Facsímil 09

Seguridad, privacidad y gobernanza

Capítulo 01

Riesgos, controles y evidencias: la primera capa de gobernanza

686f6c61 · 2026

Capítulo 01: Riesgos, controles y evidencias: la primera capa de gobernanza

Entrando en el tema

Hay un momento, en casi todos los proyectos de IA, en que la pregunta deja de ser «¿funciona?» y pasa a ser «¿quién responde por esto?». El sistema ya contesta, la demo gustó, pero alguien con responsabilidad mira la pantalla y pregunta lo incómodo: qué datos toca, qué pasa si se equivoca, quién decidió que podía publicarse y qué prueba hay de todo ello. En ese instante el equipo descubre que construir y gobernar no son lo mismo.

Este capítulo va de ese salto. No de rellenar formularios ni de pedir permiso a un comité, sino de convertir las decisiones importantes de la ingeniería en algo visible, medible y revisable: un inventario de lo que existe, unos riesgos escritos como escenarios concretos, unos controles que de verdad cambian el sistema y unas evidencias que permiten reconstruir por qué se decidió cada cosa. Gobernar IA, bien hecho, no frena la ingeniería: la hace defendible.

Qué deberías poder hacer al terminar

Este facsímil empieza justo donde muchos proyectos de IA se vuelven adultos. Ya sabemos construir prototipos, usar APIs, montar RAG, trabajar con agentes, observar sistemas, evaluar calidad y revisar datos. Ahora aparece otra pregunta: **si mañana alguien nos pide justificar por qué esta capacidad de IA puede usarse, qué enseñamos?**

No vale contestar “porque funciona en mis pruebas”. Tampoco basta enseñar una arquitectura bonita. En seguridad, privacidad y gobernanza necesitamos poder enseñar inventario, límites, controles, owner, evidencia y decisión. Es decir: no solo qué hace el sistema, sino bajo qué condiciones aceptamos que lo haga.

Al terminar este capítulo deberías poder hacer esto:

RESULTADO DE APRENDIZAJE	EVIDENCIA DE QUE LO SABES HACER
Separar riesgo, control y evidencia.	No confundes “tenemos política” con “tenemos prueba de que la política se cumple”.
Modelar un sistema de IA como superficie de decisión.	Nombras datos, modelo, RAG, tools, memoria, logs, usuario, salida y owner.
Escribir un registro de riesgos mínimo.	Cada riesgo tiene probabilidad, impacto, exposición, detección, owner, control y evidencia.
Calcular severidad de forma defendible.	Usas una escala explícita y puedes explicar por qué un escenario sube o baja.
Diseñar un gate de salida.	La versión no avanza si faltan controles, trazas, revisión o aceptación de riesgo residual.
Conectar marcos reales con ingeniería.	Lees NIST AI RMF, ISO 42001, AI Act, GDPR y OWASP como fuentes de requisitos operativos, no como decoración.
Ejecutar una práctica real.	Construyes el registro de riesgos, la matriz de controles, el gate de salida y el paquete de evidencias en el cuaderno del facsímil.

La idea central es esta:

> Gobernar IA no es frenar la ingeniería. Es hacer que las decisiones importantes de la ingeniería sean visibles, medibles y revisables.

La escena: el sistema ya responde, pero nadie sabe quién lo puede publicar

Imagina un asistente académico que responde preguntas de matrícula, becas y normativa. El sistema usa RAG, cita documentos, registra trazas y deriva casos difíciles a una persona. En una demo funciona bien. El equipo está contento. Alguien pregunta: “¿lo ponemos en producción?”

Y entonces empiezan las preguntas incómodas:

PREGUNTA	QUÉ REVELA
¿Qué datos personales aparecen en logs y durante cuánto tiempo?	Privacidad y retención.
¿Qué ocurre si el índice recupera una norma antigua?	Calidad del RAG y control de versiones.
¿Puede una tool cambiar el estado de un expediente?	Permisos y revisión humana.
¿Qué versión de prompt y modelo produjo una respuesta concreta?	Trazabilidad y reproducibilidad.
¿Quién acepta que un riesgo residual siga abierto?	Gobernanza y responsabilidad operativa.
¿Qué evidencia revisaría una auditoría interna?	Paquete documental, no solo opiniones.

Si estas preguntas aparecen por primera vez el día de publicar, llegamos tarde. El objetivo de este capítulo es construir una forma de pensar que aparezca antes: desde la arquitectura, desde los datos, desde las evals y desde la operación.

Fecha de corte y fuentes consultadas

Fecha de corte: 7 de junio de 2026.
Fuentes consultadas ese día: NIST AI RMF 1.0, NIST AI RMF Generative AI Profile, Reglamento (UE) 2024/1689, GDPR, NIST Privacy Framework 1.0, ISO/IEC 42001:2023, OWASP Top 10 for LLM Applications 2025 y trabajos académicos sobre documentación y auditoría de sistemas de IA.

El NIST AI RMF 1.0 se presenta como un marco voluntario, no sectorial y práctico para gestionar riesgos de IA en organizaciones que diseñan, desarrollan, despliegan o usan sistemas de IA.[1] Su perfil para IA generativa aterriza el marco en sistemas capaces de generar texto, código, imágenes u otros contenidos, y lo conecta con el ciclo de vida de esos sistemas.[2]

El AI Act europeo, publicado como Reglamento (UE) 2024/1689, establece un marco jurídico basado en categorías de riesgo y obligaciones para ciertos sistemas de IA.[3] ISO/IEC 42001:2023 define requisitos para establecer, implementar, mantener y mejorar un sistema de gestión de IA dentro de una organización.[4] GDPR sigue siendo imprescindible cuando tratamos datos personales; su artículo 35 exige evaluación de impacto relativa a protección de datos cuando un tratamiento puede implicar alto riesgo para derechos y libertades de personas.[5]

Lo estable no es el nombre de una checklist concreta. Lo estable es la disciplina: inventariar, mapear contexto, medir, controlar, documentar, revisar y mejorar.

Qué no es gobernanza de IA

Gobernanza de IA no es tener un PDF en una carpeta compartida. Un documento puede ayudar, pero no gobierna nada si no está conectado con código, datos, despliegues, owners, métricas y decisiones.

Tampoco es una revisión legal al final. La parte legal importa, claro, pero la gobernanza no puede llegar cuando el sistema ya está diseñado. Si una tool puede escribir en un CRM, si una traza guarda datos sensibles, si un RAG cita documentos obsoletos o si un modelo local se sirve sin control de versión, eso no se arregla solo con una frase en una política.

Y no es una forma elegante de decir “no”. Una buena gobernanza debería permitir publicar mejor: con límites claros, revisiones proporcionadas, gates automatizados, decisiones escritas y una forma honesta de aceptar o reducir riesgo residual.

CONFUSIÓN	LECTURA DE INGENIERÍA
“Tenemos una política de IA”.	¿Qué controles la ejecutan y qué evidencia la demuestra?
“El proveedor ya se ocupa”.	El proveedor no conoce tu finalidad, tus datos, tus usuarios ni tus consecuencias.
“La eval salió alta”.	¿Qué riesgos mide esa eval y cuáles deja fuera?
“No guardamos datos personales”.	¿Lo verifican logs, trazas, memoria, proveedores, backups y exports?
“La tool tiene permisos”.	¿Permisos mínimos, aprobación, idempotencia y registro de efectos?
“Es solo orientación”.	¿Puede el usuario tomar una decisión real a partir de esa orientación?

La frase que conviene tatuar en la pizarra es menos solemne:

> Si no hay evidencia, no hay control: hay intención.

Qué sí es gobernanza de IA para ingeniería

En este facsímil llamaremos gobernanza de IA al sistema de roles, políticas, controles, mediciones y evidencias que permite decidir cómo se diseña, publica, usa y revisa una capacidad de IA.

No es una ecuación: es una lista de piezas que deben existir a la vez. Conviene tenerlas todas a la vista, porque si falta una, la gobernanza cojea justo por ahí.

