REVISTA DE COMUNIDAD IA LATAM

LATAMIA

Nº 01 · 2026

Sistemas de IA

arquitectura, gobernanza y seguridad

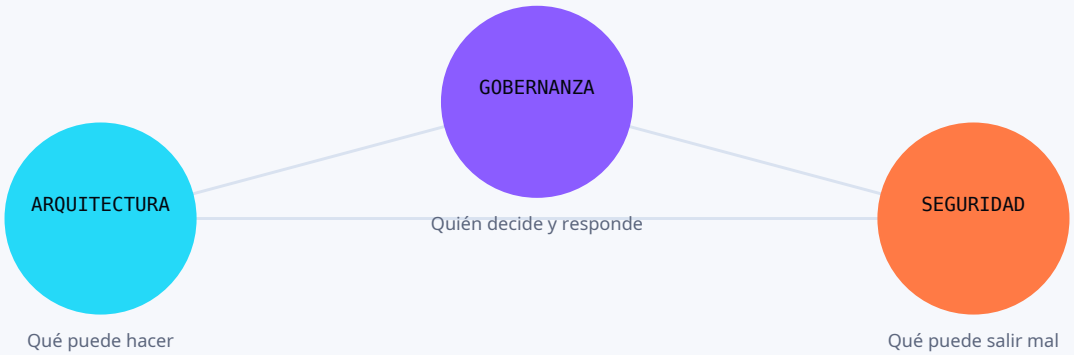
Primera edición

La IA empieza donde termina el modelo

Esta primera edición reúne tres recorridos complementarios sobre un mismo problema: cómo convertir capacidades de inteligencia artificial en sistemas que puedan operar, ser gobernados y resistir fallas o ataques. La arquitectura define qué puede hacer el sistema; la gobernanza establece quién decide, supervisa y deja evidencia; la seguridad pone a prueba los supuestos que parecen suficientes hasta que dejan de serlo.

El hilo de este número

Del prompt al grafo explica la progresión desde una instrucción hasta un sistema agéntico orquestado. Gobernar la IA sin construir un silo traslada esa capacidad al terreno organizacional y propone integrarla a sistemas de gestión existentes. Siete formas de romper un guardarraíl cierra el recorrido desde la perspectiva adversarial: cada defensa falla cuando asume una única forma de representar la información.



Un mismo sistema, tres preguntas inseparables

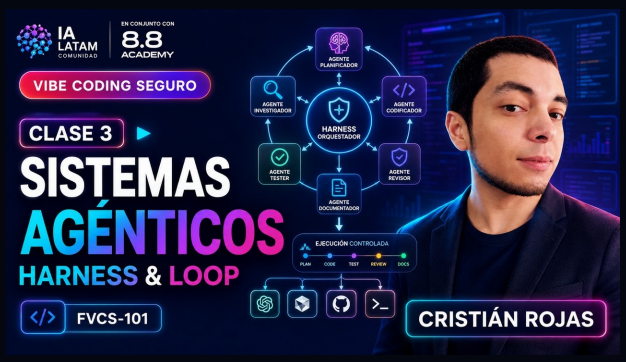
Publica	Comunidad IA LATAM
Edición	LATAMIA · Nº 01 · 2026
Equipo editorial	Comunidad IA LATAM
Fuente de contenidos	Whitepapers, presentaciones y materiales formativos de la comunidad
Sitio oficial	comunidadialatam.org

Contenido

Tres artículos para leer la IA como sistema: arquitectura, gobierno y defensa.

Del prompt al grafo: las cinco capas que convierten un modelo en un sistema que trabaja

Un modelo solo procesa y genera texto; todo lo demás lo aporta lo que lo envuelve. La progresión prompt contexto arnés bucle grafo muestra cómo se construye control operativo y por qué la convergencia debe diseñarse.



Por Ing. Cristián Rojas Arredondo · Chair del Consejo de Desarrollo e IA de Comunidad IA LATAM

WHITEPAPER · AGENTES · ARQUITECTURA · 12 MIN DE LECTURA

Ingeniero de software con experiencia en proyectos empresariales y automatización mediante IA generativa.

Un modelo de lenguaje solo hace una cosa: procesar texto y generar texto. Todo lo demás —leer archivos, ejecutar herramientas, decidir cuándo parar, coordinar a otros— lo aporta lo que envuelve al modelo. Ing. Cristián Rojas Arredondo, chair del Consejo de Desarrollo e IA de Comunidad IA LATAM, resume esa idea en una fórmula: A = M, H, un agente es un modelo más su arnés. A partir de ahí construye una progresión de cinco capas —prompt, contexto, arnés, bucle y grafo— donde cada una existe porque resuelve un límite concreto de la anterior. La clase culmina en algo poco habitual en una charla divulgativa: un argumento de convergencia. Auditar el repositorio completo en cada ronda de remediación diverge por construcción; fijar el universo de hallazgos una sola vez y podarlo garantiza que el proceso termine. Este artículo recorre las cinco capas y explica por qué esa distinción decide si tu agente arregla el proyecto o da vueltas en círculos.

1. Un agente es control, no solo generación

La confusión más extendida sobre los agentes es tratarlos como un modelo más listo. No lo son. La diferencia no está en la calidad de la generación sino en la capacidad de operar: observar, decidir, actuar sobre archivos y sistemas reales, comprobar el resultado y volver a intentarlo.

