El futuro del desarrollo de software

Del código al conocimiento: la nueva arquitectura sobre la que se construirá el software empresarial

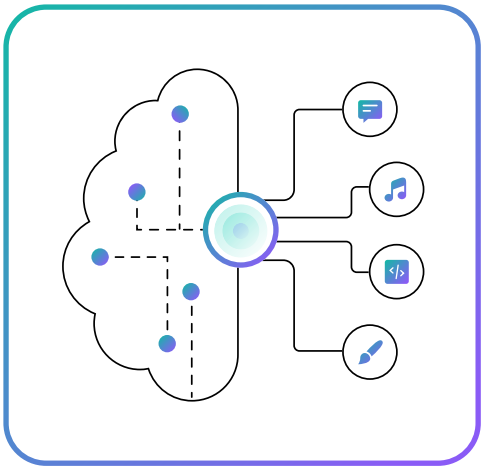
Estamos viviendo una doble revolución tecnológica.

Dos fuerzas simultáneas se están potenciando mutuamente y redefiniendo la manera en que se construye, opera y concibe el software empresarial.

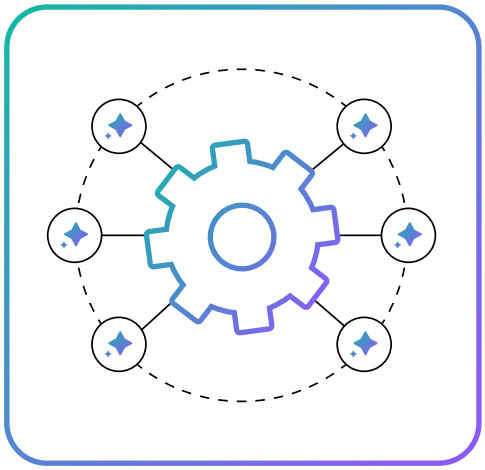
La primera revolución es la de la inteligencia artificial generativa. Desde el lanzamiento de ChatGPT en noviembre de 2022, la capacidad de generar código, procesar lenguaje natural y comunicarse con los sistemas a través del lenguaje humano dejó de ser experimental para convertirse en la nueva normalidad. Por primera vez en la historia del software, es posible programar sistemas masivamente declarando intención en lugar de escribir instrucciones.

La segunda revolución es la agéntica, y es la que menos atención recibe en la prensa a pesar de ser igual o más significativa que la primera. Los sistemas que se construyen hoy cuentan con agentes que acceden al conocimiento, razonan sobre él y actúan de forma autónoma. Esta revolución emerge sobre los cimientos de la primera y define una nueva forma de operar las organizaciones.

Nicolás Jodal, CEO de GeneXus, señala que es la primera vez en la historia que dos revoluciones tecnológicas ocurren de forma simultánea.



Revolución GenAI



Revolución Agéntica

Las últimas grandes transformaciones del sector llegaron en momentos distintos y se sintieron en capas: el smartphone y la conectividad móvil redefinieron la relación entre las personas y la tecnología a partir de 2007, creando industrias enteras que no existían y destruyendo otras que parecían inamovibles. Su impacto fue inmediato y visible para cualquier persona. Luego, a medida que las organizaciones asimilaban ese cambio, llegó la nube: **Amazon Web Services, Google Cloud y Microsoft Azure** transformaron radicalmente cómo se construye y opera el software empresarial, pasando de infraestructura propia, costosa y difícil de escalar, a un servicio bajo demanda, accesible y elástico. Su impacto fue más profundo pero más lento, y dominó la agenda del mundo empresarial durante más de una década.

En cada uno de estos casos, las organizaciones tuvieron tiempo de entender el cambio, reorganizarse y adaptarse antes de que llegara el siguiente. Hoy ese margen no existe. La revolución generativa y la revolución agéntica no se suceden, se superponen. Y lo que hace que esa superposición sea cualitativamente distinta a cualquier cambio anterior es que cada una amplifica a la otra: los agentes son más poderosos porque existe la IA generativa, y la IA generativa es más útil porque existen los agentes que la orquestan. El resultado es la multiplicación de ambos cambios. Esa es la magnitud de lo que está ocurriendo.