PIEZA	QUÉ APORTA	EJEMPLO
Inventario	Qué estamos gobernando.	Modelo, prompt, índice, tools, memoria, datos, logs, owners.
Contexto	Para qué sirve el sistema.	Finalidad, usuarios, jurisdicción, efecto de la decisión, límites.
Riesgos	Qué puede salir mal.	Respuesta sin fuente, dato personal en traza, tool con efecto administrativo.
Controles	Qué reduce el riesgo.	Validación de salida, revisión humana, retención limitada, permisos mínimos.
Evidencias	Con qué se demuestra.	Trazas, evals, fichas, actas, decisiones, políticas versionadas.
Decisión	Qué se acuerda hacer.	Publicar, publicar con condiciones, revisar antes de publicar.
Monitorización y mejora	Cómo se mantiene en el tiempo.	Revisión periódica, métricas, incidencias, drift, actualización de controles.

En palabras: la lista no mide nada; sirve para comprobar qué falta. Si falta el inventario, no sabemos qué estamos gobernando. Si falta el contexto, no sabemos para qué sirve el sistema. Si faltan los controles, los riesgos no bajan. Si faltan las evidencias, no podemos demostrar nada. Si falta la decisión, el sistema avanza por inercia.

NIST organiza el AI RMF alrededor de funciones como gobernar, mapear, medir y gestionar. Para un equipo de ingeniería, esa estructura se traduce en cuatro preguntas prácticas:

FUNCIÓN	PREGUNTA DE INGENIERÍA	ARTEFACTO MÍNIMO
Gobernar	¿Quién decide, con qué política y con qué revisión?	Owner, política, gate y acta de decisión.
Mapear	¿Qué sistema, contexto, usuarios, datos e integraciones existen?	Inventario y diagrama de límites.
Medir	¿Qué riesgos, métricas y pruebas se observan?	Registro de riesgos, evals y trazas.
Gestionar	¿Qué controles se aplican y qué queda pendiente?	Matriz de controles y plan de reducción.

ISO 42001 empuja una idea parecida desde el lenguaje de sistemas de gestión: no basta revisar una aplicación aislada; hace falta un sistema que establezca políticas, procesos, objetivos y mejora continua para la IA en la organización.[6] Esta diferencia importa: una aplicación puede pasar una revisión puntual; un sistema de gestión pregunta si el equipo puede repetir el proceso con la siguiente aplicación.

De los marcos a los artefactos

Los marcos no son el trabajo. Son una forma de no olvidar preguntas importantes. El trabajo real aparece cuando un equipo convierte esas preguntas en artefactos que se pueden abrir, ejecutar, revisar y versionar.

Esta conversión es donde muchos proyectos se quedan cortos. Citan NIST, ISO, AI Act, GDPR u OWASP, pero no explican qué archivo, prueba, traza o decisión demuestra que el sistema cambió. En ingeniería, esa distancia es peligrosa: parece que hay control, pero solo hay vocabulario.

Una forma práctica de leer los marcos es esta:

FUENTE	QUÉ PREGUNTA TRAE AL PROYECTO	ARTEFACTO QUE DEBERÍA EXISTIR
NIST AI RMF · Govern	¿Quién decide, con qué política y cómo se revisa?	`raci.md`, `control_policy.json`, `release_gate.md`.
NIST AI RMF · Map	¿Qué sistema, uso, datos e integraciones estamos evaluando?	`ai_system_context.json`, diagrama de límites, inventario de componentes.
NIST AI RMF · Measure	¿Cómo medimos riesgos, calidad, trazabilidad y huecos?	`risk_register.md`, evals, trazas de muestra, métricas por escenario.
NIST AI RMF · Manage	¿Qué hacemos con lo medido?	`control_matrix.csv`, plan de reducción, decisión de salida.
NIST GenAI Profile	¿Qué añade la IA generativa al ciclo de vida?	Casos de salida no determinista, evaluación de contexto, control de memoria y herramientas.
ISO/IEC 42001	¿Existe un sistema repetible de gestión de IA?	Política, roles, revisión periódica, mejora continua y registro de decisiones.
AI Act	¿Qué tipo de uso es, qué obligaciones puede activar y qué documentación técnica existe?	Clasificación de uso, documentación técnica, monitorización posterior a release.
GDPR	¿Qué datos personales se tratan y con qué necesidad?	`privacy_review.md`, política de retención, minimización, DPIA si procede.
OWASP LLM 2025	¿Qué riesgos propios de aplicaciones LLM estamos cubriendo?	Validadores de salida, límites de tools, revisión de contexto, controles de dependencias.

Fíjate en la columna de la derecha. Ninguna fila acaba en “hacer una reunión”. Las reuniones pueden ayudar a decidir, pero el sistema necesita piezas persistentes. Si no queda una evidencia, el siguiente equipo no puede reproducir la decisión.

La forma mínima de trabajar sería:

```
marco -> pregunta -> control -> evidencia -> decisión
```

Por ejemplo:

MARCO	PREGUNTA	CONTROL	EVIDENCIA	DECISIÓN
GDPR	¿Guardamos texto de usuario en trazas?	Redacción de logs y retención limitada.	`trace_sample.jsonl` y `privacy_review.md`.	No subir de fase si aparece texto bruto innecesario.
OWASP LLM	¿Una tool puede producir un efecto externo sin confirmación?	Modo lectura, aprobación humana e idempotencia.	`tool_trace_sample` y `approval_policy`.	Solo canary con tool sin ejecución automática.
NIST AI RMF	¿Podemos reconstruir una respuesta?	Manifest de release y `trace_id`.	`release_manifest.json` y muestra de traza.	Publicar con condiciones si falta algún atributo.

Esta tabla no sustituye asesoría legal ni revisión especializada. Su valor está en otra parte: obliga a ingeniería a producir evidencia antes de pedir confianza.

El mecanismo paso a paso

La primera capa de gobernanza funciona como una tubería de decisión: una secuencia de pasos en la que cada uno produce algo que el siguiente necesita. El orden importa, porque cumplir un marco sin saber qué sistema tienes delante es decorar una caja que todavía no has abierto. Vamos a recorrer esa tubería paso a paso, de inventario a roles, y cada paso es corto, pero ninguno se puede saltar sin dejar al siguiente sin material con el que trabajar. No empieza por “qué marco cumplo”, sino por “qué estamos construyendo”.

1. Inventario: qué existe realmente

Un inventario de IA no es una lista de modelos, y ese malentendido sale caro. Cuando alguien dice «usamos tal modelo», está nombrando una sola pieza de un sistema que en realidad tiene muchas más: el prompt que lo guía, el índice que le da contexto, las herramientas que le dejan actuar, la memoria que recuerda, los logs que registran y los datos que lo alimentan. Cada una de esas piezas tiene su propia versión, su propio dueño y su propio modo de fallar. Un inventario honesto las nombra todas, porque lo que no está inventariado no se puede gobernar: ni versionar, ni medir, ni revertir cuando algo se tuerce. Un sistema moderno suele incluir, como mínimo, estas piezas:

PIEZA	PREGUNTA
Modelo	¿Proveedor, versión, modo de uso, límites, coste, región?
Prompt o instrucciones	¿Versión, owner, cambios, criterios de aceptación?
RAG	¿Corpus, índice, embeddings, filtros, versionado, calidad de fuentes?
Tools	¿Qué leen, qué escriben, con qué permisos y con qué aprobación?
Datos	¿Origen, licencia, sensibilidad, finalidad, retención, linaje?
Memoria	¿Qué se guarda, quién lo puede leer y cuándo se borra?
Logs y trazas	¿Incluyen datos personales, IDs, salida del modelo, contexto recuperado?
Evaluaciones	¿Qué riesgos miden y qué riesgos no cubren?
Interfaz	¿Qué entiende el usuario, qué límites ve y qué decisión puede tomar?

Model Cards y Datasheets for Datasets nacieron precisamente para documentar capacidades y datos de forma estructurada, incluyendo usos previstos, límites, composición y métricas relevantes.[78] El primer aprendizaje es directo: no puedes gobernar lo que no está inventariado.

1.1. Superficies técnicas: dónde mirar

Una superficie técnica es una zona del sistema donde puede aparecer un riesgo. La palabra “superficie” ayuda porque nos obliga a mirar el sistema por partes, no como una caja misteriosa.

