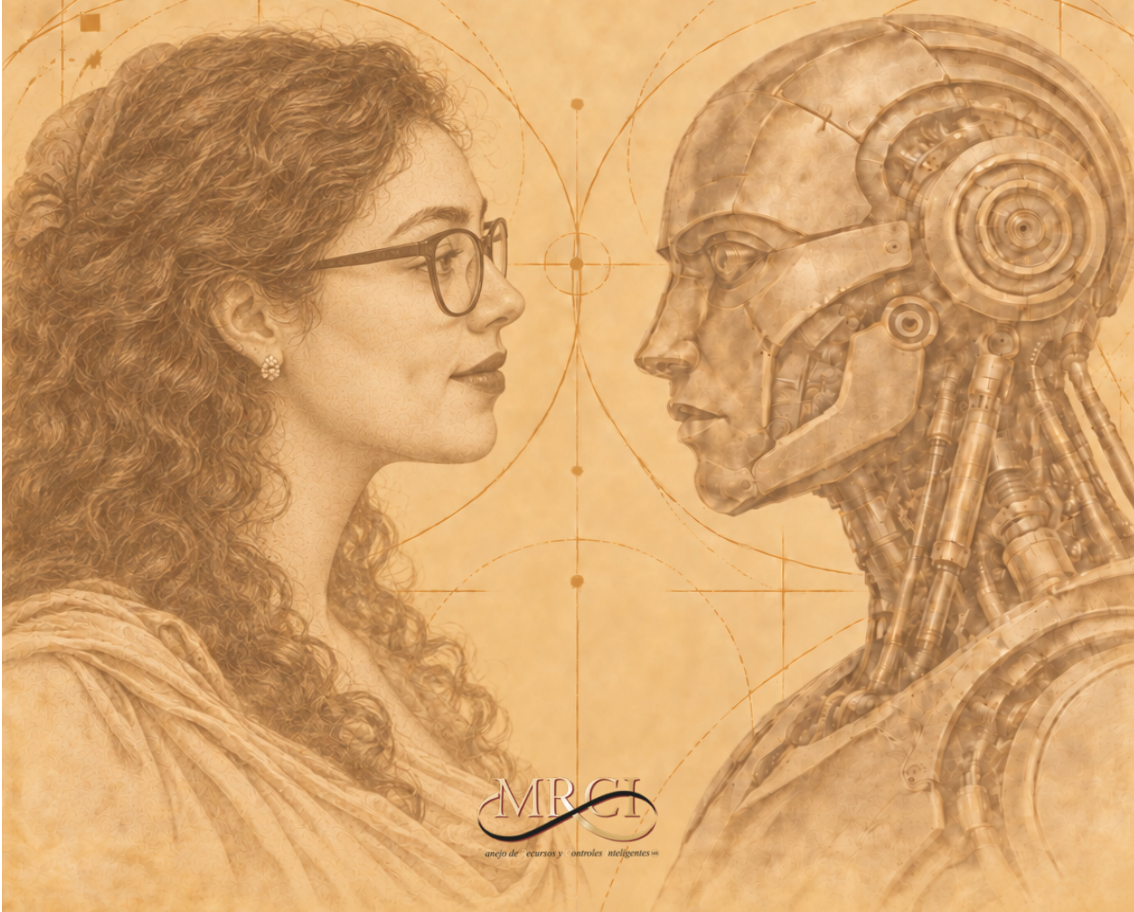
COLECCIÓN

MITOS Y REALIDADES DE LA IA

*Entre la promesa y la realidad*

# QUÉ ES UN MODELO Y CÓMO SE DISTINGUE DE UNA APLICACIÓN DE IA

Número 4



MRCI

anejo de ecursos y ontroles nteligentes

**DR. JOSÉ ANTONIO PÉREZ RAMOS**

# QUÉ ES UN MODELO Y CÓMO SE DISTINGUE DE UNA APLICCIÓN DE IA

## SOBRE EL AUTOR

Doctor en Ciencias de lo Fiscal por el Instituto de Especialización para Ejecutivos. Maestro en Fiscal por la Universidad Autónoma "Benito Juárez" de Oaxaca. Maestro en Estudios Avanzados en Derechos Humanos por la Global Open University. Licenciado en Contaduría Pública por la Universidad Autónoma "Benito Juárez" de Oaxaca y Licenciado en Filosofía por la Universidad Nacionalista de México.

Socio fundador y director general de la Firma Manejo de Recursos y Controles Inteligentes (Firma MRCI). Fiscalista del año 2009 por la revista *Defensa Fiscal*. Doctor honoris causa por la Asamblea General del Claustro Doctoral Iberoamericano (2016). Doctor honoris causa por el Instituto Mexicano de Líderes de Excelencia (2017). Doctor honoris causa por 1Million Startups, Latinomics, Leaderships Forum y la Fundación Humanist World (2018). Doctor honoris causa por el Claustro de Doctores de la AISOL (American International School of Law) (2019).

Autor de diversas obras, entre las que destacan *Ius mutantur o del derecho que cambia* (Tirant Lo Blanch, 2025), *Futurología* (MRCI, 2025) y *Hasta que la muerte nos separe* (MRCI, 2025). Ha participado como coordinador en *Cuestiones tributarias. Problemas y controversias en el México actual* (Tirant Lo Blanch, 2023), *El costo de la justicia* (Apex Iuris, 2019), *Remuneraciones estratégicas inteligentes* (MRCI, 2015) y *Retos de la justicia tributaria* (Tirant Lo Blanch-AMDF, 2025).

Titular de *Pie de Página*, programa para la divulgación de los derechos humanos en México, a través de la plataforma Legal Cloud. Expositor en foros nacionales e internacionales. Colaborador en diversas revistas jurídicas, entre las que destacan *Defensa Fiscal* y *Legal Cloud*.

# QUÉ ES UN MODELO Y CÓMO SE DISTINGUE DE UNA APLICACIÓN DE LA IA

**DR. JOSÉ ANTONIO PÉREZ RAMOS**

PRIMERA EDICIÓN, ATEMPORAL

Derechos reservados, propiedad de
Dr. José Antonio Pérez Ramos

Comentarios y opiniones: investigacion@mrci.com.mx

**Título original:** Qué es un modelo y cómo se distingue de una aplicación de IA

*Colección mitos y realidades de la inteligencia artificial.*

*Entre la promesa y la realidad.*

**Autor:** Dr. José Antonio Pérez Ramos.

Queda prohibida la reproducción total y parcial de esta obra denominada; QUÉ ES UN MODELO Y CÓMO SE DISTINGUE DE UNA APLICACIÓN DE IA, por cualquier medio, sin autorización escrita del autor.

