INFORME DE ARQUITECTURA DE IA Y ESTRATEGIA EMPRESARIAL

El efecto mosca: arquitecturas de agentes orientadas a eventos y el espejismo del coste

Reducir los costes de los LLM en la empresa, a partir del principio de inercia en un coche en movimiento

Mehmet Keskinkılıç · Noviembre de 2026 **Categoría:** Diseño de sistemas y optimización financiera

Enfoque: Máquinas de estados deterministas + inferencia dirigida

Resumen: *Muchas empresas aplazan su transformación con IA y la integración de agentes autónomos por los «costes impredecibles de las API». Una causa importante de esta reticencia no es la falta de tecnología, sino una arquitectura que activa grandes modelos en cada paso del flujo de trabajo. Inspirado en la física de una mosca dentro de un coche cerrado, este documento plantea una hoja de ruta para un «modelo de agente por aceleración»: las operaciones rutinarias se mantienen en cero tokens y el LLM solo se ejecuta ante una anomalía. Las cifras de este informe son supuestos de un escenario de ejemplo, no mediciones.*

1. La metáfora física: ¿por qué la mosca no sale despedida en la cabina cerrada?

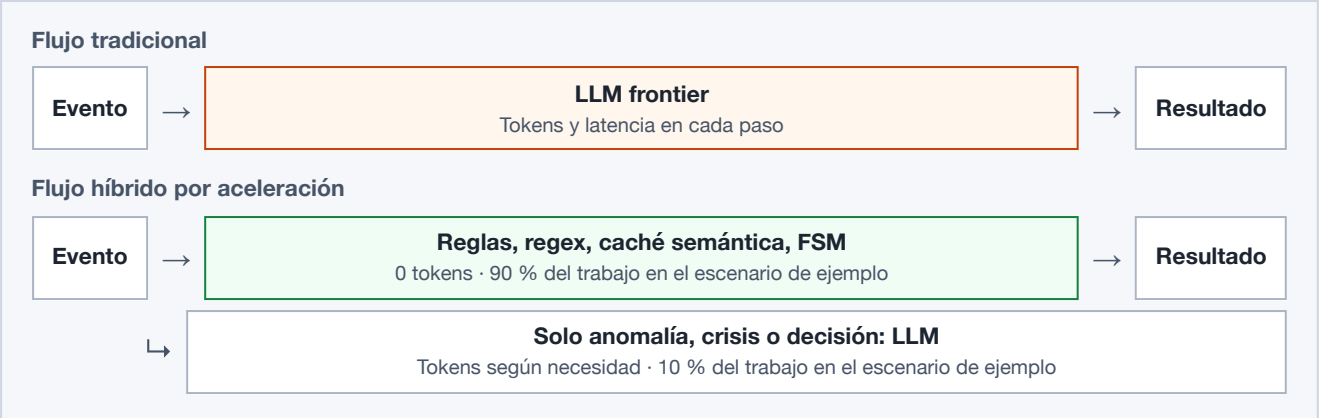
Una mosca dentro de un coche cerrado que circula a 100 km/h no hace ningún esfuerzo adicional para quedarse suspendida o volar del parabrisas al asiento trasero. Hay dos razones físicas: *la masa de aire transportada a la misma velocidad* y el principio de *inercia*. La mosca se mueve respecto al aire quieto de la cabina, no respecto a la carretera. Solo gasta mucha energía para mantener el equilibrio cuando el coche frena de golpe, acelera o una ventanilla abierta deja entrar turbulencias.

Los diseños de agentes tradicionales se comportan como si la mosca tuviera que volar fuera del coche contra un viento de 100 km/h: cada entrada rutinaria de datos, cada comprobación de estado y cada simple aprobación activa un gran modelo de lenguaje (LLM) de principio a fin, y el consumo de tokens se descontrola.

2. El error empresarial: el reflejo de conectar un LLM a cada paso

Detrás de la idea de que «la transformación con IA superará nuestro presupuesto» suele haber un diseño de agente monolítico que pasa cada evento operativo a un costoso modelo frontier. Sin embargo, una parte importante de los procesos empresariales (solicitudes de clientes, comprobación de facturas, actualizaciones del ERP, enrutamiento de tickets) puede ejecutarse con reglas previsibles y deterministas. Esa parte varía según el proceso y hay que medirla; el escenario de ejemplo de este informe supone que el 90 % del trabajo termina sin llegar a un modelo.