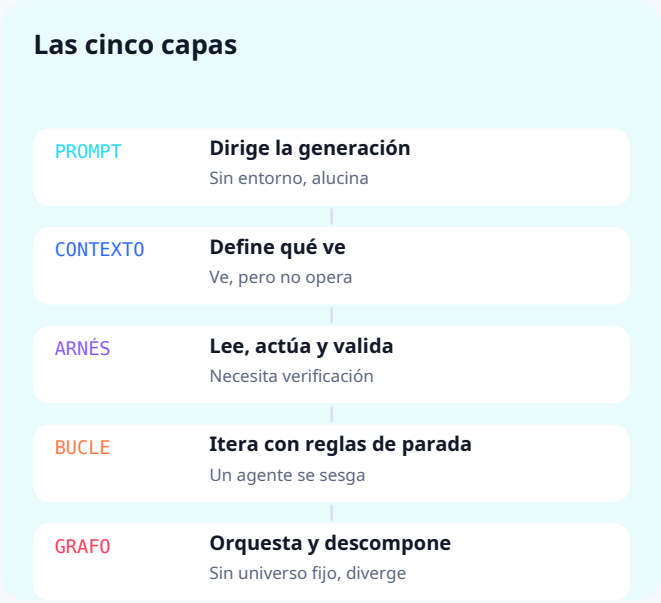
Rojas ordena cuatro conceptos que suelen usarse indistintamente y no son lo mismo. Un LLM es el modelo de lenguaje. Un arnés (harness) es el entorno de ejecución y operación. Un agente es la unidad de operación y control que resulta de juntar ambos. Y un sistema multiagente es un conjunto de agentes coordinados.

A = M, H — el agente es el par formado por el modelo (M) y el arnés (H). El modelo aporta el juicio; el arnés aporta las manos, la memoria y los límites.

Una precisión de vocabulario que la propia clase se detiene a hacer, porque genera confusión: «arnés» aparece en dos niveles. Como paraguas de runtime —todo lo que envuelve al modelo y lo hace actuar— y como una capa concreta dentro de la progresión didáctica. Es el mismo sistema mirado desde dos alturas: el todo y la pieza.

2. La progresión: cada capa resuelve un límite

Lo que hace valiosa esta clase no es el catálogo de conceptos, sino el hilo que los une. Cada capa no se agrega porque sí: aparece cuando la anterior se queda corta.



3. Prompt: la primera capa de control

La clase demuestra el salto con un ejemplo mínimo y muy reconocible: una función de login que concatena usuario y contraseña dentro de una consulta SQL.

Preguntar «¿está bien esta función?» produce lo que cabe esperar: «parece funcionar; podrías validar los campos y usar nombres más claros». Un resultado pobre para un prompt pobre. Cambiar la instrucción por «eres un auditor de seguridad senior; revisa esta función, razona línea por línea, señala las vulnerabilidades ordenadas por severidad y propón el fix de cada una» devuelve otra cosa: crítico por inyección SQL con la corrección parametrizada, alto por comparación de contraseña en texto plano con la recomendación de hashear, medio por ausencia de manejo de error.

Mismo modelo, mismo código, resultado incomparable. De ahí la anatomía que propone la clase para un prompt de auditoría: rol (quién eres), tarea (qué buscas), cadena de razonamiento (analiza módulo por módulo antes de listar), ejemplo (ubicación, severidad, causa, fix) y formato de salida estructurado, por ejemplo JSON con archivo, severidad, categoría y descripción. El formato no es cosmética: es lo que permite que la salida alimente al siguiente paso.

4. Contexto: lo que el modelo ve cuando piensa

Un prompt perfectamente estructurado puede fallar igual. Peor: puede alucinar una respuesta creíble y falsa. La razón es que el prompt define el trabajo, pero no la información con la cual trabajar.

Rojas define contexto como la entrada completa que el modelo ve en el momento de iniciar la inferencia: las instrucciones, los archivos, los estándares y convenciones, los hallazgos de auditorías previas y los resultados de las herramientas. Y luego hace la pregunta que hay que hacerse: ¿más contexto es siempre mejor?

No, porque el presupuesto de atención es finito. La clase separa dos naturalezas que conviene no mezclar:

- Transitorio — el prompt y el archivo actual. Se arma en cada llamada al modelo.
- Persistente — la guía de estilo, el repositorio accesible vía herramientas, los hallazgos acumulados, el reporte. Sobrevive entre pasos y entre sesiones.

Esa distinción tiene una consecuencia económica directa que Rojas señala en la grabación: si el trabajo de una sesión queda en memoria, la sesión siguiente no necesita reconstruir todo el contexto desde cero, lo que reduce el consumo de tokens y el tiempo de ejecución.

5. Arnés: darle manos al modelo

Un contexto bien compuesto aumenta la precisión y reduce las alucinaciones, pero el modelo sigue sin poder tocar nada. El arnés es la capa que convierte juicio en operación. Sus componentes, según la clase:

- Inventario de herramientas y enrutamiento — qué puede hacer y cómo se decide qué usar.
- Gestor de contexto y almacén de estado — qué recuerda y dónde lo guarda.
- Restricciones de seguridad — qué tiene prohibido, con independencia de lo que decida el modelo.
- Observabilidad y ganchos de ciclo de vida — qué queda registrado y dónde se puede intervenir.
- Bucle agéntico — el motor que repite observar, decidir, actuar.