PRINTED IN MEXICO
IMPRESO EN MÉXICO

## ÍNDICE

INTRODUCCIÓN ..... - 8 -
LA RESPUESTA VISIBLE OCULTA VARIAS DECISIONES..... - 10 -
QUÉ IDENTIFICA LA PALABRA MODELO ..... - 12 -
ESTRUCTURA, PARÁMETROS Y CONDICIONES DE ENTRADA ..... - 14 -
MODELO, ALGORITMO Y EJEMPLAR AJUSTADO ..... - 15 -
QUÉ AÑADE LA APLICACIÓN ..... - 16 -
UNA BUENA SALIDA PUEDE ENTREGARSE MAL ..... - 20 -
CUATRO CONFIGURACIONES POSIBLES ..... - 21 -
CLASIFICACIÓN DENTRO DE UN FORMULARIO ..... - 22 -
RECOMENDACIÓN CON SELECCIÓN Y ORDENAMIENTO ..... - 23 -
GENERACIÓN APOYADA EN DOCUMENTOS..... - 24 -
GENERACIÓN CON EJECUCIÓN DE HERRAMIENTAS ..... - 26 -
DATOS, CONTEXTO Y ESTADO..... - 27 -
ALMACENAMIENTO Y ACCESO EFECTIVO ..... - 29 -
CACHÉ, REUTILIZACIÓN Y CAMBIOS APARENTES..... - 30 -
IDENTIDAD Y VERSIONES ..... - 31 -
UNA FICHA DE EJECUCIÓN SUFICIENTE PARA LA PREGUNTA ..... - 32 -
CAMBIAR UNA PIEZA EXIGE UNA PRUEBA COMPATIBLE ..... - 34 -
DOS NIVELES DE EVALUACIÓN..... - 36 -
EL DENOMINADOR DECIDE QUÉ SE ESTÁ MIDIENDO ..... - 37 -
LA UNIDAD DE ÉXITO NECESITA CONTENIDO..... - 38 -
TRES CASOS PARA LOCALIZAR EL CAMBIO ..... - 40 -
CASO 1. EL ANEXO QUE EXISTÍA, PERO NO LLEGÓ ..... - 40 -

CASO 2. DOS MODELOS Y DOS RECUPERADORES ..... - 42 -
CASO 3. UN ARCHIVO CREADO Y UN ENVÍO PENDIENTE ..... - 44 -
MITOS QUE LA SEPARACIÓN PERMITE CORREGIR ..... - 48 -
LÍMITES Y RESPONSABILIDADES ..... - 49 -
QUÉ PUEDE SABERSE DESDE FUERA ..... - 51 -
CAPACIDAD, TIEMPO Y COSTE NO COINCIDEN ..... - 52 -
EXAMINAR UNA APLICACIÓN ..... - 53 -
CONCLUSIÓN ..... - 56 -
REFERENCIAS ..... - 57 -

## Introducción

El desarrollo y la expansión de la inteligencia artificial han transformado significativamente la manera en que las personas, las organizaciones y las instituciones interactúan con los sistemas tecnológicos. Actualmente, numerosas actividades relacionadas con la búsqueda de información, la elaboración de documentos, el análisis de datos y la automatización de procesos incorporan herramientas que utilizan inteligencia artificial. Sin embargo, esta creciente presencia tecnológica también ha generado confusiones conceptuales sobre su funcionamiento, particularmente respecto de la diferencia entre un modelo de inteligencia artificial y una aplicación que lo utiliza. Aunque ambos conceptos suelen emplearse indistintamente en el lenguaje cotidiano, representan elementos distintos cuya comprensión resulta indispensable para evaluar adecuadamente las capacidades, limitaciones y resultados de estos sistemas.

Un modelo de inteligencia artificial constituye una representación computacional diseñada para procesar determinadas entradas y producir resultados conforme a una estructura y a ciertos parámetros. Por su parte, una aplicación comprende un conjunto más amplio de componentes tecnológicos que organizan la interacción

con el usuario, administran información, incorporan herramientas y establecen las condiciones bajo las cuales se ejecutan determinadas operaciones. En consecuencia, el modelo puede constituir una parte fundamental del funcionamiento de una aplicación, pero no necesariamente determina por sí mismo la totalidad de sus capacidades ni explica todos los resultados que experimenta quien la utiliza.

Esta distinción adquiere especial relevancia ante la tendencia a atribuir directamente a la inteligencia artificial cualquier acierto, error o comportamiento observado durante su utilización. Cuando una aplicación proporciona información incorrecta, omite documentos, presenta referencias equivocadas o no ejecuta una operación solicitada, resulta frecuente considerar que el problema se encuentra exclusivamente en el modelo. No obstante, las causas pueden localizarse en diferentes etapas del proceso, como la extracción de información, la recuperación documental, la preparación de las entradas, la configuración de permisos, la ejecución de herramientas o la presentación final de los resultados. Por ello, comprender la arquitectura funcional de estos sistemas permite superar explicaciones simplificadas y desarrollar criterios de evaluación más precisos.

### La respuesta visible oculta varias decisiones

Una persona pregunta por una cláusula, adjunta dos documentos y recibe una contestación aparentemente directa; sin embargo, entre la pregunta y la respuesta pueden intervenir un lector de archivos, un buscador, un modelo generador y un programa que agrega las referencias; la experiencia es unitaria para quien la utiliza, pero esa unidad de presentación no demuestra que una sola pieza haya realizado todo el trabajo.

El problema aparece cuando se pregunta por qué falló: decir que «la IA ignoró el anexo» puede describir correctamente la experiencia sin explicar su causa, pues todavía falta saber si el archivo llegó al servidor, si pudo extraerse su texto, si el recuperador lo seleccionó y si ese contenido estuvo presente en la entrada efectiva del modelo; cada posibilidad exige una comprobación distinta y conduce a una corrección diferente.

La pregunta central de este artículo es qué permite distinguir un modelo de una aplicación que lo incorpora y por qué esa distinción cambia la forma de evaluar resultados, comparar productos y asignar tareas de mantenimiento; su argumento rector es que el nombre del



modelo identifica, cuando la documentación es suficiente, una parte relevante del funcionamiento; la aplicación añade decisiones sobre información, ejecución, controles y entrega que pueden transformar de manera sustancial la experiencia obtenida.

El estudio sigue una revisión documental conceptual y selectiva de publicaciones originales sobre sistemas de aprendizaje, recuperación de información, documentación y pruebas, junto con documentación oficial de representación y preparación de modelos; el corte de consulta es el 2 de octubre de 2026; no se presenta como revisión sistemática, ensayo de productos comerciales ni medición propia de desempeño, y los casos numéricos y organizacionales se identifican expresamente como hipotéticos.

Las fuentes cumplen funciones diferentes: los artículos científicos aportan resultados o propuestas de sus autores, la documentación técnica describe interfaces y convenciones de implementaciones concretas, y el marco del National Institute of Standards and Technology (Tabassi, 2023) ayuda a situar el sistema en un contexto de uso; ninguna de esas categorías, por sí sola, prueba

que un producto no examinado implemente determinada función o alcance un resultado específico.

### Qué identifica la palabra modelo

En el uso operativo adoptado aquí, un modelo es una representación computacional que relaciona entradas con salidas conforme a una estructura y a valores que determinan su funcionamiento; cuando es un modelo aprendido, esos valores y, en ciertos métodos, parte de la estructura resultan del ajuste a datos; puede producir una puntuación, una categoría, una representación numérica, una secuencia o una distribución de resultados posibles, sin que todos esos productos sean equivalentes.

La definición no exige que el modelo tenga forma de red neuronal ni que converse con una persona; un árbol de decisión ajustado, una regresión y un generador de lenguaje pueden ocupar el lugar de modelo en aplicaciones muy distintas; tampoco implica que la salida constituya por sí misma una decisión institucional, una afirmación verdadera o una acción material ya realizada.