© 2026 Mehmet Keskinkılıç · mehmetkeskinkilic.com



3. Los tres pilares de la arquitectura

A. Capa de filtrado determinista (Gating Layer)

Cada entrada pasa primero por controles de coste nulo o mínimo. Si coincide con un patrón estándar, una ruta de API predefinida o una validación por reglas, la operación se completa directamente en el código. No se llama a la API del LLM ni se gastan tokens.

B. Flujo de trabajo basado en una máquina de estados (Finite State Machine, FSM)

Los procesos del agente se modelan con marcos orientados a estados como LangGraph. Al pasar de un estado a otro, las reglas de decisión se ejecutan en el código. Solo una excepción inesperada, una anomalía de datos no estructurados o una bifurcación estratégica con varias opciones hacen que el sistema lance una alerta de «aceleración» y se dirija al nodo LLM.

C. Caché semántica e inferencia por diferencias (Diff)

Las consultas repetidas se detectan con modelos ligeros de embedding locales y se sirven directamente desde la caché. Incluso cuando hace falta llamar al LLM, no se reenvía cada vez todo el conjunto de documentos de la empresa; el prompt caching y el envío solo de los parámetros de estado modificados (el delta) reducen el gasto.

4. Comparación de costes y rendimiento (escenario de ejemplo)

La tabla siguiente no es la medición de una instalación real. Es un escenario de ejemplo para un centro de soporte o procesamiento de tamaño medio que gestiona 10.000 eventos operativos al día.

Métrica / Fase	Agente LLM monolítico	Modelo híbrido por aceleración	Ahorro y efecto
Flujo rutinario (75 %)	~1.500 tokens por llamada	0 tokens (ejecución a nivel de reglas y API)	100 % de tokens LLM ahorrados
Entrada similar o repetida (15 %)	El modelo razona desde cero	0 tokens (coincidencia en caché semántica)	100 % de tokens LLM ahorrados
Anomalía y decisión (10 %)	Funciona con contexto costoso	Modelo pequeño u optimizado + contexto diff mínimo	70 % de optimización de tokens
Tiempo de respuesta de ejemplo	1.800 ms – 3.500 ms	< 40 ms (rutina) / 900 ms (anomalía)	Más de un 97 % menos en rutina, cerca de un 50–75 % menos en anomalías
Coste mensual estimado	4.500 $ – 7.000 $ / mes	350 $ – 600 $ / mes	**Cerca de un 87 % – 95 % de ahorro estimado**

Supuesto de cálculo: unos 300.000 eventos al mes × ~1.500 tokens = ~450 millones de tokens, con una media aproximada de 10–16 $ por millón de tokens. Con esta base, el 10 % de anomalías del modelo híbrido cuesta unos 135–215 $ en tokens; el rango de la tabla queda por encima, como estimación prudente. La escritura de reglas, la infraestructura, el mantenimiento y la operación de la caché no se incluyen en ninguna de las dos columnas.

5. Resumen para el consejo y plan de acción

> **Conclusión estratégica:** La transformación con IA no significa «entregar todos los procesos a un LLM». Una transformación eficaz separa, en los umbrales adecuados, la fiabilidad determinista de la ingeniería de software y la capacidad de resolver problemas de los LLM. Cuando, como en el escenario de ejemplo, el 90 % del trabajo se ejecuta sin gastar tokens y la IA solo interviene en el 10 % restante de momentos inciertos, la barrera del coste puede convertirse en una ventaja.

Plan de acción

1. **Medir:** Registre las llamadas actuales al LLM por tipo de evento, con volumen, tokens, latencia y tasa de error. Sustituya las proporciones de este informe por sus propios datos.
2. **Separar:** Lleve el trabajo rutinario de gran volumen y resultado verificable a reglas y máquinas de estados.
3. **Abrir la caché con control:** Use la caché semántica con un umbral de similitud, audite muestras y evite que los datos personales o sensibles pasen de una respuesta a otra.
4. **Fijar una regla de escalado:** Envíe primero las anomalías a un modelo pequeño. Si la confianza es baja, escale a un modelo grande o a una persona.
5. **Supervisar de forma continua:** Controle la parte del trabajo completada sin modelo, las respuestas erróneas de reglas y caché, y el coste mensual total con la ingeniería incluida.

Mehmet Keskinkılıç
Especialista en operaciones aéreas y desarrollador iOS independiente
© 2026 Mehmet Keskinkılıç. Se puede compartir citando la fuente. Versión actual: mehmetkeskinkilic.com/es/writing/