ApisDom

WebAssembly en 2026: ¿Cuándo Matar el Backend?

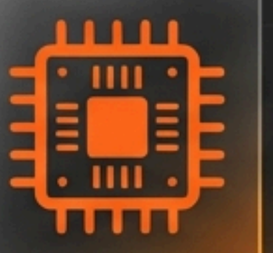
Guía de arquitectura: Rendimiento vs. Estructura.

Basado en análisis de ApisDom y estándares W3C.

EL ESTADO REAL: WASM 3.0 ES EL ESTÁNDAR

ESTÁNDAR W3C
SEPTIEMBRE 2025

WASMGC
RECOLECCIÓN DE
BASURA NATIVA



MEMORIA 64-BIT
>4GB SUPPORT

EJEMPLOS REPRESENTATIVOS

GOOGLE SHEETS

MOTOR DE CÁLCULO (WASMGC) → 2X VELOCIDAD

FIGMA

MOTOR GRÁFICO C++ → 3X VELOCIDAD DE CARGA

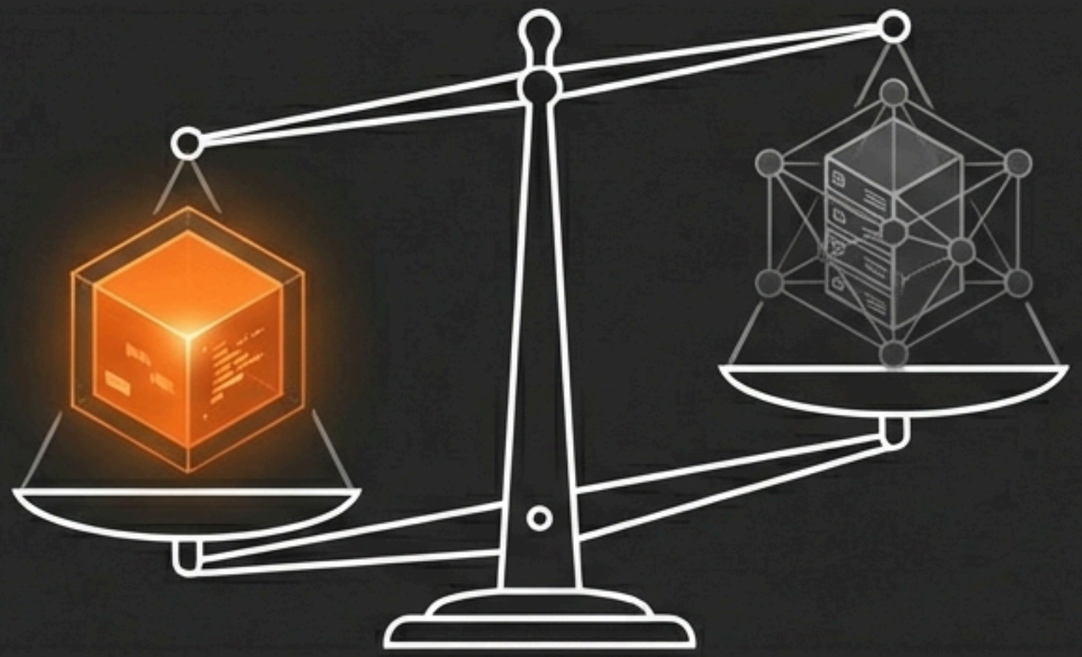
USO: OPTIMIZACIÓN DE MOTOR, NO ELIMINACIÓN DE SERVIDOR.

LA TRAMPA: RENDIMIENTO ≠ ARQUITECTURA

CAPACIDAD

JetBrains Mono

- WebGPU, OPFS
- Velocidad Nativa



ARQUITECTURA

JetBrains Mono

- Verdad
- Consistencia
- Seguridad

Wasm no elimina arquitecturas malas.
Solo las hace más rápidas.

