Enrutamiento Híbrido del IDE con LiteLLM Proxy

Estantería: Inteligencia Artificial y Agentes

Subtema: Gobernanza

mercedev.es — 2026-06-12 | Epic 2 - Fase 4

El Desafío (Síntoma)

Ante la transición a Antigravity IDE, surgió un doble problema. Por un lado, depender exclusivamente de agentes en la nube (como Gemini) generaba fricción por los límites de cuota (Errores HTTP 429 por exceso de peticiones) y saturación de contexto al analizar repositorios enteros (Context Window Stuffing). Por otro lado, forzar un aislamiento 100% local renunciaría a la potencia cognitiva de los grandes modelos de frontera de la nube para tareas complejas.

La Maniobra (Lógica)

Se adoptó el patrón de **Enrutamiento de Modelos (Model Routing)** utilizando LiteLLM no solo como librería dentro de scripts de Python, sino como un **Servidor Proxy Local** independiente a la escucha en el puerto 4000.

Se configuró un archivo maestro de infraestructura (`router.yaml`) que define una lista de prioridades para el tráfico de Inteligencia Artificial: 1. **Intento Primario:** Interfaz de Programación de Aplicaciones (API) de Gemini (Nube, para máxima inteligencia). 2. **Plan de Contingencia (Fallback):** Motor local Qwen 2.5 Coder corriendo en LM Studio (`localhost:1234`).

Finalmente, se inyectó la configuración en los ajustes de espacio de trabajo del IDE (`.vscode/settings.json`) para que Antigravity apunte siempre a nuestro proxy local, desactivando la telemetría corporativa y estableciendo un Presupuesto de Recursos estricto (4096 tokens).

El Aprendizaje / Deuda Técnica

Aplicar el Principio de Separación de Responsabilidades (Separation of Concerns) a la IA transforma radicalmente la forma de trabajar. Al extraer la lógica de reintentos y enrutamiento fuera del propio editor de código y delegarlo en LiteLLM

(como una "Sala de Máquinas" global), centralizamos la Única Fuente de Verdad (SSOT).

La desarrolladora obtiene una experiencia interactiva sin fricciones: si la nube falla o Google agota el saldo, el proxy intercepta el fallo y salta al hardware local en milisegundos de forma totalmente transparente (Degradación Elegante), garantizando la continuidad del proyecto a coste cero.