*En este whitepaper exploramos esa transformación a través de la visión de Gastón Milano, **CTO de GeneXus** y de **Glob.AI OS en Globant**, quien analiza los cambios que están ocurriendo en el desarrollo de software empresarial, los roles que están emergiendo, los activos que se vuelven estratégicos y las decisiones que las organizaciones deben tomar hoy para posicionarse correctamente frente a esta revolución.*

¿Qué software hay que construir?

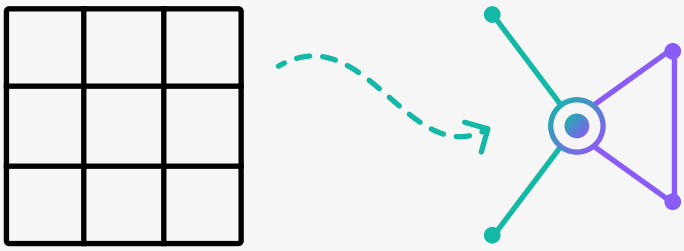
Antes de hablar de velocidad, de agentes, de generación de código, y de cómo construir software en esta nueva era, es importante entender qué tipo de software se debe construir. Esta distinción es, de acuerdo con Gastón Milano, uno de los errores más comunes en el mercado, ya que mientras no se responda el qué, cualquier conversación sobre el cómo carece de fundamento.

Lo que viene es mucho más desafiante, y así lo afirman los líderes globales:

Satya Nadella (Chairman y CEO de Microsoft)

"Toda la lógica de negocio que hoy existe como código estático pasará a ser operada por agentes de inteligencia artificial. Sólo quedarán algunos sistemas de registro, pero todo lo que hoy conocemos como SaaS y software de servicio está experimentando una profunda reinvención".

"Cuando los líderes de la transformación tecnológica global hacen afirmaciones disruptivas, la reacción inicial suele ser el escepticismo. Pero la historia demuestra que a medida que los cambios se van concretando, queda claro que no estaban equivocados", agrega Milano.



El paradigma agéntico: una nueva economía

Existe toda una infraestructura de protocolos y estándares, ninguno de los cuales es inteligencia artificial generativa sino ingeniería pura, que están madurando y definiendo qué significa construir software agéntico.

"Hoy es posible verificar si lo que construiste es verdaderamente agéntico o no. Esto refleja que el concepto de agéntico ya tiene una definición técnica concreta, chequeable y auditable. Esta maduración de protocolos nos recuerda al período de mediados de los 90, cuando internet comenzó a consolidar su stack de protocolos. Del mismo modo, hoy está madurando la infraestructura sobre la cual operará el mundo agéntico, y su consolidación es lo que hace que el concepto de agéntico deje de ser una aspiración para convertirse en algo técnicamente verificable y auditable".

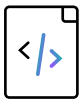
Estos protocolos incluyen:

Protocolos de descubrimiento de agentes
Mecanismos para que un agente encuentre a otro disponible en el ecosistema.

Protocolos de compra y venta
Nuevas formas de transacción económica entre agentes y sistemas.

Protocolos de utilización de herramientas
Estándares para que los agentes accedan y usen capacidades externas de forma estructurada.

Protocolos de navegación web
Mecanismos para que los agentes interactúen con interfaces web de manera autónoma.



Estándares de formato de salida
Definiciones que permiten mayor interoperabilidad y dinamismo entre sistemas.

La caída del paradigma tradicional de software

Estemos de acuerdo o no, varios de los pilares del desarrollo de software tal como lo conocemos están entrando en obsolescencia.

Una aplicación es, en esencia, una predicción. Alguien decidió de antemano qué iba a necesitar la mayoría de los usuarios y lo dejó fijo en una pantalla. Un dashboard es una apuesta sobre qué visualizaciones van a ser útiles. Una página web es un diseño pensado para el usuario promedio. Ninguna se adapta a quien la usa realmente.

"La ingeniería de software resolvió esto de la única manera que podía: construyendo para la media. Funcionó, pero siempre fue una solución de compromiso. La personalización real, la que la industria buscó durante décadas, nunca fue alcanzable con ese modelo porque preparar algo distinto para cada persona era inviable. Los sistemas conversacionales generativos rompen esa lógica. No tienen una respuesta fija para todos: generan una respuesta distinta para cada persona, en función del contexto específico que tienen de ella. Dos personas que hacen la misma pregunta pueden recibir respuestas completamente diferentes, y ambas ser correctas para cada una", explica Milano.

