Robustez en Expresiones Regulares (RegEx) para automatización

Cuadernillo | Vol. 1

mercedev.es — 2026-04-26 | Fase 10 (Empaquetado y Release)

El Desafío (Síntoma)

Durante la automatización de empaquetados mediante un script en Python (`merci-commit.py`), el analizador sintáctico (Parser) falló al intentar leer la última entrada de la bitácora. El script bloqueó el commit asumiendo que no había actualizaciones. La causa raíz fue una discrepancia tipográfica microscópica: el usuario introdujo un guion corto (`-`) en lugar del guion largo (`—`) que la Expresión Regular (RegEx - Regular Expression) esperaba estrictamente.

La Maniobra (Lógica)

En lugar de forzar al usuario a memorizar qué tipo de guion utilizar, se refactorizó la expresión regular en el código fuente para implementar una clase de caracteres tolerante: `pattern = r"###\s+(\d{4}-\d{2}-\d{2})\s+[-—–]\s+([^\n]+)"`

Al incluir `[-—–]`, el motor de RegEx acepta indistintamente guiones cortos, medios y largos, capturando el bloque de texto correctamente independientemente de la corrección tipográfica automática que el Sistema Operativo haya aplicado.

El Aprendizaje / Deuda Técnica

Este incidente ilustra a la perfección la **Ley de Postel** (Principio de Robustez): *"Sé conservador en lo que envías, pero liberal en lo que aceptas"*.

Las herramientas de CLI (Command Line Interface - Interfaz de Línea de Comandos) y los flujos DevSecOps (Development, Security, and Operations - Desarrollo, Seguridad y Operaciones) que dependen de la entrada humana (como leer un archivo Markdown) fracasarán si son excesivamente frágiles ante errores tipográficos inofensivos. Flexibilizar los analizadores sintácticos previene bloqueos operativos sin comprometer la seguridad.