Para entender la diferencia conviene separar la regla de cálculo de su utilización: un modelo que estima una puntuación de prioridad para una incidencia entrega una salida, mientras otro componente decide si esa puntuación basta para mostrar una alerta, cambiar el orden de una bandeja o solicitar revisión; la frontera debe declararse porque algunas implementaciones incorporan parte de esa transformación dentro del paquete del modelo y otras la dejan fuera.

No existe una única división física obligatoria entre todas las piezas que un proyecto llama modelo; en un trabajo de investigación, el término puede designar una formulación matemática, mientras en un repositorio puede referirse al archivo que contiene parámetros y estructura, y en una oferta comercial puede utilizarse como denominación de un servicio; antes de comparar dos afirmaciones conviene establecer cuál de esos objetos está siendo nombrado.

La especificación de ONNX ofrece un ejemplo concreto de esta necesidad: su representación de un modelo vincula un grafo de cálculo con metadatos y distingue la versión del propio modelo, la versión de la representación intermedia y los conjuntos de operadores utilizados



(ONNX, s.f.); la distinción prueba cómo un formato técnico organiza varios aspectos del objeto ejecutable, sin convertir ese formato en definición universal de toda IA.

Un grafo, en este contexto, describe operaciones y relaciones entre ellas; los parámetros contienen valores que esas operaciones utilizan, y los metadatos ayudan a identificar el contenido; el archivo resultante todavía necesita un entorno capaz de interpretarlo o ejecutarlo, así como entradas que respeten las formas, tipos y convenciones esperadas; poseer el archivo no equivale a disponer de una aplicación lista para un uso determinado.

### Estructura, parámetros y condiciones de entrada

La estructura de un modelo establece qué operaciones se relacionan y de qué manera; los parámetros fijan valores dentro de esa estructura; dos ejemplares pueden compartir diseño y diferir en los valores aprendidos, del mismo modo que dos modelos con nombres cercanos pueden pertenecer a una familia sin ser intercambiables para una tarea concreta.



La preparación de la entrada forma parte del problema técnico aunque no se incluya dentro del archivo del modelo; si durante el ajuste se normalizaron ciertas variables, su transformación posterior debe mantenerse compatible; la documentación de scikit-learn advierte que omitir en prueba o producción las transformaciones utilizadas en entrenamiento cambia el espacio de características y perjudica el desempeño, y presenta las secuencias de transformaciones como una forma de reducir esa inconsistencia (scikit-learn developers, s.f.).

Ese ejemplo permite comprender por qué una salida desacertada no demuestra necesariamente que los parámetros sean inadecuados: el programa podría haber enviado centímetros donde se esperaban metros, permutado columnas o interpretado como ausencia un cero legítimo; el modelo procesa lo que recibe según sus convenciones; establecer si esas convenciones se respetaron es un paso previo al juicio sobre su capacidad.

### Modelo, algoritmo y ejemplar ajustado

Un algoritmo describe un procedimiento, mientras un ejemplar ajustado incorpora el resultado específico de

aplicar un procedimiento a ciertos datos y decisiones de configuración; identificar el algoritmo ayuda a entender cómo se obtuvo un modelo, pero no permite reconstruir sus salidas sin conocer el ejemplar concreto y sus condiciones de ejecución.

La ficha de un modelo cumple una función documental, no ejecutiva; Mitchell et al. (2019) proponen acompañar modelos entrenados con información sobre usos previstos, condiciones de evaluación, variaciones de desempeño y limitaciones; esa propuesta hace visible algo decisivo: la capacidad debe describirse junto con el contexto donde se observó, y una ficha no sustituye la prueba de una aplicación posterior que haya cambiado las entradas o el propósito.

### Qué añade la aplicación

Una aplicación es el conjunto de programas y recursos que organiza un servicio o una tarea para sus usuarios, e incorpora modelos de IA cuando los utiliza para alguna parte de ese trabajo; puede tener una pantalla conversacional, un formulario, una integración en un sistema administrativo o un proceso sin interfaz gráfica; la



presencia de una ventana de chat no es su característica definitoria.

La aplicación decide qué solicitud admite y cómo la traduce a entradas procesables, qué modelos invoca y en qué orden, qué información añade o elimina y qué hace con las salidas; también establece cómo comunica un error, dónde guarda un resultado y si la persona puede revisarlo antes de que produzca consecuencias fuera del programa.

Un servicio de inferencia es una pieza intermedia frecuente: recibe solicitudes de ejecución y devuelve salidas del modelo a través de una interfaz de programación de aplicaciones, denominada API por sus siglas en inglés; aunque ese servicio necesite autenticación, control de carga y registros operativos, todavía puede ser únicamente un componente de una aplicación que atiende un proceso más amplio.

La infraestructura proporciona los recursos de ejecución y comunicación, como procesadores, memoria, almacenamiento y redes; su funcionamiento puede afectar tiempos de respuesta, disponibilidad y límites de tamaño; es útil distinguirla conceptualmente de la lógica de la

aplicación, aunque ambas se distribuyan en un mismo proveedor o en una misma computadora.

Sculley et al. (2015) examinan cómo las dependencias de datos, la configuración y las interacciones entre componentes generan costes de mantenimiento en sistemas de aprendizaje; su argumento se dirige al conjunto que rodea al código predictivo; la contribución pertinente para este artículo es que conservar un modelo no basta para mantener estable un sistema cuyo entorno y dependencias cambian.

La Tabla 1 ofrece una descomposición funcional sencilla: las piezas son roles, no necesariamente programas independientes, y algunas soluciones combinan varias dentro de un único proceso; su utilidad consiste en orientar una pregunta verificable sobre cada transición, sin exigir que todo producto incluya exactamente los mismos componentes.

**Tabla 1**. *Componentes de una aplicación de IA y evidencia útil para distinguirlos*

| Componente | Función posible | Evidencia útil |
|---|---|---|
| **Interfaz** | Recibir la solicitud y presentar el resultado | Entrada y salida visibles, avisos y estados mostrados |
| **Extractor** | Obtener texto y estructura de archivos | Texto extraído, páginas y elementos omitidos |
| **Recuperador** | Seleccionar información pertinente | Fragmentos seleccionados y procedencia comprobable |
| **Modelo** | Transformar la entrada efectiva en una salida | Identificador, configuración y entrada recibida |
| **Herramienta** | Realizar una operación fuera de la generación | Respuesta de ejecución y estado del recurso afectado |

| Componente | Función posible | Evidencia útil |
|---|---|---|
| **Control** | Verificar permisos y condiciones de operación | Autorización aplicable y motivo de aceptación o bloqueo |

*Nota.* Elaboración propia a partir de las funciones examinadas en el artículo; la tabla no describe un producto comercial específico.

### Una buena salida puede entregarse mal

El modelo podría obtener la cifra correcta y el programa mostrarla con una unidad equivocada; también podría generar un texto adecuado que una plantilla corte al exportarlo; en esos casos, evaluar únicamente la salida interna atribuiría un éxito a un proceso que no entregó el resultado requerido al usuario.