Cuando la velocidad de generación se vuelve indistinguible de la de una conversación humana, la predicción pre armada deja de tener ventaja. En ese momento, personalizar no cuesta más que estandarizar, y la media deja de ser el único camino posible.

¿Hacia dónde vamos?

La experiencia de usuario está migrando hacia interfaces donde el usuario simplemente le pide a un agente lo que necesita, y ese agente lo renderiza en el formato adecuado, audio, video, texto, libro, en tiempo real.

La siguiente tabla resume los cambios más significativos en lo que está perdiendo y ganando relevancia en este nuevo paradigma:

LO QUE PIERDE RELEVANCIA	LO QUE GANA RELEVANCIA
Múltiples aplicaciones	Agentes conversacionales como punto único de acceso
Interfaces de formularios y pantallas estáticas	Interfaces 100% dinámicas renderizadas en runtime
Dashboards y visualizaciones predefinidas	Visualizaciones generadas a medida según el contexto del usuario
Lógica de negocio codificada estáticamente	Lógica de negocio operada por agentes sobre bases de conocimiento
CRUD y pantallas fijas	Agentes accediendo a conocimiento en tiempo real
Pull Requests y revisión línea a línea	Supervisión arquitectónica y validación a posteriori
IDEs tradicionales	Orquestadores de flotas de agentes

Gastón Milano identifica señales concretas de hacia dónde va esta transformación:

1. "El modelo que viene no tiene formularios, no tiene pantallas fijas, no tiene CRUD. Hay agentes que acceden al conocimiento y generan en tiempo real exactamente lo que cada persona necesita, como si Netflix te renderizara el contenido en el momento en que lo pedís, en lugar de tenerlo pregrabado".

2. "Para que eso funcione bien, hay que darles a esos agentes más contexto, más conocimiento declarado sobre cómo deben comportarse, qué reglas seguir y cómo servir mejor a cada cliente. Eso incluye políticas, procesos, roles, reglas de decisión. Todo aquello que hoy vive en la cabeza de las personas, en documentos dispersos o en convenciones no escritas, debe estar codificado para que los agentes puedan actuar con eso".

3. “Cuando los agentes pueden actuar con ese conocimiento, la empresa adquiere propiedades nuevas: todo lo que sucede es versionable, auditable, derivable hacia distintos caminos, operable de forma autónoma y, al mismo tiempo, comprensible para los humanos que necesitan supervisarlo”.

4. “Esto no es una idea aislada. Investigadores como **Andrej Karpathy** (ex Director de IA en Tesla y miembro fundador de OpenAI), y otros están convergiendo en la misma dirección: las empresas del futuro no se programan, se documentan. El conocimiento organizacional deja de ser implícito y pasa a ser un activo explícito, vivo y ejecutable”.

Companies as Code:
El nuevo paradigma organizacional

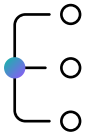
Para Gastón Milano, es fundamental tratar a las empresas como al software: del mismo modo que el código de un sistema puede versionarse, desplegarse y ejecutarse, el conocimiento de una organización puede codificarse y convertirse en el motor sobre el que operan los agentes de inteligencia artificial.

Milano denomina a esta visión **Companies as Code**: una forma de entender la empresa donde todo su conocimiento organizacional, sus políticas, procesos, roles y decisiones, está documentado, versionado y disponible para que los agentes puedan operarlo.

En conjunto, esto representa un cambio de era: pasar de empresas que operan con conocimiento implícito, disperso en la cabeza de sus empleados o enterrado en documentos, a empresas cuyo conocimiento es explícito, vivo y ejecutable. Cuando ese conocimiento está correctamente codificado, se obtienen propiedades que antes eran imposibles de tener todas juntas:

Versionable
Cada cambio en una política o proceso queda registrado. Se puede saber qué decía antes, qué dice ahora y por qué cambió.

Auditable
Todo lo que sucede en la empresa es trazable. No hay decisiones que ocurran fuera del sistema.



Derivable
A partir de ese conocimiento base se pueden explorar caminos alternativos, simular escenarios o tomar decisiones distintas con fundamento.

Operable por agentes
Los agentes no solo consultan ese conocimiento, lo ejecutan. Pueden actuar en nombre de la empresa siguiendo sus propias reglas documentadas.