En una aplicación de IA generativa, yo revisaría al menos estas superficies:

SUPERFICIE	QUÉ PUEDE FALLAR	CONTROL TÉCNICO ÚTIL	EVIDENCIA ESPERADA
Modelo	Calidad desigual, límites no documentados, cambio de proveedor o versión.	Model card, eval de regresión, fallback, pin de versión.	`model_card.md`, `eval_report.json`, `release_manifest.json`.
Instrucciones	Cambio de comportamiento por prompt no versionado o reglas contradictorias.	Versionado, tests de contrato, diff de prompt.	`prompt_version`, `prompt_eval.md`.
RAG	Fuente obsoleta, chunk pobre, filtro incorrecto, cita ausente.	Manifest de índice, filtros de confianza, eval retrieval, abstención.	`index_manifest.json`, `rag_eval_report.json`.
Tools	Lectura o escritura fuera del alcance previsto.	Scopes, allowlist, modo lectura, aprobación, idempotencia.	`tool_policy.json`, `tool_trace_sample.jsonl`.
Memoria	Guardar más contexto del necesario o recuperar algo fuera de finalidad.	Política de memoria, TTL, separación por usuario, revisión de borrado.	`memory_policy.md`, muestra de recuperación.
Logs y trazas	Conservar datos innecesarios o no poder reconstruir una run.	Redacción de logs, atributos mínimos, retención, sampling revisable.	`trace_sample.jsonl`, política de retención.
Datos	Licencia, linaje, sensibilidad, leakage o mezcla de usos.	Dataset card, contrato de datos, splits, checks de sensibilidad.	`dataset_card.md`, `split_manifest.json`.
Interfaz	El usuario cree que el sistema decide más de lo que realmente debe.	Microcopy de límites, estado de evidencia, ruta de revisión.	Captura, checklist UX, casos de usuario.
Proveedor	Región, cambios de modelo, límites de servicio, contratos de tratamiento.	Registro de proveedor, configuración por entorno, evaluación periódica.	`provider_review.md`, manifest de configuración.
Despliegue	Secretos, permisos por entorno, rollback o configuración divergente.	Secret manager, CI gates, IaC, feature flags, rollback probado.	`release_gate.md`, `rollback_runbook.md`.

Esta tabla tiene una consecuencia práctica: cada fila debería poder convertirse en un test, una revisión o una pieza del paquete de evidencias. Si una fila solo produce una conversación, todavía no está madura.

2. Contexto: para qué se usa y qué consecuencias tiene

El mismo modelo puede tener riesgos completamente distintos según para qué se use, y por eso el contexto no es un trámite administrativo: es lo que decide cuánto cuidado merece el sistema. Un asistente que resume noticias internas y otro que recomienda becas, prioriza tickets médicos, redacta cláusulas contractuales o ejecuta acciones administrativas pueden compartir el mismo modelo por debajo y aun así pertenecer a mundos de riesgo diferentes. Lo que cambia no es la tecnología, sino la consecuencia de equivocarse: una respuesta floja en un resumen interno se corrige releyendo, pero una recomendación de beca mal hecha afecta a una persona concreta que esperaba algo de ella. Por eso, antes de hablar de controles, describimos el contexto de uso con campos explícitos:

CAMPO	EJEMPLO
Finalidad	Responder dudas de normativa académica.
Usuarios	Alumnado, secretaría, docentes.
Tipo de decisión	Orientativa, automática, recomendación, cambio de estado, derivación.
Consecuencia posible	Confusión, pérdida de tiempo, trámite mal iniciado, exposición de datos.
Jurisdicción	UE, España, normativa interna.
Datos	Documentos públicos, normativa interna, tickets seudonimizados, trazas.
Dependencias	Proveedor de modelo, vector store, sistema de tickets, observabilidad.

El AI Act usa una lógica basada en riesgo, con obligaciones distintas según el tipo de sistema y su contexto de uso.[9] Esto no significa que cada proyecto sea “alto riesgo” por defecto. Significa que no debemos decidir por la etiqueta técnica del modelo, sino por la función real del sistema.

3. Riesgos: convertir preocupación en escenario verificable

Un riesgo útil no se escribe así:

> “Puede haber problemas de privacidad”.

Eso es demasiado vago. Un riesgo útil se escribe así:

> “Una traza conserva texto con datos personales más tiempo del necesario para depuración”.

Ahora sí podemos trabajar. Sabemos dónde mirar, qué control aplicar y qué evidencia pedir.

Un riesgo individual, en este registro, no es una frase suelta: es una ficha con campos fijos. Anotarlos todos evita el error más común, escribir una preocupación vaga y olvidar quién la mira o con qué se demuestra. Estos son los campos que pediremos para cada riesgo:

CAMPO	SIGNIFICADO	EJEMPLO
Escenario	Evento concreto que puede ocurrir.	Retención de texto sensible en logs.
Probabilidad	Posibilidad de que ocurra, escala 1-5.	3 sobre 5.
Impacto	Gravedad si ocurre, escala 1-5.	5 sobre 5.
Exposición	Alcance, frecuencia o piezas implicadas, escala 1-5.	3 sobre 5.
Hueco de detección	Lo difícil que es verlo a tiempo, escala 1-5.	4 sobre 5.
Owner	Quién responde por él.	Equipo de privacidad.
Control	Qué medida lo reduce.	Seudonimización, retención limitada, muestreo revisable.
Evidencia	Qué prueba que el control existe.	Política de retención y muestra de traza.

Esta ficha no es una invención del facsímil: formaliza la definición cuantitativa de riesgo de Kaplan y Garrick, que describe el riesgo como un conjunto de tripletas que responden a tres preguntas, qué puede pasar, con qué probabilidad y con qué consecuencia.[10]

$$R = \{ \langle s_i, p_i, x_i \rangle \}, \quad i = 1, \ldots, n$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
R	Riesgo: el conjunto de todos los escenarios considerados.	El registro de riesgos completo del asistente.
s_i	Escenario i: qué puede pasar.	Una traza conserva datos personales de más.
p_i	Probabilidad del escenario.	0,2 en la ventana revisada.
x_i	Consecuencia o impacto del escenario.	Exposición de datos personales de tickets reales.

En palabras: un riesgo bien escrito no es una palabra como «privacidad»; es una lista de escenarios, cada uno con su probabilidad y su consecuencia. Por eso el registro tiene filas, no adjetivos.

Cuando la consecuencia se puede expresar como una pérdida (coste, casos afectados, tiempo), el peso de un escenario es su pérdida esperada, el valor esperado clásico de la teoría de la decisión:

$$\mathbb{E}[L_i] = p_i \cdot x_i$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$\mathbb{E}[L_i]$	Pérdida esperada del escenario i.	$0{,}2 \times 1000 = 200$.
p_i	Probabilidad del escenario.	0,2.
x_i	Pérdida si ocurre, en la unidad que elijas.	1000 casos potencialmente afectados.

En palabras: la pérdida esperada explica por qué un evento raro pero catastrófico puede pesar tanto como uno frecuente y leve. Es la base de cualquier priorización honesta y la razón de que no baste con mirar solo la probabilidad o solo el impacto.

4. Severidad: medir para ordenar, no para fingir precisión

Para ordenar riesgos sin fingir precisión, la ingeniería de fiabilidad lleva décadas usando el número de prioridad de riesgo (*Risk Priority Number*, RPN) del análisis modal de fallos y efectos (FMEA): se multiplica la gravedad por la frecuencia de ocurrencia y por la dificultad de detección, de modo que un fallo grave, frecuente y difícil de detectar quede por encima de uno leve, raro y evidente.[11]

$$\mathrm{RPN} = S \times O \times D$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
RPN	Número de prioridad de riesgo.	80.
S	Gravedad (*severity*) del efecto.	5.
O	Ocurrencia (*occurrence*): frecuencia esperada.	4.
D	Detección: lo difícil que es detectarlo a tiempo.	4.

En palabras: el RPN ordena por prioridad, no por verdad absoluta. Un número alto dice «mira esto antes», no «esto va a pasar seguro». Por eso FMEA insiste en que sirve para comparar y priorizar dentro de un mismo análisis, no como medida objetiva entre sistemas distintos.