La situación inversa también importa: una aplicación puede corregir un formato, comprobar una suma o pedir revisión antes de presentar una salida defectuosa; el resultado final puede ser aceptable aunque el primer



intento del modelo no lo fuera, por lo que atribuir el desempeño observado exclusivamente a ese intento elimina del análisis el efecto de controles y correcciones.

La distinción no busca proteger al modelo de la crítica ni responsabilizar siempre al programa que lo rodea; busca que cada afirmación tenga el objeto adecuado; la misma aplicación puede contener errores de extracción, interpretación y presentación en una sola ejecución, y describir la primera causa identificada como explicación total puede dejar abiertos fallos adicionales.

### Cuatro configuraciones posibles

Las siguientes configuraciones son construcciones explicativas; representan relaciones funcionales plausibles y no implementaciones reales medidas en esta investigación; se presentan para mostrar qué permanece igual al reutilizar un modelo y qué puede variar cuando se modifica la aplicación, sin hacer depender la distinción de una marca comercial.

### Clasificación dentro de un formulario

Una mesa de atención recibe reportes sobre equipos y utiliza un clasificador para proponer la categoría «hardware», «software» o «conectividad»; el formulario valida que exista una descripción, la convierte a la representación esperada, obtiene puntuaciones del modelo y muestra la categoría elegida; una regla adicional remite a revisión los casos donde la diferencia entre las dos puntuaciones mayores es pequeña.

El modelo de este ejemplo produce puntuaciones, mientras la aplicación establece cómo traducirlas a una ruta de atención; si el equipo modifica el umbral de revisión, cambia la cantidad de asuntos que pasan a una persona sin que necesariamente se alteren los parámetros; por eso «ahora resuelve más casos automáticamente» es una afirmación sobre el proceso y requiere examinar el cambio de criterio, además de la calidad predictiva.

Supóngase, de manera ilustrativa, que un reporte recibe puntuaciones de 0.52, 0.45 y 0.03, y que la regla exige una diferencia mínima de 0.10 entre las dos mayores; la diferencia observada es 0.07 y el caso se remite a revisión;

si la regla cambia a 0.05, el mismo resultado del modelo produce una asignación automática, sin que los números por sí solos acrediten probabilidades bien calibradas.

### Recomendación con selección y ordenamiento

Una aplicación de consulta bibliográfica recibe una búsqueda y debe mostrar cinco documentos; una primera etapa obtiene candidatos, un modelo los ordena y un filtro excluye materiales que el usuario no tiene permiso de consultar; la lista visible combina pertinencia estimada, disponibilidad y reglas de acceso, de modo que su ausencia de cierto documento admite varias explicaciones.

El modelo de ordenamiento puede ser adecuado y no llegar a examinar una obra que el buscador inicial omitió; también puede colocarla en primer lugar y que el filtro la retire correctamente por falta de acceso; solicitar «un mejor modelo» antes de revisar esas transiciones podría aumentar costes sin resolver la causa de la experiencia observada.

Si la organización cambia de cinco a diez resultados, la cobertura visible puede crecer aunque el orden calculado



sea idéntico; si cambia el catálogo o los permisos, dos personas podrían recibir listas diferentes con el mismo modelo y la misma consulta; la comparación debe registrar el universo disponible y la política aplicada, además del identificador del componente que ordena.

### Generación apoyada en documentos

Una aplicación documental puede extraer el texto de archivos, dividirlo en unidades de consulta, recuperar los pasajes pertinentes y añadirlos a la entrada de un generador; en esa configuración, parte de la información utilizada procede de documentos recuperados durante la solicitud y no tiene que haber sido incorporada a los parámetros del generador.

Lewis et al. (2020) estudian una propuesta que combina generación con memoria paramétrica y recuperación sobre un índice externo; también examinan el cambio de esa memoria externa en una prueba de actualización; el trabajo proporciona un antecedente concreto para distinguir ambas rutas, pero su entrenamiento conjunto y sus variantes experimentales no deben atribuirse automáticamente a todas las aplicaciones que hoy utilizan



la denominación RAG, sigla inglesa de generación aumentada por recuperación.

En el ejemplo aquí planteado, agregar un manual al catálogo puede hacer que sus pasajes estén disponibles para una consulta posterior sin ajustar los parámetros del generador; todavía será necesario que se indexe correctamente, que el recuperador lo encuentre y que el contenido relevante se entregue; el resultado actualizado permite afirmar que la aplicación utilizó nueva información cuando hay evidencia del recorrido, pero no basta para afirmar que el modelo la aprendió permanentemente.

Una aplicación que envía el documento completo y otra que selecciona fragmentos también pueden producir resultados diferentes con el mismo generador; esa diferencia no demuestra que la primera sea siempre superior; la selección puede reducir ruido y la lectura extensa puede conservar relaciones que se perderían al fragmentar, por lo que cada solución necesita casos que permitan observar la ventaja y el coste de su elección.

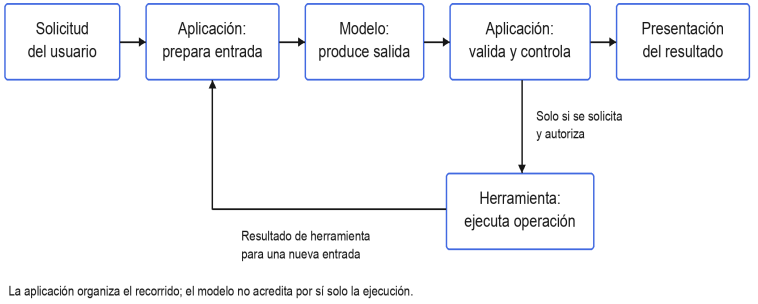
### Generación con ejecución de herramientas

Una cuarta configuración permite que el modelo produzca una solicitud estructurada para consultar una base, hacer un cálculo o crear un archivo; la aplicación valida esa solicitud, comprueba permisos y la envía a una herramienta cuando corresponde; el resultado de la operación puede regresar al modelo para que redacte la respuesta final o ser mostrado directamente por la interfaz.

Una solicitud de herramienta es una salida que debe interpretarse conforme a un contrato de datos, con campos y tipos definidos; no equivale a la operación externa descrita; si un campo exige el identificador de una carpeta, un texto que parece un nombre de carpeta puede ser insuficiente, y si falta autorización el programa debe impedir la acción aunque los argumentos sean técnicamente válidos.

La Figura 1 representa esa separación y el retorno del resultado, sin sugerir que toda pregunta active herramientas; el bloque de control permanece fuera del texto que genera el modelo en este esquema, lo que permite contrastar la propuesta, la autorización y la evidencia de ejecución como hechos distintos.

**Figura 1**. *Recorrido conceptual de una solicitud con herramientas opcionales*




*Nota.* Diagrama de elaboración propia; las flechas muestran intercambios de información y la ruta inferior solo se recorre si se solicita y autoriza una herramienta. No representa un producto real ni resultados experimentales.

## Datos, contexto y estado

Decir que una aplicación «tiene» información puede significar que el archivo está almacenado, que su contenido fue indexado, que ciertos fragmentos se recuperaron o que efectivamente llegaron al modelo; son estados relacionados, pero ninguno implica automáticamente el siguiente; un diagnóstico que los



confunda puede pasar por alto el punto preciso donde se interrumpió el recorrido.

