Building Applications with AI Agents

Designing and Implementing Multiagent Systems

Michael Albada

November 22, 2025

Capítulo 1: Introducción a los Agentes

Nos encontramos en medio de una profunda transformación tecnológica, impulsada por la llegada de los **agentes autónomos**. Estos no son simplemente programas, sino sistemas de software inteligentes dotados de la capacidad de ejercer un razonamiento independiente, tomar decisiones por sí mismos e interactuar eficazmente dentro de entornos dinámicos y cambiantes. A diferencia del software tradicional, que opera siguiendo instrucciones rígidas, los agentes autónomos están diseñados para interpretar contextos, adaptarse a nuevos escenarios y ejecutar acciones sofisticadas con una mínima supervisión humana.

Definiendo Agentes de IA

Un **agente autónomo** es, en esencia, un sistema inteligente diseñado para analizar datos de manera independiente, interpretar el entorno que lo rodea y tomar decisiones informadas por el contexto. Sin embargo, a medida que el término "agente" ha ganado popularidad, su significado se ha diluido. Es crucial entender que la agencia existe en un espectro.

En un extremo se encuentran los **verdaderos agentes autónomos**, que demuestran una capacidad de decisión significativa, un razonamiento basado en el contexto y comportamientos adaptativos. En el otro extremo, muchos sistemas etiquetados como "agentes" en realidad se limitan a ejecutar guiones deterministas o flujos de trabajo estrictamente controlados. Por lo tanto, la prueba de fuego para un verdadero agente es si demuestra una toma de decisiones real en lugar de limitarse a seguir un guion estático.

El vertiginoso avance de los agentes autónomos se debe principalmente a las innovaciones en los **modelos de fundación** y el **aprendizaje por refuerzo**. Mientras que los casos de uso tradicionales de los modelos de fundación se centraban en generar resultados legibles por humanos, los avances más recientes les permiten generar salidas estructuradas, como firmas de funciones y selecciones de parámetros. Esto es fundamental, ya que permite que los frameworks de orquestación ejecuten estas funciones, dando a los agentes la capacidad de buscar datos, manipular sistemas externos y realizar acciones concretas en el mundo real.

A lo largo de este libro, utilizaremos el término **"sistema agéntico"** para describir no solo al agente en sí, sino a toda la funcionalidad de soporte que le permite operar eficazmente. Esto incluye las herramientas a las que tiene acceso, su memoria, el modelo de fundación subyacente, la capa de orquestación y toda la infraestructura de apoyo.

Mirando hacia el futuro, la aparición de protocolos como el *Model Context Protocol* y el *Agent-to-Agent Protocol* permitirá a los agentes utilizar herramientas remotas y colaborar entre sí para resolver problemas complejos. Esto abre enormes oportunidades para una automatización sofisticada, pero también conlleva la profunda responsabilidad de diseñar, medir y gestionar estos sistemas de forma reflexiva, garantizando que sus acciones se alineen con los valores humanos y que operen de forma segura.

La Revolución del Preentrenamiento

El Machine Learning (ML) tradicional, aunque poderoso, siempre ha estado limitado por la cantidad y calidad de los datos disponibles. Los profesionales del ML a menudo pasan la mayor parte de su tiempo no en el entrenamiento de modelos, sino en la ardua tarea de recolectar y limpiar los datos necesarios.

La increíble historia de éxito de los modelos generativos ha puesto fin a esta era. Entrenados con volúmenes masivos de datos, estos modelos preentrenados pueden adaptarse a una amplia gama de tareas sin necesidad de entrenamiento adicional. Esto representa un cambio de paradigma que reduce drásticamente el costo y la complejidad de construir aplicaciones habilitadas para IA y ML.

Los recientes avances en los modelos de lenguaje grandes (LLMs), como GPT-5, Claude de Anthropic o Gemini de Google, han elevado aún más el rendimiento en tareas difíciles. Estos modelos de fundación ofrecen robustas capacidades de comprensión del lenguaje natural y generación de contenido, mejorando la funcionalidad de los agentes a través de varias capacidades clave. Estas incluyen la **comprensión del lenguaje natural** para interpretar intuitivamente las peticiones de los usuarios, la **interacción consciente del contexto** para mantener respuestas relevantes a lo largo de interacciones prolongadas, y la **generación de contenido estructurado** para producir texto, código y otros formatos esenciales para tareas analíticas y creativas.

Integrados con sofisticados frameworks de orquestación, estos modelos pueden realizar tareas prácticas como la **interpretación contextual y la toma de decisiones** para navegar situaciones ambiguas, el **uso de herramientas** para llamar a otro software, la **planificación adaptativa** para ejecutar acciones complejas de múltiples pasos, y la **automatización de tareas rutinarias**, liberando a los trabajadores humanos para que se centren en actividades más matizadas.