LUZ VERDE: CUÁNDO ELIMINAR EL BACKEND

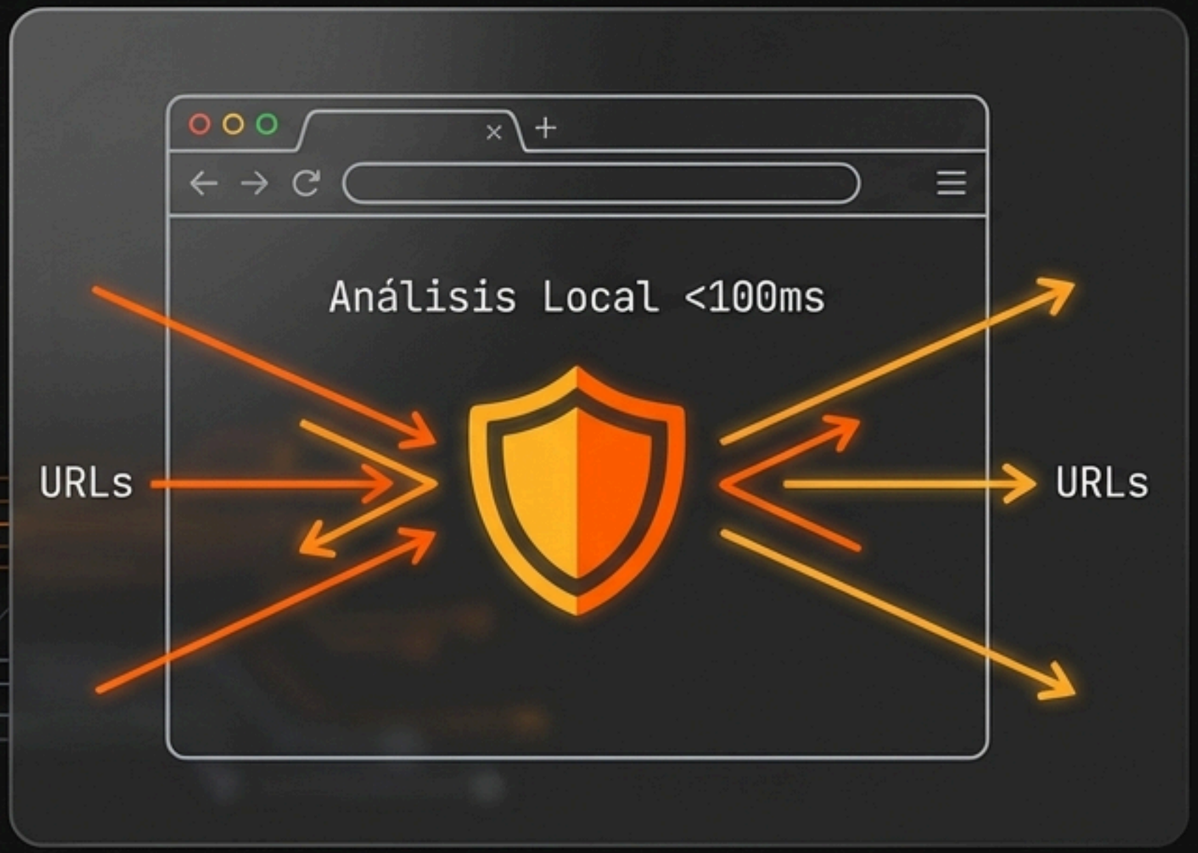
- ✓ **CÓMPUTO PURO**: Archivos locales (PDF, Imágenes).
- ✓ **PRIVACIDAD TOTAL**: El dato nunca abandona el dispositivo.
- ✓ **IA LOCAL**: Modelos de inferencia <500MB.

CASO DE ESTUDIO: DOCTVAULT & FORMATVAULT

- Firma y cifrado AES-256 local.
- Procesamiento de imágenes ilimitado.
- Resultado: 0 latencia, 0 coste de nube.

EL CASO DE LA PRIVACIDAD ABSOLUTA

Brújula Security & Protección Anti-Phishing



Privacidad por Diseño (GDPR Art. 25):

- Análisis sin enviar URLs a la nube.
- El dato NUNCA sale del cliente.
- La única arquitectura honesta para datos sensibles.

LUZ ROJA: CUÁNDO EL BACKEND ES OBLIGATORIO

- ✕ **ESTADO COMPARTIDO**: Sincronización entre usuarios.
- ⚠ **VERIFICACIÓN EXTERNA**: Auditoría, Compliance, Antifraude.
- ⚠ **DATASETS ILIMITADOS**: El navegador no es un Data Lake.

⚠ **ADVERTENCIA DE SEGURIDAD:**

El sandbox protege al usuario del código malicioso, pero no protege al sistema de un usuario malicioso.