Los datos de entrenamiento pertenecen al proceso que produjo o ajustó el modelo; el contexto de una ejecución comprende la información suministrada para esa ejecución; la base documental de la aplicación es un recurso externo, y el estado de la sesión conserva información operativa como mensajes anteriores, selecciones o resultados parciales; estos cuatro lugares deben describirse por su función, incluso cuando la interfaz los presente juntos.

La separación tiene una consecuencia sencilla: recuperar una preferencia guardada y volver a enviarla al modelo puede personalizar una respuesta sin modificar sus parámetros; la experiencia de continuidad proviene entonces de la combinación entre almacenamiento y preparación de entradas, y no autoriza a deducir que cada conversación haya reentrenado el componente.

También puede ocurrir que una parte del estado desaparezca o no se incluya por límites de la ejecución; una persona puede ver el historial completo y el modelo

recibir únicamente una selección o un resumen; esa posibilidad es un criterio de investigación del incidente, no una afirmación sobre todos los productos, y solo puede confirmarse con documentación o registros adecuados del entorno examinado.

### Almacenamiento y acceso efectivo

Un archivo que figura en un catálogo puede no estar disponible para una cuenta concreta, para una región de servicio o para una herramienta determinada; el acceso debe evaluarse en la identidad y el momento de la operación, ya que demostrar que un administrador puede abrirlo no prueba que el proceso ejecutado en nombre del usuario tenga ese mismo permiso.

Por otra parte, disponibilidad no significa integridad de lectura: un PDF con texto en columnas puede extraerse en orden incorrecto, una imagen escaneada puede requerir reconocimiento óptico y una tabla puede perder relaciones entre encabezados y cifras; el contenido nominal del archivo y la representación que llega al modelo son objetos distintos cuya correspondencia debe comprobarse cuando el resultado dependa de esas relaciones.



### Caché, reutilización y cambios aparentes

Una caché conserva resultados para reutilizarlos bajo condiciones definidas; si una aplicación devuelve una respuesta previamente calculada, el usuario puede recibir una salida sin que esa solicitud haya ejecutado nuevamente el modelo; la existencia de esta ruta debe comprobarse en la implementación, y su relevancia práctica aparece cuando se espera que un documento recién modificado cambie la respuesta.

Un error de invalidación puede mantener una respuesta antigua aunque el archivo ya esté actualizado; culpar al generador por ignorar el cambio sería prematuro si ni siquiera hubo una nueva ejecución; en sentido inverso, una aplicación que elimina correctamente la respuesta almacenada puede cambiar su salida con el mismo modelo, sin que esa variación permita hablar de una actualización del componente.



### Identidad y versiones

Para afirmar que dos ejecuciones utilizaron «el mismo modelo» se necesita especificar el nivel de identidad: misma familia, misma versión anunciada, mismo artefacto de parámetros o misma configuración efectiva; la precisión necesaria depende de la pregunta; para una explicación introductoria puede bastar la familia, mientras que una comparación causal requiere un registro mucho más específico.

Un nombre de servicio puede funcionar como alias que el proveedor resuelve a una versión concreta; en una aplicación hipotética, mantener el alias «modelo-principal» no impediría cambiar el componente que lo atiende; sin documentación o registros del proveedor, el observador externo no puede resolver esa incertidumbre solamente porque el nombre mostrado en la interfaz siga igual.

Además del modelo, conviene identificar las instrucciones, la preparación de entradas, el índice documental, las herramientas, los permisos y la versión del programa que organiza la tarea; no es necesario convertir cada conversación en una auditoría exhaustiva, pero sí



reconocer que un único número de versión de la aplicación puede ocultar cambios en varias piezas a la vez.

Amershi et al. (2019) describen, en un estudio de equipos de Microsoft, dificultades de ingeniería asociadas a la gestión y el versionado de datos y a la reutilización de componentes de aprendizaje; el alcance es un estudio de caso organizacional, no una medición universal de todos los desarrollos; su utilidad aquí es mostrar por qué datos, modelo y software necesitan tratamiento conjunto durante el mantenimiento.

### Una ficha de ejecución suficiente para la pregunta

Supóngase que una aplicación identifica su versión como 3.2 y el modelo como M7, pero cambia el extractor documental entre dos días de trabajo; si empieza a omitir notas al pie, mantener esos dos identificadores no basta para reproducir el incidente; el registro necesita la versión del extractor y la representación de entrada afectada, además del modelo y la solicitud.

La Tabla 2 propone una ficha mínima por capas que debe ajustarse al riesgo del trabajo; su finalidad es hacer posible

una comparación y no imponer la conservación de todo dato en cualquier entorno; los ejemplos de identificación son ficticios y no corresponden a versiones reales de un proveedor.

**Tabla 2**. *Qué debe distinguirse al registrar versiones y estado*

| Capa | Pregunta que resuelve | Evidencia proporcionada |
|---|---|---|
| **Modelo** | ¿Qué componente produjo la salida? | Identificador de versión o artefacto y configuración pertinente |
| **Aplicación** | ¿Qué lógica organizó la tarea? | Versión del programa, instrucciones y ruta elegida |
| **Información** | ¿Qué material estuvo disponible y se utilizó? | Versión documental, extracción y fragmentos efectivamente enviados |

| Capa | Pregunta que resuelve | Evidencia proporcionada |
|---|---|---|
| **Ejecución** | ¿Qué condiciones existían en ese momento? | Fecha, sesión, resultados parciales y respuestas de herramientas |
| **Acceso** | ¿Qué podía hacerse en nombre de esa cuenta? | Perfil autorizado y decisión del control, sin secretos |

*Nota.* Propuesta de elaboración propia; conservar únicamente el detalle necesario y autorizado para la finalidad de comprobación.

### Cambiar una pieza exige una prueba compatible

Sustituir el generador manteniendo la aplicación permite investigar una pregunta diferente de sustituir el recuperador manteniendo el generador; si ambas piezas cambian simultáneamente, la mejora global puede observarse, pero atribuirla a una sola exige pruebas adicionales; la expresión «el nuevo modelo funciona



mejor» necesita una comparación donde el modelo sea realmente la diferencia relevante.

También debe comprobarse la compatibilidad de salidas: un componente nuevo puede utilizar campos distintos, ordenar categorías de otra forma o expresar una negativa que el programa anterior interprete como resultado válido; una integración que solo verifica que llegó una respuesta deja sin controlar el significado que esa respuesta tendrá en los siguientes pasos.

Congelar componentes facilita reproducir un incidente, pero introduce el riesgo de seguir utilizando información o dependencias desactualizadas; actualizar todo permanentemente dificulta reconstruir qué ocurrió; la solución práctica propuesta es conservar versiones identificables, evaluar cambios deliberados y poder volver a una configuración conocida cuando la situación lo requiera, con un registro que explique qué se mantuvo y qué se sustituyó.



### Dos niveles de evaluación

Una evaluación de modelo puede proporcionar entradas previamente preparadas y comprobar sus salidas contra un criterio; una evaluación de aplicación debe comenzar en una solicitud realista y terminar en un resultado verificable para el usuario; ambas son necesarias porque una describe capacidad bajo condiciones delimitadas y la otra examina si el proceso completo produce el resultado requerido.

