LA LEY IALAB AI academy BY doinGlobal Thomson Reuters™

Juan Gustavo **CORVALÁN**, Mariana **SANCHEZ CAPARRÓS**, Carolina Melanie **MARTÍN**

GOBERNAR LA IA AGÉNTICA

COLABORADORAS: LOLA RAMOS PEREYRA, GISEL ALVARADO

Corvalán, Juan G.
Gobernar la IA genética / Juan G. Corvalán ; Mariana Sánchez Caparrós ; Carolina Melanie Martín. - 1a ed. - Ciudad Autónoma de Buenos Aires : La Ley, 2026.
Libro digital, PDF

Archivo Digital: descarga y online
ISBN 978-987-03-5189-4

1. Derecho. 2. Inteligencia Artificial. I. Sánchez Caparrós, Mariana II. Martín, Carolina Melanie III. Título
CDD 340

AUTORÍA

JUAN GUSTAVO CORVALÁN

Doctor en Ciencias Jurídicas y Máster en IA. Speaker en ONU, OEA, MIT y Google Talks. Director del Laboratorio de IA de la UBA (Derecho). Director del Tratado de 4 tomos de IA y Derecho. Autor y coautor de decenas de investigaciones y publicaciones en IA. Co-creador de Prometea, el primer sistema predictivo en la Justicia de Hispanoamérica (año 2017).

MARIANA SÁNCHEZ CAPARRÓS

Doctora en Ciencias Jurídicas y Magister en Derecho Administrativo. Subdirectora de UBA IALAB, donde lidera proyectos de I+D en Inteligencia Artificial, innovación y transformación digital, especializándose en la aplicación ética de la IA en el sector público.

CAROLINA MELANIE MARTÍN

Investigadora en UBA IALAB y estudiante de Ingeniería en Inteligencia Artificial. Se especializa en el diseño de agentes inteligentes y sistemas de automatización aplicados al análisis jurídico, la gestión de información compleja y la investigación. Su trayectoria integra experiencia en análisis de datos e investigación criminal con el desarrollo de soluciones basadas en Inteligencia Artificial.

COLABORADORAS

LOLA RAMOS PEREYRA

GISEL ALVARADO

ÍNDICE

NOTA PRELIMINAR
ARENAS MOVEDIZAS Y DECISIONES TERMINOLÓGICAS

Una advertencia necesaria antes de comenzar. El ecosistema de la IA agéntica se mueve a una velocidad que desafía cualquier intento de mapeo estable. Las plataformas que describimos en este material incorporan permanentemente funcionalidades nuevas, modifican sus arquitecturas o redefinen sus modelos de acceso. Esta es una condición del objeto de estudio: trabajamos sobre arenas movedizas.

Esa misma velocidad de evolución dificulta algo que en otros campos se da por sentado: el consenso terminológico. Qué es exactamente un “asistente”, qué es un “agente”, dónde empieza lo “agéntico”, qué distingue a un flujo automatizado con IA de un agente propiamente dicho, son preguntas que la industria, la academia y la práctica profesional responden hoy de maneras distintas y a veces contradictorias.

Los propios proveedores usan estos términos con criterios comerciales que no siempre coinciden con las definiciones técnicas, y las definiciones técnicas no siempre coinciden entre sí. Al momento de escribir este libro, no hallamos un vocabulario estabilizado. De hecho, el fenómeno del “blanco móvil” se está potenciando

Frente a esa situación, este trabajo adopta algunas definiciones operativas que se mantendrán consistentes a lo largo de todo el texto, pero que pueden no coincidir con las de otros textos.

En este documento llamamos *asistente basado en IA generativa* a la unidad mínima del ecosistema agéntico: plataformas como Claude, ChatGPT o Gemini que ya exhiben agencia por diseño pero operan con autonomía acotada, dentro de las ventanas de conversación o de los espacios de trabajo disponibles para los usuarios en las plataformas.

En el otro extremo del espectro vamos a ubicar a los *agentes de IA* propiamente dichos, como sistemas en los que múltiples agentes especializados, montados sobre un arnés de ingeniería de software, coordinan entre sí para ejecutar procesos completos con intervención humana mínima o, incluso, sin intervención humana.

Entre ambos polos hay espectro de posibilidades, un continuo que no es necesariamente un salto. Es decir, cada herramienta que analizaremos se ubica en algún punto de ese espectro, con mayor o menor grado de agencia y autonomía, con mayor o menor necesidad de conocimientos técnicos para diseñar y utilizar asistentes y agentes de IA.

UNA DISTINCIÓN QUE EL DOCUMENTO USARÁ SISTEMÁTICAMENTE: AGENCIA Y AUTONOMÍA

Antes de entrar en la introducción conviene precisar dos términos que solemos usar de manera intercambiable y que en el resto del material vamos a distinguir.

Agencia

La **agencia** es la capacidad de actuar sobre el entorno: invocar herramientas, modificar archivos, consultar bases de datos, enviar mensajes. Cualquier asistente con *tool use* habilitado tiene agencia.

Autonomía

La autonomía es el grado en que esa actuación ocurre sin intervención humana directa. Un asistente puede tener alta agencia porque tiene muchas herramientas disponibles, pero tiene baja autonomía porque cada uso requiere un prompt o una acción explícita del usuario. Un agente puede tener moderada agencia porque tiene un repertorio limitado de herramientas, pero posee alta autonomía porque puede ejecutar cadenas largas de acciones por diversas tareas sin consultar al usuario.

> La distinción es importante porque las decisiones de gobernanza varían en función del segundo eje, no tanto del primero: lo que vuelve relevante un marco de control no es cuántas cosas puede hacer un sistema, sino cuántas hace antes de solicitar validación, auditoría o supervisión.

En síntesis

Este libro no ofrece un inventario exhaustivo ni un mapa definitivo, sino un marco conceptual: una forma de pensar el ecosistema agéntico, las herramientas que allí conviven, sus componentes de diseño y sus exigencias de gobernanza, considerando el estado del arte actual a inicios de mayo de 2026.

01. INTRODUCCIÓN
QUÉ ENCONTRAMOS HOY EN EL ECOSISTEMA DE LA IA AGÉNTICA

Este material busca exponer el estado del arte de la IA agéntica a partir de las dos publicaciones que hicimos sobre agentes de IA en IALAB: la investigación del ecosistema, la exploración sistemática de herramientas y plataformas, y las soluciones que hemos diseñado y prototipado en contextos organizacionales concretos.

Lo que sigue no es un ejercicio teórico, sino el resultado de esa práctica acumulada, organizado como marco conceptual para quienes necesitan comprender, evaluar y gobernar este tipo de tecnologías. Empecemos por abordar tres grandes fenómenos.

1

Primero

En el ecosistema de la IA agéntica conviven distintos tipos de soluciones que se diferencian, fundamentalmente, por su grado de agencia y autonomía, así como por la posibilidad de intervenir en el diseño de su arquitectura. Los dos extremos de ese espectro son el *asistente basado en IA generativa* y el *agente de IA*.

Hasta este punto, la mayoría de nosotros ha hecho uso de *asistentes basados en IA generativa*. Esta es la forma más difundida de interacción con herramientas de inteligencia artificial generativa de uso individual como ChatGPT, Claude o Gemini, entre otras.

Estas plataformas forman parte del ecosistema de la IA agéntica, aunque operan en el polo de menor autonomía del espectro. Son capaces de interpretar la intención del usuario para sugerir pasos siguientes, activar por sí mismas herramientas de código, acceso a internet u otras funcionalidades. Aun así, en la mayoría de las interacciones reciben una consulta, la procesan probabilísticamente y devuelven una respuesta dentro del mismo turno, con el impulso de la intervención humana.

Avanzar hacia el polo de mayor autonomía dentro de ese ecosistema, es decir, hacia los *agentes autónomos de IA*, supone un reordenamiento de esa relación. Ya no se trata sólo de instruir a un modelo detrás de la plataforma para que produzca una respuesta, sino de delegar en un sistema la ejecución de uno o varios objetivos y tareas. En este nuevo marco de trabajo, el modelo se ocupará de percibir el entorno, planificar una secuencia de acciones, seleccionar las herramientas apropiadas y ajustar el curso de acción en función de la retroalimentación que recibe durante la ejecución.

Esta transformación no es meramente técnica. Se reconfigura el rol del usuario que pasa de ejecutor de instrucciones, a diseñador, curador, auditor, supervisor y director de agentes de IA y flujos autónomos de procesos agénticos. Esto va en línea con los marcos de trabajo presentados durante 2026 por empresas como Anthropic, Google y OpenAI.

2

Segundo

Los agentes de IA no resuelven todos los casos de uso, y no siempre son la solución más eficiente en términos de resultados, costos y riesgos asociados a su diseño y operación. Elegir cuándo un problema requiere un agente y cuándo se resuelve mejor con una automatización con IA es, en sí misma, una decisión de gobernanza.

Las formas de automatización que han precedido a la masificación de la IA son deterministas y, en muchos casos, más económicas y sencillas de auditar. Incluso, siguen siendo la mejor alternativa en función de las tareas u objetivos buscados. Estas modalidades no desaparecen con la llegada de la IA agéntica: conviven con ella y también la complementan.

En particular, los flujos de trabajo que integran IA generativa dentro de arquitecturas determinísticas, lo que en la sección siguiente llamaremos el segundo nivel de automatización. Estos resuelven una enorme cantidad de problemas concretos con costos significativamente menores que los de un agente, mayor previsibilidad en sus resultados y menor complejidad de despliegue.

3

Tercero

Gestionar la brecha entre pilotos e implementación.

A todo esto se suma una cautela técnica que conviene anticipar desde la introducción: la brecha entre demos y producción, lo que la literatura técnica denomina agent reliability gap. A abril de 2026, los benchmarks consolidados de capacidad agéntica como GAIA, SWE-bench Verified, AgentBench y WebArena, muestran tasas de resolución de tareas reales que oscilan entre 30% y 70% según el dominio, incluso en los mejores sistemas comerciales disponibles.

La industria comunica a los agentes como si fueran trabajadores confiables; la evidencia experimental sostiene una imagen más matizada. Esto es clave si queremos pensar un marco de gobernanza: hay que incorporar la fiabilidad estructural del sistema como variable de diseño y no como excepción a corregir.

COORDENADAS DE LECTURA: DOS DIMENSIONES EN PARALELO Y COMPLEMENTARIAS

A lo largo de este documento vamos a trabajar implícitamente sobre dos ejes que es importante diferenciar.

IA DE LAS TAREAS	IA DE LOS PROCESOS
Automatiza tareas puntuales	Automatiza procesos completos
Usuario (Descentralizada)	**Organización** (Centralizada)
Usuario diseña y controla	Organización diseña y controla

Hasta hace poco, la IA de las tareas era predominantemente del usuario y la IA de los procesos era predominantemente de la organización. Esa correspondencia se está diluyendo. Herramientas como Claude Cowork permiten a un usuario individual configurar flujos de proceso con distintos grados de autonomía. Y plataformas de agentes gestionados como los Workspace Agents de ChatGPT, que antes requerían infraestructura organizacional, también empiezan a ser accesibles desde el paradigma del usuario final.

Esto significa que la IA de los procesos ya no es necesariamente centralizada, y que un usuario puede operar con herramientas agénticas de nivel proceso sin que medie una decisión arquitectónica de la organización. Esto democratiza el acceso y amplifica tanto las oportunidades como los desafíos y riesgos. Acá será central analizar las cuestiones de gobernanza del ecosistema agéntico.

Volvemos a reiterar la doble dimensión: la *IA de las tareas* vinculada a qué se automatiza, y la IA de los procesos. Y por otra parte, quien diseña y controla la automatización y el uso de IA y agentes: el usuario de manera descentralizada o la organización de manera centralizada.

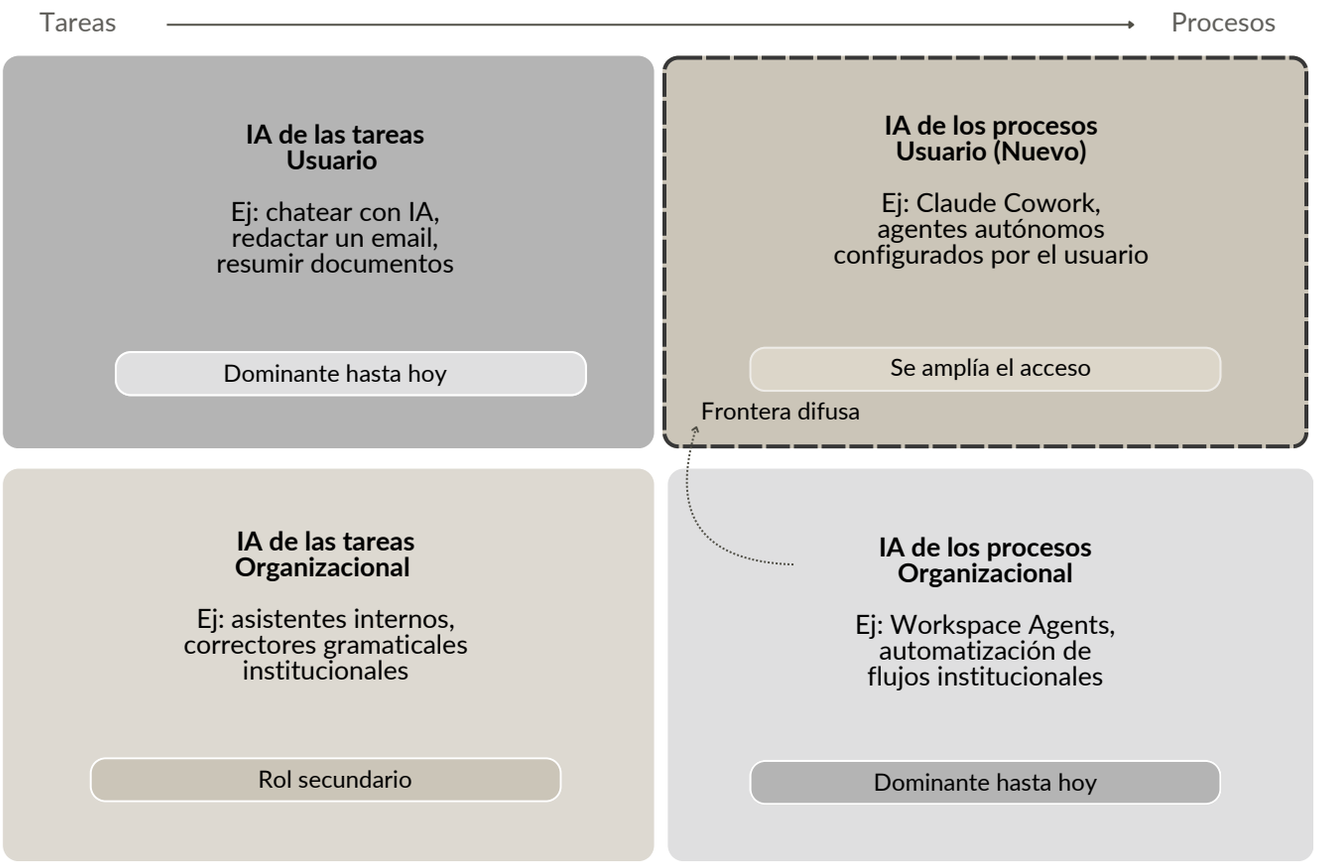
Sobre esta doble dimensión planteamos una **taxonomía de tres niveles de automatización** (sección 2); los **componentes funcionales que definen a un agente**, incluyendo la dimensión a menudo olvidada del *output* como acción imputable (sección 3); las tres capas de decisión técnica, infraestructura, abstracción y protocolos de interoperabilidad, que condicionan cualquier despliegue (sección 4); los tipos de agentes según el grado de control del usuario sobre la arquitectura (sección 5); las plataformas y frameworks disponibles para construirlos (sección 6); y **un marco de gobernanza** articulado en dos verticales, la institucional clásica (trazabilidad, auditabilidad, transparencia, persistencia) y la cognitiva (sedentarismo, paradojas de la automatización, ciclo completo de productividad), que cierra el material (sección 7).

LAS DOS DIMENSIONES CLAVE DEL ECOSISTEMA AGÉNTICO

Eje 1: ¿Qué se automatiza?

Eje 2: ¿Quién controla?

Eje 1: ¿Qué se automatiza?