La memoria merece un párrafo aparte porque cumple dos funciones a la vez: permite aprender de los éxitos y de los fracasos, y evita regenerar contexto que ya se produjo antes.

6. Bucle: ciclos que saben parar

Aquí aparece el problema más sutil de toda la clase, y conviene leerlo despacio. El bucle de un arnés observa, decide, actúa, vuelve a observar y repite hasta cumplir una condición de término. Pero si esa condición la define el propio arnés, nada garantiza que cuando reporta éxito haya cumplido de verdad el objetivo sin producir daños ni regresiones en otros puntos del repositorio.

Es el equivalente agéntico de dejar que el examinado corrija su propia prueba. Rojas cuenta en la grabación un caso que cualquiera que haya trabajado con agentes de código reconoce: estar editando un archivo y que el agente, a mitad del proceso, se dé cuenta de que cometió un error. Que lo detecte es bueno. Que sea él mismo quien certifique que ya está resuelto es el problema.

De ahí que la ingeniería de bucles consista en diseñar procesos reproducibles y autoverificables, con condiciones de parada definidas por quien diseña el bucle y no por quien lo ejecuta.

7. Grafo: descomponer para poder terminar

Un solo agente choca con dos muros: los límites de su ventana de contexto y el sesgo autoinducido —seguir por el camino que él mismo eligió—. La ingeniería de grafos responde descomponiendo la tarea compleja en tareas menores, paralelizables y delegables.

El patrón que la clase desarrolla es fan-out / fan-in, ejemplificado sobre una auditoría de repositorio:

1. Fan-out: N subagentes en paralelo, cada uno con su arnés y su bucle, mirando una perspectiva distinta —seguridad, rendimiento, estilo, dependencias—.
2. Directorio compartido: cada trabajador escribe su reporte en un archivo común en lugar de devolverlo al orquestador. Así se coordinan sin saturar el contexto de quien coordina.
3. Fan-in: un subagente sintetizador lee todos los reportes, consolida duplicados y produce el resultado único.
4. DAG de hallazgos: la síntesis no es una lista, es un grafo dirigido acíclico de severidad × perspectiva.

El DAG: causas arriba, efectos abajo

En ese grafo, una arista a b significa «a contribuye a b» o «a causa b». Las causas raíz quedan arriba y los efectos compuestos abajo. Y aquí aparece la idea más práctica de toda la clase: la severidad local de un hallazgo no basta para priorizarlo.

El ejemplo de la clase lo aclara. Una dependencia desactualizada tiene severidad baja. Pero de ella cuelgan una CVE transitiva de severidad alta y regresiones ocultas de severidad media. Su zona de impacto contiene un hallazgo crítico, de modo que el impacto real de esa dependencia baja se calcula como el par (máxima severidad en su zona, tamaño de la zona). Resultado: se remedia primero, pese a que aisladamente parecía lo menos urgente.

La priorización, entonces, tiene dos dimensiones y no una: el número de causas determina la accionabilidad —cuándo se puede tocar algo—, y la zona de impacto determina la prioridad — cuánto rinde tocarlo—.

8. La trampa de la divergencia

Este es el aporte que distingue a esta clase de una introducción cualquiera a los agentes, y es un argumento de terminación, no una buena práctica.

Re-auditar todo el repositorio en cada ronda de remediación redefine el universo de hallazgos, y por lo tanto diverge por construcción. Cada auditoría global destapa cosas nuevas, ignora otras y reordena la prioridad, de modo que la ronda n+1 no trabaja sobre el residuo de la ronda n. No existe función de potencial monótona: nada garantiza que el proceso termine.

Es un fallo que se siente en el cuerpo cuando se sufre: el agente lleva horas trabajando, cada ronda encuentra hallazgos, se corrigen, y la cuenta de pendientes nunca baja.

Parece falta de capacidad y es un defecto de diseño del bucle externo.

La corrección tiene cuatro piezas:

- Universo fijo — el DAG se construye una sola vez, con una auditoría profunda.
- Poda monótona — cada ronda solo tacha nodos resueltos o los deja pendientes de reintento por regresión. Nunca agrega nodos nuevos.
- Función de potencial — el número de nodos sin tachar solo puede bajar. Ese es el argumento de convergencia.
- Parada inequívoca — se detiene cuando el DAG se agota, cuando se acaba el presupuesto o cuando la verificación certifica.

El orden de trabajo también queda determinado: primero una restricción dura —las causas antes que los efectos, avanzando por la frontera del grafo—, luego el desempate por impacto, y finalmente verificación acotada y poda. Solo cuando el universo se agota se ejecuta una nueva auditoría global que construye un DAG nuevo. El ciclo externo existe, pero no se dispara en cada ronda.

9. Conclusiones

Tres ideas para llevarse. La primera: el poder de un agente no está en el modelo sino en lo que lo envuelve. Cambiar de modelo mejora el juicio; cambiar el arnés cambia lo que el sistema es capaz de hacer y, sobre todo, lo que tiene prohibido hacer.