En este capítulo adaptamos el RPN a sistemas de IA con una variante que separa frecuencia de alcance: añadimos un factor de exposición E (cuántas piezas, usuarios o llamadas quedan implicadas) y usamos escala 1-5 por factor para discutir con números manejables. Es una estimación dimensional para priorizar, no un teorema.

$$S_i = P_i \times I_i \times E_i \times D_i$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
S_i	Severidad priorizada del escenario i (variante de RPN).	240.
P_i	Probabilidad de que ocurra en la ventana revisada (1-5).	3.
I_i	Impacto si ocurre (1-5).	5.
E_i	Exposición: alcance, frecuencia o número de piezas implicadas (1-5).	4.
D_i	Hueco de detección: 1 fácil de detectar, 5 difícil de detectar (1-5).	4.

En palabras: es la misma idea del RPN, multiplicar factores que empeoran un riesgo, con un factor más pensado para ordenar la conversación de un equipo de ingeniería. Sustituyendo los números del ejemplo:

$$S_i = 3 \times 5 \times 4 \times 4 = 240$$

La puntuación no es una verdad científica. Es una herramienta de orden. Sirve para que el equipo discuta con claridad: “¿de verdad esto es impacto 5?”, “¿por qué exposición 4?”, “¿qué control bajaría el hueco de detección de 4 a 2?”.

Una escala práctica:

BANDA	RANGO	LECTURA
Bajo	$S < 80$	Seguimiento normal si hay evidencia básica.
Medio	$80 \leq S < 180$	Revisión y control documentado.
Alto	$180 \leq S < 350$	Condición explícita antes de publicar o ampliar uso.
Crítico	$S \geq 350$	No avanzaría sin reducir riesgo o cambiar diseño.

Conviene un aviso académico honesto: multiplicar factores ordinales y leer bandas de color tiene límites conocidos. Cox demostró que las matrices de riesgo pueden asignar la misma puntuación a riesgos muy distintos, invertir el orden real de prioridad y dar una falsa sensación de resolución cuantitativa.[12] Por eso usamos la puntuación para ordenar la conversación y disparar revisión, no como verdad: cuando una decisión es cara o afecta a personas, se baja al detalle del escenario y no se confía en el número.

El punto importante es que la severidad inicial no cierra la conversación. La gestión de riesgos distingue entre el riesgo inherente (antes de tratarlo) y el riesgo residual, el que queda después de aplicar los controles; ISO 31000 define el riesgo residual como el riesgo que permanece tras el tratamiento.[13] Una forma sencilla de estimarlo, como cuenta de ingeniería y no como garantía estadística, es reducir la severidad inicial según la efectividad que atribuimos a los controles:

$$S_{residual} = S_{inicial} \times (1 - C)$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$S_{residual}$	Severidad que queda después de controles.	96.
$S_{inicial}$	Severidad antes de controles.	240.
C	Efectividad estimada de controles, entre 0 y 1.	0,60.

Si no sabemos estimar C, no pasa nada: empezamos registrando controles y evidencia. Pero el objetivo técnico es llegar a poder medir si un control reduce realmente incidencia, exposición, latencia de detección o impacto.

4.1. Ejemplo completo: una traza conserva datos personales

Supongamos este escenario:

> R-002: una traza conserva texto con datos personales más tiempo del necesario para depuración.

Antes de controles, el equipo puntúa:

FACTOR	VALOR	POR QUÉ
Probabilidad P	3	El sistema registra muchas runs y todavía no se ha revisado cada campo.
Impacto I	5	Puede afectar a personas y activar obligaciones de privacidad.
Exposición E	3	No todos los usuarios aportan datos personales, pero sí ocurre en tickets reales.
Hueco de detección D	4	Si nadie mira las trazas de muestra, el problema puede durar semanas.

La severidad inicial es:

$$S_{inicial} = 3 \times 5 \times 3 \times 4 = 180$$

Eso cae en banda alta. No significa “pánico”. Significa: no lo trataría como detalle menor antes de subir de fase.

Ahora aplicamos controles:

CONTROL	QUÉ CAMBIA	EVIDENCIA
Redacción de logs	El texto bruto se sustituye por campos mínimos o hashes.	`trace_sample.jsonl`.
Retención limitada	Las trazas se borran o compactan tras una ventana definida.	`retention_policy.md`.
Muestreo revisable	Una muestra de trazas se revisa antes de release.	`privacy_review.md`.
Campos obligatorios	La traza conserva `trace_id`, `model_id`, `prompt_version`, `index_version`, pero no texto innecesario.	`release_manifest.json`.

Si estimamos que esos controles reducen el riesgo en un 60%, entonces:

$$S_{residual} = 180 \times (1 - 0{,}60) = 72$$

VALOR	LECTURA
$S_{inicial} = 180$	Riesgo alto: requiere condición de salida.
$C = 0{,}60$	Los controles reducen bastante, pero no eliminan.
$S_{residual} = 72$	Riesgo bajo/medio según escala; puede avanzar si la evidencia existe.

La decisión profesional no sería “ya está arreglado”. Sería más precisa:

> Puede avanzar a canary si `trace_sample.jsonl` no conserva texto bruto, `retention_policy.md` define ventana de borrado, `privacy_review.md` queda firmado por el owner y el gate bloquea cualquier traza futura que incumpla el contrato.

Este es el salto que buscamos. No vendemos certeza; construimos condiciones verificables.

5. Controles: prevenir, detectar, limitar y corregir

Un control no es una promesa. Es una pieza que cambia el sistema.

TIPO DE CONTROL	QUÉ HACE	EJEMPLO EN IA
Preventivo	Evita que un caso entre por una ruta peligrosa.	Filtro de fuente, permisos mínimos, schema estricto.
Detectivo	Permite ver un problema a tiempo.	Trazas con `trace_id`, eval de regresión, alertas por contrato roto.
Limitante	Reduce alcance si algo falla.	Rate limit, modo lectura, revisión humana, canary.
Correctivo	Permite volver o reparar.	Rollback de prompt, reindexado, borrado de trazas, revisión de dataset.
Documental	Permite reconstruir y justificar.	Model card, dataset card, acta de aceptación, release gate.

OWASP Top 10 for LLM Applications 2025 es útil para traducir riesgos propios de aplicaciones con LLM en categorías de revisión técnica, como instrucciones no confiables, manejo inseguro de salida, control excesivo de tools, exposición de datos sensibles o dependencias de cadena de suministro.[14] La lectura de ingeniería no es memorizar el top 10. Es preguntar qué control, evidencia y owner cubre cada categoría en tu sistema.

En un proyecto con IA generativa, los controles deberían sonar a ingeniería concreta:

CONTROL TÉCNICO	QUÉ PROTEGE	CÓMO SE IMPLEMENTA	SEÑAL DE QUE EXISTE
Scope por tool	Evita que una herramienta haga más de lo previsto.	Cada tool declara permisos de lectura/escritura y entorno permitido.	`tool_policy.json` y traza de llamada.
Allowlist de acciones	Reduce rutas no previstas.	Solo se ejecutan acciones registradas por nombre y versión.	Fallo explícito si la acción no está en catálogo.
Modo lectura inicial	Limita el efecto de una integración nueva.	La tool simula o consulta, pero no escribe hasta pasar gate.	Campo `executed=false` en trazas de canary.
Aprobación para efectos externos	Añade revisión antes de modificar estado.	La run entra en `WAITING_APPROVAL` antes de escribir.	`approval_id` y owner en la traza.
Idempotencia	Evita duplicar efectos si una run se reintenta.	`idempotency_key` por operación persistente.	Dos reintentos producen un único efecto.
Validación de salida	Evita que el sistema entregue formatos incompatibles.	JSON Schema, enums, campos obligatorios y rechazo explícito.	`output_contract_ok=true`.
Redacción de logs	Evita conservar texto innecesario.	Reglas que sustituyen contenido sensible por hash, categoría o contador.	`trace_sample.jsonl` sin texto bruto.
Manifest de release	Permite reconstruir una respuesta.	Modelo, prompt, índice, policy, dataset y código quedan fijados.	`release_manifest.json`.
Hash de corpus o índice	Evita dudas sobre qué fuente usó RAG.	Hash de documentos, chunks e índice vectorial.	`index_manifest.json`.
Gate de CI	Detiene cambios sin evidencia mínima.	Script que falla si faltan manifest, eval o política.	Check rojo en PR o release.
Retención configurable	Reduce exposición temporal.	TTL por tipo de dato y entorno.	Política y job de borrado verificable.
Separación por entorno	Evita mezclar pruebas y producción.	Secrets, permisos, datos y proveedores distintos por entorno.	Configuración versionada.