Comprensible para humanos
Todo esto no sirve si solo lo entienden las máquinas. El sistema tiene que poder explicarse y ser supervisado por las personas.

¿Qué hay que hacer?

La transición hacia una arquitectura agéntica no ocurre de un solo movimiento. Milano identifica tres pasos concretos que las organizaciones deben dar para posicionarse correctamente en esta nueva era.

Paso 1: Codificar el conocimiento

El conocimiento más valioso de una empresa no vive en bases de datos relacionales ni en código. Vive en una wiki: un conjunto de ideas conectadas entre sí a través de una ontología, es decir, una estructura que define cómo se relacionan los conceptos. Cuando ese conocimiento está codificado en ese formato, se vuelve operable. Los agentes pueden navegarlo, interpretarlo y actuar sobre él. Las personas pueden combinarse con esos agentes trabajando sobre la misma base. La empresa deja de depender de que el conocimiento esté en la cabeza de alguien y empieza a tenerlo como un activo real.

“**GeneXus** es un buen ejemplo de esto, porque opera así. Su wiki interno contiene desde cómo hacer una buena presentación hasta cómo responder preguntas difíciles, cómo conducir reuniones, cómo cuidar la salud. En esta nueva era, esos activos se vuelven consumibles, expandibles y escalables por agentes, de formas que antes no eran posibles”.

Para quienes trabajan en el ecosistema GeneXus esto no es una novedad: la Knowledge Base (KB) siempre fue un activo central. Lo que cambia es que ahora ese activo puede conectarse directamente con la capa de ejecución. El conocimiento de la empresa no solo se documenta, se ejecuta.

“La gran mayoría de la industria arranca desde cero. No tiene políticas codificadas, no tiene procesos versionados, no tiene conocimiento organizacional en un formato que los agentes puedan consumir. Quien ya tiene una KB de GeneXus tiene algo que el resto todavía no sabe cómo construir. Esa distancia no es menor: es la diferencia entre tener el activo estratégico del próximo ciclo tecnológico y estar empezando a entender qué es”.

En este contexto, las KBs presentan características que las hacen especialmente valiosas en el mundo agéntico:

Son versionables
Al igual que el código, se pueden rastrear sus cambios históricos.

Son consumibles por agentes
Con GeneXus for Agents, las KBs pueden ser accedidas y utilizadas por agentes de IA de manera directa.

Son guiadas por humanos
Los humanos proveen la intención, el juicio y la identidad organizacional; los agentes ejecutan.

"Desarrollar software nunca fue el objetivo central de una organización, sino el medio para construir los activos digitales que le permiten operar y servir a sus clientes. La diferencia es que hoy ese medio tiene que estar enmarcado dentro de la misma lógica que rige el resto de la organización: una base de conocimiento como fuente de verdad y agentes que operan sobre ella. Cuando eso ocurre, el desarrollo de software deja de ser una actividad técnica aislada y se convierte en una operación más de la empresa, tan gestionable, versionable y auditable como cualquier otra".

Gastón Milano

Paso 2: Modelar Bases de Conocimiento + Agentes

La arquitectura del software del futuro, según Milano, se articula en torno a dos elementos fundamentales: **Una Base de Conocimiento (KB)** que codifica todo el saber de la organización, sus procesos, políticas, datos, reglas de negocio y contexto; y **agentes de IA que operan sobre esa KB** para ejecutar tareas, tomar decisiones y generar valor.

El modelo redefine los roles de cada actor en la organización:

Rol	Función	Herramienta
Humanos	Dirigen la intención, la ideación y la identidad empresarial. Toman decisiones estratégicas.	Base de conocimiento + juicio humano
IA Generativa	Ejecuta tareas, genera contenido, procesa datos.	LLMs + harness de ingeniería
Agentes	Operan procesos completos de forma autónoma, consumen la KB.	Protocolos agénticos + KB + herramientas
AI Architect	Diseña el stack, orquesta agentes, define niveles de autonomía.	Orquestadores + plataformas multi-agente

> El tiempo humano ya no se gasta escribiendo código; se gasta definiendo el qué, construyendo la base de conocimiento y orquestando los agentes que ejecutan. Gastón Milano lo ilustra con su propia práctica: desde noviembre de 2025 no escribe una sola línea de código. Su trabajo es orquestar.