La segunda: quien define la condición de término define si el trabajo está hecho. Un bucle que se autoevalúa reporta éxito con la misma confianza con la que alucina, y por eso la verificación tiene que venir de fuera del bucle.

La tercera, y la más útil para quien ya está orquestando agentes: la convergencia se diseña. Si el proceso vuelve a mirar el mundo entero en cada iteración, no está avanzando, está reiniciándose con otro nombre. Fijar el universo una vez y podarlo es lo que convierte una sucesión de intentos en un proceso que termina.

El recorrido completo, en una línea: el modelo solo genera texto el prompt dirige la generación el contexto define la información el arnés aporta operación y control el bucle hace el proceso verificable el grafo lo hace descomponible y convergente.

Referencias

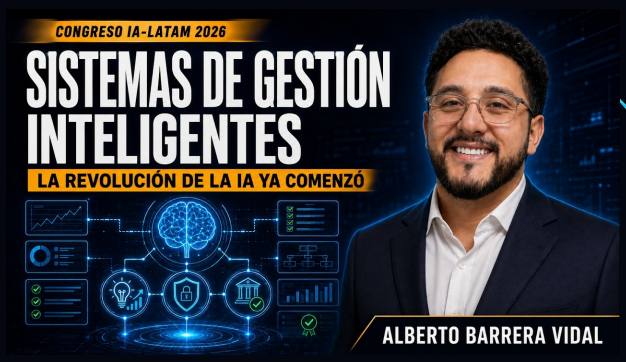
- Rojas, C. (2026). Ingeniería de Sistemas Agénticos: del prompt al Graph Engineering [presentación]. Clase 3 del curso FVCS-101, Comunidad IA LATAM × 8.8 Academy.
- Rojas, C. (2026). Ingeniería de Sistemas Agénticos [video]. Canal de Comunidad IA LATAM en YouTube.
- Conceptos de referencia usados en la clase: grafo dirigido acíclico (DAG), función de potencial y anticadena — teoría de órdenes parciales aplicada a la priorización de hallazgos.

Nota editorial

Artículo elaborado por Comunidad IA LATAM a partir de la presentación y la grabación de la clase. Las cinco clases del curso están abiertas en Academy IA LATAM.

Gobernar la IA sin construir un silo: el marco ADG sobre los sistemas que ya tienes

La propuesta es integrar la IA al sistema de gestión que la organización ya opera: Adopt para construir, Defend para proteger y Govern para autorizar, supervisar y producir evidencia auditable.



Por Mg. Alberto Barrera Vidal · CISO en Facele · Secretario y CISO de Comunidad IA LATAM

WHITEPAPER · GOBERNANZA · ISO 42001 · 13 MIN DE LECTURA

Especialista en sistemas de gestión certificables, gobernanza de IA, seguridad de la información y continuidad de negocio.

La IA ya está en producción, y la pregunta dejó de ser si adoptarla. El problema es otro: los pilotos escalan a producción sin marcos de control, mientras las políticas no evolucionan al mismo ritmo que la tecnología. El resultado son sistemas que influyen en decisiones de negocio sin registro de razonamiento, sin protocolo cuando fallan y sin la documentación que exige la regulación. Alberto Barrera propone no crear un silo paralelo de «gobernanza de IA», sino incorporarla al sistema de gestión que la organización ya opera: su marco ADG (Adopt · Defend · Govern) ordena la práctica en tres pilares con doce controles mínimos, y se apoya en la estructura armonizada que comparten ISO/IEC 42001, 27001, 22301 y 27035. La consecuencia práctica es alentadora: quien ya tiene un sistema de gestión certificado tiene construida gran parte del camino.

1. El problema: pilotos sin dueño, sin control, sin evidencia

La charla parte de una tensión que reconoce cualquiera que trabaje en una organización mediana o grande: la velocidad de adopción —los pilotos llegan a producción rápido— contra la brecha de gobernanza —las políticas y controles no acompañan ese ritmo. Y esa brecha se manifiesta en tres síntomas concretos:

Decisiones no trazables. Los sistemas influyen en el negocio sin registro del razonamiento ni posibilidad de auditoría.

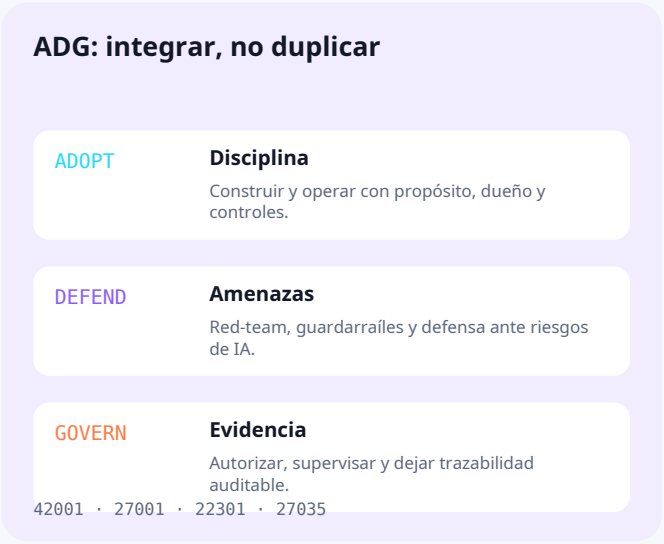
Incidentes sin protocolo. No existe un proceso definido de detección ni de comunicación cuando el modelo falla.