Tipos de Agentes

La popularidad del término "agente" ha llevado a su aplicación a una amplia gama de sistemas de IA. Para clarificar esta confusión, la información los categoriza en siete tipos prácticos, que reflejan cómo se están utilizando estas tecnologías hoy en día:

Lista para examen – Tipos de agentes (7):

- **Agentes de tareas empresariales**: Automatizan flujos de trabajo de negocio predefinidos, como la automatización robótica de procesos de UiPath o las integraciones de aplicaciones de Zapier. Ejecutan secuencias de acciones deterministas con un razonamiento contextual mínimo.

- **Agentes conversacionales**: Esta categoría incluye chatbots y agentes de servicio al cliente que interactúan con los usuarios a través de interfaces de lenguaje natural. Están optimizados para la gestión de diálogos y el reconocimiento de intenciones.

- **Agentes de investigación**: Su función es realizar tareas de recopilación, síntesis y resumen de información. Escanean documentos o la web para proporcionar resultados estructurados que ayuden a los analistas humanos. Perplexity AI es un buen ejemplo.

- **Agentes de análisis**: Se centran en interpretar conjuntos de datos estructurados para generar ideas, cuadros de mando e informes. A menudo se integran con almacenes de datos empresariales, permitiendo a los usuarios consultar datos complejos en lenguaje natural, como es el caso de Power BI Copilot.

- **Agentes desarrolladores**: Herramientas como GitHub Copilot asisten a los desarrolladores generando, refactorizando y explicando código. Se integran profundamente en los flujos de trabajo de los IDE para aumentar la productividad.

- **Agentes de dominio específico**: Están afinados para dominios profesionales especializados, como el legal (Harvey) o el médico (Hippocratic AI). Combinan conocimientos específicos del dominio con flujos de trabajo estructurados.

- **Agentes de navegación web**: A diferencia de la automatización robótica tradicional que sigue pasos prescritos, estos agentes combinan la comprensión del lenguaje, la percepción visual y la planificación dinámica para interactuar con sitios web de forma autónoma.

Además de estos, se espera que los **agentes de voz** y los **agentes de video** ganen una adopción significativa. El enfoque principal de este libro, sin embargo, recaerá en los agentes construidos en torno a modelos de lenguaje que utilizan texto y código.

Selección del Modelo

La proliferación de modelos potentes, tanto de proveedores comerciales como de la comunidad de código abierto, crea un dilema: ¿cómo elegir el modelo adecuado para potenciar un sistema agéntico?

La buena noticia es que la competencia está impulsando la innovación, el rendimiento y la reducción de costos. La verdad es que no hay una respuesta única. Un punto de

partida muy razonable es simplemente utilizar el último modelo de propósito general de un proveedor líder como OpenAI o Anthropic. Como se puede ver en la Tabla 1-1, estos modelos ofrecen un rendimiento sólido "de fábrica" y requieren poca personalización para muchas aplicaciones.

Modelo	Puntuación media	MMLU-Pro	GPQA	IFEval	WildBench	Omni-MATH
GPT-5 mini (2025-08-07)	0.819	0.835	0.756	0.927	0.855	0.722
o4-mini (2025-04-16)	0.812	0.820	0.735	0.929	0.854	0.720
o3 (2025-04-16)	0.811	0.859	0.753	0.869	0.861	0.714
GPT-5 (2025-08-07)	0.807	0.863	0.791	0.875	0.857	0.647
Qwen3 235B...	0.798	0.844	0.726	0.835	0.866	0.718
Grok 4 (0709)	0.785	0.851	0.726	0.949	0.797	0.603
Claude 4 Opus (20250514)	0.780	0.875	0.709	0.849	0.852	0.616
gpt-oss-120b	0.770	0.795	0.684	0.836	0.845	0.688
Kimi K2 Instruct	0.768	0.819	0.652	0.850	0.862	0.654
Claude 4 Sonnet...	0.766	0.843	0.706	0.840	0.838	0.602

Tabla 1-1: *Tabla de clasificación HELM Core Scenario (agosto 2025). Rendimiento comparativo de los 10 mejores modelos en tareas de razonamiento y evaluación.*

Sin embargo, los modelos más grandes no siempre son la opción más eficiente. Para tareas bien definidas, de baja latencia o sensibles al costo, modelos mucho más pequeños pueden ofrecer un rendimiento casi equivalente a una fracción del costo. Esto ha llevado a una tendencia creciente: la selección automatizada de modelos, donde los sistemas enrutan las consultas simples a modelos rápidos y económicos, reservando los modelos grandes y costosos para el razonamiento más complejo. Este futuro **multimodelo** es casi una certeza, por lo que diseñar sistemas con flexibilidad desde el principio dará sus frutos

más adelante.