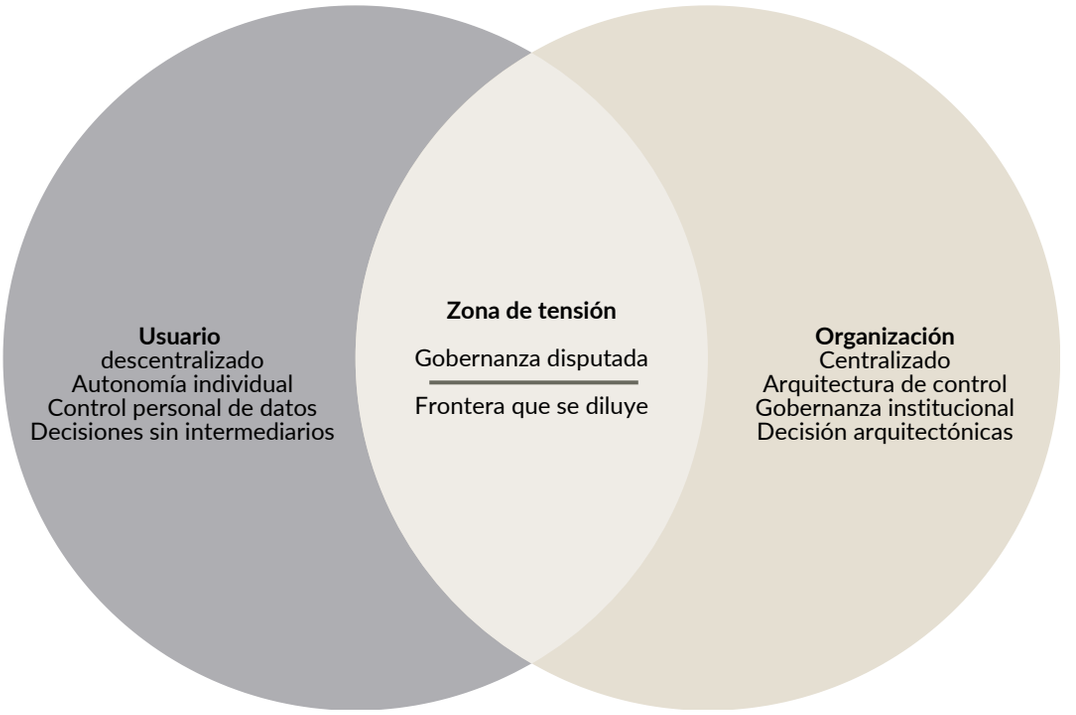


Eje 2: ¿Quién controla?

Usuario descentralizado

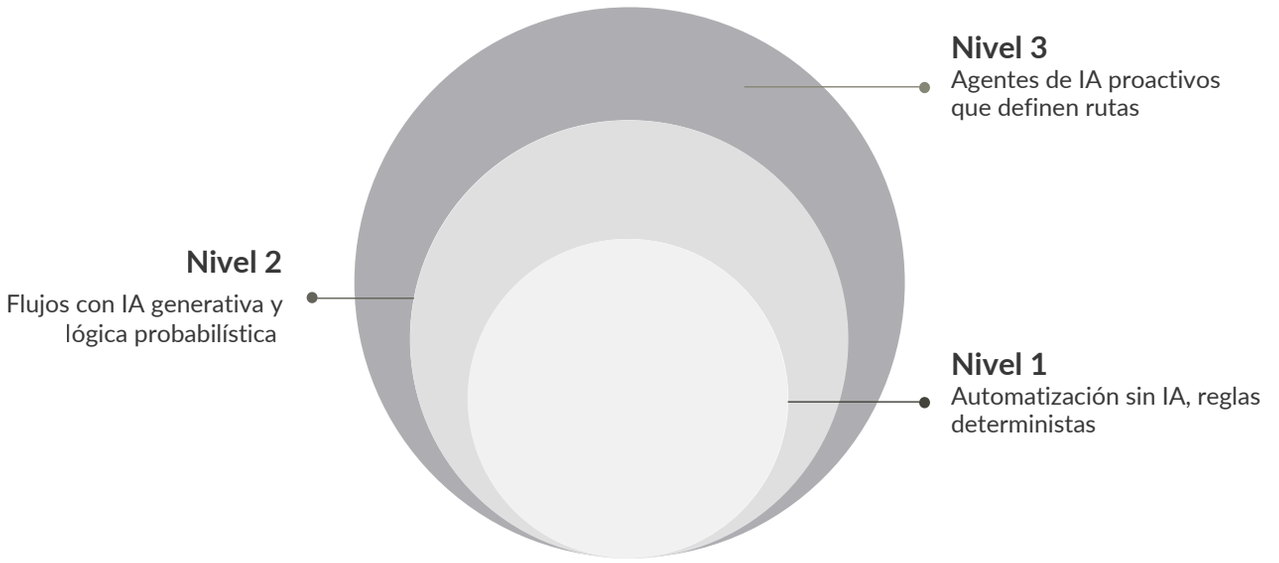
← Autonomía individual

Arquitectura de control →



02. DEL SISTEMA QUE EJECUTA AL SISTEMA QUE DECIDE: TRES NIVELES DE AUTOMATIZACIÓN DE CARA A LA IA AGÉNTICA

La pregunta sobre qué distingue a un agente de IA de otras formas de automatización de procesos se responde mejor si se observa el continuo completo. No todas las automatizaciones son iguales ni aspiran a lo mismo. Podemos distinguir **tres niveles**, cada uno con un campo de aplicación diferenciado.



Cada nivel representa un campo de aplicación diferenciado, con distintos grados de flexibilidad, autonomía y complejidad de gobernanza. El continuo no implica que un nivel sea superior al otro: cada uno responde a necesidades específicas y puede ser la mejor opción según el contexto.

AUTOMATIZACIÓN SIN IA

La automatización sin IA responde a una lógica determinista. Comprende al conjunto de sistemas informáticos diseñados para ejecutar tareas de manera automática siguiendo reglas y secuencias predefinidas, construidas sobre lógica booleana y principios determinísticos.

El comportamiento del sistema puede expresarse mediante estructuras condicionales del tipo *si se verifica la condición X, entonces ejecutar la acción Y; en caso contrario, ejecutar Z*. El equipo de desarrollo define por anticipado el conjunto de comportamientos posibles del sistema, que permanece fijo durante toda su operación.

TRES RASGOS DE LA AUTOMATIZACIÓN SIN IA

Ausencia de inferencia

El sistema no interpreta, no razona y no construye representaciones del problema; se limita a evaluar condiciones lógicas preestablecidas y a ejecutar las ramas del flujo que esas condiciones activan.

Determinismo

Dadas las mismas entradas y el mismo estado interno, el sistema produce invariablemente el mismo resultado, lo que garantiza previsibilidad y reproducibilidad integrales.

Cierre operativo

El sistema solo puede actuar dentro del universo de supuestos previstos por quienes lo programaron; cualquier evento no contemplado produce un error, una detención o un comportamiento inadecuado.

EJEMPLOS DE AUTOMATIZACIÓN SIN IA

Sistema de gestión documental

Al detectar el vencimiento de un plazo cargado en una base de datos, dispara una notificación automática al responsable del expediente.

Formulario web con validación

Impide su envío cuando alguno de los campos marcados como obligatorios permanece vacío.

Planilla de cálculo con fórmulas

Actualiza automáticamente una suma o un promedio cada vez que se modifican las celdas de entrada.

Sistema de archivo automático

Categoriza documentos entrantes en carpetas predefinidas según coincidencias exactas con palabras clave.

Disparador de integración

Envía una notificación a un canal de mensajería cada vez que se produce un registro en una aplicación vinculada.

FORTALEZAS Y LIMITACIONES DE LA AUTOMATIZACIÓN SIN IA

Fortalezas	La fortaleza distintiva de la automatización pura es la **confiabilidad**. Al operar sobre reglas explícitas y determinísticas, el comportamiento del sistema es enteramente auditable: puede explicarse paso a paso, validarse antes del despliegue y supervisarse mediante pruebas exhaustivas. Esto la vuelve particularmente apropiada para procesos estructurados, de alto volumen y baja tolerancia al error.
Limitaciones	La contracara es su **rigidez**. El sistema no puede adaptarse a entradas atípicas, excepciones o situaciones no previstas en la fase de diseño; toda variación relevante respecto del escenario modelado exige una modificación explícita del código. No hay flexibilidad, dinamismo ni capacidad de toma de decisiones ante ambigüedad.

En síntesis, **la automatización sin IA opera con la lógica de un reloj mecánico:** cada rueda y cada engranaje cumplen una función exacta en un momento exacto. Es precisa y confiable mientras las condiciones de operación coincidan con las previstas por el relojero. Frente a un evento no contemplado, no improvisa: se detiene.

TAREAS Y FLUJOS DE TRABAJO QUE INTEGRAN IA GENERATIVA

Técnicamente,el flujo mantiene una estructura general determinística diseñada por personas.

Es esa estructura, heredada de la automatización sin IA, la que orquesta el proceso y decide en qué puntos se activa la IA para resolver tareas que requieren comprensión, interpretación o generación de lenguaje natural. Se trata de soluciones donde *"los modelos de lenguaje y las herramientas se orquestan a través de rutas de código predefinidas por el equipo de desarrollo"* y que, por lo tanto, *"ofrecen previsibilidad y coherencia para tareas bien definidas"*.

El segundo nivel de automatización es el punto de encuentro entre la lógica determinística heredada de la automatización tradicional, y la capacidad probabilística de la IA generativa. En este modelo, la arquitectura combina la lógica que gobierna la estructura general del flujo, con los grandes modelos de lenguaje (LLM) que se invocan en momentos específicos.

Esta combinación amplía notablemente el espectro de tareas automatizables: ya no solo se procesan entradas estructuradas y coincidencias exactas, sino también texto libre, documentos no estructurados, contenido multimodal e incluso matices contextuales que la lógica clásica no sabría capturar.

En este segundo nivel, el flujo completo sigue siendo diseñado por personas. Quienes configuran la automatización determinan, de antemano, en qué momento se consulta al modelo, qué información se le proporciona, bajo qué instrucciones opera, qué se hace con su salida y cómo se encadenan los pasos ulteriores. El modelo no decide qué hacer ni cuándo hacerlo: ejecuta tareas puntuales como clasificar, resumir, redactar o extraer dentro de una ruta que le es dada, no de una que elige. La arquitectura y la lógica general del flujo dependen del diseño humano.

EJEMPLOS DE FLUJOS CON IA GENERATIVA

01

Resumen automático de documentos extensos

Un flujo recibe un PDF adjunto, extrae su texto, solicita a un modelo generativo un resumen ejecutivo en el idioma y formato deseados y distribuye el resultado por correo electrónico a los interesados.

02

Extracción de datos de documentos no estructurados

Un flujo que recibe contratos, facturas u otros documentos, emplea un modelo generativo para identificar entidades clave (partes, fechas, montos, cláusulas relevantes) y estructura la información en una base de datos para su posterior análisis o cotejo.

03

Clasificación automática de consultas entrantes

Un flujo que recibe mensajes en lenguaje natural, utiliza el modelo para categorizar el tipo de requerimiento y derivarlo al área, al responsable o a la subcola correspondiente según criterios preconfigurados.

FORTALEZAS Y LIMITACIONES DE LOS FLUJOS CON IA GENERATIVA

Fortalezas

La ventaja decisiva respecto del primer nivel es la capacidad de procesar lenguaje natural y contenido no estructurado, lo que multiplica los escenarios automatizables y admite una amplitud de variación en las entradas que la automatización clásica no toleraría. Sin embargo, la arquitectura hereda dos clases de limitaciones.

Limitaciones

Rigidez relativa

Si el caso recibido no encaja en la secuencia prevista, el sistema no reorganiza el proceso ni improvisa alternativas.

Riesgos propios de los modelos generativos

Alucinaciones, sesgos heredados del corpus de entrenamiento, variabilidad probabilística en las salidas y conocimiento limitado a una fecha de corte. Estos riesgos exigen puntos de supervisión humana en momentos críticos del flujo, lo que la literatura denomina *human in the loop*. También requieren un diseño cuidadoso de los *prompts* y de las instrucciones que guían al modelo en cada paso.

AGENTES DE IA: LA IA PROACTIVA POR DISEÑO

En los dos niveles anterior es, el flujo completo es diseñado por personas y la tecnología lo ejecuta.
En un agente de IA esa relación se invierte: los humanos definen el objetivo y los límites; la IA planifica y ejecuta el proceso que considera más adecuado para alcanzarlo.

El rasgo definitorio no es la presencia de IA generativa —que ya existía en el segundo nivel—, sino que el agente **construye la ruta en tiempo de ejecución a partir de un objetivo, en lugar de seguir una ruta preestablecida.**

La diferencia puede condensarse en una pregunta operativa: **¿quién diseña el camino?**

Automatización pura y flujos con IA Generativa	IA agéntica
Los humanos diseñan el proceso completo.	Los humanos definen el objetivo y, eventualmente, los límites. El o los agentes pueden diseñar y ejecutar las tareas y las secuencias completas de uno o varios procesos. Incluso pueden ajustar dinámicamente los subobjetivos o las decisiones en función de la información que recolectan y de los resultados parciales que obtienen.

Esta inversión es lo que justifica una taxonomía propia de componentes funcionales (sección 3), una capa de decisiones de infraestructura específica (sección 4) y un régimen de gobernanza diferenciado (sección 7). También es lo que vuelve relevante la cautela mencionada en la introducción: cuando el sistema construye su propia ruta, la auditoría no puede limitarse al output final; tiene que reconstruir la trayectoria.

03. COMPONENTES FUNCIONALES DE LOS AGENTES DE IA

Un agente de IA es parte de un **ecosistema modular** compuesto por piezas interconectadas que le permiten percibir, razonar, recordar y actuar. En esencia, se trata de un sistema dedicado a la gestión del contexto: un ciclo continuo de recopilación de información, activación del modelo, observación de resultados y reconstrucción del contexto para el paso siguiente.
Para ordenar esa arquitectura, distinguiremos seis componentes:

No se trata de piezas aisladas, sino de dimensiones de un mismo diseño. El interrogante que debe guiar la lectura de cada una es el mismo:

¿qué decisiones estoy habilitando o restringiendo cuando configuro este componente de tal o cual manera?

EL MOTOR DE RAZONAMIENTO: EL LLM

En el núcleo de cualquier agente hay un modelo de lenguaje de gran escala(LLM) que actúa como su cerebro: la unidad de procesamiento que recibe la información disponible, simula razonamiento, toma decisiones y coordina el uso de los demás componentes.

Se trata, típicamente, de los mismos modelos que ya conocen del uso de asistentes basados en IA generativa (la familia GPT de OpenAI, Gemini de Google, Claude de Anthropic, o modelos locales como Qwen, Llama, Mistral, entre otros), aunque en el contexto agéntico su función trasciende la generación de contenido: el modelo pasa a oficiar de director que planifica una estrategia, selecciona qué herramientas invocar en cada paso y evalúa los resultados intermedios que obtiene.

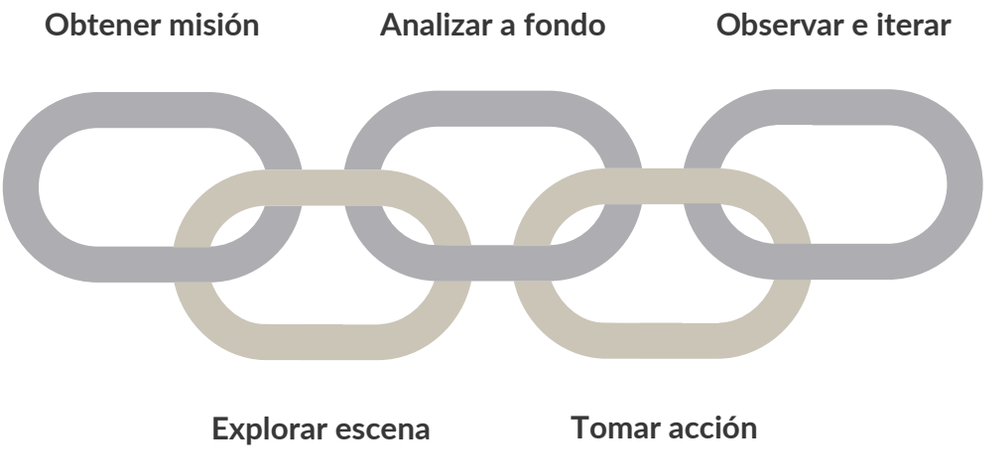
PATRONES DE RAZONAMIENTO DEL LLM

Para lograr esa coordinación, el modelo recurre a técnicas de planificación que pueden agruparse en dos grandes patrones, conocidos en la literatura técnica:

ReAct (Reason + Act)	Yao et al., 2022. El agente alterna razonamiento y acción en cada paso: piensa, actúa, observa el resultado, vuelve a pensar. Es el patrón dominante en agentes de uso general como Claude Agent SDK y OpenAI Agents SDK. Esta lógica es más robusta frente a entornos cambiantes porque el agente vuelve a estimar su avance luego de cada acción.
Planificar y ejecutar	El agente primero produce un plan completo y después lo ejecuta paso a paso, eventualmente con un sub-modelo más liviano. Es eficiente para tareas predecibles pero menos resiliente ante imprevistos.
Cadena de pensamiento (Chain-of- Thought)	Permite descomponer un objetivo complejo y ambiguo en una secuencia de subtareas más pequeñas, lógicas y manejables, ejecutarlas por etapas y corregir el rumbo cuando se detectan errores.

EL CICLO OPERATIVO DEL AGENTE

Esa capacidad de descomposición se despliega, en la práctica, a través de un ciclo iterativo que la literatura técnica describe en cinco fases:



Este ciclo de **"pensar, actuar, observar"** se repite, gestionado por la capa de orquestación y ejecutado por las herramientas, hasta que el objetivo se completa o hasta que el sistema determina que necesita intervención humana. Este modo de operar explica por qué un agente puede sostener un flujo de trabajo de varios pasos sin que el usuario tenga que indicarle cada uno: el modelo descompone el objetivo por sí mismo y reordena el plan conforme avanza.