Paso 3: La economía de tokens

Uno de los aspectos prácticos que Milano destaca es la necesidad de gestionar eficientemente el costo de los tokens. Las grandes empresas no tienen los mismos subsidios que los usuarios individuales; el costo de operar flotas de agentes a escala empresarial puede ser muy significativo y se convierte en un factor estratégico por derecho propio.

Por esta razón surge el concepto de tokenomics: la economía de tokens. Las skills y el conocimiento codificado de manera precisa permiten reducir drásticamente la cantidad de tokens consumidos por cada operación, bajando los costos y aumentando la eficiencia. Esto convierte a la calidad de la base de conocimiento en un factor económico crítico. Una KB bien estructurada no solo hace que los agentes trabajen mejor: hace que trabajen más barato. Y a escala, esa diferencia define la viabilidad del modelo.

El problema del vibe coding y la necesidad de supervisión

El vibe coding es el término acuñado por **Andrej Karpathy** en febrero de 2025, para referirse a la forma de desarrollar software donde el programador le describe a una IA lo que quiere en lenguaje natural, acepta el código generado sin revisarlo en detalle, pega los errores directamente de vuelta al modelo para que los resuelva, y deja que el sistema crezca de forma orgánica, muchas veces más allá de lo que el propio desarrollador comprende completamente.

Milano describe este problema con una metáfora visual: "El vibe coding puede verse como un queso suizo: cada rebanada tiene agujeros. Cuando se combinan muchas rebanadas, los agujeros se acumulan y los errores crecen de forma exponencial. Esto puede funcionar para prototipos rápidos o proyectos personales, pero en sistemas empresariales de misión crítica, donde el código tiene que ser predecible, mantenible y auditable, el vibe coding sin una base de conocimiento sólida detrás produce resultados poco confiables".

La solución a este problema tiene varias capas:

Reducción de agujeros con mejores herramientas
El ingreso de herramientas como Claude Code, Codex, Gemini CLI y otros harnesses ha reducido significativamente la cantidad de errores en la generación automática de código.

Contexto real y preciso
La única forma de reducir aún más los errores es proveer a los agentes con el contexto correcto y completo, que proviene de una base de conocimiento bien estructurada.

Supervisión arquitectónica humana
Los humanos deben seguir ideando las arquitecturas correctas y supervisando los resultados para mantener software predecible y funcional.

El vibe coding sin contexto es el síntoma. La base de conocimiento es la solución. Y el AI Architect es quien une ambas cosas.

AI Architect: El nuevo rol estratégico

El AI Architect es el rol que emerge cuando la generación de código deja de ser el cuello de botella. Es quien define el qué y el por qué, y quien orquesta los agentes que se encargan del resto.

La aparición del AI Architect no es casual: es la consecuencia directa de tres factores que convergieron al mismo tiempo:

1. Los modelos alcanzaron un umbral de competencia suficiente para tareas bien definidas.

2. La ingeniería alrededor del modelo maduró, con herramientas como Claude Code, Codex y Gemini CLI que resolvieron cómo hacer el loop agéntico, cómo darles las herramientas correctas a los agentes, cómo comprimir la memoria y cómo gestionar el file system.

3. Y los agentes trabajan en paralelo las 24 horas, produciendo volúmenes de código que ningún equipo humano puede igualar.

El resultado es que el cuello de botella dejó de ser la generación de código para convertirse en la definición de qué construir y con qué conocimiento hacerlo. Ahí es donde opera el AI Architect.

Sus responsabilidades centrales son:

- **Diseñar el nivel de autonomía de cada componente del sistema**
  Determinar qué partes requieren supervisión humana activa, cuáles pueden operar con sugerencias al humano, y cuáles pueden funcionar de manera completamente autónoma. Esta decisión, que parece técnica, es en realidad estratégica: define cuánto control retiene la organización y dónde delega en los agentes.

- **Orquestar flotas de agentes**
  Definir cómo se coordinan decenas o cientos de agentes trabajando en paralelo, cómo se les asignan tareas y cómo se gestionan sus resultados. Gastón Milano opera habitualmente con 30, 40 o 50 agentes corriendo simultáneamente durante la noche. La orquestación de esa flota es su trabajo central.