Para un clasificador, la primera prueba puede medir las categorías acertadas sobre reportes ya representados; la segunda incorpora recepción del reporte, validación, ejecución, enrutamiento y posibilidad de corregir una asignación; si la aplicación pierde reportes antes de clasificarlos, un excelente porcentaje sobre los que sí llegaron no resuelve el problema operativo.

Breck et al. (2017) presentan una rúbrica de 28 pruebas y necesidades de supervisión que abarca datos, desarrollo del modelo, infraestructura y seguimiento; su planteamiento incluye contrastar medidas fuera de producción con efectos reales de uso; aquí se retoma esa separación como fundamento para no equiparar una

métrica del componente con la aptitud del servicio completo, sin convertir la rúbrica en certificación automática.

### El denominador decide qué se está midiendo

Considérese un ejemplo hipotético de 100 preguntas documentales: en 80 llega al generador el pasaje suficiente y en 72 de esas 80 la respuesta satisface el criterio de contenido; el porcentaje condicional de acierto es 72/80, equivalente a 90 %, pero la proporción de solicitudes iniciales con ese resultado es 72/100, equivalente a 72 %; ambas cifras pueden ser correctas y responder preguntas distintas.

Supóngase además que seis de las 72 respuestas correctas se entregan con un vínculo de fuente roto, y que el criterio de éxito exige contenido correcto y referencia accesible; quedan 66 solicitudes completamente resueltas de las 100 iniciales; informar únicamente 90 % ocultaría tanto la pérdida previa de información como el defecto de entrega, aunque esa cifra siguiera describiendo correctamente el subconjunto condicionado.

El cálculo no pretende estimar una tasa real ni afirmar que las etapas sean independientes: se han definido conteos anidados dentro de un escenario construido; si los fallos se superponen o una respuesta se obtiene por otra ruta, debe reconstruirse cada caso antes de multiplicar proporciones o sumar errores, pues los porcentajes de conjuntos diferentes no se combinan libremente.

La selección de casos también importa: excluir todas las solicitudes rechazadas, los archivos difíciles o las operaciones sin permiso podría mejorar artificialmente el indicador observado; un informe claro distingue resultados correctos, fallos, negativas justificadas y casos no evaluables, de acuerdo con reglas definidas antes de mirar qué alternativa sale favorecida.

### La unidad de éxito necesita contenido

En una tarea de creación de archivos, «respondió» es un criterio insuficiente; el archivo debe existir, abrirse, contener el material solicitado y estar disponible en el destino autorizado; para una búsqueda, encontrar un documento nominalmente relacionado no acredita que el pasaje responda la pregunta, y para una acción externa

una afirmación de ejecución no equivale a confirmación del recurso afectado.

La evaluación puede separar dimensiones que no conviene fundir: corrección del contenido, soporte documental, cumplimiento de formato, terminación del proceso y comportamiento ante falta de permiso; una puntuación agregada es útil solo si conserva visibles los fallos que por su gravedad no deben compensarse con buena redacción u otras ventajas menores.

Ribeiro et al. (2020) proponen pruebas de comportamiento mediante capacidades y transformaciones de entradas, incluidas situaciones donde la salida debería mantenerse o cambiar de una forma esperada; para la distinción que nos ocupa, esa idea permite formular pares de casos que investiguen una función específica; las adaptaciones documentales que siguen son propuestas del presente artículo, no resultados reportados por esos autores.

Un caso puede repetirse con el mismo texto y distinto nombre de archivo para comprobar si una asociación indebida altera la respuesta; otro puede cambiar una fecha sustantiva para observar si el resultado se actualiza; las



expectativas necesitan justificación: sustituir una palabra no siempre conserva el significado y exigir invariancia sin revisar esa condición generaría falsas alarmas.

### Tres casos para localizar el cambio

Los tres casos siguientes son hipotéticos, con datos creados exclusivamente para explicar el método de diagnóstico; no describen empresas, expedientes ni pruebas ejecutadas; su interés está en mostrar qué observación permitiría distinguir hipótesis rivales y qué conclusión seguiría sin respaldo, incluso después de encontrar una explicación plausible.

### Caso 1. El anexo que existía, pero no llegó

Una organización carga un manual de ocho páginas y un anexo de dos páginas; el manual establece un plazo general de diez días y el anexo introduce una excepción de cinco días para cierta categoría; una persona pregunta por un caso que pertenece a esa categoría y la aplicación responde diez días, citando únicamente el manual.



La primera comprobación consiste en abrir los dos archivos y verificar que el anexo aportado es efectivamente el pertinente; después se examina el texto extraído; si la copia del anexo está escaneada y el extractor devuelve una cadena vacía, la hipótesis de pérdida de entrada tiene un soporte concreto, mientras que inferir que el generador «no entendió la excepción» excedería la evidencia disponible.

Para continuar se entrega al mismo componente, en un entorno de prueba, una entrada controlada que incluya ambos pasajes y la pregunta original; si ahora responde cinco días, aumenta la evidencia a favor de que la ausencia del anexo contribuyó al fallo inicial; ese resultado no prueba que el componente interpretará bien todas las excepciones, ni que la reparación de la extracción sea suficiente para cualquier documento.

La corrección debería comprobarse desde el punto de entrada original: volver a cargar archivos con esa característica y observar que el proceso detecta o extrae el anexo, selecciona la excepción y conserva su referencia; una prueba que pegue manualmente el texto



solo evalúa la ruta reparada de forma parcial, porque evita precisamente el paso que había fallado.

Si el anexo sí fue extraído, se revisa si quedó fuera de la recuperación; si llegó a la entrada efectiva, la investigación pasa a interpretación o seguimiento de la solicitud; y si la salida interna menciona correctamente cinco días pero la plantilla muestra diez, el defecto queda después del modelo, lo que ilustra por qué no debe fijarse el diagnóstico antes de reconstruir el recorrido.

### Caso 2. Dos modelos y dos recuperadores

Una oficina desea comparar dos generadores, M1 y M2, para responder preguntas sobre procedimientos internos; inicialmente ejecuta M1 con el recuperador R1 y M2 con R2, de manera que cada alternativa difiere en dos componentes; si la segunda combinación produce más respuestas útiles, se habrá comparado una solución contra otra, pero todavía no se habrá aislado la contribución del generador.

Para mostrar la diferencia, supóngase un conjunto fijo de 100 preguntas y documentos idénticos, con criterios de



corrección establecidos previamente; la combinación M1 con R1 resuelve 60, M2 con R1 resuelve 64, M1 con R2 resuelve 78 y M2 con R2 resuelve 80; estos conteos son ilustrativos, no observaciones experimentales ni estimaciones sobre modelos existentes.

Dentro del escenario, cambiar M1 por M2 con R1 representa cuatro casos adicionales, mientras hacerlo con R2 representa dos; cambiar R1 por R2 con M1 representa dieciocho casos adicionales y hacerlo con M2 representa dieciséis; la comparación inicial entre 60 y 80 muestra veinte casos de diferencia entre soluciones completas, pero atribuir los veinte al cambio de modelo sería incorrecto.

Los números sugieren que, en ese conjunto construido, revisar la recuperación merece atención prioritaria; no demuestran significación estadística ni superioridad universal, porque hacen falta los resultados por caso, repeticiones cuando procedan, representatividad y un análisis de incertidumbre; tampoco permiten sumar mecánicamente mejoras como si los componentes no interactuaran.