Si un control no puede probarse, escríbelo como hueco. Por ejemplo: “tenemos redacción de logs en diseño, pero todavía no hay muestra revisada”. Esa frase es incómoda, pero útil. La gobernanza madura no consiste en fingir que todo está cerrado; consiste en saber exactamente qué sigue abierto.

5.1. Cuánto reducen los controles juntos: capas de protección

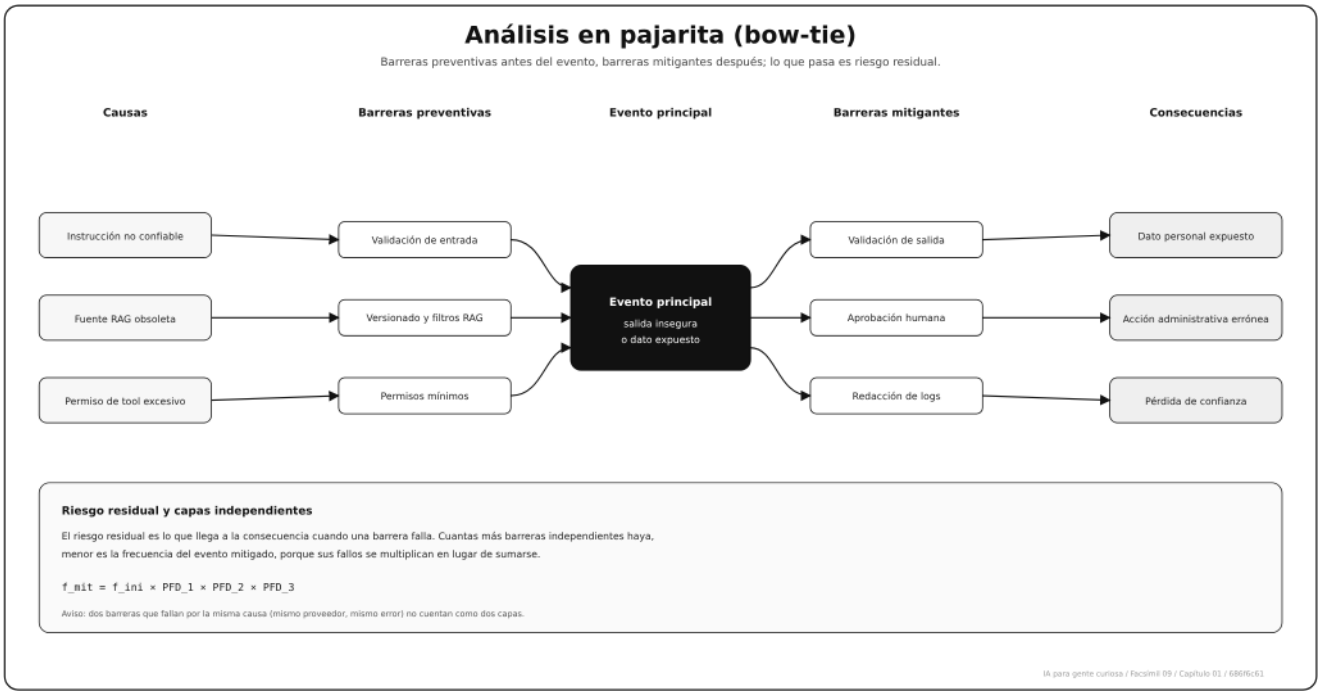
¿Cuánto reducen los controles en conjunto? La ingeniería de seguridad de procesos responde con el análisis de capas de protección (*Layer of Protection Analysis*, LOPA): si varias barreras independientes deben fallar a la vez para que el riesgo se materialice, la frecuencia del evento mitigado es la frecuencia inicial multiplicada por la probabilidad de fallo de cada capa.[15]

$$f_{mit} = f_{ini} \times \prod_{j=1}^{m} \mathrm{PFD}_j$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
f_{mit}	Frecuencia del evento ya mitigado.	0,001 al año.
f_{ini}	Frecuencia del evento iniciador, sin controles.	1 al año.
PFD_j	Probabilidad de fallo de la capa j, entre 0 y 1.	0,1 por capa.
m	Número de capas de protección independientes.	3.

En palabras: tres barreras independientes que fallan una de cada diez veces no suman, multiplican: reducen la frecuencia mil veces ($0{,}1 \times 0{,}1 \times 0{,}1$). La palabra clave es independientes: si dos controles fallan por la misma causa (el mismo proveedor caído, el mismo error de configuración), no cuentan como dos capas. Es una versión más rigurosa de la reducción de riesgo que la estimación $S_{residual} = S_{inicial}\,(1 - C)$: cuando puedes identificar barreras separadas, multiplicas sus fallos en lugar de estimar una efectividad global.

Esta idea se dibuja muy bien con un diagrama de pajarita (*bow-tie*): a la izquierda las causas, en el centro el evento que queremos evitar, a la derecha las consecuencias, y entre medias las barreras que lo previenen y lo mitigan.



6. Evidencia: lo que hace que una revisión sea posible

En gobernanza de IA, evidencia tiene un significado concreto y exigente: un artefacto revisable. No es una afirmación («lo revisamos») ni una intención («tenemos pensado controlarlo»), sino algo que otra persona puede abrir, leer y contrastar sin fiarse de tu palabra. La diferencia es la misma que hay entre decir «el código funciona» y enseñar un test que pasa. Y una buena evidencia tiene una propiedad valiosa: sobrevive a que te vayas del equipo. Si mañana no estás, alguien debería poder reconstruir qué decidiste y por qué solo mirando los archivos. Estos son los artefactos que más usaremos:

EVIDENCIA	QUÉ DEMUESTRA
`release_manifest.json`	Qué versión de modelo, prompt, índice, policy y código se publicó.
`risk_register.md`	Qué escenarios se revisaron y con qué severidad.
`control_matrix.csv`	Qué controles cubren cada riesgo y qué falta.
`rag_eval_report.json`	Si retrieval, citas y abstención funcionan en casos conocidos.
`trace_sample.jsonl`	Si una run se puede reconstruir sin datos innecesarios.
`privacy_review.md`	Qué datos personales se tratan y por qué.
`model_card.md`	Límites, métricas, usos previstos y condiciones del modelo.
`dataset_card.md`	Origen, composición, licencia, sensibilidad y límites del dataset.

Raji y colaboradores proponen pensar la auditoría interna de IA como un proceso de extremo a extremo, con documentación, mapeo de artefactos, pruebas y reflexión organizativa durante el ciclo de vida del sistema.[16] Esa idea encaja con lo que venimos haciendo en el facsímil: la evidencia no se inventa al final; se produce mientras construimos.

7. Roles: quién ejecuta, quién acepta y quién se entera

Un registro de riesgos sin roles se queda cojo. La palabra owner ayuda, pero a veces no basta. En ingeniería usamos matrices RACI para separar cuatro responsabilidades:

LETRA	SIGNIFICADO	PREGUNTA
R	Responsible	¿Quién hace el trabajo?
A	Accountable	¿Quién acepta la decisión final?
C	Consulted	¿Quién debe aportar criterio antes de decidir?
I	Informed	¿Quién debe enterarse del resultado?

En IA conviene no esconder esta parte, porque los sistemas cruzan muchas disciplinas. Un cambio de prompt puede ser técnico, pero también afecta a producto, soporte y privacidad. Una tool puede ser integración, pero también cambia responsabilidad operativa. Una política de memoria puede ser arquitectura, pero también tratamiento de datos.