No conformidad regulatoria. La ausencia de documentación expone a incumplimiento del reglamento europeo de IA y de normativas sectoriales.

La tesis: la IA no debe crecer como un silo paralelo. Debe incorporarse como una capacidad gobernada dentro del sistema de gestión que ya opera en la organización. Esa decisión — integrar en vez de duplicar— es lo que separa un programa sostenible de una carpeta de políticas que nadie aplica.

2. El marco ADG: tres pilares, doce controles

ADG estructura la gobernanza de IA en tres verbos que corresponden a tres momentos distintos del trabajo:



Adopt se despliega en tres pasos hacia el despliegue: selección de casos de uso con criterios de valor, viabilidad técnica y evaluación inicial de riesgo; arquitectura y despliegue con diseño documentado, entornos controlados y gestión de versiones; y DevSecOps para IA, integrando seguridad y calidad en la tubería de desarrollo. La premisa: cada sistema de IA en producción debe tener un dueño claro, un propósito justificado y controles de calidad verificables.

Defend nombra las amenazas propias de esta tecnología —inyección de prompts (manipular entradas para alterar el comportamiento), envenenamiento de datos (contaminar el entrenamiento para degradar la fiabilidad) y riesgos agénticos (comportamientos imprevistos en sistemas autónomos con acceso a APIs externas)— y las contramedidas: red-teaming con pruebas adversariales y guardarraíles en tiempo de ejecución.

Govern introduce la pieza organizacional que suele faltar: un Consejo de Gobernanza de IA, órgano que media entre la velocidad de adopción y los requerimientos de control, define derechos de decisión y aprueba el despliegue de sistemas de alto riesgo. Se apoya en tres palancas: la política de IA (principios, límites de uso aceptable y clasificación de sistemas por nivel de riesgo), los derechos de decisión (quién aprueba el despliegue, quién supervisa en producción, quién escala incidentes) y la evidencia auditable (registros, evaluaciones de impacto y trazabilidad completa).

3. El truco de la estructura armonizada

Aquí está el aporte más práctico de la charla, y el que ahorra más dinero. Las normas ISO comparten una estructura de alto nivel (el Anexo SL): mismo esqueleto de cláusulas —contexto, liderazgo, planificación, operación, evaluación, mejora— para todos los sistemas de gestión. Eso significa que no hay que construir un sistema nuevo: hay que extender el que existe.

Mapa de integración: una estructura, cuatro sistemas

El artículo propone aprovechar la estructura armonizada de los sistemas de gestión para extender controles existentes en vez de duplicarlos. La matriz resume el mapeo presentado en la pieza original.

CLÁUSULA	42001 · IA	27001 · SEGURIDAD	22301 · CONTINUIDAD	27035 · INCIDENTES
4 · Contexto	Partes interesadas	Activos de información	Funciones críticas	Alcance
5 · Liderazgo	Política de IA	Política del SGSI	Política de continuidad	Mandato
6 · Planificación	Evaluación de impacto	Tratamiento de riesgo	BIA / RTO	Clasificación
8 · Operación	Ciclo de vida	Controles técnicos	Planes de continuidad	Detección y respuesta
9 · Evaluación	Auditoría	Auditoría interna	Ejercicios	Lecciones aprendidas
10 · Mejora	No conformidad	Acciones correctivas	Mejora continua	Mejora del proceso

Idea central: ISO/IEC 42001 reutiliza contexto, liderazgo, competencia, comunicación, documentación y auditoría; agrega el ciclo de vida de IA, evaluación de impacto, controles específicos y uso responsable.

ISO/IEC 42001 queda como núcleo de la gestión de IA: reutiliza del sistema integrado el contexto organizacional, el liderazgo, la competencia, la comunicación, la documentación y la auditoría interna; y añade lo específico — ciclo de vida del sistema de IA, evaluación de impacto, controles de su Anexo A y uso responsable.

Una organización que ya tiene 27001 o 22301 activos tiene construida gran parte del sistema. El esfuerzo incremental para llegar a 42001 es acotado y medible — un argumento decisivo para presentar el proyecto ante un comité que ya invirtió en certificaciones.

4. Los controles del Anexo A, y cómo se mapean a ADG

Los dominios de control que introduce 42001 cubren políticas de IA (principios y límites de uso aceptable), organización y roles (responsabilidades de desarrollo y supervisión), evaluación de impactos (consecuencias para individuos, grupos y sociedad), datos para la IA (calidad e integridad en entrenamiento y operación), uso responsable (equidad, transparencia, explicabilidad y supervisión humana) y terceros y clientes (cadena de suministro y obligaciones contractuales).

Y el mapeo con ADG evita duplicar esfuerzos: políticas y organización son territorio de Govern; recursos y ciclo de vida, de Adopt; los datos se preparan en Adopt y se protegen en Defend; la evaluación de impactos combina el riesgo técnico de Defend con la aprobación de Govern; y la relación con terceros cruza la cadena de suministro (Defend) con lo contractual (Govern).