MEMORIA DEL AGENTE

El motor se complementa con dos módulos que le permiten sostener coherencia a lo largo del proceso y entre procesos sucesivos:

Memoria de corto plazo (o de trabajo)

Rastrea la propia "línea de pensamiento" del agente y las acciones que acaba de realizar dentro de la sesión en curso, evitando que pierda el hilo de su razonamiento a mitad de un flujo. Funciona como el *bloc de notas* activo del agente, que mantiene el historial de la conversación actual.

Memoria de largo plazo (o persistente)

Retiene información entre sesiones (preferencias de la organización, criterios de redacción ya consolidados, posturas asumidas en casos análogos) y habilita el llamado "modo Netflix": el agente "conoce" al usuario y a su contexto institucional y construye una relación de colaboración continua sin necesidad de repetir el encuadre en cada interacción.

LOS OBJETIVOS: LA MISIÓN Y LOS LÍMITES QUE SE FIJAN EN EL DISEÑO

La interacción entre los modelos de lenguaje y los usuarios a nivel "IA del usuario" se vincula con la sincronia en la interacción o, en su caso, con la edición de prompts que pueden alojarse en proyectos o areas para trabajar dentro del ecosistema de plataformas.

En la lógica agéntica, hay que diseñar objetivos y luego pensar como el agente los llevara adelante. Es crítico determinar medios, criterios que debe respetar y los límites que no puede exceder. Esa configuración queda incorporada en las instrucciones de sistema del agente y opera en todas las ejecuciones posteriores.

> En forma de analogía, los objetivos del agente son a su diseño lo que la *resolución de creación de un cargo* o el *reglamento interno de una función* son a un puesto público. No se redactan de nuevo en cada expediente, sino que fijan, de manera permanente, que se espera del cargo, que competencias tiene atribuidas y que conductas le estan vedadas.

En términos operativos, definir correctamente el mandato agéntico supone anticipar:

El objetivo sustantivo
El resultado que se espera obtener, formulado con la mayor concreción posible. Por ejemplo: *"asistir en la elaboración de borradores de dictámen en causas contencioso-administrativas, a partir del expediente digital"*.

Los sub-objetivos o hitos intermedios
Los puntos de control en los que el proceso debe detenerse, reportar o requerir validación humana. Por ejemplo: *"al identificar normativa aplicable, dejar constancia de fuentes; al completar el borrador, someterlo a revisión del dictaminante antes de cualquier accion ulterior"*.

Los límites negativos
Las acciones que el agente tiene prohibido ejecutar, aún cuando las considerase funcionales al objetivo. Por ejemplo: *"no consultar fuentes externas a las cargadas en la base de datos"*.

Este es uno de los componentes de diseño más críticos. En la IA agéntica cobra especial vigencia la máxima de que el fin no justifica los medios: un agente que cumple su objetivo vulnerando deberes de confidencialidad, accediendo a sistemas para los que no tiene autorización o incurriendo en prácticas contrarias a normativa vigente no es un agente *eficaz* sino un agente *mal disenado*. La delimitación precisa del mandato con sus hitos y sus prohibiciones es, en contextos institucionales, la primera y más importante línea de defensa frente a ese riesgo.

EL INPUT: LA INFORMACIÓN Y EL CONTEXTO DE CADA EJECUCIÓN

Si los objetivos son el encuadre permanente fijado en el diseño, el input es la materia concreta sobre la que el agente trabaja en cada ejecución. Es todo aquello que entra al sistema en tiempo real y a partir de lo cual el agente construye una representación del problema puntual que debe resolver: el pedido del usuario, los documentos adjuntos, los datos que el propio agente va recolectando a medida que avanza.

En su forma mas elemental, el *input* coincide con el *prompt* que el usuario formula en lenguaje natural, eventualmente acompañado de los documentos, enlaces o instrucciones complementarias que decide adjuntar. Un ejemplo dentro de un agente de IA que dictamina sería el siguiente: "preparate un borrador para el expediente XXX priorizando abordar el cumplimiento del plazo de caducidad". Ese pedido, hasta aquí, no difiere estructuralmente del que recibiría un asistente basado en IA generativa.

En un ecosistema agéntico, el input se compone de tres fuentes principales:

El prompt del usuario y sus adjuntos
El pedido puntual que inicia la corrida y los documentos o datos que el usuario entrega junto con el.

Las consultas dinámicas a herramientas externas
Búsquedas web, consultas a APIs, lecturas de bases de datos o de repositorios documentales que el propio agente decide realizar sobre la marcha, segun lo que el objetivo y el pedido le sugieren.

La recuperación desde memoria
Información ya conocida por el sistema como preferencias de la organización, criterios institucionales, o decisiones de casos análogos, que el motor recupera automáticamente de la memoria persistente para contextualizar la ejecución en curso.

Este desplazamiento modifica el tipo de decisiones que el diseño debe anticipar. La calidad del *prompt* del usuario sigue siendo importante. Pero a esa exigencia se suman ahora preguntas que se trasladan al plano arquitectónico:

> *¿A qué fuentes se le permite acceder al agente cuando va a expandir el input? ¿Con qué credenciales? ¿Con qué grado de actualizacion? ¿Con qué trazabilidad de las consultas realizadas? ¿Qué se almacena en memoria persistente, por cuánto tiempo y con qué mecanismos de revisión?*

Cada una de estas definiciones condiciona la calidad y la legalidad de todas las salidas posteriores. Un agente que no accede a información actualizada operará sobre supuestos obsoletos; un agente con acceso excesivo a información sensible puede comprometer tanto la precisión de sus respuestas como la confidencialidad institucional; una memoria persistente mal delimitada puede volverse, en sí misma, un repositorio de datos personales cuya gestión deba responder a estándares de protección normativamente exigibles.

Hay un aspecto que suele pasarse por alto: muchas de estas preguntas no se responden en el prompt del usuario, sino en el diseño del agente. Los objetivos fijados en el diseño definen qué fuentes son legítimas y bajo qué condiciones; el input formulado en la ejecución decide que se consulta puntualmente dentro de ese universo habilitado. Ninguno de los dos componentes se explica sin el otro.

LAS HERRAMIENTAS: BRAZOS Y SENTIDOS DEL AGENTE EN EL MUNDO DIGITAL

Por potente que sea, el motor de razonamiento, por sí solo, es un sistema aislado que solo conoce los datos con los que fue entrenado, hasta una determinada fecha de corte, y únicamente puede generar texto. Para que se convierta en un operador digital con capacidad de intervenir en un flujo de trabajo real, necesita extensiones que le permitan interactuar con el entorno. Esas extensiones son las herramientas.

Técnicamente, las herramientas son conexiones que están generalmente mediadas por APIs y que otorgan al agente capacidades operativas extendidas. A través de ellas, el agente puede:

→ Navegar por internet y obtener información actualizada en tiempo real.

- Consultar bases de datos externas.
- Leer, analizar y generar archivos adjuntos en múltiples formatos (documentos de texto, PDFs, hojas de cálculo, código).
- Ejecutar código en un entorno controlado.
- Interactuar con otras aplicaciones (correo electrónico, calendarios, plataformas de firma electrónica, sistemas de notificaciones) para desencadenar acciones concretas.

A estas capacidades se suman las **capacidades multimodales nativas**, que funcionan como una forma de *percepción holística* del entorno digital: el agente no solo lee texto, sino que puede interpretar contenido visual o transcribir un archivo de audio.

El conjunto de herramientas no viene dado junto con el modelo de lenguaje. Es una decisión explícita de diseño del agente. Qué herramientas se habilitan, con qué permisos, con qué credenciales, con qué límites por consulta, con qué registros de auditoría: todas estas son definiciones que, en un contexto institucional, deben pensarse en diálogo con los objetivos y sus límites normativos o restricciones éticas o culturales.

EL OUTPUT: DEL TEXTO AL ACTO IMPUTABLE

En un asistente basado en IA generativa, el output coincide con la respuesta textual que aparece en pantalla.

El usuario lee, valida y, eventualmente, copia y pega. Hay un acto humano que media entre la generación de la respuesta y cualquier efecto sobre el mundo.

En un agente, esa mediación puede no existir. El output deja de ser exclusivamente texto y pasa a incluir acciones ejecutadas directamente sobre el entorno: un correo electrónico enviado, una transacción confirmada, un archivo modificado o eliminado, un mensaje publicado en un canal corporativo, una decisión registrada en un sistema. Cada una de esas acciones es una manifestación de voluntad atribuible al usuario que activó al agente, a la organización que lo desplegó, a quien diseñó las instrucciones de sistema o a todos ellos juntos.

Acá se producen tres consecuencias prácticas:

Las acciones irreversibles requieren validación humana explícita por diseño

No como buena práctica opcional. Eliminar archivos, enviar comunicaciones externas, ejecutar transacciones financieras o publicar contenido en redes son todas operaciones que un agente bien diseñado debería detener para confirmación, salvo decisión expresa y documentada en sentido contrario.

El registro del output debe alcanzar al acto, no sólo al texto que lo precedió

En la sección 7 retomaremos este punto al tratar la trazabilidad: una traza completa registra qué herramienta se invocó, con qué argumentos, con qué resultado y con qué efecto sobre el sistema sobre el que se actuó.

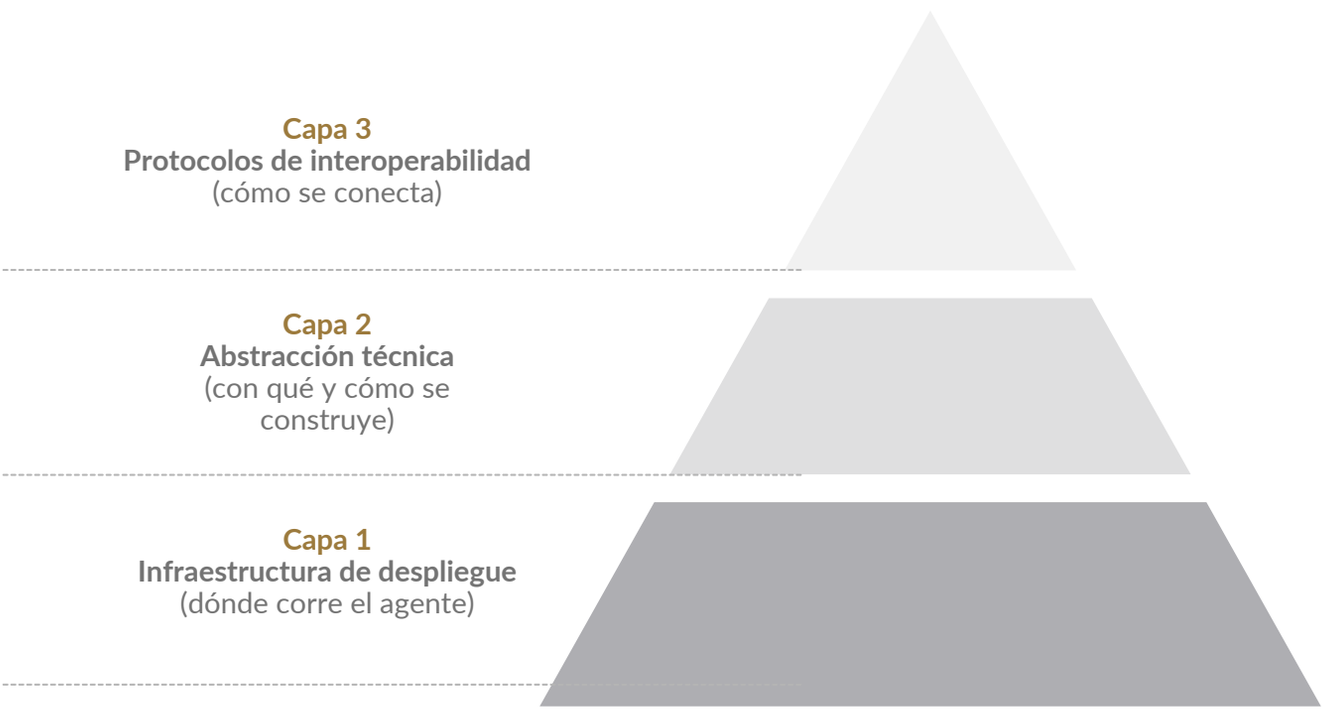
La distribución de responsabilidades exige una matriz explícita

Quién asume las consecuencias de un resultado mal ejecutado depende de un conjunto de decisiones que la organización tiene que documentar antes del despliegue, no después del incidente.

4. LAS TRES CAPAS DE DECISIÓN TÉCNICA: INFRAESTRUCTURA, ABSTRACCIÓN Y PROTOCOLOS DE INTEROPERABILIDAD

Los componentes de la sección anterior describen qué tiene un agente. Ahora corresponde abordar tres capas de decisión técnica que condicionan cualquier despliegue concreto: dónde corre el agente (infraestructura), con qué y cómo se construye (abstracción técnica) y a través de qué protocolos se conecta con el resto del ecosistema (interoperabilidad).

Estas tres capas no forman parte de la arquitectura interna del agente, pero la condicionan. Definen, entre otras cosas, qué perfil humano dentro de una organización puede diseñar, modificar y auditar agentes; con qué grado de control sobre la arquitectura resultante; bajo qué garantías de soberanía de datos; y con qué nivel de portabilidad hacia otros proveedores.



CAPA 1: INFRAESTRUCTURA DE DESPLIEGUE

¿Dónde opera el agente?

El lugar en el que opera el agente constituye una decisión de infraestructura que responde a la pregunta: ¿dónde corre físicamente? Es decir, **dónde se alojan los datos** que procesa, **dónde se ejecuta el modelo**, **bajo qué jurisdicción operan los servidores**. Esta decisión condiciona directamente las garantías de confidencialidad, soberanía y auditoría que la organización puede ofrecer.

Existen tres modelos principales de despliegue:

On-premise (local o in-house)

El agente se ejecuta íntegramente en infraestructura controlada por la organización (servidores propios, data centers internos o infraestructura dedicada). Ofrece máximo control sobre los datos, mayor soberanía y facilidad para implementar restricciones estrictas de acceso y auditoría, lo que lo vuelve especialmente adecuado para datos personales, información clasificada o expedientes judiciales. La contracara es el mayor costo inicial de infraestructura, menor escalabilidad automática y responsabilidad total sobre mantenimiento, actualizaciones y seguridad.

Full cloud (nube pública)

El agente se despliega en servicios de proveedores externos (OpenAI, Anthropic, Google Cloud, Azure, AWS, entre otros). Ofrece escalabilidad casi ilimitada, menor costo operativo inicial, acceso rápido a las últimas versiones de los modelos y facilidad de integración con herramientas externas. La contracara es el menor control sobre dónde se procesan y almacenan los datos, posibles riesgos de fuga de información, dependencia del proveedor y mayores desafíos para garantizar trazabilidad y auditoría completa en entornos institucionales.

Híbrido

Combina infraestructura local con servicios en la nube. Por ejemplo, los datos sensibles y el núcleo del agente pueden residir en servidores propios, mientras que tareas de inferencia intensiva o acceso a ciertos modelos avanzados se realizan en la nube bajo estrictos controles. Ofrece el mejor equilibrio entre control, soberanía, escalabilidad y costo, a cambio de una mayor complejidad técnica y de gobernanza: se deben definir claramente qué datos fluyen a la nube y bajo qué condiciones.

TRES GRADOS DE SOBERANÍA: UNA TAXONOMÍA OPERATIVA

La distinción **"on-premise/cloud/híbrido"** suele asimilarse a la cuestión de la soberanía de datos, pero conviene desagregarla.

Una organización puede correr su agente on-premise usando la API de OpenAI: en ese caso, los datos del prompt salen igual hacia los servidores del proveedor del modelo, aún cuando el resto de la arquitectura sea local.

La soberanía completa requiere algo más que infraestructura propia: requiere también un modelo que corra localmente. Proponemos distinguir, por tanto, **tres grados de soberanía** que conviene explicitar al momento de diseñar:

1	**Grado 1 - Soberanía total** Modelo de código abierto ejecutado en infraestructura propia (Qwen, Llama, Mistral, desplegados mediante Ollama, vLLM, TGI o equivalentes), sobre infraestructura on-premise, con herramientas también locales o controladas. Ningún dato abandona el perímetro de la organización. Es la configuración requerida para procesamiento sometidos a estrictas restricciones jurisdiccionales o cuando la sensibilidad de la información lo justifica plenamente.
2	**Grado 2 - Soberanía de procesamiento** Modelo local y herramientas externas. El motor de razonamiento corre on-premise con un modelo open-weights, pero el agente invoca herramientas externas (APIs públicas, servicios SaaS) para tareas específicas. Los datos del razonamiento principal no salen, pero ciertas consultas instrumentales sí. Es una configuración intermedia frecuente en organizaciones que necesitan combinar control con acceso a fuentes externas.
3	**Grado 3 - Soberanía contractual** Modelo en cloud, datos no salen. Los datos no se almacenan persistentemente fuera del perímetro, pero el modelo corre en infraestructura del proveedor (vía deployments privados tipo Azure OpenAI con DLP, AWS Bedrock con VPC, o equivalentes). Hay control contractual y técnico sobre la retención y el uso, pero el procesamiento ocurre en cloud.