De Operaciones Síncronas a Asíncronas

Los sistemas de software tradicionales operan de forma **síncrona**, ejecutando tareas paso a paso y esperando a que cada acción termine antes de comenzar la siguiente. Este enfoque, aunque sencillo, puede generar ineficiencias significativas, especialmente cuando se espera una entrada externa o se procesan grandes volúmenes de datos.

Por el contrario, los agentes autónomos están diseñados para una operación **asíncrona**. Pueden gestionar múltiples tareas en paralelo, adaptarse rápidamente a nueva información y priorizar acciones de forma dinámica según las condiciones cambiantes. Este procesamiento asíncrono mejora drásticamente la eficiencia.

Las implicaciones prácticas de este cambio son sustanciales. Por ejemplo, los correos electrónicos pueden llegar con borradores de respuesta ya preparados, o los ingenieros de software pueden recibir tickets de problemas acompañados de código sugerido para solucionarlos. En cada caso, los agentes no solo están acelerando los flujos de trabajo, sino que están cambiando la naturaleza misma del trabajo. Esta evolución transforma los roles humanos de ejecutores de tareas a **gestores de tareas**, permitiendo que las personas se centren en la supervisión estratégica y la toma de decisiones de alto valor.

Aplicaciones Prácticas y Casos de Uso

Para mantener este libro anclado en ejemplos claros y específicos, se hará referencia frecuente a siete agentes de ejemplo del mundo real, cuyos sistemas de evaluación están disponibles públicamente. Estos son: un agente de atención al cliente, un agente de servicios financieros, un agente de admisión de pacientes, un agente de soporte técnico de TI, un agente de revisión de documentos legales, un agente analista de SOC y un agente de cadena de suministro.

Lista para examen – Ejemplos de agentes del libro:

1. Agente de atención al cliente.
2. Agente de servicios financieros.
3. Agente de admisión de pacientes (triaje en salud).
4. Agente de soporte técnico de TI (help desk).
5. Agente de revisión de documentos legales.
6. Agente analista del Centro de Operaciones de Seguridad (SOC).

7. Agente de cadena de suministro y logística.

Flujos de Trabajo y Agentes

Elegir entre un simple script, un flujo de trabajo determinista, un chatbot tradicional o un agente autónomo completo puede ser la diferencia entre una solución elegante y un sistema sobrediseñado y difícil de mantener. La elección correcta depende de la variabilidad de las entradas, la complejidad del razonamiento requerido y las restricciones de rendimiento.

- **Cuándo usar código simple**: Si las entradas son totalmente predecibles y cada posible salida puede ser descrita de antemano (como analizar un archivo de log con un formato fijo), el código tradicional es más rápido, barato y fácil de probar.

- **Cuándo usar un flujo de trabajo**: Si la lógica puede expresarse en un conjunto finito de pasos o ramas y se necesita un control auditable sobre cada paso (como procesar facturas de un conjunto conocido de proveedores), un motor de flujos de trabajo ofrece un control más claro que un LLM.

- **Cuándo usar un chatbot o RAG**: Si la necesidad principal es permitir a los usuarios hacer preguntas sobre una base de conocimiento (como un manual de producto), un sistema de Generación Aumentada por Recuperación (RAG) es la opción correcta. Estos sistemas recuperan información relevante y generan respuestas, pero no deciden ni ejecutan acciones de seguimiento de forma autónoma.

- **Cuándo usar un agente autónomo**: Se llega a los agentes autónomos cuando ni el código simple, ni los flujos de trabajo rígidos, ni el RAG son suficientes. Esto ocurre en situaciones donde las entradas son no estructuradas, novedosas o muy variables, y se requiere una planificación dinámica de múltiples pasos o un aprendizaje continuo. Un ejemplo es un centro de soporte que recibe correos electrónicos de formato libre con problemas diversos y complejos.

Resumen rápido de enfoques (para examen):

Enfoque	Cuándo usarlo en una frase
Código tradicional	Entradas totalmente predecibles, necesidad de latencia muy baja y lógica 100% determinista.
Flujo de trabajo (Workflow)	Procesos con pasos y ramas conocidos de antemano, donde importa la auditoría paso a paso.
Chatbot / Sistema RAG	Responder preguntas sobre una base de conocimiento, sin ejecutar acciones complejas de seguimiento.

Enfoque	Cuándo usarlo en una frase
Agente autónomo	Entradas no estructuradas o novedosas, planificación dinámica de múltiples pasos y aprendizaje continuo.

La siguiente tabla resume estas diferencias clave.