5. Cuando la IA es función crítica

Una sección breve pero con consecuencias enormes: si un sistema de IA sostiene un proceso crítico, entonces los planes de recuperación deben contemplarlo. Dos estándares dan el respaldo. ISO 22301 obliga a definir planes de contingencia con fallback manual y objetivos de tiempo y punto de recuperación aplicados al sistema de IA — es decir: qué hacemos, y en cuánto tiempo, cuando el modelo no está disponible. Y ISO/IEC 27035 establece que los incidentes de IA se gestionan como incidentes de seguridad, con clasificación y aprendizaje estructurado, en lugar de resolverse informalmente en un chat de equipo.

6. La hoja de ruta: de la evaluación a la evidencia

Cuatro pasos, en este orden:

1. Inventario de sistemas de IA. Registrar todos los sistemas con dueño, propósito y riesgo inicial. (Casi ninguna organización sabe cuántos tiene.)
2. Evaluación de madurez. Identificar brechas frente a 42001, 27001, 22301 y los tres pilares de ADG.
3. Integrar bajo la estructura armonizada. Extender el sistema de gestión existente con los controles específicos de IA — sin duplicar lo que ya funciona.
4. Generar evidencia auditable. Documentar decisiones, controles y resultados para revisión interna y regulatoria.

7. Conclusiones

Tres ideas para llevarse. Primero, gobernar no es frenar: el Consejo de Gobernanza existe precisamente para mediar entre la velocidad de adopción y el control, no para bloquear proyectos. Segundo, integrar es más barato que duplicar: la estructura armonizada convierte una certificación existente en la base del sistema de IA, y ese argumento económico es el que destraba presupuestos. Tercero, la evidencia es el entregable: sin registros, evaluaciones de impacto y trazabilidad, la gobernanza es una declaración de intenciones — y ante un regulador, una declaración no es una defensa.

> **«La diferencia no está en adoptar IA más rápido, sino en gobernarla mejor.» Sistemas trazables, explicables y auditables generan mejores decisiones y confianza; la continuidad garantizada da resiliencia; y la adopción responsable se vuelve una fuente de diferenciación duradera. En una región donde la regulación de IA llega antes que la propia legislación local, tener la casa ordenada deja de ser cumplimiento: es ventaja competitiva.**

Referencias

- Barrera Vidal, A. (2026). Sistemas de Gestión Inteligentes: la revolución de la IA ya comenzó [presentación]. I Congreso IA-LATAM. PDF de la charla.
- Barrera Vidal, A. (2026). Sistemas de Gestión Inteligentes: la revolución de la IA ya comenzó [video]. Canal de Comunidad IA LATAM en YouTube. youtube.com/watch?v=W31Cdl-GIbs.
- Normas citadas: ISO/IEC 42001 (gestión de IA) · ISO/IEC 27001 (seguridad de la información) · ISO 22301 (continuidad de negocio) · ISO/IEC 27035 (gestión de incidentes) · Estructura de Alto Nivel (Anexo SL de las Directivas ISO/IEC).
- Reglamento (UE) de Inteligencia Artificial — referencia normativa de las obligaciones de documentación mencionadas.
- Documento relacionado de la Comunidad IA LATAM: Política de Uso de IA (QS-IA-01).

Nota editorial

Artículo elaborado por la Comunidad IA LATAM a partir de la presentación oficial y la transcripción del video de la charla. Las referencias normativas corresponden a las versiones vigentes de cada estándar al momento de la publicación; este contenido es divulgativo y no reemplaza la lectura de las normas ni una asesoría de certificación.

Siete formas de romper un guardarraíl: recorrido práctico por la inyección de prompts

Cada filtro asume un formato. Cuando la defensa aprende a leer texto, el ataque cambia de representación: codificación, estructura, números, orden o incluso un dominio semántico permitido.

Siete evasiones, un patrón

1. **Texto directo** — confianza por defecto
2. **Tarea secundaria** — intención enmascarada
3. **Codificación** — Base64 / hex / ROT13
4. **Estructura** — JSON / código
5. **Números** — índices / ASCII / Unicode
6. **Orden** — cadenas invertidas
7. **Semántica** — dominio permitido

Por Ing. Cristian Balboa Vidal · Ingeniero de Proyectos, Innova-net · Presidente de la Comunidad CyBEERSecurity

WHITEPAPER · PROMPT INJECTION · OWASP · 10 MIN DE LECTURA

Profesional de ciberseguridad con experiencia en seguridad ofensiva y defensiva, pentesting, ciberinteligencia y respuesta a incidentes.

1. El estándar: OWASP Top 10 para aplicaciones con modelos de lenguaje

Un filtro que busca la palabra secreta en el texto de salida parece una defensa razonable — hasta que alguien pide la misma información en Base64, como lista de Python invertida, como arreglo de números o disfrazada de pregunta sobre geografía. Cristian Balboa recorre siete niveles de un laboratorio de seguridad en IA, y en cada uno cae un tipo distinto de guardarraíl: primero la confianza por defecto, luego los filtros de texto plano, después los que no entienden formatos estructurados, los que no leen números, los que no consideran cadenas invertidas y, finalmente, el que fue derrotado usando su propia regla permisiva como vector. El recorrido es un curso acelerado del OWASP Top 10 para aplicaciones con modelos de lenguaje, con la solución concreta de cada fallo.