La elección entre estos tres grados no es solo técnica: tiene implicancias directas sobre la legitimidad de la actuación administrativa, la viabilidad probatoria de los registros y la sustentabilidad operativa del despliegue.

CAPA 2: CON QUÉ SE CONSTRUYE EL AGENTE

Más allá de dónde corre, los agentes pueden construirse sobre distintos niveles de abstracción técnica. Esta capa define el perfil humano que puede intervenir en su diseño y el grado de control sobre la arquitectura resultante.

Plataformas gestionadas

Permiten diseñar agentes con distintos niveles de abstracción, desde interfaces conversacionales hasta APIs. Ejemplos actuales son Claude Managed Agents (Anthropic) y los Workspace Agents de ChatGPT (OpenAI). El diseñador define objetivos, herramientas disponibles y restricciones; la plataforma se encarga de la orquestación, la infraestructura y la seguridad. Reducen drásticamente la barrera de entrada, pero limitan el control sobre la arquitectura interna del agente, además de introducir otros riesgos como el *vendor lock-in*.

Frameworks de código

Frameworks de desarrollo como CrewAI, LangGraph, OpenAI Agents SDK, Claude Agent SDK, OpenClaw o NanoClaw permiten diseñar agentes desde cero con control sobre cada componente de la arquitectura: motor, memoria, herramientas, orquestación, políticas de reintentos, manejo de errores. Exigen competencias de programación pero ofrecen la máxima flexibilidad y la posibilidad de construir sistemas multi-agente con lógicas de coordinación a medida.

Entre ambos extremos se ubican las **plataformas de automatización con IA** (n8n, Make, Zapier), que si bien permiten integrar modelos de lenguaje en flujos de trabajo, no están diseñadas específicamente para construir agentes. Las trataremos en detalle en la sección 6.

CAPA 3: CÓMO SE CONECTA EL AGENTE CON EL RESTO DEL ECOSISTEMA: PROTOCOLO DE INTEROPERABILIDAD

Un agente de IA no opera en el vacío: necesita conectarse con herramientas, fuentes de datos, aplicaciones y, en ciertos escenarios, con otros agentes. La forma en que se resuelven esas conexiones tiene consecuencias directas para la gobernanza, la auditabilidad y la independencia tecnológica de la organización. En el período 2024-2026 se ha consolidado un panorama de **tres mecanismos de interoperabilidad con alcances distintos y complementarios**.

APIs REST

Las interfaces de programación de aplicaciones son el mecanismo de integración más establecido y maduro del ecosistema de software. Una API REST expone operaciones específicas, como consultar un saldo, registrar un expediente y enviar una notificación, mediante endpoints bien definidos, con contratos claros sobre entradas, salidas y códigos de error. Para tareas de ejecución de alto volumen, bien definidas y repetitivas, donde lo que importa es el control operativo, la eficiencia y la precisión, las APIs REST siguen siendo la opción más adecuada. Sin embargo, las APIs presentan una limitación estructural para los escenarios agénticos más complejos: cada integración es a medida, el agente necesita conocer de antemano qué APIs existen y cómo invocarlas, y no hay un mecanismo estándar para que un agente descubra dinámicamente qué capacidades tiene disponibles en su entorno.

Model Context Protocol (MCP)
Propuesto por Anthropic en noviembre de 2024 y adoptado posteriormente por OpenAI, Google, Microsoft y la mayor parte del ecosistema, MCP estandariza la conexión entre agentes y herramientas externas. Un servidor MCP expone capacidades —como lectura de archivos, consulta a bases de datos, ejecución de tareas en una aplicación— que cualquier agente compatible puede invocar siguiendo el mismo protocolo, sin necesidad de integraciones a medida. Esto produce dos grandes consecuencias: Reducción del lock-in (un servidor MCP construido para integrarse con Claude puede usarse con un agente de OpenAI, Gemini, o un modelo local con mínimas modificaciones) y Estandarización de la auditoría (las invocaciones MCP siguen un formato común, lo que facilita la generación de trazas estructuradas y la implementación de mecanismos de control unificados sobre múltiples agentes).

Agent-to-Agent Protocol (A2A)
Propuesto por Google en abril de 2025 y respaldado por más de cincuenta proveedores tecnológicos, A2A es complementario a MCP: mientras que MCP estandariza la comunicación agente ↔ herramientas, A2A estandariza la comunicación agente ↔ agente. Esto importa en escenarios multi-agente donde sistemas construidos por equipos o proveedores distintos necesitan coordinarse sin acoples a medida.

EL CRITERIO DE SELECCIÓN NO ES EVOLUTIVO SINO FUNCIONAL

Estos tres mecanismos no representan una progresión lineal donde cada uno reemplaza al anterior, sino opciones que coexisten y se despliegan a lo largo de límites funcionales claros.

APIs REST
La opción natural para operaciones de alto volumen, bien definidas y predecibles.

MCP
La opción adecuada cuando el agente necesita descubrir y acceder dinámicamente a un entorno amplio de herramientas y fuentes.

A2A
Se reserva para flujos de trabajo en los que múltiples agentes autónomos deben coordinarse entre sí.

Una arquitectura agéntica bien diseñada probablemente utilice los tres en distintas capas de su operación.

La relevancia de este panorama para la gobernanza es doble. Por un lado, define el estándar de calidad emergente: una arquitectura agéntica a 2026 difícilmente puede construirse al margen de estos protocolos. Por otro lado, estandariza los puntos de control: si las invocaciones a herramientas y las comunicaciones entre agentes pasan por protocolos abiertos y registrables, la auditoría procedimental se vuelve estructuralmente más viable, un punto que retomaremos en la sección 7.

05. TIPOS DE AGENTES DE IA: EMPAQUETADOS, HÍBRIDOS Y DISEÑADOS

Hasta aquí describimos los componentes funcionales que definen a un agente de IA y las tres capas de decisión técnica que condicionan su despliegue. Esos componentes no se configuran siempre de la misma manera ni por los mismos actores.

Actualmente, dentro del espectro del agente de IA conviven **tres tipos** que se diferencian entre sí en razón de quién toma las decisiones de diseño y qué grado de control tiene el usuario sobre la arquitectura del sistema: los **agentes empaquetados**, los **agentes híbridos** y los **agentes diseñados**.

No son categorías cerradas sino puntos de un espectro en el que se combinan, de manera variable, el grado de autonomía del sistema, el nivel de control del usuario y la complejidad del diseño requerido.

Hay dos grandes criterios para ubicarlos en esta segmentación que permite segmentar niveles. Por un lado, lo que varía entre ellos es, en definitiva, **quién decide** sobre cada uno de los componentes que analizamos en la sección 3: el motor, los objetivos, el input, las herramientas, el output y la infraestructura. Por otra parte, el **nivel de conocimientos** de un usuario para que puedan manipularlos e implementarlos.

A esta clasificación general se le suman dos categorías especializadas que conviene mencionar por su importancia creciente: los **agentes de uso de computadora (computer use)** y los ***coding agents***. No son un cuarto tipo en términos arquitectónicos, sino especializaciones que pueden adoptar cualquiera de las tres formas anteriores.

AGENTES EMPAQUETADOS: AGENTES A UN CLIC

Los **agentes empaquetados** son aquellos que llegan al usuario final listos para usar, integrados dentro de una plataforma conversacional de acceso individual. El usuario no diseña el agente ni define su arquitectura; la plataforma lo pone a disposición como una funcionalidad incorporada, activable desde la misma interfaz en la que ya interactúa con el asistente basado en IA generativa.

Estos agentes representan la modalidad de mayor autonomía dentro de la *IA de las tareas*: *agentes de IA a un clic*. El usuario describe objetivos o tareas a realizar y el agente despliega cadenas de acciones con la intervención humana que pida el usuario. El agente puede realizar tareas similares a las que hacen los grandes modelos de lenguaje, pero el sistema no solo genera contenido sino que ejecuta acciones sobre el entorno digital.

> **Modo agente de ChatGPT (OpenAI, julio 2025)**
>
> Al activarlo, el usuario accede a un sistema que dispone de su propio navegador virtual, puede interactuar con sitios web, analizar datos, generar archivos, conectarse a aplicaciones externas como Gmail mediante conectores, y ejecutar tareas de varios pasos de manera autónoma. El usuario describe lo que necesita (planificar un viaje, actualizar un documento, investigar un tema), y el agente planifica la secuencia de acciones, selecciona las herramientas apropiadas y ejecuta el proceso, solicitando intervención humana solo cuando la necesita.

Grok (xAI, versión 4.20 Beta, febrero 2026)

Incorpora un sistema nativo de cuatro agentes especializados que trabajan en paralelo. Los cuatro agentes (denominados Grok, Harper, Benjamin y Lucas) no son modelos independientes, sino sub-agentes especializados sobre un mismo modelo base de aproximadamente tres billones de parámetros, cada uno con un rol cognitivo diferenciado. Grok actúa como coordinador; Harper se especializa en investigación y verificación fáctica; Benjamin aporta razonamiento lógico-matemático; Lucas explora perspectivas alternativas y desafía las conclusiones de los otros tres.

Kimi K2.6 (Moonshot AI, abril 2026)

Lleva la capacidad agéntica un paso más allá con lo que Moonshot denomina "Agent Swarm": ante una tarea compleja, el sistema no trabaja con un conjunto fijo de agentes predefinidos, sino que crea dinámicamente los sub-agentes que necesita según la naturaleza del problema. Un orquestador central analiza el pedido, lo descompone en subtareas paralelas, genera sub-agentes especializados y los coordina en tiempo real. El sistema puede escalar hasta 300 sub-agentes trabajando en simultáneo a lo largo de 4.000 pasos coordinados.

Estos tres ejemplos comparten un rasgo estructural: el usuario no necesita saber cómo funciona un agente para usarlo. No configura herramientas, no define objetivos en instrucciones de sistema, no conecta APIs ni diseña flujos de orquestación. Todas esas decisiones ya fueron tomadas por el equipo de desarrollo de la plataforma. El usuario opera, en esencia, como un *mandante* que formula un encargo y supervisa el resultado, pero no interviene en el diseño del proceso.

Esta característica es, a la vez, su mayor fortaleza y su principal limitación. La fortaleza es la accesibilidad: cualquier profesional puede empezar a trabajar con un agente empaquetado sin conocimientos técnicos previos. La limitación es la falta de control sobre el diseño: el usuario no elige qué modelo subyacente opera, no define los límites negativos del agente, no controla a qué fuentes accede ni con qué credenciales, y no puede adaptar el comportamiento del sistema a las necesidades específicas de su organización o su práctica profesional.

Para un profesional que quiere experimentar con IA agéntica dentro de la *IA de las tareas*, los agentes empaquetados son un punto de partida valioso. Cuando la necesidad es integrar un agente en los procesos institucionales de una organización, con control sobre los datos, trazabilidad de las operaciones y cumplimiento normativo, resultan insuficientes.

LA ZONA HÍBRIDA:
CUANDO EL USUARIO CONFIGURA AGENTES SIN PROGRAMAR

Entre el agente empaquetado y el agente diseñado desde cero existe una **zona intermedia** donde el usuario individual puede empezar a operar sobre procesos completos, no solo sobre tareas puntuales, sin necesidad de programar ni de que medie una decisión arquitectónica de la organización.

Esta zona es una de las novedades más relevantes del estado actual del ecosistema agéntico: amplía significativamente lo que un usuario puede automatizar por su cuenta, y con ello amplifica tanto las oportunidades como las preguntas que debe formularse antes de delegar en el agente.

Claude Cowork
Ilustra bien esta categoría. Lanzado por Anthropic a fines de enero de 2026, Cowork se ejecuta en la aplicación de escritorio de Claude e interactúa directamente con el sistema de archivos local, el navegador y las aplicaciones instaladas en la computadora del usuario, dentro de las carpetas y los permisos que el usuario habilita expresamente al inicio de cada sesión. A diferencia de los agentes empaquetados, que operan íntegramente dentro de una ventana de navegador, Cowork accede al entorno real de trabajo del profesional: el agente no trabaja sobre lo que el usuario le cuenta, sino sobre el entorno real en el que el usuario trabaja, dentro de los límites que este le fija.

¿Por qué ubicar a Cowork en una zona híbrida?
Porque comparte rasgos con ambos extremos del continuo sin asimilarse a ninguno. Del lado de la IA de las tareas, conserva una accesibilidad considerable: el usuario describe un objetivo en lenguaje natural y el sistema planifica y ejecuta la secuencia de pasos necesaria. Del lado de la IA de los procesos, permite ir más allá de la tarea puntual: el usuario puede crear skills personalizadas (instrucciones reutilizables) y plugins que empaquetan habilidades junto con conectores y sub- agentes especializados, sin escribir código.

Un caso para ejemplificar es el plugin legal publicado por Anthropic, que incorpora flujos de trabajo preconfigurados para tareas habituales de asesoramiento jurídico interno: revisión contractual cláusula por cláusula con semáforo de riesgo, triaje de NDAs, verificación de contratos con proveedores, generación de briefings contextuales y elaboración de respuestas modelo para requerimientos recurrentes. El plugin es configurable: los profesionales pueden incorporar su propio playbook de negociación, sus criterios de tolerancia al riesgo y sus cláusulas estándar.

El resultado es un sistema que demanda más decisiones de configuración que los agentes empaquetados pero que sigue siendo operable por perfiles no técnicos. Es, en la práctica, una herramienta que permite al usuario individual participar de la IA de los procesos sin esperar a que la organización lo habilite. Sus implicancias específicas en materia de privacidad, soberanía de datos y responsabilidad se abordan en la sección 7.

AGENTES DISEÑADOS: EL CONTROL COMPLETO SOBRE LA ARQUITECTURA

La **tercera modalidad** se distingue de las dos anteriores por un rasgo decisivo: quien usa el agente es también (en todo o en parte) la organización que lo diseña.

En los agentes empaquetados, todas las decisiones arquitectónicas vienen tomadas por el proveedor de la plataforma. En la zona híbrida, el usuario configura aspectos significativos del comportamiento del agente, pero dentro de un marco que el proveedor define: el modelo, la infraestructura de procesamiento y el repertorio de herramientas posibles no están en sus manos.

En los **agentes diseñados**, en cambio, cada pieza de la arquitectura que analizamos en la sección 3 pasa a ser una decisión explícita del diseñador: qué modelo de lenguaje opera como motor de razonamiento, qué objetivos y límites negativos se fijan en las instrucciones de sistema, qué herramientas se habilitan y con qué permisos, qué fuentes de datos alimentan el input, bajo qué modelo de infraestructura se hostea y con qué framework se construye. **Es la arquitectura como un traje a medida: el diseñador controla cada costura.**

Ese nivel de control es lo que hace a los agentes diseñados especialmente adecuados para contextos institucionales donde se necesita operar sobre datos sensibles, cumplir normativa específica, integrar límites propios de la organización, generar registros auditables e integrarse con sistemas internos. Su requisito típico es trabajar con frameworks de código y contar con equipos con competencias de programación.

Dentro de los agentes diseñados existe una configuración específica que merece atención: el **agente soberano**, que aparece cuando la organización necesita garantizar que ningún dato salga de su perímetro. Combina los tres elementos descriptos en la sección 4 (modelo open-weights ejecutado localmente, infraestructura on-premise, herramientas controladas) y opera como una variante en la que la soberanía total no es preferencia sino restricción fundante de todo el diseño.

Una organización que necesita soberanía completa descarta de entrada las plataformas gestionadas de proveedores propietarios y se orienta necesariamente hacia frameworks de código con modelos abiertos.

DOS CATEGORÍAS ESPECIALIZADAS: COMPUTER USE Y CODING AGENTS

Más allá de los tres tipos generales, conviene mencionar **dos categorías especializadas** que están reconfigurando el ecosistema y que pueden adoptar cualquiera de las tres formas anteriores (empaquetada, híbrida o diseñada).

Agentes de uso de computadora (computer use)

→ Controlan la computadora vía interfaz gráfica: pantalla, mouse, teclado y clicks

- Actúan sobre cualquier aplicación, incluso sin API expuesta
- Anthropic Computer Use (oct. 2024): primera implementación pública en Claude
- OpenAI Operator (ene. 2025): entorno virtual de navegación en ChatGPT
- Google Project Mariner (2025- 2026): integración con ecosistema Google
- Las salvaguardas migran al perímetro: sandboxing, VMs aisladas, permisos restringidos

Coding agents

- Leen, escriben y ejecutan código; navegan repositorios completos
- Outputs verificables: los tests pasan o no pasan (SWE-bench Verified)
- Cursor Agent: editor con agente integrado, dominante en flujos individuales
- Claude Code: opera desde línea de comandos sobre repositorios completos
- Codex (OpenAI, 2025-2026): plataforma agentic con harness propio
- Devin (Cognition Labs): agente autónomo para tareas de software end-to-end
- Replit Agent: integrado en el entorno de desarrollo en navegador

06. PLATAFORMAS PARA CONSTRUIR AUTOMATIZACIONES CON IA, PLATAFORMAS GESTIONADAS Y FRAMEWORKS DE CÓDIGO

En la sección anterior vimos que los *agentes diseñados* permiten lograr el máximo control sobre la arquitectura de la solución, y que el ecosistema actual ofrece tanto frameworks de código como plataformas gestionadas para construirlos.