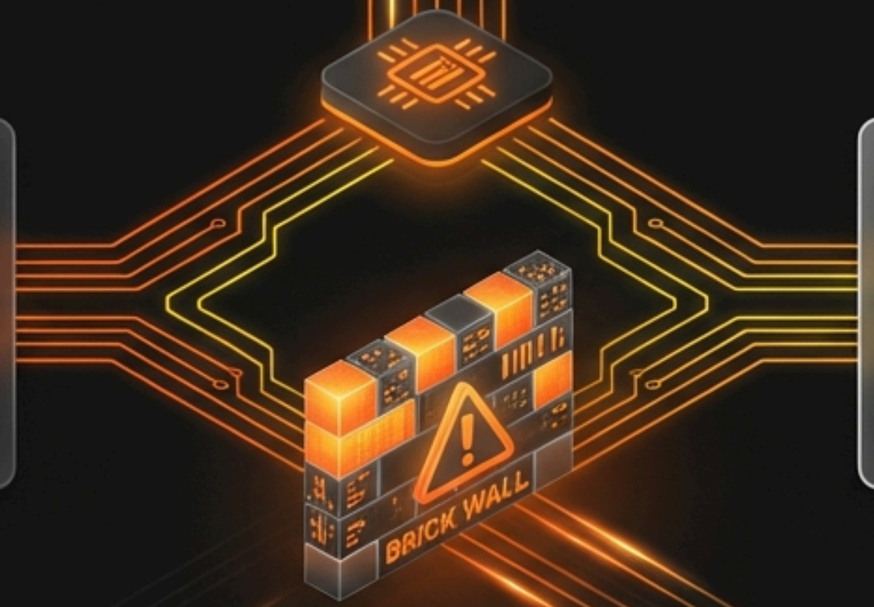
EL ANTI-PATRÓN: EL CICLO DEL FRACASO

LA LETRA PEQUEÑA TÉCNICA (I): HEADERS

EL DILEMA COOP/COEP

MULTI-THREADING REAL

Requiere SharedArrayBuffer + Headers COOP/COEP



ECOSISTEMA DE TERCEROS

Google Maps, Analytics, Widgets

Decisión: O tienes threading real, o tienes integraciones fáciles. Difícil tener ambos.

LA LETRA PEQUEÑA TÉCNICA (II): LATENCIA

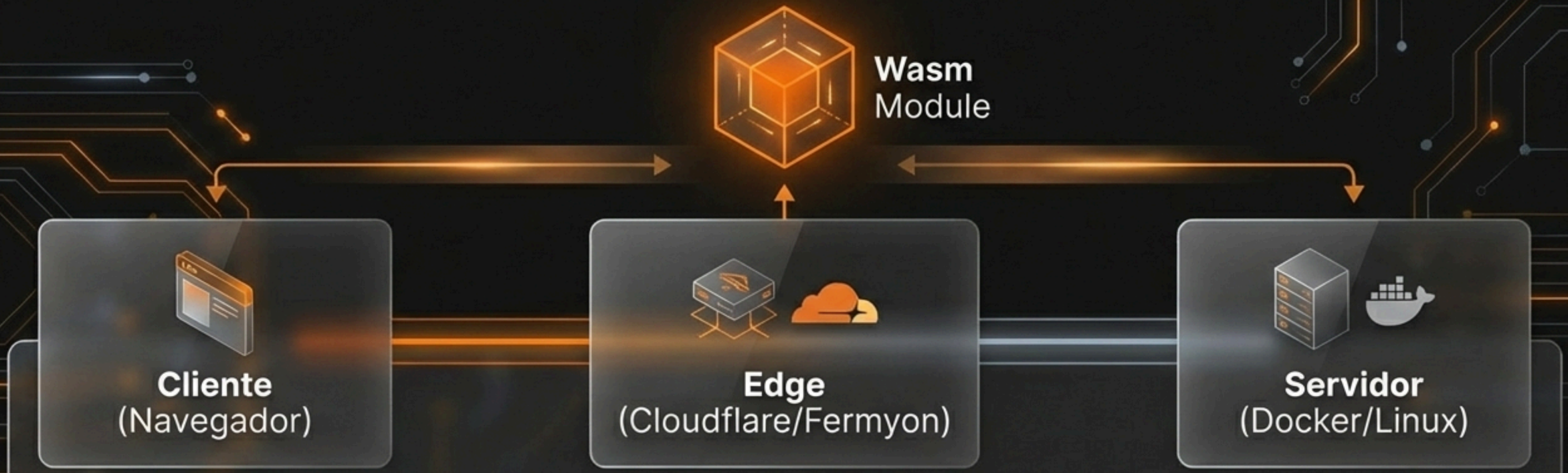
COLD START

EL PUENTE JS <-> Wasm

⚠ Diseño “Chatty” (muchas llamadas pequeñas) = Cuello de botella.

✓ Diseño “Chunky” (pocas llamadas grandes) = Rendimiento óptimo.