Además de contar éxitos, conviene conservar qué preguntas cambian: una alternativa podría ganar en consultas simples y perder en las pocas excepciones de mayor impacto; si el criterio de adopción prohíbe cierto tipo de error grave, una mejora del promedio no sustituye la revisión de esos casos ni la decisión sobre si el riesgo residual resulta aceptable.

Un último control consiste en mantener comparable el presupuesto de información y ejecución: si una configuración recibe más documentos, dispone de más intentos o utiliza una herramienta adicional, esos cambios deben describirse; pueden ser decisiones legítimas de diseño, pero convierten la comparación en una prueba de configuraciones más amplias y no únicamente del modelo base.

### Caso 3. Un archivo creado y un envío pendiente

Una persona solicita preparar un informe y dejarlo disponible para revisión; la aplicación usa un modelo para redactar, una herramienta para crear el archivo y otra para exportarlo; el modelo puede terminar su borrador antes de que finalice la escritura, y la creación del archivo editable



puede completarse antes de que exista una copia en el formato adicional solicitado.

Si el programa solo recibe un identificador de trabajo pendiente, la respuesta correcta debe reflejar ese estado; declarar que el informe final está listo confundiría aceptación con terminación; la comprobación debe localizar el archivo, abrirlo y verificar que corresponde a la versión prevista, además de distinguir qué salida sigue pendiente.

Supóngase ahora que la herramienta crea el archivo y la exportación falla por un carácter incompatible en el nombre; el texto sustantivo puede ser correcto y el modelo no necesita reescribirlo; corregir el nombre permitido y repetir la exportación es una reparación de integración, cuya eficacia se comprueba sobre el archivo resultante y no mediante una nueva declaración del asistente.

El envío a una tercera persona constituye otra operación: incluso si el archivo existe, estar autorizado para prepararlo no implica permiso para compartirlo; una aplicación que conserva el borrador y solicita la autorización correspondiente está cumpliendo un límite del



proceso, y esa detención no debe contarse como incapacidad de redacción del modelo.

También pueden presentarse fallos por reintento: si la operación de creación se completó pero se perdió la confirmación, repetirla sin consultar el estado puede producir duplicados; la comprobación del identificador de operación y del recurso existente permite distinguir una ejecución incierta de una ejecución inexistente, y evita que la urgencia por terminar agrave el incidente.

La Tabla 3 reúne síntomas que motivan investigación; las hipótesis orientan la búsqueda de evidencia y no reemplazan el diagnóstico; conserva la idea de revisar componentes separados, con una precisión adicional sobre los estados de operaciones que pueden tardar o fallar después de ser aceptadas.

**Tabla 3**. *Síntomas que necesitan diagnóstico por componente*

| Síntoma | Hipótesis posible | Comprobación propuesta |
|---|---|---|
| **Omite un anexo** | Su contenido no llegó al modelo | Cotejar archivo, extracción, recuperación y entrada efectiva |
| **Cita otro documento** | La procedencia quedó mal vinculada | Comparar pasaje, identificador y vínculo presentado |
| **Afirma que creó un archivo** | Solo produjo texto o la ejecución sigue pendiente | Abrir el recurso real y verificar estado y contenido |
| **No puede enviar** | Falta permiso o una integración necesaria | Revisar capacidades autorizadas y respuesta de la herramienta |
| **Cambió el resultado** | Variaron entradas, estado o componentes | Comparar registros de ambas ejecuciones |
| **Ignora una actualización** | Se reutilizó información antigua | Revisar versión del índice, recuperación y caché, si existe |



*Nota.* Elaboración propia; varias causas pueden concurrir en el mismo incidente y una hipótesis requiere evidencia antes de aceptarse.

## Mitos que la separación permite corregir

«Si usa el mismo modelo, debe dar el mismo resultado» omite la preparación de entradas, la configuración, el estado y las operaciones posteriores; la comparación necesita controlar esas condiciones y reconocer las fuentes de variación de la ejecución; las respuestas distintas son un dato que requiere explicación, no prueba automática de que hubo un modelo diferente.

«La nueva versión aprendió mis documentos porque ahora los cita» confunde una experiencia actualizada con una modificación de parámetros; una aplicación puede incorporar documentos al contexto o a un índice externo; para afirmar aprendizaje del modelo haría falta evidencia de un proceso de ajuste, mientras que la mera presencia de una cita solo permite examinar qué fuente se utilizó en esa respuesta.



«Un modelo más capaz arregla cualquier aplicación» pasa por alto los fallos que ocurren antes y después de su ejecución; ningún aumento de capacidad permite interpretar un anexo que nunca llegó por la ruta utilizada, aunque otro diseño sí pudiera recuperarlo; tampoco corrige automáticamente una salida guardada en el destino equivocado o un permiso mal configurado.

«Una función anunciada es una acción realizada» confunde disponibilidad con uso efectivo; un servicio puede ofrecer consulta de archivos y no haberla ejecutado en la solicitud observada; del mismo modo, conocer la definición de una herramienta no demuestra que se disponga de una sesión válida, de la autorización adecuada o de una confirmación de éxito.

### Límites y responsabilidades

La división por componentes es una herramienta analítica, pero las interacciones impiden tratar siempre cada pieza como si fuera independiente; un modelo puede haberse adaptado a cierto formato de entrada y un cambio del extractor puede afectar su desempeño sin que exista un error sintáctico; localizar el cambio ayuda a investigar la

causa, aunque no garantiza que una reparación aislada conserve todos los comportamientos deseados.

La separación física tampoco resuelve por sí sola la responsabilidad organizacional: un proveedor puede desarrollar varias piezas y otro encargarse de su operación, mientras la entidad usuaria define el propósito y las condiciones de supervisión; este artículo propone identificar funciones y evidencias, sin deducir responsabilidades jurídicas de un diagrama técnico ni sustituir el análisis aplicable a una relación contractual concreta.

El marco AI RMF 1.0 (Tabassi, 2023) caracteriza los sistemas de IA como sociotécnicos y relaciona sus riesgos con la interacción entre aspectos técnicos, personas y contexto de despliegue; esa perspectiva permite reconocer que una aplicación técnicamente funcional puede incorporarse a un procedimiento mal diseñado; la prueba del software no resuelve por sí sola si la tarea, los criterios o los efectos sobre las personas son adecuados.

Una revisión humana puede formar parte de la aplicación en sentido amplio del proceso, pero necesita condiciones

efectivas: acceso a la información, tiempo suficiente y posibilidad real de corregir o detener la operación; incluir un botón de aprobación sin esos recursos no demuestra que la persona pueda detectar un error que el diseño le impide ver.

### Qué puede saberse desde fuera

En servicios cerrados, el usuario puede observar solicitudes y resultados sin conocer la entrada exacta que recibió el modelo, las rutas internas o los identificadores completos; en ese caso, la conclusión debe limitarse al comportamiento observable; es legítimo decir que la aplicación falló una tarea comprobada, aunque todavía no se pueda determinar cuál de sus componentes originó el defecto.