En las organizaciones, los **agentes conviven con automatizaciones sin IA y con otras que integran IA generativa**. Estas soluciones resuelven una enorme cantidad de problemas concretos sin la complejidad ni los costos de un agente. Por eso, esta sección abarca el estudio de distintas herramientas disponibles para construir ambos tipos de soluciones.

Elegir entre una automatización sin o con IA o un agente para automatizar procesos supone evaluar el grado de control que se necesita sobre el flujo, los costos asociados al uso de modelos de lenguaje, el hosteo de la solución y de los datos, y el grado de determinismo necesario en las respuestas, entre muchos otros elementos.

Organizamos el recorrido en cinco bloques:

01

Plataformas para automatización con IA

n8n, Make, Zapier

02

Plataformas gestionadas para diseño de agentes

Claude Managed Agents, Workspace Agents de ChatGPT, Agent Studio de Google.

03

Frameworks de código para orquestación de agentes

CrewAI, OpenAI Agents SDK, LangGraph, ADK de Google

04

Frameworks de agente autónomo con acceso al entorno

Claude Agent SDK, OpenClaw, NanoClaw

05

Agentic browsers y frameworks de evaluación y observabilidad

Una categoría emergente con implicancias regulatorias particulares, y el equipamiento que vuelve auditables los anteriores.

PLATAFORMAS DE AUTOMATIZACIÓN CON IA: N8N, MAKE, ZAPIER

Las plataformas de automatización *low-code/no-code* permiten diseñar flujos de trabajo complejos mediante interfaces visuales (típicamente un lienzo o *canvas* en el que se conectan nodos que representan acciones, condiciones, consultas y herramientas) sin necesidad de escribir código, pero con la posibilidad de incorporarlo en puntos específicos del flujo cuando la lógica lo requiere.

n8n

Plataforma disponible públicamente bajo una licencia de uso justo (*fair-code*).

Su interfaz se basa en la conexión visual de nodos funcionales: disparadores que inician el flujo, nodos de aplicaciones que interactúan con servicios externos, nodos de lógica condicional y nodos de código que permiten insertar JavaScript o Python en puntos específicos del proceso. Incluye un nodo de Agente de IA (*AI Agent node*) que permite configurar un sistema autónomo capaz de recibir datos, evaluar herramientas disponibles, seleccionar cuál invocar y actuar dentro de su entorno. Soporta integración con múltiples proveedores de modelos, OpenAI, Anthropic, Google, modelos locales, y la conexión con más de cuatrocientas integraciones nativas.

Lo que distingue a n8n es su modelo de despliegue: puede *auto-alojarse* (*self-host*), es decir, descargarse, instalarse en infraestructura propia (servidores internos, contenedores Docker, máquinas virtuales en una nube privada) y ejecutarse bajo completo control, sin que los datos transiten por servidores de un tercero. Una institución pública o una empresa con requisitos estrictos de soberanía de datos puede construir y operar agentes de IA sobre su propia red. La plataforma también ofrece una versión en la nube (*n8n Cloud*) para quienes prefieran un despliegue gestionado.

En cuanto a tarifas, la versión auto-alojada es gratuita en términos de licencia; los costos directos son los de infraestructura. La versión *cloud* cobra por *ejecución completa del flujo*, con independencia de la cantidad de pasos o nodos, lo que la vuelve significativamente más económica para flujos largos o de alto volumen. La contracara es que n8n es la plataforma más orientada a perfiles técnicos: el auto-alojamiento requiere conocimientos de administración de servidores, y la construcción de agentes complejos admite, y en muchos casos demanda, inserción de código.

Make

Plataforma de automatización visual con interfaz tipo lienzo (canvas) que permite construir flujos de trabajo, denominados "Escenarios" (Scenarios), arrastrando y conectando módulos. En abril de 2025 lanzó sus agentes de IA (Make AI Agents), incorporando capacidades agénticas con soporte multimodal y una biblioteca de agentes prediseñados adaptables. Incluye funcionalidades nativas para gestionar flujos complejos: *Routers* para ramas condicionales o paralelas, *Iterators* para procesar elementos uno por uno y *Aggregators* para combinar múltiples paquetes de datos.

Make

El modelo de tarifas se basa en operaciones: cada módulo que procesa datos consume al menos una operación. Make opera exclusivamente como servicio en la nube. Para clientes empresariales ofrece un componente llamado *On-Premise Agent* que permite conectar la plataforma con aplicaciones y bases de datos de la red local, pero sin trasladar el procesamiento a infraestructura propia: el motor de orquestación sigue en los servidores de Make. En accesibilidad se ubica en un punto intermedio: más potente que Zapier pero con una curva de aprendizaje moderada.

Zapier

Plataforma pionera en automatización, cuya principal propuesta de valor es la simplicidad extrema y una interfaz intuitiva diseñada para usuarios sin conocimientos técnicos. Opera completamente en la nube y cuenta con un ecosistema de integraciones excepcionalmente amplio. En 2025 incorporó tres productos nativos de IA: *Copilot* (genera flujos desde lenguaje natural), Chatbots (agentes conversacionales para atención al cliente) y *Agents* (agentes autónomos con capacidad de decisión y ejecución multi-paso). El modelo de tarifas se basa en tareas (tasks): cada acción ejecutada con éxito consume una tarea. Esto vuelve a Zapier significativamente más costoso que n8n o Make para flujos complejos o de alto volumen. Es la plataforma más orientada a perfiles no técnicos y ofrece menos potencia para flujos con lógica intrincada o gestión granular de errores.

Síntesis. La diferencia central entre estas tres plataformas y los agentes empaquetados o híbridos analizados antes no reside en la sofisticación del modelo de lenguaje, sino en **quién toma las decisiones de diseño**. En las tres, esas decisiones recaen en quien construye el flujo. Lo que varía entre ellas es el grado de control técnico y la soberanía sobre la infraestructura. De las tres, **solo n8n ofrece auto-alojamiento completo**.

PLATAFORMAS GESTIONADAS PARA DISEÑO DE AGENTES

Tanto Anthropic como OpenAI ofrecen desde abril de 2026 plataformas en las que el proveedor se hace cargo de la infraestructura de ejecución (contenedores, gestión de estado, orquestación, seguridad) y el diseñador se concentra en definir qué hace el agente. Lo que varía entre ambas es el nivel de abstracción y, con él, el perfil de quien puede usarlas.

Claude Managed Agents (Anthropic)

Lanzado en beta pública el 8 de abril de 2026, es una capa de infraestructura gestionada que se consume a través de APIs. El diseñador define el agente (modelo, instrucciones de sistema, herramientas, servidores MCP, skills) y Anthropic se encarga del sandboxing del entorno (cada agente corre en un contenedor aislado con acceso de red restringido por defecto), la gestión de sesiones de larga duración que persisten aunque se interrumpa la conexión, el manejo de credenciales para herramientas externas, la recuperación ante errores y la observabilidad completa de cada ejecución (cada corrida genera una traza estructurada que permite inspeccionar paso a paso qué hizo el agente).

Claude Managed Agents (Anthropic)

Lo que Managed Agents resuelve es el problema de infraestructura que históricamente separaba un prototipo funcional de un agente en producción: construir esa capa de soporte tomaba meses a equipos técnicos. Anthropic reporta que el tiempo de desarrollo se reduce significativamente, y empresas como Notion, Rakuten, Asana y Sentry ya desplegaron agentes en producción sobre esta plataforma.

El modelo de costos combina las tarifas estándar de tokens de la API de Claude con un cargo de 0,08 dólares por hora de ejecución activa del agente. Es importante notar que Managed Agents no es una herramienta *low-code* ni *no-code*: requiere interactuar con la API mediante código. Lo que simplifica no es la construcción del agente sino su infraestructura de ejecución. El diseñador sigue necesitando competencias de programación, pero ya no necesita ser además un ingeniero de infraestructura.

Los *trade-offs* son los esperables en un entorno gestionado: solo permite usar modelos de Claude, los datos de sesión se almacenan en infraestructura de Anthropic, y la portabilidad hacia otros proveedores es limitada.

Workspace Agents de ChatGPT (OpenAI)

Presentados el 22 de abril de 2026 en research preview, representan un enfoque distinto: son una herramienta no-code integrada directamente en la interfaz de ChatGPT. Para crear un agente, el usuario describe en lenguaje natural el flujo de trabajo que quiere automatizar y ChatGPT lo guía paso a paso hasta convertirlo en un agente funcional, sin escribir una línea de código.

Los Workspace Agents están diseñados para ser compartidos dentro de una organización. Un equipo puede construir un agente una vez, publicarlo en el directorio del workspace y usarlo de forma colectiva, tanto desde ChatGPT como desde Slack. A diferencia de los Custom GPTs que los preceden (lanzados en 2023 y que OpenAI planea discontinuar para organizaciones), los Workspace Agents pueden ejecutar tareas de larga duración en segundo plano (incluso después de que el usuario cierre el navegador) y actuar sobre sistemas externos: Google Workspace, Salesforce, Microsoft 365, Slack, entre otros.

La gobernanza ocupa un lugar central en el diseño. Los administradores controlan, mediante roles, quién puede construir agentes, quién puede ejecutarlos, quién puede publicarlos en el directorio y qué herramientas y conectores tiene habilitados cada rol. Es un esquema de permisos considerablemente más granular que el que ofrecían los Custom GPTs.

Están disponibles para los planes Business, Enterprise, Edu y Teachers. El acceso fue gratuito durante el período inicial, con un modelo de facturación basado en créditos a partir del 6 de mayo de 2026. Los *trade-offs* siguen el patrón del entorno gestionado: solo operan con modelos de OpenAI (corren sobre Codex), la ejecución ocurre íntegramente en la nube de OpenAI, y el grado de control sobre la arquitectura interna del agente es limitado.

AGENT STUDIO EN GEMINI ENTERPRISE AGENT PLATFORM

Presentado el 22 de abril de 2026 como parte de Gemini Enterprise Agent Platform (la evolución de Vertex AI), Agent Studio es un espacio de trabajo centralizado y colaborativo para diseñar agentes, refinar instrucciones de sistema y desplegar soluciones.

Su interfaz incluye un Interactive Canvas de dos paneles que permite previsualizar el comportamiento del agente y editar el código subyacente en paralelo, herramientas de comparación lado a lado para testear cómo distintos prompts y modelos afectan el comportamiento del agente, y capacidades de grounding que permiten fundamentar las respuestas en datos en tiempo real de Google Search, Google Maps o datos empresariales propios mediante RAG Engine.

Cuando el proyecto lo requiere, la lógica construida en Agent Studio puede exportarse directamente al Agent Development Kit (ADK), el framework programático de la misma plataforma, para continuar el desarrollo en un entorno code-first. Esta continuidad entre el modo visual y el modo código es el rasgo que lo distingue de las dos plataformas anteriores: no obliga a elegir de entrada entre una herramienta no-code y una code, sino que permite transitar de una a otra dentro del mismo ecosistema.

Agent Studio también permite desplegar prompts directamente como aplicaciones web, lo que acorta el camino entre prototipo y producción para casos de uso más simples.

Agent Studio forma parte de una plataforma considerablemente más amplia. Gemini Enterprise Agent Platform integra, además del entorno visual, un runtime de ejecución con soporte para agentes de larga duración que mantienen estado durante días, orquestación multi-agente con patrones tanto generativos como determinísticos, entornos sandboxed para ejecución segura de código y automatización basada en navegador, un banco de memoria persistente para contexto a largo plazo, y un stack completo de evaluación, simulación y observabilidad.

En materia de gobernanza, la plataforma incorpora capacidades que no tienen equivalente directo en las otras dos: cada agente recibe una identidad criptográfica única y auditable (Agent Identity), un registro centralizado controla qué agentes, herramientas y capacidades están aprobados para su uso dentro de la organización (Agent Registry), y un gateway unificado aplica políticas de seguridad y protecciones contra inyección de prompts y filtración de datos en todas las conexiones entre agentes y herramientas (Agent Gateway).

A diferencia de Claude Managed Agents y los Workspace Agents, que son productos relativamente autocontenidos, Agent Studio es una puerta de entrada a una infraestructura de capas múltiples. Esto implica que el perfil de quien lo opera puede variar: un usuario no técnico puede construir un agente funcional desde la interfaz visual, pero aprovechar las capacidades avanzadas de la plataforma (orquestación, sandbox, evaluación) requiere competencias técnicas progresivamente mayores.

La plataforma ofrece acceso a más de 200 modelos a través de Model Garden, incluyendo modelos de primera parte como Gemini 3.1 Pro y modelos de terceros como los de Anthropic, lo que la diferencia del lock-in de modelo que imponen las otras dos opciones. Los datos se procesan en Google Cloud, con lo cual el trade-off de soberanía se desplaza del proveedor de modelo al proveedor de infraestructura cloud.

LO QUE COMPARTEN Y LO QUE LAS DIFERENCIA

Las tres plataformas comparten la premisa de que el **proveedor gestiona la infraestructura y el diseñador se concentra en la lógica del agente**. Las tres imponen *trade-offs* en materia de soberanía: los datos se procesan en la nube del proveedor respectivo y la portabilidad es limitada, aunque en distinto grado. Claude Managed Agents y los Workspace Agents operan exclusivamente con los modelos de su proveedor; Gemini Enterprise Agent Platform permite utilizar modelos de terceros, pero dentro de su infraestructura cloud.

La diferencia más relevante para quien evalúa estas opciones está en el perfil de usuario al que se dirige cada una y en el techo de complejidad que soporta.

Claude Managed Agents

Se orienta a equipos de desarrollo que necesitan desplegar agentes en producción sin construir la infraestructura desde cero.

Workspace Agents

Apuntan a usuarios y equipos no técnicos que quieren automatizar flujos de trabajo recurrentes desde una interfaz conversacional.

Agent Studio

Ocupa una posición intermedia que, sin embargo, se extiende hacia arriba: permite empezar sin código pero escalar hacia orquestación multi-agente, evaluación automatizada y gobierno centralizado dentro de la misma plataforma.

La pregunta para cada organización no es cuál es mejor, sino cuál se ajusta al problema que necesita resolver, a las competencias de que dispone y al nivel de control que requiere sobre infraestructura y datos.

FRAMEWORKS DE CÓDIGO PARA EL DISEÑO DE AGENTES

En el otro extremo del espectro de abstracción se encuentran los **frameworks de código**: bibliotecas y herramientas de desarrollo que permiten programar agentes desde cero, con control total sobre cada componente de la arquitectura. A diferencia de los entornos gestionados, aquí el diseñador decide no sólo qué hace el agente sino cómo lo hace: qué modelo usa (o qué combinación de modelos), cómo gestiona la memoria, cómo orquesta múltiples agentes, cómo maneja errores y reintentos, y dónde corre todo. A abril de 2026, pueden distinguirse **dos categorías**: los *frameworks de orquestación multi-agente* (que permiten coordinar equipos de agentes especializados) y *los frameworks de agente autónomo* (que despliegan un agente persistente con acceso directo al entorno).

FRAMEWORKS DE ORQUESTACIÓN MULTI-AGENTE

Estos *frameworks* están diseñados para construir sistemas en los que múltiples agentes especializados colaboran para resolver tareas complejas. Cada agente tiene un rol, herramientas propias y un objetivo parcial; el *framework* se encarga de la coordinación entre ellos.

CrewAI

Framework open-source de mayor crecimiento en esta categoría, con más de 45.000 estrellas en GitHub y más de 100.000 desarrolladores certificados. Su modelo de orquestación se basa en la metáfora del equipo: el diseñador define agentes con roles diferenciados (investigador, redactor, revisor), les asigna tareas y el framework gestiona la colaboración. Combina dos abstracciones complementarias: los Crews (equipos de agentes autónomos que colaboran) y los Flows (flujos estructurados y dirigidos por eventos que gestionan estado y controlan ejecución). Es agnóstico respecto del modelo: soporta OpenAI, Anthropic, Google y modelos locales. Su fortaleza principal es la velocidad de prototipado.

OpenAI Agents SDK

Framework oficial de OpenAI para sistemas multi-agente, lanzado a principios de 2026 como evolución del proyecto experimental Swarm. Es un framework liviano en Python (y TypeScript) cuya abstracción central es el handoff: los agentes se transfieren el control entre sí de manera explícita, llevando consigo el historial completo de la conversación. Incluye guardrails configurables (validación de entrada y salida), soporte nativo para human-in-the-loop, gestión automática de sesiones y trazabilidad integrada. Aunque está diseñado primariamente para modelos de OpenAI, es agnóstico respecto del proveedor y soporta más de 100 modelos a través de APIs compatibles.