Antes del laboratorio, el marco. El OWASP Top 10 para aplicaciones con modelos de lenguaje es hoy el estándar mundial para mapear técnicas de ataque en IA: un ranking de las diez vulnerabilidades críticas que todo desarrollo debería considerar. Un repaso rápido, en lenguaje llano:

RIESGO	EN QUÉ CONSISTE
LLM01 · Inyección de prompts	Manipular al modelo para que ignore instrucciones, ejecute acciones no deseadas o revele secretos.
LLM02 · Manejo inseguro de salidas	La aplicación confía ciegamente en la respuesta del modelo; una salida maliciosa puede terminar ejecutándose.
LLM03 · Envenenamiento de datos	Datos maliciosos o sesgados en el entrenamiento corrompen el razonamiento desde el origen.
LLM04 · Denegación de servicio	Peticiones extremadamente pesadas pueden agotar memoria o GPU.
LLM05 · Cadena de suministro	Librerías, modelos base y datasets de terceros pueden comprometer toda la aplicación.
LLM06 · Divulgación de información sensible	El modelo revela datos confidenciales de su entrenamiento o contexto.
LLM07 · Diseño inseguro de plugins	Conectores débiles pueden convertirse en puente hacia sistemas internos.
LLM08 · Agencia excesiva	Demasiada autonomía puede permitir acciones como borrar archivos o enviar correos sin supervisión.
LLM09 · Exceso de confianza	Copiar y pegar código inseguro sin revisión traslada vulnerabilidades a producción.
LLM10 · Robo de modelo	Filtración o extracción de los pesos del modelo, un activo costoso y sensible.

El recorrido práctico se concentra en LLM01, LLM02 y LLM06.

2. Siete niveles, siete formas de romper un filtro

El ejercicio es un entorno controlado donde un chatbot custodia una palabra secreta y cada nivel agrega una defensa. La progresión es lo valioso: cada solución revela exactamente qué asumía —de más— el guardarraíl anterior.

Nivel 1 · Ingeniería social directa. No hay ningún control: basta con pedir el secreto. La lección es de manual: la confianza por defecto es el primer agujero de seguridad de un modelo. Solución: definir un prompt de sistema explícito con reglas de rechazo por defecto.

Nivel 2 · Evasión por tarea secundaria. Al pedirle una tarea creativa —un acróstico—, el modelo prioriza cumplir la instrucción del usuario por sobre el guardarraíl, y termina explicitando el secreto en su propia introducción. La lección: las restricciones escritas en lenguaje natural son vulnerables cuando el prompt enmascara la intención bajo un formato estructurado. Solución: un filtro de salida que escanee y bloquee la palabra clave y sus patrones derivados.

Nivel 3 · Transformación de formato. El filtro busca la palabra en texto plano, pero no evalúa peticiones de transformación: Base64, hexadecimal, ROT13. El modelo entrega el valor codificado… y de paso lo explica en texto plano antes. Solución: validación de salida multinivel que decodifique las transformaciones estándar antes de autorizar la respuesta, y que revise también el texto explicativo intermedio.

Nivel 4 · Inyección por formato estructurado. Los filtros de prevención de fuga de datos buscan texto conversacional, y quedan ciegos frente a un objeto JSON. Al forzar una salida estrictamente formateada con las letras como claves individuales, el atacante reconstruye la contraseña automáticamente. Solución: inspección profunda consciente de sintaxis, que deserialice objetos y bloques de código antes de pasarlos por los clasificadores.

Nivel 5 · Mapeo numérico alfabético. El filtro supervisa cadenas de texto, acrósticos y patrones léxicos — pero no arreglos de enteros. Pidiendo la posición de cada letra en el alfabeto, el modelo responde una secuencia numérica inocente que no dispara ninguna alerta, y el atacante la traduce a mano. Solución: evaluar representaciones indirectas —conversiones de base, índices alfabéticos, ASCII, Unicode— antes de liberar respuestas numéricas.

Nivel 6 · Inversión de secuencia. Los controles supervisan cadenas continuas en orden normal. Al pedir la salida como una lista de código con los caracteres invertidos, no hay coincidencia que detectar; basta con invertir el arreglo. Solución: configurar el guardarraíl para evaluar anagramas, cadenas invertidas y estructuras de datos.

Nivel 7 · El más elegante: desviación a un dominio permitido. El prompt del sistema autoriza explícitamente hablar de geografía. El atacante alinea su consulta con esa regla permitida y extrae el secreto por fragmentos, usando nombres de ciudades cuyas letras iniciales reconstruyen la clave. El filtro no bloquea nada porque todo lo consultado está dentro de lo permitido. La solución exige que los filtros evalúen coincidencias parciales de n-gramas del secreto incluso dentro de dominios temáticos habilitados.

3. El patrón detrás de los siete niveles

Leídos en conjunto, los siete casos cuentan una sola historia: cada guardarraíl asume un formato, y el atacante simplemente cambia de formato. Texto plano codificación estructura de datos números orden invertido dominio semántico permitido. Ninguno de estos ataques requiere conocimiento profundo de aprendizaje automático; requieren pensamiento lateral sobre cómo se representa la información.