- **Diseñar el stack tecnológico completo**
  Elegir qué LLMs usar, qué plataformas de orquestación, qué bases de datos, cómo combinarlos. El AI Architect puede combinar Claude, Codex, Gemini o cualquier otro modelo según lo que cada situación requiera, sin estar atado a un proveedor.

- **Garantizar la calidad sin revisión línea a línea**
  Es imposible revisar 100.000 líneas de código generadas en un fin de semana por agentes. El AI Architect diseña mecanismos de validación sistémica y supervisión a posteriori que reemplazan los flujos de revisión tradicionales.

- **Mantener la visión end-to-end**
  Es imposible revisar 100.000 líneas de código generadas en un fin de semana por agentes. El AI Architect diseña mecanismos de validación sistémica y supervisión a posteriori que reemplazan los flujos de revisión tradicionales.

Por otra parte, el AI Architect también define los distintos niveles de automatización que coexisten en un mismo sistema:

- **Intervención humana directa**
  Tareas que todavía requieren que una persona actúe sin asistencia de agentes: despliegues de infraestructura, configuración de claves de seguridad, instalación de dispositivos IoT, robótica física.

- **Asistencia con sugerencias**
  El agente propone, el humano decide. El sistema amplifica la capacidad humana sin reemplazar el juicio.

- **Humano como auditor eventual**
  El agente actúa de forma autónoma y el humano revisa excepciones y resultados, no cada paso del proceso.

- **Autonomía completa**
  El agente opera sin intervención humana en el ciclo normal. Un ejemplo concreto: cuando GeneXus genera el código de una reorganización de base de datos, nadie lo revisa manualmente. Se confía en que el sistema funciona, porque el contexto y las reglas están correctamente definidos.

Implicaciones para la Comunidad GeneXus

Para Milano, en el mercado actual, todos los usuarios de GeneXus son naturalmente ‘AI Architect’. ¿Por qué? Porque el usuario GeneXus ya tiene incorporado el mindset correcto:

- Sabe que primero hay que construir una base de conocimiento.
- Sabe que hay que tener los datos correctos antes de construir.
- Entiende el concepto de generar desde especificaciones.
- Piensa de forma end-to-end sin estar fragmentado en silos tecnológicos.

Además, los usuarios GeneXus nunca revisaron pull requests ni siguieron flujos de GitHub. Lo que en otro contexto podría parecer una limitación resulta ser exactamente la mentalidad correcta para este momento, porque esos flujos de revisión tradicionales están muriendo.

“No hay forma de revisar 100.000 líneas de código generadas en un fin de semana por agentes, y la industria entera está buscando cómo resolver eso. Los usuarios de GeneXus ya operaban sin esa dependencia. Sin embargo, hay algo que la comunidad todavía necesita incorporar, específicamente el Impact Analysis de datos. Ahora existe una tendencia a tirarlo directamente a Claude Code sin ese análisis previo, y eso es un error que se paga caro. Inculcar esa práctica en la comunidad es una tarea pendiente”.

Milano anticipa que habrá bases de conocimiento verticales por industria. Esto significa que los partners con expertise sectorial tendrán una oportunidad única de construir activos de conocimiento profundos y diferenciados para sus industrias, que luego se convertirán en ventajas competitivas difíciles de replicar.

GeneXus ya está trabajando en asegurar el "future proofing" de las bases de conocimiento existentes de sus clientes. El objetivo es que esas KBs puedan evolucionar hacia el mundo agéntico sin perder el valor acumulado. La estrategia incluye:

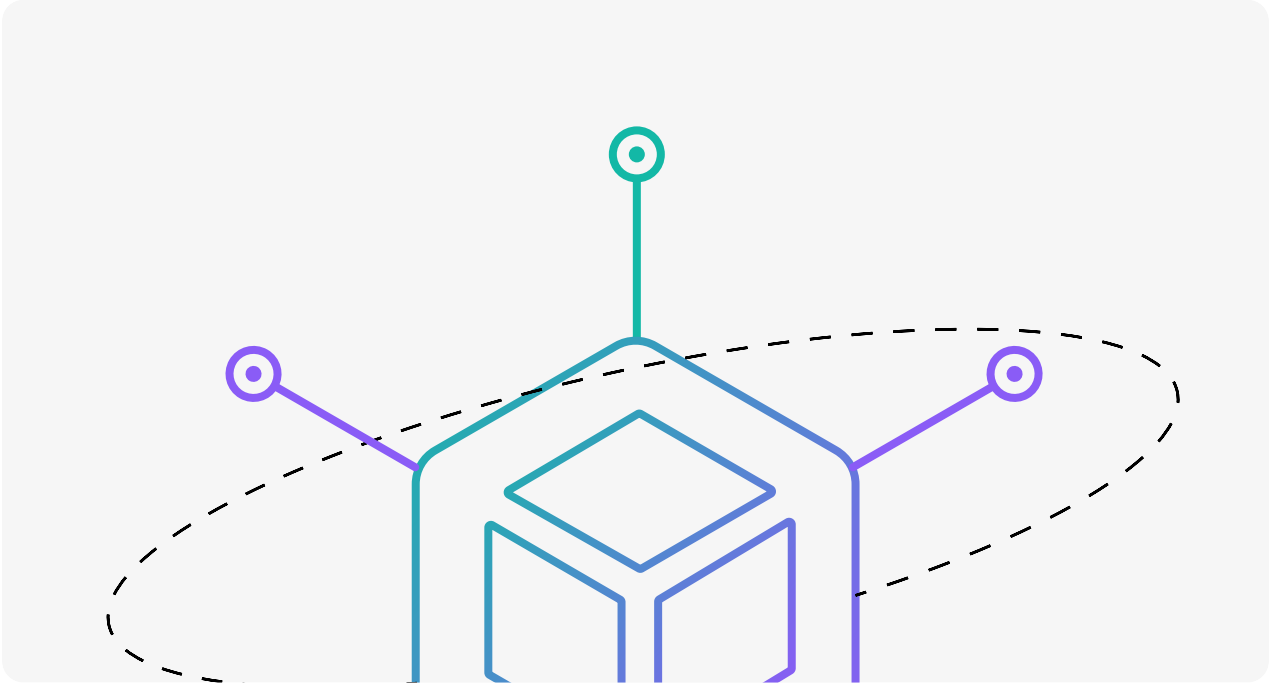
- GeneXus for Agents: herramienta que hace las KBs consumibles por agentes externos.
- Segunda iteración hacia la agéntica completa: una versión evolucionada de las KBs que permita que los agentes trabajen directamente con el conocimiento existente de los clientes.
- Herramientas de orquestación que permiten combinar GeneXus con otros generadores y agentes del mercado.

Volver al futuro

GeneXus fue fundada en Uruguay en 1988 con una idea que entonces parecía radical: que el conocimiento del negocio debía ser el centro del desarrollo de software, y que los sistemas debían generarse a partir de ese conocimiento. Mientras el resto de la industria construía código, GeneXus construía bases de conocimiento. Mientras otros optimizaban cómo escribir más rápido, GeneXus trabajaba en eliminar la escritura manual de código generando sistemas a partir del conocimiento declarado.

Hoy, en el momento en que la industria global converge hacia exactamente lo que GeneXus lleva años practicando, el modelo evoluciona una vez más: plataforma más servicios, con bases de conocimiento verticales por industria, implementación, governance y operación continua. Un modelo recurrente donde el conocimiento codificado es el activo central y los agentes son el motor de ejecución.

Lo que durante años fue una forma particular de trabajar resulta ser hoy la arquitectura correcta para la era agéntica.



El momento de actuar es ahora

El mundo está madurando a una velocidad extraordinaria. Los protocolos agénticos se consolidan. Las herramientas evolucionan. Y la brecha entre las organizaciones que ya están construyendo sus bases de conocimiento y las que todavía están evaluando si hacerlo se agranda cada día.

Las organizaciones que lideren la próxima era del software empresarial son las que entienden antes que el activo estratégico de esta era es el conocimiento codificado, y las que tienen la guía correcta para construirlo y operarlo con agentes.

Ese es exactamente el trabajo que hacemos. Desde Globant y GeneXus acompañamos a organizaciones en esa transición: desde la construcción de su primera base de conocimiento hasta el diseño de arquitecturas agénticas completas, la selección del stack tecnológico y la formación de los equipos que van a operar en este nuevo paradigma.

Si tu organización está buscando dar ese paso, el momento de empezar la conversación es ahora.

Contáctanos