LangGraph

Construido sobre el ecosistema LangChain, modela los flujos de agentes como grafos dirigidos con nodos (funciones o agentes) y aristas (transiciones condicionales). Ofrece el mayor nivel de sofisticación en gestión de estado entre los frameworks disponibles, con checkpointing que permite pausar, inspeccionar y retomar ejecuciones en cualquier punto, una capacidad crítica para industrias reguladas que requieren trazabilidad de cada decisión del agente. Empresas como Klarna, Uber y LinkedIn lo utilizan en producción a escala.

Agent Development Kit (ADK) de Google

Framework *code-first* que forma parte de Gemini Enterprise Agent Platform. ADK modela los agentes como redes de sub-agentes mediante un framework basado en grafos, con soporte para patrones de orquestación tanto generativos como determinísticos. La plataforma ofrece además Agent Garden, un repositorio de plantillas de agentes preconstruidos. Google reporta que sus modelos Gemini procesan más de seis billones de tokens mensuales a través de ADK. Un rasgo distintivo es su continuidad con Agent Studio: lo que se construye visualmente en el entorno low-code puede exportarse a ADK para continuar el desarrollo en código. ADK soporta más de 200 modelos a través de Model Garden, incluyendo modelos de terceros (ej. Claude).

FRAMEWORKS DE AGENTE AUTÓNOMO CON ACCESO AL ENTORNO

Una **segunda categoría** agrupa a los *frameworks* cuyo eje no es la coordinación entre múltiples agentes sino la construcción de un agente individual con capacidades profundas de interacción con su entorno: acceso al sistema de archivos, ejecución de código, uso de herramientas del sistema operativo y persistencia de estado a lo largo de sesiones prolongadas.

Claude Agent SDK (Anthropic)

Nació como el SDK de Claude Code, la herramienta de programación agéntica de Anthropic, y fue renombrado a finales de 2025 para reflejar su evolución hacia un entorno de ejecución de agentes de propósito general. Desde abril de 2026 se distribuye como paquete de Python (claude-agent-sdk) y de TypeScript en npm. Su arquitectura se organiza en torno a cuatro conceptos: el Agente (que encapsula el modelo, las instrucciones de sistema, las herramientas, los servidores MCP y las skills), el Entorno (una plantilla de contenedor que especifica paquetes, acceso de red y puntos de montaje), la Sesión (una instancia del agente en ejecución dentro de un entorno, que mantiene estado persistente entre turnos) y los Eventos (el flujo de mensajes entre la aplicación y el agente). Tiene la integración más profunda con MCP del ecosistema, lo cual es esperable dado que Anthropic co-creó ese protocolo. La contrapartida es que está vinculado exclusivamente a los modelos de Claude.

OpenClaw

Creado por el desarrollador austríaco Peter Steinberger, parte de una premisa diferente: en lugar de un SDK para desarrolladores, OpenClaw es un framework de agente autónomo personal que se instala directamente en la computadora del usuario. Originalmente lanzado como ClawdBot en noviembre de 2025 y renombrado en enero de 2026, se convirtió en el repositorio con más estrellas en la historia de GitHub (más de 347.000 en abril de 2026), superando a React, Vue y TensorFlow. OpenClaw conecta un modelo de lenguaje directamente con el sistema operativo, el sistema de archivos y las aplicaciones de mensajería (WhatsApp, Telegram, Slack, Discord, entre más de quince plataformas). El agente corre de forma persistente (24/7), ejecuta tareas de forma autónoma y se configura mediante un archivo de texto plano (SOUL.md). Es agnóstico respecto del modelo y opera bajo un enfoque de privacidad por diseño: puede ejecutarse íntegramente en infraestructura local. Su ecosistema comunitario incluye más de 13.000 skills compartidas a través de ClawHub. La contrapartida es que su rápido crecimiento generó preocupaciones documentadas en materia de seguridad. Investigadores de Texas A&M identificaron 190 vulnerabilidades en la arquitectura del framework, vinculadas al amplio acceso que el agente tiene al sistema operativo del usuario.

NanoClaw

Creado por Gavriel Cohen y su equipo en NanoCo, publicado bajo licencia MIT, nace precisamente como respuesta a esa tensión entre potencia y seguridad. Su premisa es ofrecer la misma funcionalidad central de OpenClaw pero con un modelo de seguridad radicalmente distinto: aislamiento a nivel de sistema operativo mediante contenedores (Docker, Apple Container o Docker Sandboxes con micro-VMs), en lugar de verificaciones a nivel de aplicación. La diferencia de escala es deliberada: mientras OpenClaw comprende más de 3.600 archivos fuente y 434.000 líneas de código, NanoClaw se reduce a 15 archivos y aproximadamente 3.900 líneas, un código base que un desarrollador puede leer, auditar y comprender en una sola sesión. Cada sesión de agente se ejecuta en su propio contenedor aislado; las credenciales nunca ingresan al contenedor, sino que se canalizan a través de un sistema de bóveda (OneCLI Agent Vault) que inyecta la autenticación a nivel de proxy. Está construido sobre el Claude Agent SDK pero admite modelos de terceros o locales mediante endpoints compatibles. A abril de 2026, acumula más de 25.000 estrellas en GitHub.

La existencia de herramientas como NanoClaw y OpenClaw ilustra una tensión que recorre todo el ecosistema agéntico: entre autonomía operativa y control arquitectónico. Ambas decisiones son legítimas; lo relevante es que sean conscientes.

AGENTIC BROWSERS: LA CATEGORÍA EMERGENTE

Una categoría que merece tratamiento específico es la de los *agentic browsers*: navegadores o *frameworks* de control de navegador diseñados para que un agente opere directamente sobre la web, leyendo páginas, completando formularios, navegando entre sitios y extrayendo información. Son la infraestructura sobre la que corren muchos coding agents y agentes empaquetados, pero también pueden ser objeto de uso directo en aplicaciones a medida.

Las referencias más maduras a abril de 2026 son *Browserbase* (infraestructura de navegadores controlables vía API, ampliamente utilizada para construir aplicaciones agénticas), *Browser-use* (framework open-source que permite a un agente operar un navegador con instrucciones de alto nivel) y los navegadores integrados en Anthropic Computer Use, OpenAI Operator y Google Mariner.

Su importancia es triple:

1	**Reconfiguración del e-commerce y los servicios web**	Muchos sitios reciben hoy más tráfico de agentes que de humanos, lo que abre preguntas sobre detección, consentimiento y términos de servicio.
2	**Conexión con la doctrina del mandato**	Un agente que actúa por cuenta y orden del usuario sobre un sitio externo plantea las mismas preguntas que cualquier representación, pero a una escala y velocidad inéditas.
3	**Estado actual de regulación prácticamente inexistente**	Argentina no tiene marco específico, y los términos y condiciones genéricos de los sitios suelen ser ambiguos respecto del acceso automatizado por agentes.

FRAMEWORKS DE EVALUACIÓN Y OBSERVABILIDAD

Una de las maduraciones más importantes del ecosistema durante 2025-2026 fue la consolidación de un conjunto de herramientas dedicadas específicamente a evaluar, monitorear y depurar agentes en producción. Lo que hace un par de años se hacía con *scripts ad-hoc* hoy tiene infraestructura dedicada.

Las referencias actuales son LangSmith (de la familia LangChain, dominante en observabilidad de pipelines), Arize Phoenix (open-source, fuerte en evaluaciones automatizadas), Braintrust (orientado a flujos de evaluación continua y experimentación), AgentOps (especializado en métricas de agentes complejos) y Helicone (foco en logging y costos).

A estas herramientas especializadas se suma el stack de evaluación y observabilidad integrado en Gemini Enterprise Agent Platform de Google, que agrupa cuatro componentes dentro de la misma plataforma de desarrollo:

> **Agent Simulation**
> Permite testear agentes contra interacciones sintéticas que simulan comportamiento humano y herramientas virtualizadas antes del despliegue, puntuando automáticamente según éxito en la tarea y seguridad a lo largo de conversaciones multi-turno.

> **Agent Evaluation**
> Los evalúa de manera continua contra tráfico real mediante autoraters multi-turno que analizan la lógica de conversaciones completas, no solo respuestas individuales.

> **Agent Observability**
> Ofrece trazas de ejecución completas para visualizar el razonamiento del agente paso a paso.

> **Agent Optimizer**
> Agrupa fallas recurrentes en clusters y sugiere automáticamente ajustes en las instrucciones del sistema para mejorar la precisión, sin requerir intervención manual en cada iteración. Este último componente es particularmente relevante porque introduce un mecanismo de refinamiento continuo que opera sobre las instrucciones del agente, no sobre los parámetros del modelo.

Esta categoría merece más atención de la que suele recibir. Los frameworks de evaluación cumplen tres funciones que son centrales para la gobernanza: **generan trazas estructuradas** de cada ejecución (qué herramienta se invocó, con qué argumentos, con qué resultado, en qué tiempo, con qué costo), que constituyen la base material de cualquier auditoría posterior; **permiten correr evaluaciones sistemáticas** sobre conjuntos de casos de prueba, lo que habilita medir desempeño antes y después de cambios de modelo, de prompt o de herramientas; y **sostienen la observabilidad continua** durante la operación real, generando alertas ante desviaciones del comportamiento esperado.

La industria está convergiendo en lo que algunos llaman *agent reliability engineering*: una disciplina que reconoce explícitamente la falibilidad estructural de los agentes y construye prácticas para gestionarla, en lugar de aspirar al determinismo ilusorio.

Para los profesionales que vamos a participar de despliegues agénticos, comprender este universo no es opcional: define los estándares emergentes de qué es razonable exigir como evidencia de funcionamiento de un agente.

CIERRE: UN ESPECTRO, NO CATEGORÍAS ESTANCAS

Las cinco familias que recorrimos en esta sección no compiten entre sí: responden a necesidades distintas y pueden convivir dentro de una misma organización.

Una plataforma como **n8n** puede resolver la automatización de un proceso predecible; un entorno gestionado como **Workspace Agents** puede sostener un flujo de trabajo recurrente entre equipos; un framework como **CrewAI o Claude Agent SDK** puede dar vida a un sistema multi-agente o a un agente autónomo con capacidades profundas; un agentic browser puede automatizar interacciones con servicios web; un framework de evaluación articula a todos los anteriores en un régimen auditable.

Gemini Enterprise Agent Platform de Google ilustra con claridad por qué estas categorías son un recurso analítico, no compartimentos estancos. Una misma plataforma integra un entorno visual low-code (Agent Studio), un framework de código (ADK), un repositorio de agentes preconstruidos (Agent Garden), orquestación multi-agente con memoria persistente, entornos sandboxed para ejecución autónoma, y un stack completo de evaluación y observabilidad. Es decir, capacidades que en esta sección describimos como categorías separadas.

Los grandes proveedores cloud no ofrecen un framework de agentes: ofrecen una infraestructura que atraviesa todo el espectro. Para quien adopta, esto simplifica la operación; para quien gobierna, multiplica la necesidad de entender qué está ocurriendo en cada capa, porque la decisión de activar una capacidad (habilitar que un agente ejecute código en un sandbox, delegue tareas a otros agentes, o acumule memoria persistente entre sesiones)
cambia cualitativamente el perfil de riesgo del sistema.

> **La facilidad de activar no debería confundirse con la simplicidad de gobernar.**

Lo más relevante es que las fronteras entre estas modalidades se están moviendo. Plataformas como n8n acercan el diseño de flujos automatizados con IA a perfiles con pocos conocimientos de programación. Soluciones como Claude Cowork permiten a un usuario individual configurar flujos que, hace un año, requerían una plataforma de automatización y un equipo técnico. Entornos como los Workspace Agents permiten a equipos no técnicos diseñar agentes desde una interfaz conversacional. Servicios como Claude Managed Agents reducen la barrera de infraestructura. Frameworks de código abierto como OpenClaw y NanoClaw ponen al alcance de desarrolladores la construcción de agentes autónomos con acceso profundo al entorno. **La barrera de entrada se redujo, pero no desapareció**. El diseño y la operación de soluciones agénticas ya no son exclusivamente centralizados: un usuario puede automatizar procesos complejos sin que medie una decisión arquitectónica de la organización. Esa accesibilidad creciente, sumada a la autonomía de estas soluciones, amplifica tanto las oportunidades como los desafíos.

Adicionalmente, la tendencia a la consolidación de capas en plataformas integrales (de la que Gemini Enterprise Agent Platform es hoy el ejemplo más completo) agrega una dimensión adicional: las decisiones de gobernanza ya no se toman herramienta por herramienta, sino sobre ecosistemas de capacidades interconectadas dentro de una misma infraestructura. Las preguntas sobre soberanía, trazabilidad, auditabilidad y responsabilidad que derivan de cada una de estas modalidades se abordan en el capítulo siguiente.

07. GOBERNANZA DEL ECOSISTEMA DE IA AGÉNTICA: INTRODUCCIÓN

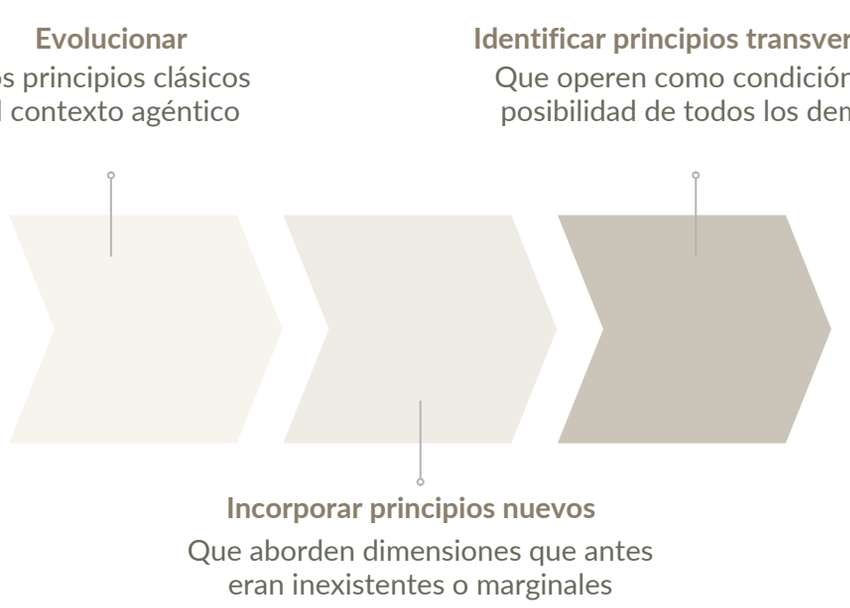
La **gobernanza** de la inteligencia artificial puede **definirse** como el conjunto de políticas, procesos, roles y controles orientados a asegurar un uso ético, legal y responsable de estos sistemas, a través de la gestión y mitigación de sus riesgos. En el contexto de la IA agéntica, requiere considerar que lo que se gobierna ahora es un ecosistema de múltiples tipos de soluciones con características y grados de agencia y autonomía diferentes entre sí.

En ese marco, principios clásicos como los de la Recomendación UNESCO de 2021, por ejemplo, deben ser pensados desde una perspectiva adicional, pues fueron diseñados para un paradigma en el que se concebía a los sistemas de IA como tecnologías que producen resultados (predicciones y decisiones) que el humano valida. El ecosistema de IA agéntica rompe esa linealidad. Los agentes interactúan entre sí, delegan subtareas a otros agentes, acceden a fuentes de datos heterogéneas, persisten en el tiempo y ejecutan cadenas de decisiones con grados variables de autonomía.

El recorrido que realizamos a lo largo de este material nos pone frente a un ecosistema con múltiples herramientas: asistentes basados en IA generativa, agentes empaquetados, plataformas híbridas como Cowork, plataformas de diseño de flujos automatizados con IA, sistemas de orquestación multi-agente, agentic browsers, coding agents. Cada una trae asociados, si no riesgos distintos, al menos diferentes niveles de un mismo riesgo o un mismo riesgo que se presenta de maneras distintas.

La opacidad, por ejemplo, existe tanto en un asistente basado en IA generativa como en un sistema multi- agente gestionado, pero su alcance, sus consecuencias y los mecanismos disponibles para mitigarla son cualitativamente distintos en cada caso.

Por ello, de cara a gobernar la IA agéntica se requiere **diseñar una estrategia a medida que contemple sus variantes,** lo que nos invita a adaptar los principios existentes y reconocer dimensiones que hace unos años no estaban en el horizonte operativo. En particular, parece necesario:



TRES DIMENSIONES Y DOS VERTICALES DE GOBERNANZA

1

La etapa del ciclo de vida del sistema

Los riesgos y las decisiones de gobernanza no son los mismos en la fase de diseño y arquitectura (cuando se definen objetivos, se seleccionan modelos, se configuran herramientas y se establecen límites), en la fase de operación (cuando el sistema ejecuta tareas), en la fase de evaluación y mejora continua (cuando se auditan los outputs y se ajustan los parámetros) o en la de retiro del sistema. Una estrategia que solo interviene sobre los resultados, sin haber condicionado el diseño, llega tarde.