También cabe reconocer información declarada por el proveedor como tal: si la documentación afirma utilizar determinada versión, esa declaración tiene un alcance distinto de la comprobación independiente de un artefacto; el rigor consiste en conservar la diferencia entre lo observado, lo documentado por terceros y lo inferido, en



lugar de presentar todos esos elementos con el mismo grado de certeza.

La ausencia de acceso interno no impide toda evaluación: puede probarse si los archivos abren, si los vínculos conducen al pasaje correcto, si una solicitud no autorizada se detiene y si una modificación controlada del documento cambia la respuesta como debería; lo que se limita es la atribución causal fina, no la posibilidad de juzgar el resultado que efectivamente recibe el usuario.

### Capacidad, tiempo y coste no coinciden

Una aplicación puede mejorar la calidad utilizando varios intentos, consultas o verificaciones; ese beneficio debe compararse con el tiempo y los recursos necesarios para completar la tarea; la velocidad del modelo aislado no equivale a la latencia total si el proceso espera extracción, búsqueda, herramientas o colas de servicio.

Considérese una secuencia hipotética con 1.2 segundos de recuperación, 2.5 de generación, 0.8 de ejecución y 0.5 de presentación; si no hay solapamiento, la suma es 5.0 segundos; reducir a la mitad la generación la deja en 1.25



segundos y el total pasa a 3.75, una reducción de 1.25 segundos, equivalente al 25 % del tiempo inicial, en lugar de una reducción del 50 % del proceso completo.

El cálculo solo describe una ejecución serial idealizada y no mide un producto; si las etapas se ejecutan en paralelo, si hay reintentos o si el servicio acumula solicitudes, la suma simple deja de representar el recorrido; su función es mostrar que optimizar una pieza y optimizar la tarea son objetivos relacionados cuya diferencia debe medirse con la unidad adecuada.

### Examinar una aplicación

La primera pregunta práctica es cuál es el resultado que se necesita: una propuesta de categoría, una respuesta sustentada, un documento editable o una operación confirmada; esa definición evita exigir a un modelo una función que corresponde a una integración ausente y evita aceptar como éxito una salida que todavía deja sin resolver la tarea.

La segunda es qué recorrido conecta la entrada con ese resultado: conviene identificar datos aportados,



transformaciones necesarias, modelos utilizados y operaciones externas; no hace falta conocer cada detalle de implementación para empezar, pero sí reconocer las transiciones de las que depende la afirmación de cumplimiento y pedir evidencia proporcional a su importancia.

La tercera pregunta se dirige a los límites: qué ocurre con archivos ilegibles, fuentes contradictorias, herramientas indisponibles o permisos insuficientes; una demostración donde todo funciona muestra una ruta favorable, mientras que esas situaciones permiten comprobar si la aplicación comunica incertidumbre, conserva estados y permite recuperarse sin inventar resultados.

La cuarta se refiere a cambios y continuidad: quién puede modificar el modelo, el programa, el índice y los permisos, cómo se informa una actualización y qué evidencia queda para comparar versiones; en trabajos repetidos, disponer de una versión identificable y de casos de regresión resulta más útil que una descripción genérica de capacidades que no pueda vincularse con una ejecución concreta.



La quinta es qué se verificará después de cada tarea: abrir el documento, cotejar las referencias, revisar el destino o confirmar el estado de la operación; la revisión debe centrarse en las consecuencias que importarían si hubiera un fallo, y no en pedir una explicación adicional al mismo texto como sustituto de la comprobación externa.

La decisión final debe corresponder al conjunto realmente disponible: una aplicación sencilla con fuentes accesibles, controles claros y resultados comprobables puede ser más adecuada para cierto trabajo que otra con un modelo mejor puntuado pero sin las integraciones necesarias; esa comparación exige documentar las necesidades concretas y los resultados observados, sin convertir la simplicidad ni la complejidad en virtudes automáticas.



## Conclusión

Modelo y aplicación corresponden a niveles distintos de análisis: el primero identifica una representación computacional y sus condiciones de ejecución, mientras la segunda organiza entradas, recursos, controles y resultados para una tarea; su relación explica por qué una misma capacidad puede producir experiencias diferentes y por qué un fallo visible no revela de inmediato su causa.

Los casos examinados muestran tres consecuencias: mejorar el modelo no repara automáticamente una entrada perdida, comparar dos aplicaciones no aísla por sí solo la capacidad de sus modelos y una declaración de finalización no acredita una acción externa; la evaluación del componente y la del proceso completo deben complementarse, con criterios que distingan calidad de contenido, integridad de entrega y límites de autorización.

Comprender esa diferencia permite usar la IA con preguntas más concretas y comprobaciones mejor dirigidas; en lugar de depender exclusivamente del prestigio de una denominación, se puede examinar qué trabajo realiza cada parte y qué evidencia respalda el resultado; ese cambio de enfoque no elimina la

incertidumbre, pero vuelve más precisa la decisión sobre cuándo confiar, qué revisar y dónde intervenir cuando algo falla.

## Referencias

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., y Zimmermann, T. (2019). Software engineering for machine learning: A case study [Ingeniería de software para aprendizaje automático: Un estudio de caso]. En *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (pp. 291–300). IEEE. https://doi.org/10.1109/ICSE-SEIP.2019.00042

Breck, E., Cai, S., Nielsen, E., Salib, M., y Sculley, D. (2017). The ML test score: A rubric for ML production readiness and technical debt reduction [La puntuación de pruebas de aprendizaje automático: Una rúbrica para la preparación para producción y la reducción de deuda técnica]. En *2017 IEEE International Conference on Big Data (Big Data)* (pp. 1123–1132). IEEE. https://doi.org/10.1109/BigData.2017.8258038

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., y Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks [Generación aumentada mediante recuperación para tareas de procesamiento del lenguaje que requieren conocimiento]. En H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, y H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 9459–9474). Curran Associates. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I. D., y Gebru, T. (2019). Model cards for model reporting [Fichas de modelos

para documentar modelos]. En *Proceedings of the Conference on Fairness, Accountability, and Transparency* (pp. 220–229). Association for Computing Machinery. https://doi.org/10.1145/3287560.3287596

ONNX. (s.f.). *Open Neural Network Exchange intermediate representation (ONNX IR) specification* [Especificación de la representación intermedia de ONNX]. Recuperado el 2 de octubre de 2026, de https://onnx.ai/onnx/repo-docs/IR.html

Ribeiro, M. T., Wu, T., Guestrin, C., y Singh, S. (2020). Beyond accuracy: Behavioral testing of NLP models with CheckList [Más allá de la exactitud: Pruebas de comportamiento de modelos de procesamiento del lenguaje con CheckList]. En D. Jurafsky, J. Chai, N. Schluter, y J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4902–4912). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.442

scikit-learn developers. (s.f.). *Common pitfalls and recommended practices* [Errores frecuentes y prácticas recomendadas]. Recuperado el 2 de octubre de 2026, de https://scikit-learn.org/stable/common_pitfalls.html

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., y Dennison, D. (2015). Hidden technical debt in machine learning systems [Deuda técnica oculta en los sistemas de aprendizaje automático]. En C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, y R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 28, pp. 2503–2511). Curran Associates. https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html

Tabassi, E. (2023). *Artificial intelligence risk management framework (AI RMF 1.0)* [Marco de gestión de riesgos de inteligencia artificial] (NIST AI 100-1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.100-1