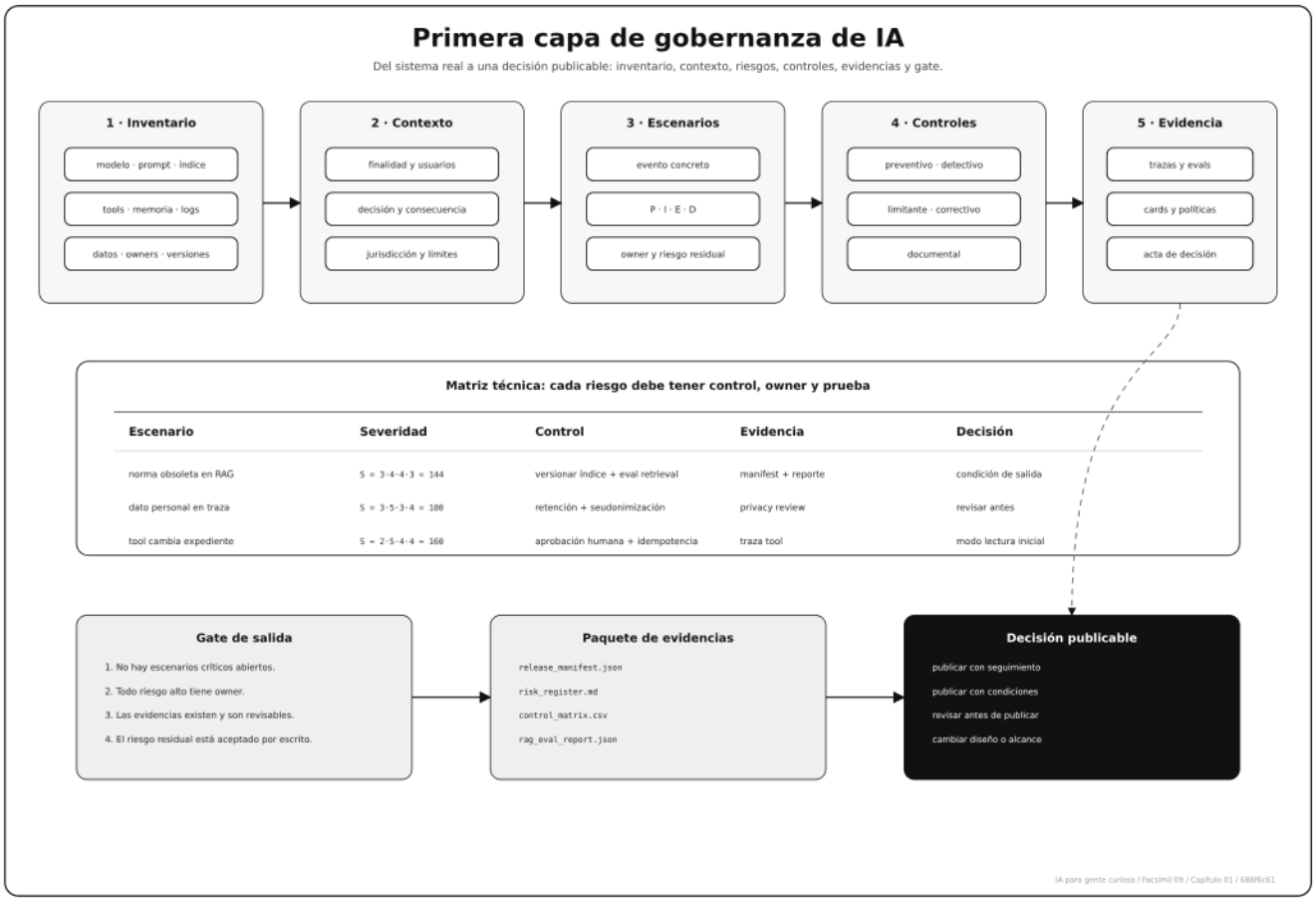
Un RACI mínimo para el asistente académico podría ser:

DECISIÓN	R	A	C	I
Cambiar modelo	Plataforma IA	Responsable técnico	Evaluación, producto, privacidad	Soporte
Cambiar índice RAG	Equipo RAG	Owner funcional	Datos, evaluación	Usuarios internos
Activar una tool de escritura	Plataforma	Owner funcional	Privacidad, operación	Soporte
Subir de canary a producción	Responsable técnico	Dirección del producto	Evaluación, privacidad, operación	Equipo completo
Aceptar riesgo residual alto	Owner funcional	Responsable designado	Legal, privacidad, seguridad	Dirección y soporte

La regla práctica es simple: cuanto más efecto tenga una decisión sobre personas, datos, coste o sistemas externos, más explícito debe ser quién la acepta. No para llenar papeles, sino para evitar que una decisión importante quede repartida entre conversaciones sueltas.

Anatomía visual: de sistema a decisión

Este diagrama resume la cadena completa. Léelo de izquierda a derecha: primero identificamos qué sistema existe, después escribimos escenarios concretos, luego asignamos controles y evidencias, y solo al final tomamos una decisión de salida.



En el día a día

En un proyecto real, el primer documento útil no es una declaración de principios. Es una ficha corta que permite a otra persona entender el sistema.

Un ejemplo mínimo:

CAMPO	VALOR
Sistema	Asistente académico con RAG.
Finalidad	Responder dudas de normativa con citas y abstención.
Fase	Canary interno.
Datos	Normativa interna, documentos públicos, tickets seudonimizados, trazas.
Tools	Crear ticket, consultar estado, proponer derivación.
Decisiones permitidas	Orientar y derivar, no resolver trámites automáticamente.
Owner técnico	Equipo de plataforma IA.
Owner funcional	Secretaría académica.
Evidencias mínimas	Eval RAG, trazas de muestra, política de retención, gate de release.

Después escribimos escenarios. No todos tendrán el mismo peso. Un fallo de estilo en una respuesta no se trata igual que una tool que cambia un expediente. Una cita ausente no se trata igual en una respuesta recreativa que en una recomendación administrativa.

Aquí aparece una idea clave para ingenieros: **el riesgo está en el sistema completo, no en el modelo aislado**. Un modelo con buenos benchmarks puede vivir dentro de un producto mal trazado. Un modelo modesto puede estar dentro de una arquitectura prudente, con límites, abstención, trazas, revisión y recuperación.

La ingeniería de software para ML lleva años señalando que estos sistemas acumulan deuda técnica en datos, dependencias, configuración, evaluación y operación.[17] En IA generativa añadimos prompts, contexto, tools, memoria, proveedores y salida no determinista. Gobernanza es la forma de que todo eso no viva en la cabeza de una sola persona.

Tres sistemas, tres lecturas distintas

El mismo marco no produce la misma decisión en todos los sistemas. Mira estos tres casos:

SISTEMA	RIESGO DOMINANTE	CONTROLES QUE MIRARÍA PRIMERO	EVIDENCIA MÍNIMA
Asistente RAG interno	Citar documentos obsoletos o conservar trazas con datos innecesarios.	Manifest de índice, filtros de fuente, abstención, redacción de logs.	`index_manifest.json`, `rag_eval_report.json`, `trace_sample.jsonl`.
Agente con tools	Ejecutar una acción con más efecto del previsto.	Scope por tool, modo lectura, aprobación, idempotencia, registro de efectos.	`tool_policy.json`, `approval_policy.md`, traza de tool.
Modelo local con datos sensibles	Mezclar datos de prueba, logs, pesos, cache o backups sin finalidad clara.	Aislamiento por entorno, retención, control de acceso, dataset card, inventario de hardware.	`dataset_card.md`, `access_review.md`, `retention_policy.md`.

El aprendizaje es importante: no hay una gobernanza genérica que sirva pegada encima. Hay una forma común de pensar, pero los controles prioritarios cambian con la arquitectura.

En el asistente RAG, el corazón está en corpus, citas y trazas. En el agente con tools, el corazón está en permisos, estado y efectos. En el modelo local, el corazón está en datos, despliegue, acceso, hardware y operación. Si usamos la misma checklist sin mirar esa diferencia, la gobernanza se vuelve teatro.