LA EVOLUCIÓN: WASI Y EL EDGE



WASI (WebAssembly System Interface)

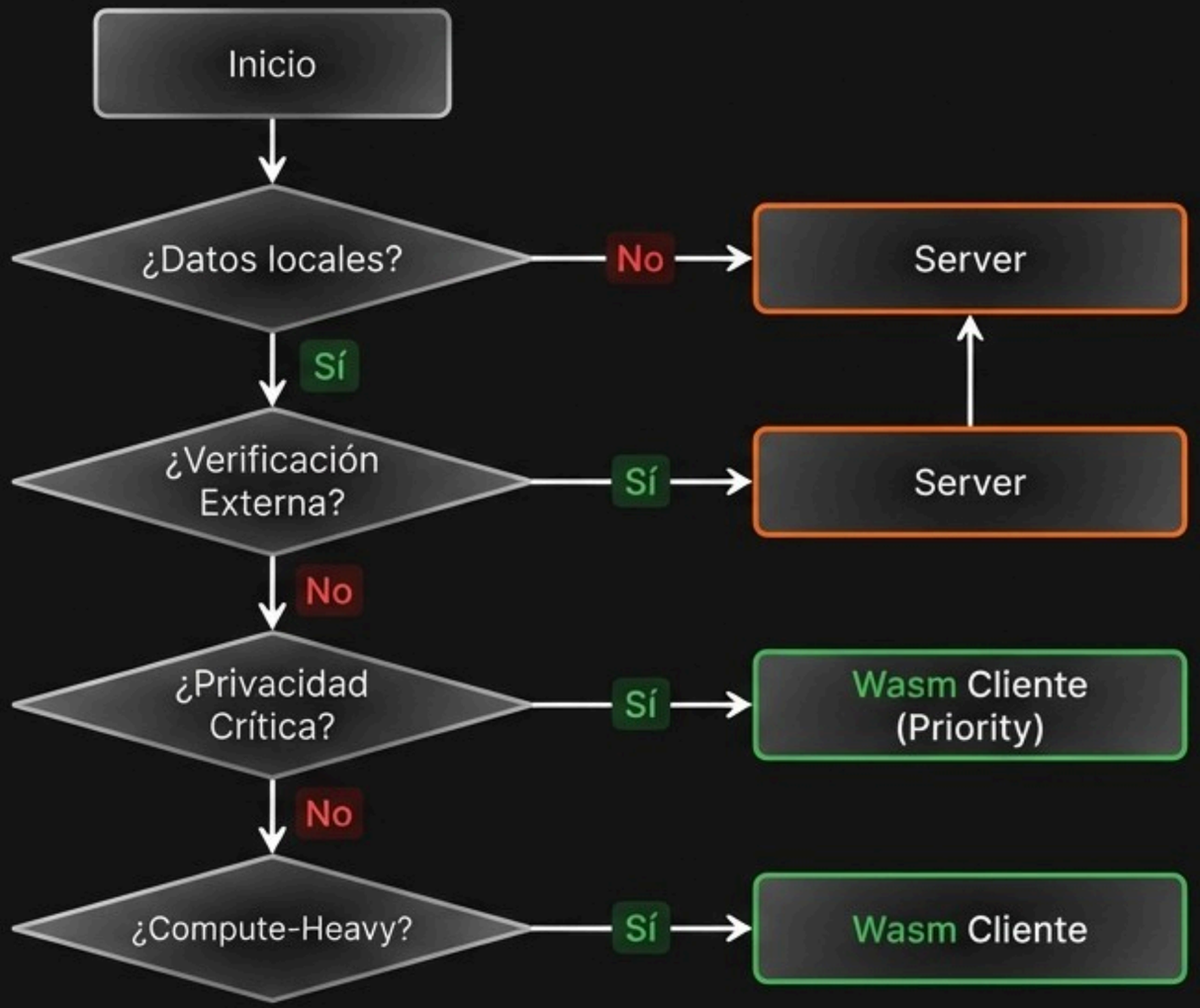
El mismo módulo se ejecuta en cualquier lugar.

La decisión ya no es de reescritura, es de despliegue.

MATRIZ DE DECISIÓN: CARGAS DE TRABAJO

Carga de Trabajo	Wasm Cliente	Backend / Edge
Cómputo sobre datos locales	✓ Ideal	✗ Overhead
Privacidad (Zero-Knowledge)	✓ Única opción	✗ Incompatible
Estado Compartido / Sync	✗ No aplica	✓ Necesario
Verificación Externa / Fraude	✗ Inseguro	✓ Necesario
Acceso a Recursos OS	✗ Sandbox	✓ WASI

Árbol de Decisión Arquitectónica



La Tecnología Sirve al Problema

Inteligencia Artificial Transparente

El sistema que funciona no es el más local.
Es el que tiene cada pieza donde corresponde.