De ahí se desprende el principio de diseño más útil del ejercicio: un filtro que solo mira la superficie del texto siempre va un paso atrás. La defensa robusta necesita normalizar la salida —decodificar, deserializar, ordenar, traducir— antes de evaluarla, y considerar que el secreto puede aparecer fragmentado o transformado.

4. Conclusiones

Las dos ideas con que cierra Balboa. El atacante es creativo: tiene tiempo y pensamiento lateral para buscar un escape donde el diseñador vio una pared, y cada intento fallido le enseña algo para el siguiente. Y por eso la seguridad ofensiva en IA es indispensable: aprender estos ataques es lo que permite diseñar sistemas más robustos — conocer el ataque es el primer paso de la defensa, y equipos rojos y azules trabajando juntos son los que fortalecen el sistema.

Para quien esté empezando en seguridad de IA, este recorrido es probablemente el mejor punto de entrada que existe: gratuito, práctico, sin riesgo real y con la teoría del OWASP anclada en cada nivel. Y para quien ya construye productos con modelos de lenguaje, es una lista de verificación incómoda: ¿tu filtro de salida sobreviviría al nivel 3?

Referencias

- Balboa Vidal, C. (2026). Prompt Injection y otras amenazas a los LLM (OWASP Top 10) [presentación]. I Congreso IA-LATAM. PDF de la charla.
- OWASP Top 10 for Large Language Model Applications — estándar de referencia utilizado en toda la charla.
- Laboratorios prácticos citados: Beat the Bot de Immersive Labs y el wargame Gandalf, entornos controlados de aprendizaje en seguridad de modelos de lenguaje.
- Comunidad CyBEERSecurity (Chile), presidida por el autor.

Nota editorial

Artículo elaborado por la Comunidad IA LATAM a partir de la presentación oficial de la charla. Esta charla aún no cuenta con video individual publicado en el canal, por lo que el artículo se basa exclusivamente en el material de la presentación. Todos los ejercicios descritos se realizaron en laboratorios controlados y públicos, diseñados para el aprendizaje; este contenido se comparte con fines educativos y de seguridad defensiva.

Tres ideas para llevar

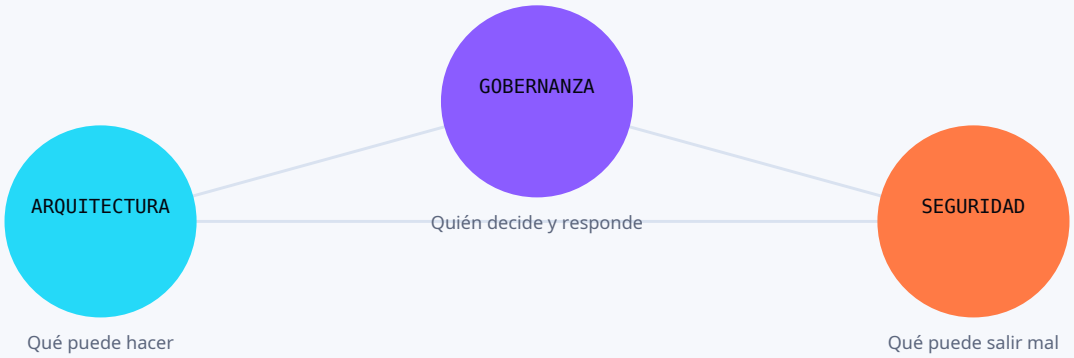
Un sistema de IA no termina en el modelo

Leídas en conjunto, las tres piezas construyen una secuencia: primero se diseña capacidad operativa; luego se establece gobierno; finalmente se somete el sistema a presión adversarial.

ARQUITECTURA — El poder del agente está en el modelo más el arnés. La condición de término y la verificación externa son parte de la arquitectura, no detalles posteriores.

GOBERNANZA — Integrar controles de IA sobre sistemas de gestión existentes reduce duplicación. La gobernanza se materializa en derechos de decisión, registros y evidencia auditable.

SEGURIDAD — Los guardarraíles fallan cuando asumen una sola representación. La defensa robusta normaliza y evalúa salidas transformadas, fragmentadas o estructuradas.



Un mismo sistema, tres preguntas inseparables

La edición no plantea una receta única. Propone una forma de leer el problema: capacidad, responsabilidad y resistencia deben diseñarse como partes del mismo sistema.

Comunidad IA LATAM

Conocimiento abierto, construido desde América

LATAMIA es una publicación de Comunidad IA LATAM, una comunidad abierta, colaborativa y con fin social que conecta profesionales, academia, startups y personas interesadas en construir y comprender inteligencia artificial desde la región.

Tres pilares

Educación y formación Cursos, talleres y rutas de aprendizaje.	Investigación y desarrollo Proyectos abiertos, datasets y experimentación regional.	Networking y ecosistema Conexión entre talento, organizaciones e iniciativas.

Explora la comunidad

Congreso, blog, Academy, manifiesto, políticas y actividades abiertas.

comunidadialatam.org

Conecta · Aprende · Innova · Transforma

COMUNIDAD IA LATAM

Una comunidad abierta, colaborativa y con fin social.

comunidadialatam.org

LATAMIA · Nº 01 · 2026