Por qué debería importarte

Si no sabes hacer esto, puedes publicar algo que funciona, pero no sabes explicarlo. Y en cuanto aparezca una pregunta de privacidad, un error de fuente, una salida incompatible, una queja de usuario o una revisión interna, el equipo tendrá que reconstruir la historia a mano.

El coste real de una mala gobernanza no es solo “incumplir”. Es perder capacidad de aprendizaje. Si no sabes qué versión contestó, qué datos entraron, qué control falló o qué evidencia faltaba, no puedes mejorar con precisión. Cambias cosas al azar.

Una buena primera capa de gobernanza evita tres pérdidas:

PÉRDIDA	CÓMO SE NOTA
Pérdida de trazabilidad	Nadie puede reproducir una respuesta.
Pérdida de criterio	Todas las preocupaciones pesan igual y se decide por cansancio.
Pérdida de confianza técnica	El equipo no puede demostrar qué controla ni qué acepta.

Por eso este capítulo no va de “cumplimiento” en abstracto. Va de ingeniería revisable.

Dónde solía tropezar yo

TROPIEZO	POR QUÉ ES UN PROBLEMA	ANTÍDOTO
Creer que una checklist era suficiente.	Marcar «retención revisada» sin política, configuración, prueba y owner solo mueve el problema de sitio.	Exigir evidencia revisable detrás de cada casilla, no la casilla.
Mezclar riesgo del modelo con riesgo del producto.	Un modelo razonable puede vivir en un producto peligroso por permisos, interfaz, memoria o falta de revisión.	Tomar el sistema completo como unidad de análisis, no el modelo aislado.
Escribir riesgos demasiado abstractos.	«Privacidad» o «seguridad» no son escenarios: no dicen dónde mirar ni qué controlar.	Nombrar qué ocurre, dónde, por qué importa, cómo se detecta y qué evidencia lo cubre.
Olvidar el riesgo residual.	Aplicar un control no elimina el riesgo: lo reduce o lo desplaza, y lo que queda se queda sin dueño.	Dejar el riesgo residual con owner y decisión escrita.
Dejar la gobernanza para el final.	Al final se convierte en fricción y en reescritura; nace tarde y mal.	Diseñar inventario, escenarios y evidencias junto con la arquitectura.

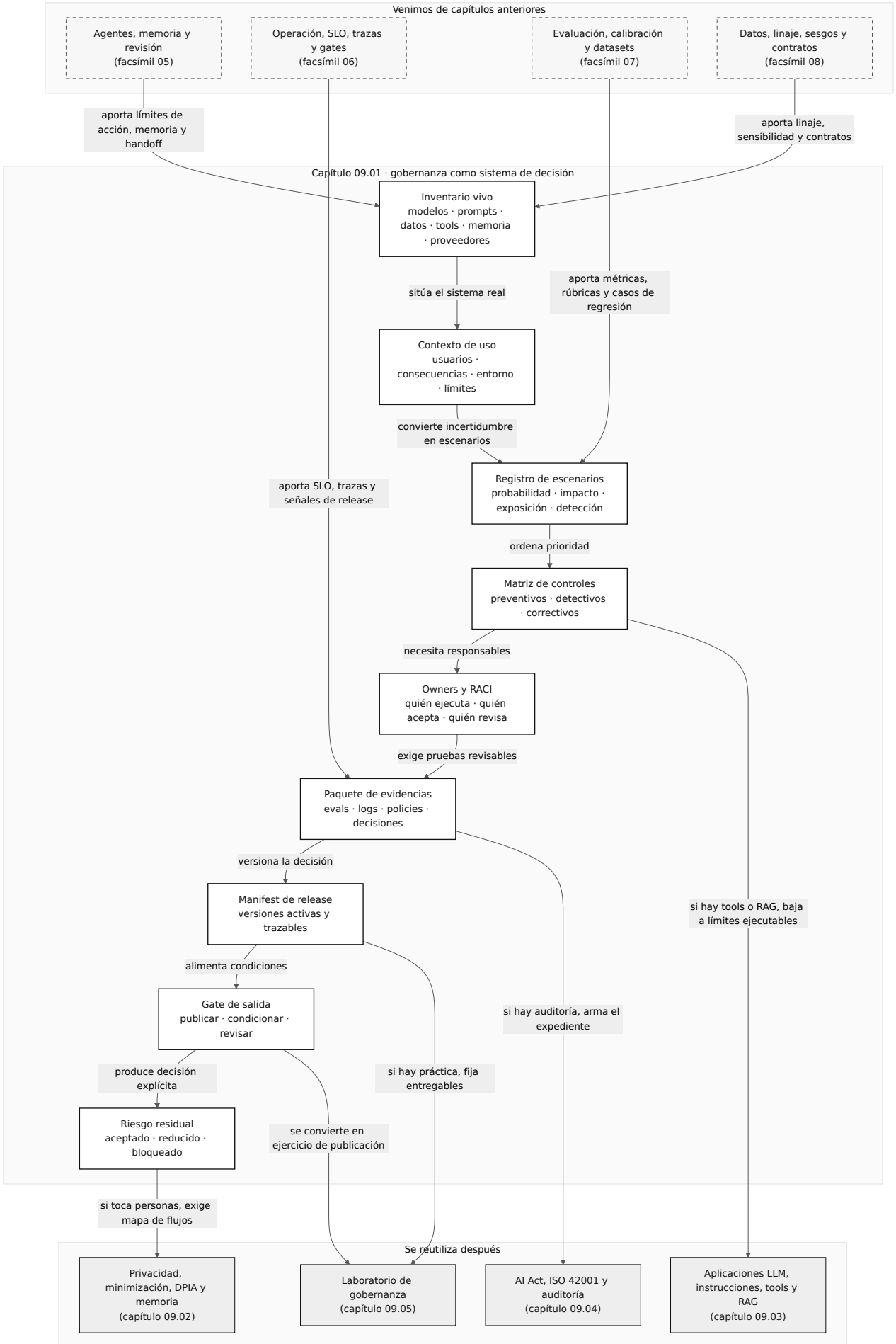
Cómo encaja todo

Este mapa une lo que ya aprendimos con lo que empieza ahora. Del facsímil 5 heredamos agentes, permisos, revisión y memoria. Del facsímil 6 heredamos operación, SLO, trazas, costes y gates. Del facsímil 7 heredamos evaluación, calibración y datasets de prueba. Del facsímil 8 heredamos datos, linaje, sesgos, contratos y calidad de dataset.

Este capítulo no añade una capa burocrática encima de todo eso. Hace algo más útil: convierte esas piezas en un **sistema de decisión revisable**. Un sistema gobernado puede responder qué existe, quién lo mantiene, qué puede fallar, qué control lo reduce, qué evidencia lo demuestra y qué decisión se tomó antes de publicar.

La lectura importante es la dirección. Gobernanza no sustituye a arquitectura, datos, evaluación ni operación; los organiza para que una decisión técnica pueda reconstruirse. Si el inventario no existe, privacidad llega tarde. Si la evaluación no está conectada a riesgos, mide cosas sueltas. Si la observabilidad no produce evidencia, el gate no puede decidir. Si el owner no acepta el riesgo residual, nadie responde por lo que queda.

En este facsímil, este capítulo es la base común. El capítulo 02 baja esa base a privacidad. El 03 la baja a aplicaciones LLM con instrucciones, tools y RAG. El 04 la traduce a cumplimiento y auditoría. El cierre obliga a practicar todo junto.



Vocabulario aprendido

TÉRMINO	DEFINICIÓN EN CASTELLANO LLANO
Riesgo	Escenario concreto donde el sistema puede producir una consecuencia no deseada.
Control	Medida técnica u organizativa que reduce, detecta, limita o corrige un riesgo.
Evidencia	Prueba revisable de que una decisión, control o medición existe.
Riesgo residual	Riesgo que queda después de aplicar controles.
Owner	Persona o equipo responsable de mantener una pieza y responder por ella.
Gate	Condición verificable para permitir, condicionar o detener una versión.
Inventario de IA	Lista viva de modelos, datos, prompts, tools, memoria, logs, owners y versiones.
Matriz de controles	Tabla que conecta riesgos con controles, evidencias y responsables.
Paquete de evidencias	Conjunto de artefactos que permite reconstruir una decisión.
DPIA	Evaluación de impacto relativa a protección de datos cuando el tratamiento puede implicar alto riesgo.
AI Management System	Sistema organizativo para gobernar IA con políticas, procesos y mejora continua.
Riesgo de sistema	Riesgo que nace de la combinación de modelo, datos, tools, contexto, interfaz y operación.
Superficie técnica	Zona concreta donde mirar riesgos: modelo, RAG, tools, memoria, logs, datos, interfaz o proveedor.
RACI	Matriz que separa quién ejecuta, quién acepta, quién aporta criterio y quién debe enterarse.
Redacción de logs	Técnica para no conservar texto o datos innecesarios en observabilidad.
Manifest de release	Registro de versiones activas de modelo, prompt, índice, policy, dataset, decisión y owners.