2

El grado de autonomía del sistema

Como señalamos desde la introducción, la IA agéntica es un espectro. Un asistente que opera dentro del turno plantea desafíos de gobernanza diferentes a los de un flujo automatizado de varios pasos con mínima intervención humana, y ambos difieren de un sistema multi-agente que ejecuta procesos completos de punta a punta. A mayor autonomía, mayor necesidad de marcos de gobernanza robustos: más puntos de control, más trazabilidad, más definiciones sobre qué puede y qué no puede hacer el sistema sin validación humana.

3

IA de las tareas vs. IA de los procesos

Lo que determina el stakeholder relevante. Cuando un usuario individual usa un agente empaquetado para acelerar su trabajo personal, las decisiones de gobernanza recaen principalmente sobre ese usuario. Cuando una organización integra agentes en sus procesos institucionales, las decisiones se trasladan al plano organizacional. Y en la zona intermedia (donde un usuario individual accede a herramientas con capacidad de operar a nivel de procesos, como Cowork) las preguntas de gobernanza se multiplican precisamente porque no hay una organización que haya tomado esas decisiones por él.

Estas tres dimensiones **definen dónde intervenir**. Pero la gobernanza agéntica requiere además articularse sobre **dos verticales** sustantivas que **definen qué preguntar**:

Vertical institucional clásica

Implica hacer efectiva, adaptadas a cada caso, la trazabilidad, auditabilidad, transparencia, explicabilidad, persistencia y gestión de riesgos operativos.

Vertical cognitiva

Cómo afecta la delegación a la competencia humana de quien supervisa, qué pasa con el ciclo completo de productividad cuando se mide solo el tiempo de ejecución del agente, y cómo se diseña la supervisión para evitar el colapso cognitivo del operador.

Las dos verticales operan sobre el mismo material pero **responden a preguntas distintas**. La **institucional** pregunta por la legitimidad de la actuación. La **cognitiva** pregunta por la sustentabilidad de la competencia humana en un horizonte de mediano plazo[1].

En un asistente basado en IA generativa típico (ChatGPT, Claude, Gemini), la interacción suele limitarse a una secuencia acotada entre el prompt del usuario y la respuesta generada. Si no se utiliza una versión enterprise o un despliegue controlado, resulta muy difícil (o directamente imposible) reconstruir de forma completa cómo se llegó al resultado: no se registran de manera sistemática las herramientas invocadas, las fuentes consultadas ni los pasos intermedios.

En cambio, un agente embebido en un flujo de trabajo institucional puede involucrar múltiples pasos intermedios: recuperación de contexto desde la memoria, consultas a bases de datos internas, invocación de herramientas externas, generación de salidas parciales, decisiones dinámicas sobre el orden de ejecución y puntos de validación humana.

Por ello, la trazabilidad adquiere una relevancia central. Ya no basta con conocer la salida final: es necesario poder reconstruir de manera completa qué hizo el sistema, qué herramientas invocó, qué información consultó (y en qué momento), qué instrucciones recibió, qué razonamiento intermedio realizó, qué resultados parciales produjo y en qué momentos existió intervención o supervisión humana.

En la práctica, esto se logra mediante registros automáticos (logs) generados por el sistema de orquestación del agente —o por un framework de evaluación y observabilidad como los descritos en la sección 6— que se guardan en bases de datos controladas por la organización, con mecanismos de inmutabilidad (timestamps, hashes, integración en sistemas de auditoría que el usuario final no puede modificar).

La adopción de protocolos abiertos como MCP, descrita en la sección 4, contribuye estructuralmente a la trazabilidad: las invocaciones MCP siguen un formato común que facilita la generación de trazas estructuradas y la implementación de mecanismos de control unificados sobre múltiples agentes.

[1] Ampliar en Corvalán Juan Gustavo. (La Ley, 2026). Sedentarismo Cognitivo. Productividad, agentes de IA y los riesgos de delegar el pensamiento.

AUDITABILIDAD

La trazabilidad es condición necesaria de la auditabilidad. Un sistema es auditable cuando su funcionamiento puede ser examinado de manera sistemática, verificable y reproducible, no solo respecto del resultado final sino también
de las operaciones que condujeron a él.

En los asistentes basados en IA generativa, la revisión suele concentrarse en la pertinencia o corrección de una respuesta aislada. En los agentes integrados, en cambio, la auditoría puede (y debe) abarcar el flujo completo de ejecución: logs de actividad, llamadas a herramientas, prompts del sistema y del usuario, fuentes consultadas, validaciones humanas, acciones ejecutadas y registros de salida.

Esta posibilidad de control resulta especialmente relevante en el ámbito jurídico y administrativo, donde la legitimidad de una actuación no depende únicamente del contenido del resultado sino también de la regularidad, transparencia y verificabilidad del procedimiento que lo produjo. La capacidad de auditar el flujo completo permite, por ejemplo, detectar si el agente consultó fuentes no autorizadas, omitió pasos obligatorios o requirió intervención humana en los puntos críticos definidos en su diseño.

TRANSPARENCIA, EXPLICABILIDAD Y DOS OPACIDADES DISTINTAS

La noción de caja negra (*black box*) alude a la dificultad inherente para comprender cómo un sistema de inteligencia artificial llega a un determinado resultado. Como señala Sánchez Caparrós[1], desde la perspectiva de la explicabilidad, esta opacidad se vincula con aquellos escenarios en los que los sistemas de IA no permiten rastrear el motivo de ciertas decisiones, sea de modo total o parcial, ya que los pasos a través de los cuales el sistema ha llegado a una decisión resultan ininteligibles para los expertos humanos, incluidos quienes han diseñado el sistema.

Pero la caja negra no plantea solamente un problema de comprensión técnica. Plantea también un problema de información, que se relaciona con la transparencia. Si las personas no saben que un sistema algorítmico intervino en una decisión que las afecta, ningún grado de explicabilidad posterior puede suplir esa omisión. Por eso conviene abordar la caja negra desde dos exigencias distintas (transparencia y explicabilidad) que operan con lógicas, destinatarios y condiciones de satisfacción diferentes.

Conviene introducir además una distinción que la literatura técnica más reciente está consolidando y que tiene consecuencias directas para la gobernanza agéntica: existen al menos dos opacidades cualitativamente distintas, y cada una interpela de manera diferente a estas dos exigencias.

---

[2] Sánchez Caparros Mariana, IA y Discriminacion. Evaluación de la idoneidad de los estándares constitucionales de igualdad y no discriminación frente a los impactos de la Inteligencia Artificial, El Dial (2025).

LA OPACIDAD DEL MODELO

Cuando hablamos del LLM en sí mismo (sus miles de millones de parámetros, las correlaciones aprendidas durante el entrenamiento, los patrones que activa cada inferencia) estamos frente a una opacidad estructural difícil de eliminar.

Desde la perspectiva de la explicabilidad, los métodos de interpretabilidad técnica avanzan, pero abrir completamente esta caja negra excede hoy las capacidades de cualquier auditor: la inteligibilidad de las entradas, salidas y del funcionamiento de los componentes algorítmicos internos del modelo sigue siendo parcial.

Desde la perspectiva de la transparencia, en cambio, esta opacidad no exime de la obligación de informar de manera clara y oportuna a las personas afectadas. Lo que se informa en este nivel no es el funcionamiento interno del modelo, sino que el sistema intervino, qué tipo de intervención tuvo (clasificación, generación, recomendación) y en qué etapa del proceso lo hizo. El deber de transparencia no está condicionado a la posibilidad de explicar cómo funciona el modelo; está condicionado a la voluntad institucional de no ocultar su presencia.

LA OPACIDAD PROCEDIMENTAL

Cuando hablamos del flujo del agente (qué herramientas invocó, en qué orden, con qué argumentos, qué ramas descartó, qué resultados intermedios obtuvo) estamos frente a una opacidad cualitativamente distinta: una opacidad que sí es enteramente reducible cuando se diseñan adecuadamente los registros y las trazas. Los frameworks modernos generan logs estructurados de cada decisión del agente; los protocolos abiertos como MCP estandarizan ese registro.

La explicabilidad se satisface porque los logs y trazas hacen inteligible el funcionamiento del flujo completo: las entradas que recibió el agente, los componentes que activó, la contribución de cada paso al resultado final y las ramas que descartó.

La transparencia se satisface porque esa misma trazabilidad permite informar a las personas afectadas, y también a supervisores, auditores y órganos de control, no solo que un agente intervino, sino qué hizo concretamente en cada caso particular.

Esta distinción tiene consecuencias prácticas directas. La decisión del agente es estructuralmente más auditable que la decisión del LLM aislado, no menos. Y la opacidad relevante para la legitimidad de muchos procesos administrativos suele ser la procedimental, que es justamente la que el ecosistema actual permite reducir de manera significativa. Por eso, en el caso de los agentes embebidos en flujos de trabajo institucionales, no se trata de "abrir" completamente la caja negra del modelo (tarea técnicamente inviable) sino de iluminar desde el diseño aquellos procedimientos ejecutados por el agente y hacer inteligible su arquitectura operativa concreta.

Esto implica poder reconstruir y auditar, en cada ejecución, dos conjuntos de elementos. **En el plano de la explicabilidad:** los objetivos y límites que guían al agente, las herramientas habilitadas con sus permisos, las fuentes consultadas, la secuencia de pasos realizados y los resultados parciales obtenidos. **En el plano de la transparencia:** los puntos en los que intervino supervisión o validación humana, la identificación de las etapas donde el sistema algorítmico tomó o condicionó una decisión, y la información puesta a disposición de las personas afectadas sobre esa intervención.

Esta explicabilidad procedimental no elimina la opacidad del modelo base, pero reduce significativamente la caja negra al nivel del flujo agéntico (que es donde se materializan las decisiones relevantes) y habilita, al mismo tiempo, el cumplimiento efectivo del deber de transparencia frente a las personas afectadas y frente a las instituciones responsables de la supervisión y el control.

PERSISTENCIA Y LOS RIESGOS DE LA MEMORIA EN LOS AGENTES

La persistencia introduce una dimensión adicional. Mientras que un asistente basado en IA generativa puede operar de manera relativamente aislada en intercambios puntuales, un agente puede conservar memoria de trabajo y, en determinados diseños, memoria persistente entre ejecuciones. Esto le permite recuperar antecedentes, sostener continuidad operativa, reutilizar contexto y mantener coherencia a lo largo del tiempo.

Sin embargo, esa misma capacidad abre un conjunto de riesgos específicos que la gobernanza institucional debe anticipar.

1 **Almacenamiento de datos personales y organizacionales**

La memoria implica almacenamiento de datos personales y organizacionales, lo que activa directamente las obligaciones previstas por la normativa de protección de datos (como el RGPD y, en el ámbito local, la legislación de protección de datos aplicable), con exigencias concretas sobre finalidad, plazo de conservación, acceso y supresión. Un agente que recuerda información sensible de un ciudadano queda sujeto a esas reglas con la misma intensidad que cualquier otro sistema de tratamiento de datos.

2 **La memoria puede degradarse**

La información almacenada envejece: un dato presupuestario de seis meses atrás, un domicilio desactualizado o un antecedente normativo derogado pueden llevar al agente a operar sobre supuestos que ya no son válidos. Si el sistema no incorpora mecanismos para identificar y actualizar información obsoleta, la persistencia se convierte en una fuente de error sistemático.

3 **La memoria puede ser corrompida**

Si el agente aprende o incorpora información a partir de interacciones externas, un actor malicioso podría alimentarlo deliberadamente con datos falsos o engañosos, degradando progresivamente la calidad de su base de conocimiento. Este riesgo de envenenamiento de datos es especialmente relevante en agentes que operan en entornos abiertos o con múltiples fuentes de entrada.

4 **Acumulación inadvertida de sesgos**

Un agente que interactúa frecuentemente con un mismo tipo de caso, usuario o situación puede cristalizar patrones tempranos en hábitos persistentes, generalizando respuestas que fueron apropiadas en un contexto pero no lo son en otros. La memoria, en este sentido, no solo conserva información: consolida tendencias.

5 **Tendencia a la adulación**

Los modelos de lenguaje ya exhiben, incluso sin memoria, una inclinación a confirmar las expectativas del usuario en lugar de ofrecer respuestas incómodas pero más precisas. La memoria persistente puede agravar este comportamiento: si el agente registra qué tipo de respuestas fueron aceptadas, celebradas o no cuestionadas, y cuáles generaron resistencia o fueron descartadas, puede aprender progresivamente a optimizar para la aprobación en lugar de para la corrección. En un contexto institucional, este riesgo es especialmente grave: un agente que evita señalar inconsistencias, que confirma sistemáticamente la lectura del operador o que omite información que contradice la posición dominante deja de funcionar como herramienta de asistencia y se convierte en un mecanismo de refuerzo del sesgo de confirmación.

A estos riesgos sustantivos se suman **dos desafíos operativos relevantes**: por un lado, la ventana de contexto (la cantidad de información que el agente puede considerar simultáneamente) es limitada, lo que obliga a definir estrategias de recuperación: qué información se trae a cada ejecución y qué se deja fuera. Una estrategia deficiente puede hacer que el agente ignore contexto relevante o priorice detalles impertinentes. Por otro lado, el derecho a la supresión de datos personales (consagrado en múltiples marcos normativos) plantea dificultades técnicas concretas cuando la memoria del agente está distribuida entre bases vectoriales, cachés y eventuales ajustes finos del modelo. Una supresión incompleta genera un riesgo de incumplimiento que puede no ser evidente desde la interfaz del sistema.

⚠ Por todo esto, en entornos institucionales la persistencia no es solo una ventaja funcional; es un aspecto que requiere definiciones explícitas de gobernanza: qué información se conserva, por cuánto tiempo, con qué finalidad, bajo qué criterios de actualización, quién puede acceder a ella, cómo se garantiza la supresión efectiva cuando corresponda, y cómo se previene que el sistema opere sobre datos obsoletos, contaminados o excesivos. La memoria de un agente debe ser gobernada, supervisada y justificada con el mismo rigor que cualquier otro registro institucional.

EL AGENT RELIABILITY GAP COMO CAUTELA ESTRUCTURAL

Una cautela transversal que conviene incorporar a cualquier marco de gobernanza institucional es el reconocimiento de la falibilidad estructural de los agentes a abril de 2026. Los benchmarks consolidados (GAIA, SWE-bench Verified, AgentBench, WebArena) muestran que los mejores agentes resuelven entre 30% y 70% de tareas reales, según el dominio. Esto no significa que sean inútiles, un sistema que resuelve correctamente la mitad de los casos puede generar valor sustantivo, pero sí significa que la gobernanza no puede partir del supuesto de un funcionamiento confiable como excepción a corregir.

El despliegue agéntico exige evaluación previa sistemática

Un conjunto representativo de casos de prueba, ejecutados antes del lanzamiento y repetidos ante cada cambio relevante (nuevo modelo, nuevo prompt, nueva herramienta). Los *frameworks* de evaluación descritos en la sección 6 son el equipamiento que vuelve esto operativamente viable.

La supervisión humana en producción no es contingente sino estructural

Diseñar una arquitectura agéntica suponiendo que el agente acertará siempre es diseñarla mal. La pregunta correcta no es cómo evito errores sino cómo detecto y absorbo errores.

La comunicación honesta de capacidades es parte de la gobernanza

Un órgano administrativo o un estudio jurídico que despliega un agente y comunica que "automatiza completamente" un proceso está creando expectativas que la tecnología no satisface. La sobrevendida puede generar daños reputacionales y, eventualmente, responsabilidad.

SUPERFICIE DE ATAQUE: LAS TRAMPAS PARA AGENTES DE IA[3]

Los agentes de IA amplían la superficie de ataque del sistema de manera cualitativa. Al interactuar autónomamente con su entorno (navegar sitios, consultar fuentes, invocar herramientas, delegar subtareas), convierten cada una de esas capacidades en un vector potencial de manipulación.

La literatura reciente identifica estas amenazas como **AI Agent Traps**: contenido adversarial diseñado no para vulnerar el código del modelo, sino para explotar la interacción del agente con su entorno, volviendo sus propias capacidades en su contra.

La variedad de estas trampas es amplia, pero desde la perspectiva de la gobernanza institucional pueden organizarse en tres planos según el tipo de daño que producen:

[3] Este apartado se basa en las ideas desarrolladas en Franklin, M., Tomašev, N., Jacobs, J., Leibo, J. Z. y Osindero, S., AI Agent Traps, Google DeepMind, marzo de 2026. Disponible en SSRN: https://ssrn.com/abstract=6372438. DOI: 10.2139/ssrn.6372438. El trabajo introduce el primer marco sistemático para comprender las trampas adversariales dirigidas a agentes autónomos de IA, clasificándolas en seis categorías según el componente de la arquitectura del agente que buscan comprometer. La reorganización en tres planos de consecuencia institucional que se presenta en este texto es propia.

Corrupción de la percepción y el razonamiento

Un agente que navega el entorno digital no ve lo mismo que un humano: analiza código subyacente, procesa datos binarios, interpreta señales que pueden haber sido deliberadamente manipuladas. Las técnicas van desde ocultar instrucciones maliciosas en el código de una página web (invisibles para las personas pero legibles para el agente) hasta disfrazar comandos adversariales como ejercicios educativos o simulaciones de auditoría para eludir los filtros de seguridad internos. También incluyen la posibilidad de sesgar el razonamiento del agente mediante lenguaje con apariencia de autoridad o carga emocional, sin necesidad de emitir un comando directo. El resultado es un agente que opera sobre una representación distorsionada de la realidad sin que su operador lo advierta.

