ size: A4; margin: 2.5cm; }}
            body {{ font-family: -apple-system, BlinkMacSystemFont,
"Segoe UI", sans-serif; line-height: 1.6; color: #334155; }}
            h1, h2, h3 {{ color: #ea580c; border-bottom: 1px solid
#e2e8f0; padding-bottom: 0.3em; }}
            pre {{ background: #f1f5f9; padding: 1em; border-radius:
4px; white-space: pre-wrap; font-size: 0.9em; }}
            code {{ font-family: monospace; background: #f1f5f9;
padding: 0.2em 0.4em; border-radius: 3px; font-size: 0.9em; }}
        </style>
    </head>
    <body>{html_content}</body>
    </html>
    """
    HTML(string=html_string).write_pdf(pdf_file)
    print(f"✅ Éxito. PDF generado listo para imprimir en:
{pdf_file.relative_to(REPO_ROOT)}")

if __name__ == "__main__":
    main()
```

El Aprendizaje / Deuda Técnica

Las herramientas de "usar y tirar" o de conveniencia local no deben formar parte del pipeline CI/CD ni del orquestador. Mantenerlas como scripts efímeros respeta la pureza de la infraestructura base, pero preservar su código como un activo en la Biblioteca (Art de Coté) garantiza que el conocimiento invertido no se evapore y pueda ser reaprovechado en futuras necesidades.