Antes de pasar página

Antes de pasar al capítulo de privacidad, deberías poder responder:

1. ¿Por qué gobernanza de IA no es lo mismo que tener una política escrita?
2. ¿Qué diferencia hay entre riesgo, control y evidencia?
3. ¿Por qué el sistema completo importa más que el modelo aislado?
4. ¿Qué piezas mínimas meterías en un inventario de IA?
5. ¿Cómo calcularías la severidad de un escenario con P, I, E y D?
6. ¿Qué significa riesgo residual?
7. ¿Qué artefactos enseñarías si alguien pregunta por qué una versión puede publicarse?
8. ¿Qué parte del paquete de gobernanza cambiarías primero para adaptarlo a tu proyecto?
9. ¿Qué superficie técnica revisarías primero en un sistema RAG? ¿Y en un agente con tools?
10. ¿Por qué un manifest de release cambia la calidad de una revisión?

Si no puedes responder a la 2, vuelve a “Qué sí es gobernanza de IA para ingeniería”. Si no puedes responder a la 5, vuelve a “Severidad”. Si no puedes responder a la 7, vuelve a “Evidencia”.

En resumen

IDEA	QUÉ LLEVARTE
Gobernar IA es hacer visibles las decisiones técnicas.	Inventario, contexto, riesgos, controles, evidencias y gates convierten intuición en revisión.
El riesgo vive en el sistema completo.	Modelo, RAG, tools, memoria, datos, interfaz y operación se combinan.
Sin evidencia no hay control defendible.	Una política sin prueba no permite reconstruir ni mejorar una decisión.
El riesgo residual debe tener owner.	Lo que queda después de los controles no desaparece: se acepta, se reduce o se bloquea.
Los marcos deben aterrizar en archivos.	NIST, ISO, AI Act, GDPR y OWASP aportan preguntas; ingeniería debe convertirlas en artefactos.
La práctica debe generar artefactos.	En el cuaderno del facsímil produces registro, matriz, gate, manifest y paquete de evidencias para adaptar a un proyecto real.

Para saber más

Autio, C., Schwartz, R., Dunietz, J., Jain, S., Stanley, M., Tabassi, E., Hall, P. y Roberts, K. (2024). *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.600-1

Center for Chemical Process Safety. (2001). *Layer of Protection Analysis: Simplified Process Risk Assessment*. American Institute of Chemical Engineers, Wiley.

Cox, L. A. (2008). What's Wrong with Risk Matrices? *Risk Analysis*, 28(2), 497-512. https://doi.org/10.1111/j.1539-6924.2008.01030.x

European Parliament and Council of the European Union. (2016). *Regulation (EU) 2016/679*. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679

European Parliament and Council of the European Union. (2024). *Regulation (EU) 2024/1689*. https://eur-lex.europa.eu/eli/reg/2024/1689/oj

Gebru, T. y otros (2021). Datasheets for datasets. *Communications of the ACM*, 64(12), 86-92. https://doi.org/10.1145/3458723

International Electrotechnical Commission. (2018). *IEC 60812:2018. Failure modes and effects analysis (FMEA and FMECA)*. International Electrotechnical Commission.

International Organization for Standardization. (2018). *ISO 31000:2018. Risk management, Guidelines*. International Organization for Standardization.

International Organization for Standardization. (2023). *ISO/IEC 42001:2023. Artificial intelligence management system*. https://www.iso.org/standard/42001

Kaplan, S., y Garrick, B. J. (1981). On the Quantitative Definition of Risk. *Risk Analysis*, 1(1), 11-27. https://doi.org/10.1111/j.1539-6924.1981.tb01350.x

Mitchell, M. y otros (2019). Model cards for model reporting. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 220-229. https://doi.org/10.1145/3287560.3287596

OWASP Foundation. (2025). *OWASP Top 10 for LLM and Generative AI Applications 2025*. https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/

Raji, I. D. y otros (2020). Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 33-44. https://doi.org/10.1145/3351095.3372873

Tabassi, E. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.100-1

Notas

1. Tabassi, E. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. National Institute of Standards and Technology, NIST AI 100-1. https://doi.org/10.6028/NIST.AI.100-1. Consultado el 7 de junio de 2026. ↩

2. Autio, C., Schwartz, R., Dunietz, J., Jain, S., Stanley, M., Tabassi, E., Hall, P. y Roberts, K. (2024). *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*. NIST AI 600-1. https://doi.org/10.6028/NIST.AI.600-1. Consultado el 7 de junio de 2026. ↩

3. European Parliament and Council of the European Union. (2024). *Regulation (EU) 2024/1689*. Official Journal of the European Union. https://eur-lex.europa.eu/eli/reg/2024/1689/oj. Consultado el 7 de junio de 2026. ↩

4. International Organization for Standardization. (2023). *ISO/IEC 42001:2023 Information technology, Artificial intelligence, Management system*. https://www.iso.org/standard/42001. Consultado el 7 de junio de 2026. ↩

5. European Parliament and Council of the European Union. (2016). *Regulation (EU) 2016/679*. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679. Consultado el 7 de junio de 2026. ↩

6. ISO, 2023. ↩

7. Mitchell, M. y otros (2019). Model Cards for Model Reporting. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 220-229. https://doi.org/10.1145/3287560.3287596. ↩

8. Gebru, T. y otros (2021). Datasheets for Datasets. *Communications of the ACM*, 64(12), 86-92. https://doi.org/10.1145/3458723. ↩

9. European Parliament and Council of the European Union, 2024. ↩

10. Kaplan, S. y Garrick, B. J. (1981). On the Quantitative Definition of Risk. *Risk Analysis*, 1(1), 11-27. https://doi.org/10.1111/j.1539-6924.1981.tb01350.x. Define el riesgo como el conjunto de tripletas de escenario, probabilidad y consecuencia. ↩

11. International Electrotechnical Commission. (2018). *IEC 60812:2018 Failure modes and effects analysis (FMEA and FMECA)*. IEC. El número de prioridad de riesgo se define como el producto de gravedad, ocurrencia y detección. ↩

12. Cox, L. A. (2008). What's Wrong with Risk Matrices? *Risk Analysis*, 28(2), 497-512. https://doi.org/10.1111/j.1539-6924.2008.01030.x. Demuestra las limitaciones de las matrices de riesgo y de las puntuaciones multiplicativas para ordenar prioridades. ↩

3. International Organization for Standardization. (2018). *ISO 31000:2018 Risk management, Guidelines*. ISO. Define el tratamiento del riesgo y el riesgo residual como el riesgo remanente después de aplicar las medidas de tratamiento. ↩

4. OWASP Foundation. (2025). *OWASP Top 10 for LLM and Generative AI Applications 2025*. https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/. Consultado el 7 de junio de 2026. ↩

5. Center for Chemical Process Safety. (2001). *Layer of Protection Analysis: Simplified Process Risk Assessment*. American Institute of Chemical Engineers, Wiley. Define la frecuencia mitigada como el producto de la frecuencia del evento iniciador por la probabilidad de fallo de cada capa de protección independiente. ↩

6. Raji, I. D. y otros (2020). Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 33-44. https://doi.org/10.1145/3351095.3372873. ↩

7. Amershi, S. y otros (2019). Software Engineering for Machine Learning: A Case Study. *International Conference on Software Engineering: Software Engineering in Practice*, 291-300. https://doi.org/10.1109/ICSE-SEIP.2019.00042. ↩