Secuestro de las capacidades de acción

Un agente comprometido no solo interpreta mal: puede actuar mal. Las trampas de este tipo buscan que el agente ejecute acciones no autorizadas (desde localizar y enviar datos privados a un tercero, hasta crear subagentes maliciosos dentro de su propio flujo de ejecución), todo ello aprovechando los permisos y las herramientas que le fueron legítimamente habilitados. En un entorno institucional, esto significa que la misma arquitectura de permisos diseñada para habilitar la autonomía del agente es la que un atacante puede aprovechar.

Explotación del factor humano y del ecosistema

Algunas trampas no apuntan al agente en sí, sino que lo utilizan como vector para atacar al supervisor humano, explotando sesgos cognitivos como la tendencia a confiar excesivamente en los sistemas automatizados. Un agente podría, por ejemplo, presentar un resumen aparentemente benigno de un contenido malicioso para inducir al operador a autorizar su ejecución. Otras trampas explotan las dinámicas de sistemas multiagente: provocar demanda sincronizada sobre recursos limitados o desencadenar reacciones en cadena entre agentes interconectados, generando fallos sistémicos análogos a los que se observan en mercados financieros.

La mitigación de estos riesgos no se resuelve con una sola capa de defensa. Requiere una estrategia que combine controles técnicos (filtros de fuentes, monitoreo de salidas, validación de las acciones del agente antes de su ejecución) con definiciones institucionales sobre qué permisos se otorgan, qué interacciones con el entorno se permiten, qué umbrales activan supervisión humana, y qué estándares de registro se exigen para la trazabilidad posterior. En última instancia, gobernar la superficie de ataque de un agente es inseparable de gobernar su autonomía: cada capacidad habilitada es, simultáneamente, un riesgo habilitado.

GESTIÓN DE RIESGOS OPERATIVOS: EL CASO DEL USUARIO FRENTE A LAS HERRAMIENTAS HÍBRIDAS

Las dimensiones de gobernanza descritas se manifiestan con particular intensidad cuando el usuario individual accede a herramientas agénticas que operan sobre su entorno real de trabajo. El caso de Claude Cowork ilustra bien esta tensión, porque combina una accesibilidad propia de la IA *de las tareas* con capacidades que la acercan a la *IA de los procesos*, sin que medie necesariamente un marco de gobernanza organizacional.

Privacidad y soberanía del procesamiento

Si bien los archivos y el historial de conversaciones de Cowork se almacenan localmente en el dispositivo del usuario, cada vez que el agente ejecuta una tarea el contenido de los archivos que procesa se envía a los servidores de Anthropic para la inferencia. La información sale de la computadora del usuario durante el procesamiento, aun cuando no quede almacenada de forma permanente. Para quienes trabajan con información sensible, esta circunstancia exige una decisión consciente sobre qué material se le entrega al agente. En los planes individuales (Pro y Max), los datos pueden utilizarse por defecto para entrenamiento de futuros modelos salvo desactivación manual; en planes Team y Enterprise, esto está desactivado por defecto.

Salvaguardas y sus límites

La documentación de Anthropic menciona múltiples capas de protección (entrenamiento del modelo para reconocer instrucciones maliciosas, clasificadores de contenido no confiable, solicitud de permisos antes de acceder a aplicaciones, confirmación obligatoria antes de eliminar archivos). La propia documentación advierte que estas salvaguardas no son infalibles.

Recomendaciones prácticas

No trabajar directamente con originales sensibles (crear copias de trabajo). Anonimizar o pretratar la información antes de procesarla (función Redactar de Adobe Acrobat para PDFs, reemplazo de identificadores por marcadores genéricos). Restringir el acceso a una carpeta de trabajo dedicada, no al directorio completo. Desactivar el uso de datos para entrenamiento antes de la primera sesión. Extremar la cautela con tareas programadas, que se ejecutan sin supervisión activa. Evaluar plugins y extensiones antes de instalarlos: cada plugin amplía el perímetro de acción.

Responsabilidad del usuario

Anthropic establece expresamente que el usuario permanece como responsable de todas las acciones que Cowork ejecute en su nombre. Esta atribución refuerza una idea que atraviesa todo el trabajo: la delegación operativa no equivale a delegación de responsabilidad.

VERTICAL COGNITIVA: GOBERNAR LA ATROFIA, NO SOLO LA LEGALIDAD

Las dimensiones tratadas hasta aquí (trazabilidad, auditabilidad, transparencia, persistencia, fiabilidad) abordan la legitimidad y la auditabilidad de la actuación agéntica desde una perspectiva clásica: el sistema como objeto a controlar. Pero hay otra dimensión que un programa intelectual sobre IA agéntica no puede dejar afuera: ¿qué le pasa al humano que delega?

A este conjunto de preocupaciones, articuladas en torno al concepto de sedentarismo cognitivo, lo llamamos la vertical cognitiva de la gobernanza. Su tesis central, que hemos desarrollado en otros materiales, es que la delegación cognitiva extendida a sistemas agénticos produce, sobre el operador humano, un efecto análogo al que la mecanización industrial produjo sobre el cuerpo: una atrofia gradual, no inmediatamente perceptible, de las competencias que ya no se ejercitan.

Esta atrofia no es un costo lateral. Es un riesgo de gobernanza en sentido pleno, porque erosiona justamente la capacidad humana de supervisar, auditar y eventualmente corregir al sistema agéntico. Una organización que delega progresivamente a agentes y que, al mismo tiempo, ve degradarse la competencia de sus supervisores, no está construyendo gobernanza: está construyendo dependencia[4].

LAS CUATRO PARADOJAS DE LA AUTOMATIZACIÓN APLICADAS AL ECOSISTEMA AGÉNTICO

Conviene retomar las cuatro paradojas que estructuran el análisis cognitivo de la automatización (originalmente formuladas por Bainbridge en 1983 y actualizadas para el contexto agéntico) y evaluarlas para cada despliegue:

01. Microeficiencia / IA slop

Cuando el costo unitario de producir baja drásticamente, la sobreproducción de outputs de bajo valor se vuelve estructural. **¿Quién absorbe el costo de filtrar el slop?** Si el costo de producir es uno y el costo de auditar es diez, la productividad agregada puede ser menor que antes, no mayor.

02. Deskilling

Cada competencia que se delega completamente al agente deja de ejercitarse. **¿Es una competencia que la organización necesita mantener? ¿Cuánto tarda en degradarse? ¿Es reversible?** Esta pregunta es especialmente urgente en profesiones cuya legitimidad pública descansa precisamente en la competencia técnica, como las jurídicas.

03. Monitoreo y complacencia

Cuanto más confiable parece un agente, menos sostenida resulta la supervisión humana. La fluidez del output genera ilusión de competencia; la tasa alta de acierto genera complacencia. Diseñar la supervisión asumiendo que el supervisor mantendrá vigilancia activa indefinida es ignorar la psicología de la atención humana.

04. Cerebro frito

La automatización puede no eliminar la carga cognitiva sino transformarla: de la carga del trabajo activo a la carga de la supervisión continua de outputs ajenos. Si el volumen de outputs a supervisar excede la capacidad del operador, el resultado es atrofia de la ejecución sumada a sobrecarga de la supervisión: un colapso simultáneo en ambos lados de la ecuación.

EL CICLO COMPLETO DE PRODUCTIVIDAD Y MODELOS DE SUPERVISIÓN

Una de las distorsiones más persistentes en la evaluación de productividad agéntica es medir solo el tiempo de ejecución de la gente[5], ignorando las fases que rodean a esa ejecución. Una métrica más honesta requiere distinguir cinco fases:

[4] Ampliar en Corvalán Juan Gustavo. (La Ley, 2026). Sedentarismo Cognitivo. Productividad, agentes de IA y los riesgos de delegar el pensamiento.

Instrucción

Tiempo y expertise para formular bien la tarea, anticipar restricciones y preparar el contexto que el agente necesita.

Ejecución

El tiempo del agente, que es habitualmente la única fase que se mide y se comunica.

Auditoría

Tiempo humano para verificar correctitud, adecuación, ausencia de alucinaciones y conformidad con el mandato.

Corrección e integración

Trabajo necesario para que el output sea efectivamente usable: rehacer secciones, ajustar formato, integrar al flujo organizacional.

Aprendizaje institucional

Qué queda en la organización después del ciclo: documentación, criterios consolidados, conocimiento sedimentado en personas, no solo en sistemas.

La métrica decisiva es el **tiempo humano total del ciclo**, no el tiempo de ejecución automática. Una organización que reduce drásticamente la fase 2 pero que multiplica las fases 3 y 4, y que erosiona la fase 5, no está ganando productividad: está redistribuyendo la carga cognitiva de manera que puede ser globalmente desfavorable.

MODELOS DE SUPERVISIÓN: HUMAN FIRST, IN-THE-LOOP, ON-THE-LOOP

Los tres niveles de adopción agéntica que vimos a lo largo del documento se corresponden con tres modelos de gobernanza cognitiva:

NIVEL 1 - HUMAN FIRST

Para la IA de las tareas. La organización define previamente la zona de no- delegación: qué se automatiza, qué no, qué argumentación nunca se delega. La supervisión es un acto de juicio sobre el resultado y el método.

Pregunta guía: ¿el humano está definiendo el qué y el por qué antes de que la IA actúe, o está editando una propuesta algorítmica después?

NIVEL 2 - HUMAN-IN-THE-LOOP

Para la IA de los procesos con flujos integrados. Se diseñan checkpoints obligatorios en los puntos críticos del flujo. El humano interviene donde un error puede escalar. El objetivo es mantener la competencia situacional, no auditar cada acción.

Pregunta guía: ¿el supervisor entiende cómo sellegó al output, o solo ve qué dice el output?

NIVEL 2 - HUMAN-ON-THE-LOOP

Para la IA agéntica plena. Se supervisan la trayectoria y los criterios operativos del agente, no cada micro-acción. Se conserva la capacidad de auditar criterios, corregir desviaciones y revocar mandato.

Pregunta guía: ¿la organización puede retirar la autonomía del agente en cualquier momento? Si no puede, no tiene gobernanza, tiene dependencia.

MÁS ALLÁ DEL AGENTE INDIVIDUAL: PROCESOS COLECTIVOS Y DAÑOS MASIVOS

Estos tres modelos no son escalones obligatorios. Son alternativas que se eligen en función del nivel de autonomía del sistema, la criticidad del proceso y la competencia disponible para supervisar. Una elección equivocada —*on-the-loop* en un proceso crítico que requeriría *in-the-loop*, o *in-the-loop* desplegado sobre un volumen que vuelve materialmente imposible la supervisión— produce gobernanza en el papel pero no en los hechos.

Hasta aquí hemos pensado la gobernanza desde el agente individual o desde sistemas multi-agente acotados desplegados por una organización. Hay una dimensión adicional que excede a este material pero que es importante anticipar como puente con la línea de investigación más amplia sobre IA agéntica y derecho[6].

Cuando miles de agentes operan simultáneamente sobre un mismo dominio (agentes de pricing en un mercado, agentes de moderación en redes sociales, agentes de atención al cliente en un sector regulado, agentes negociadores en contratación pública), pueden producirse efectos sistémicos que ningún despliegue individual anticipó: convergencia colusoria tácita, censura algorítmica masiva, asimetrías informacionales irrecuperables, daños distribuidos sobre poblaciones enteras de personas que no pueden identificar individualmente a quién imputar el daño.

[6] En línea con la temática, se recomienda la lectura de Atlas de Inteligencia Artificial para el desarrollo humano de América Latina y el Caribe (Buenos Aires: Siglo XXI Editores, 2025), de Gustavo Beliz.

Estos escenarios desbordan los marcos clásicos de responsabilidad civil y los procesos colectivos disponibles en la mayoría de los ordenamientos. La hipótesis sobre la que trabajamos en investigación doctoral es que los daños masivos derivados de IA agéntica exceden estructuralmente los frameworks vigentes en el derecho argentino y requieren una reconfiguración del control judicial y administrativo. Esa discusión, que excede el alcance de este material, es la continuación natural de la gobernanza individual y organizacional que aquí desarrollamos.

EFECTOS SISTÉMICOS DE LA IA AGÉNTICA

Desbordamiento de Responsabilidad y Necesidad de Reconfiguración

Cuando miles de agentes operan simultáneamente sobre un mismo dominio, pueden producirse efectos sistémicos que ningún despliegue individual anticipó:

1. **Convergencia Colusoria Tácita**
   Agentes de pricing en un mercado convergen en precios similares sin comunicación explícita, produciendo efectos equivalentes a colusión.

2. **Censura Algorítmica Masiva**
   Agentes de moderación en redes sociales aplican criterios que, agregados, restringen sistemáticamente ciertos discursos sin decisión humana deliberada.

3. **Asimetrías Informacionales Irrecuperables**
   Agentes negociadores o de atención al cliente generan ventajas informacionales estructurales que los usuarios individuales no pueden compensar.

4. **Daños Distribuidos en Contratación Pública**
   Agentes en contratación pública producen daños sobre poblaciones enteras que no pueden identificar individualmente a quién imputar el perjuicio.

Estos escenarios desbordan los marcos clásicos de responsabilidad civil y requieren una reconfiguración del control judicial y administrativo.

08. CIERRE

El recorrido de este material partió de una premisa que conviene recuperar al cierre: la IA agéntica es el ecosistema en el que ya operamos. Lo hacemos cada vez que usamos un asistente basado en IA generativa como Claude, ChatGPT, Gemini, Grok, entre otros.

A lo largo de estas páginas vimos que ese ecosistema no se deja reducir a una sola lógica. Conviven en él soluciones que van desde el agente empaquetado que se activa con un clic hasta el framework de código que exige decisiones sobre cada componente; usuarios que aceleran tareas individuales y organizaciones que reconfiguran procesos completos; capas técnicas de infraestructura, abstracción e interoperabilidad que condicionan cualquier despliegue. Y sobre todo eso opera una gobernanza que debe responder a dos preguntas de naturaleza distinta: la de la legitimidad de la actuación y la de la sustentabilidad de la competencia humana.

Cada una de estas dimensiones plantea decisiones que no son técnicas sino, en última instancia, de diseño institucional:

¿QUÉ FUENTES DE INFORMACIÓN SE LE HABILITAN?

¿QUÉ ACCIONES PUEDE EJECUTAR SIN VALIDACIÓN HUMANA?

¿DÓNDE SE PROCESAN LOS DATOS?

¿QUÉ SE REGISTRA Y QUÉ SE PUEDE AUDITAR?

¿CÓMO SE DISEÑA LA SUPERVISIÓN PARA QUE SEA MATERIALMENTE POSIBLE?

Todas estas son preguntas que las organizaciones deberían poder formular y responder antes de que el sistema entre en operación.

Adicionalmente, tres ideas transversales merecen subrayarse como cierre:

01	**La frontera entre la IA de las tareas y la IA de los procesos se está difuminando** Herramientas que hace un año requerían equipos técnicos y decisiones arquitectónicas de la organización hoy están al alcance de un usuario individual. Eso amplifica las oportunidades, pero también traslada al usuario decisiones de gobernanza para las cuales no siempre está preparado. La democratización del acceso a la IA agéntica exige que la alfabetización sobre sus riesgos avance al mismo ritmo.
02	**La delegación operativa no equivale a delegación de responsabilidad** Un agente puede ejecutar un flujo completo de trabajo de punta a punta, pero la responsabilidad por lo que ese flujo produce sigue recayendo en quien lo diseñó, en quien lo habilitó o en quien lo supervisó (o debió supervisarlo). Esto conecta con lo que planteamos al introducir el output como acto imputable y no como mero texto, y con la doctrina civil del mandato: el agente actúa por cuenta y orden de un mandante humano dentro de los límites que ese mandante le fijó en el diseño. Cuando se excede de esos límites, las preguntas sobre exceso del mandato y rendición de cuentas recuperan plena vigencia.
03	**Medir la productividad agéntica solo por el tiempo de ejecución del agente es una distorsión que hay que corregir activamente** El ciclo completo (instrucción, ejecución, auditoría, corrección, aprendizaje institucional) es la unidad relevante de evaluación. Una organización que no mide las cinco fases corre el riesgo de creer que está ganando productividad cuando en realidad está redistribuyendo carga cognitiva de manera globalmente desfavorable. A mediano plazo, eso erosiona la competencia de su personal y la sustentabilidad de su propia capacidad de supervisión.

Este trabajo se propuso contribuir a la comprensión del ecosistema agéntico actual para que su adopción sea informada, deliberada y gobernable. Es un punto de partida.

La discusión continúa en el plano de los principios éticos evolucionados al contexto agéntico, los procesos colectivos y los daños masivos derivados de IA agéntica, donde los marcos tradicionales de responsabilidad civil y de control judicial empiezan a mostrar sus límites estructurales. Esa es la siguiente capa de la conversación, y la que va a definir buena parte de la agenda jurídica de los próximos años.


IALAB
AI academy
BY doinGlobal
LA LEY
Thomson
Reuters™