Característica	Código Tradicional	Flujo de Trabajo (Workflow)	Agente Autónomo
Estructura de Entrada	Esquemas totalmente predecibles	Mayormente predecible con ramas finitas	Altamente no estructurada o entradas novedosas
Explicabilidad	Transparencia total; fácilmente auditable	Rastro de auditoría explícito rama por rama	Componentes de caja negra que requieren herramientas adicionales
Latencia	Latencia ultra baja	Latencia moderada	Latencia más alta
Adaptabilidad y Aprendizaje	Ninguna	Limitada	Alta (aprendizaje a partir de la retroalimentación)

Tabla 1-2: Distinguiendo flujos de trabajo y agentes del código tradicional.

Principios para Construir Sistemas Agénticos Eficaces

La creación de agentes autónomos exitosos requiere un enfoque que priorice cinco principios clave:

Lista para examen – Principios clave de sistemas agénticos:

1. **Escalabilidad**: Asegurar que los agentes puedan manejar cargas de trabajo crecientes mediante arquitecturas distribuidas y una optimización eficiente de los recursos.

2. **Modularidad**: Diseñar agentes con componentes independientes e intercambiables conectados a través de interfaces claras, lo que simplifica el mantenimiento y facilita la adaptación.

3. **Aprendizaje continuo**: Equipar a los agentes con mecanismos para aprender de la experiencia y de la retroalimentación del usuario, permitiendo que su rendimiento evolucione y se mantenga relevante.

4. **Resiliencia**: Desarrollar arquitecturas robustas capaces de manejar con gracia errores, amenazas de seguridad y condiciones inesperadas para garantizar operaciones fiables.

5. **A prueba de futuro**: Construir sistemas en torno a estándares abiertos e infraestructura escalable para adaptarse rápidamente a las tecnologías emergentes y a las expectativas cambiantes de los usuarios.

Organización para el Éxito en la Construcción de Sistemas Agénticos

La facilidad con la que se puede experimentar con modelos de fundación a través de APIs simples ha estimulado una amplia experimentación. Sin embargo, esto a menudo conduce a la fragmentación. El éxito requiere equilibrar la flexibilidad para la experimentación con la alineación suficiente para la escalabilidad. En las fases iniciales, las organizaciones deben fomentar la exploración. Con el tiempo, a medida que surgen patrones exitosos, la alineación estratégica se vuelve crítica.

Adoptar estándares abiertos como OpenAPI ayuda a evitar la dependencia de un proveedor y garantiza la flexibilidad futura. Además, es crucial compartir eficazmente el conocimiento adquirido de los experimentos, tanto exitosos como fallidos, para acelerar el aprendizaje organizativo.

Frameworks Agénticos

Existen numerosos frameworks para el desarrollo de agentes autónomos. Entre los más destacados se encuentran:

- **LangGraph**: Un framework de orquestación modular basado en grafos dirigidos. Es ideal para construir sistemas robustos con un control de flujo explícito e inspeccionable. *Trade-off:* ofrece mucho control y robustez, pero exige algo más de configuración y lógica personalizada por parte del equipo técnico.

- **AutoGen**: Ofrece una potente orquestación multiagente, asignación dinámica de roles e interacción flexible basada en mensajes, lo que lo hace ideal para sistemas de investigación que involucran diálogos complejos entre agentes. *Trade-off:* muy potente para escenarios complejos, pero puede resultar excesivo o más complejo de lo necesario para casos de uso simples.

- **CrewAI**: Es fácil de aprender y permite una configuración rápida para la creación

de prototipos. Es una excelente opción para los desarrolladores que desean iniciarse rápidamente en agentes prácticos centrados en el ser humano. *Trade-off:* facilita el arranque rápido, pero ofrece menos control fino y personalización profunda que frameworks más "bajos nivel" como LangGraph.

- **OpenAI Agents SDK**: Proporciona una profunda integración con el ecosistema de OpenAI, lo que lo hace perfecto para los equipos que ya utilizan la API de OpenAI y buscan una forma rápida de construir agentes seguros que utilicen herramientas. *Trade-off:* muy cómodo si ya estás en el ecosistema de OpenAI, pero te acopla fuertemente a su infraestructura (menos portable a otros entornos).

Aunque cada framework tiene sus ventajas, este libro se centrará principalmente en **LangGraph**, elegido por su enfoque directo pero potente para el desarrollo de sistemas de agentes.

Conclusión

Los agentes autónomos representan un desarrollo transformador en la inteligencia artificial, capaces de realizar tareas complejas y dinámicas con un alto grado de autonomía. Este capítulo ha esbozado los conceptos fundamentales de los agentes, ha destacado sus avances sobre los sistemas de ML tradicionales y ha discutido sus aplicaciones prácticas y limitaciones. A medida que profundicemos en el diseño y la implementación de estos sistemas, quedará claro que su integración reflexiva en diversos dominios tiene el potencial de impulsar una innovación y eficiencia significativas.