Tecnología

IA en sistemas críticos: por qué la nube no alcanza

Infraestructura local y arneses de software: el diseño técnico para pasar de la consulta a la ejecución autónoma.

26 de noviembre de 2026

El techo invisible de las APIs comerciales en proyectos de escala

La incorporación de inteligencia artificial en el desarrollo de software suele arrancar por el camino más accesible: asistentes conversacionales y APIs comerciales en la nube. Para resolver consultas aisladas, autocompletar funciones o escribir fragmentos acotados de código, ese esquema funciona sin mayores fricciones. El problema aparece cuando una organización intenta dar el salto siguiente y **usar esa misma arquitectura para refactorizar sistemas complejos o saldar deuda técnica profunda**.

En ese umbral, depender de proveedores externos choca contra **tres barreras estructurales** que comprometen tanto la viabilidad técnica como la previsibilidad financiera de un proyecto. La primera es **el costo exponencial por consumo de tokens**: analizar, procesar y reescribir repositorios extensos a través de servicios en la nube escala rápidamente hacia facturaciones de miles de dólares por iniciativa, convirtiendo la modernización tecnológica en un gasto variable difícil de acotar. La segunda es **el resguardo de la propiedad intelectual**: operar sobre APIs de terceros implica enviar código fuente y lógica central del negocio a servidores externos, supeditando activos críticos a políticas de privacidad ajenas. La tercera, y más determinante en el plano de la ingeniería, es **la saturación de la ventana de contexto** cuando una tarea exige continuidad.

La "ansiedad operativa" de los modelos y el diseño de arneses

Los modelos comerciales están optimizados para interacciones relativamente acotadas, pero cuando enfrentan un proceso largo **se quedan sin espacio operativo para sostener todo el mapa del problema en simultáneo**. Mariano Casero, CTO de Finnegans, describe ese comportamiento como una falla de diseño en la forma en que hoy se consumen estos sistemas: "Cuando un modelo percibe que su ventana de contexto empieza a

agotarse en medio de una tarea extensa, entra en una suerte de **ansiedad operativa: apura el cierre del trabajo y empieza a entregar código incompleto, fallas lógicas o alucinaciones**".

Mover la ejecución desde la nube hacia **servidores propios con modelos de código abierto** resuelve dos de las tres restricciones iniciales: elimina la facturación variable por tokens (reduciendo el costo operativo al consumo energético del hardware) y garantiza que el código nunca salga de la red interna. Sin embargo, la limitación de memoria de contexto persiste incluso en infraestructura local.

Superar ese límite exige dejar de tratar a la IA como un chat sin estado y construir una capa de ingeniería alrededor del modelo. En Finnegans, el equipo de arquitectura abordó este problema mediante el diseño de **arneses (harnesses): estructuras de software que envuelven al modelo y gobiernan su ejecución de manera continua**. El arnés administra el estado del desarrollo entre sesiones, lo que permite que el sistema inicie ciclos nuevos conservando memoria exacta de lo que ya hizo, retome tareas de forma iterativa y autónoma a lo largo de horas o días sin perder coherencia técnica y concluya procesos extensos sin caer en atajos forzados por la saturación de memoria.

Saldar años de deuda técnica en quince días

La eficacia de esta arquitectura se puso a prueba frente a un problema concreto de mantenimiento evolutivo: un **repositorio complejo de aproximadamente sesenta componentes** responsables de la interfaz visual de una plataforma empresarial, como botones y grillas de datos. Durante años, esa estructura funcionó con un alto nivel de acoplamiento: modificar el comportamiento o la estética de un simple botón exigía intervenciones engorrosas que podían alterar el resto de la interfaz.

Desarticular ese bloque en componentes independientes era una deuda técnica postergada durante mucho tiempo. Resolverlo mediante horas de desarrollo humano implicaba un esfuerzo altísimo sobre tareas repetitivas, mientras que delegarlo a APIs comerciales de IA resultaba económicamente inviable por el volumen de contexto requerido. Al ejecutar un modelo open source dentro de la infraestructura propia y coordinarlo mediante arneses de software, **el proceso se llevó adelante de manera autónoma en un plazo estimado de entre diez y quince días** (dato sujeto a validación interna). El sistema desarmó el repositorio completo en módulos aislados, **sin transferir una sola línea de código fuera de la red interna y sin costos por tokens externos**. Hoy, cada actualización visual sobre esos componentes se ejecuta de forma independiente y predecible.

Soberanía tecnológica y nuevas fronteras de ingeniería

Cuando la ejecución de tareas pesadas deja de depender de medidores externos y ventanas de memoria frágiles, las organizaciones recuperan margen para encarar proyectos de ingeniería que antes quedaban archivados por plazos o presupuestos inabordables. Sobre esta misma arquitectura de infraestructura local y arneses de ejecución, los equipos de desarrollo ya avanzan en líneas de investigación aplicadas a otros desafíos estructurales, como la generación automatizada de tests unitarios y la migración de bases de datos desde SQL Server hacia PostgreSQL.

A medida que la inteligencia artificial **pasa de asistir al desarrollador individual a ejecutar transformaciones profundas sobre plataformas de misión crítica**, la ventaja competitiva deja de estar en contratar la API más potente del mercado. La pregunta que empieza a ordenar las decisiones de arquitectura (y de inversión) en el software empresarial es qué organizaciones cuentan hoy con el diseño técnico necesario para **gobernar esa autonomía con precisión, costos previsibles y control real sobre su propio código**.