



Introducción a LLMs y Agentes

Arquitecturas y Aplicaciones en Ingeniería de Software

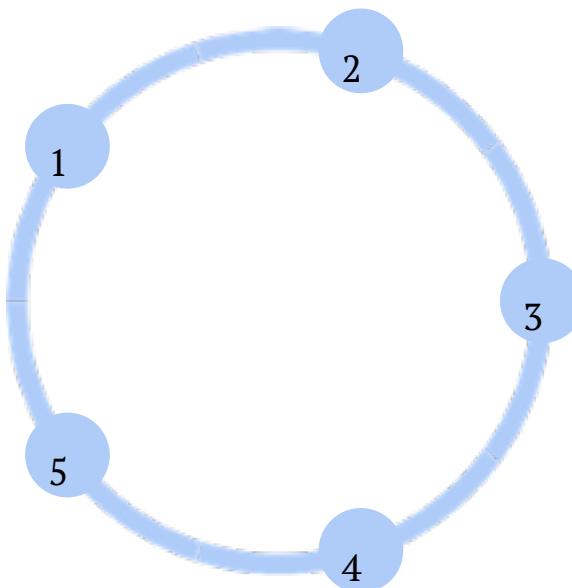
IV ESCUELA DE INFORMÁTICA

J. ANDRÉS DÍAZ PACE
2025

Sistemas Multi-agente

Agentes especializados
Entidades con roles específicos y habilidades diferenciadas (investigador, crítico, sintetizador, etc.)

Mecanismos de toma de decisiones
Procesos para resolver conflictos y determinar acciones basadas en múltiples inputs



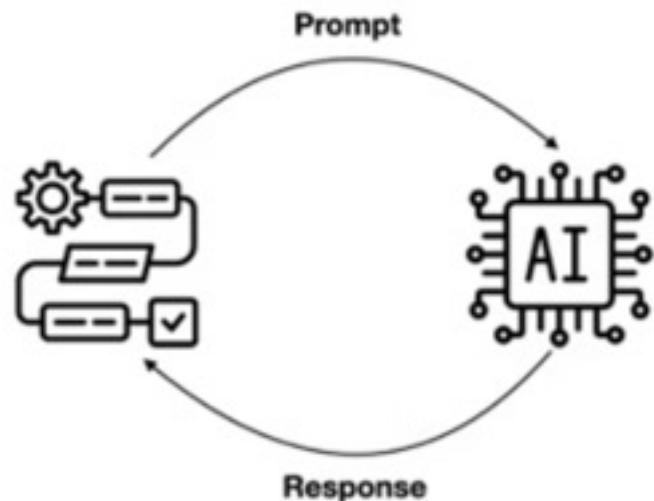
Protocolos de comunicación
Mecanismos estructurados para el intercambio de información entre agentes

Orquestador central
Componente que coordina las interacciones y el flujo de trabajo entre agentes

Memoria compartida
Repositorio común de información accesible para todos los agentes del sistema

Ingeniería de LLMs

El objetivo es construir aplicaciones de software soportadas/potenciadas por LLMs (por ej., chatbots, generadores, etc.)



Architecting LLM applications

The language model is just one part of the technical architecture



Adapted from: [Machine Learning: The High Interest Credit Card of Technical Debt \(Google\)](#)

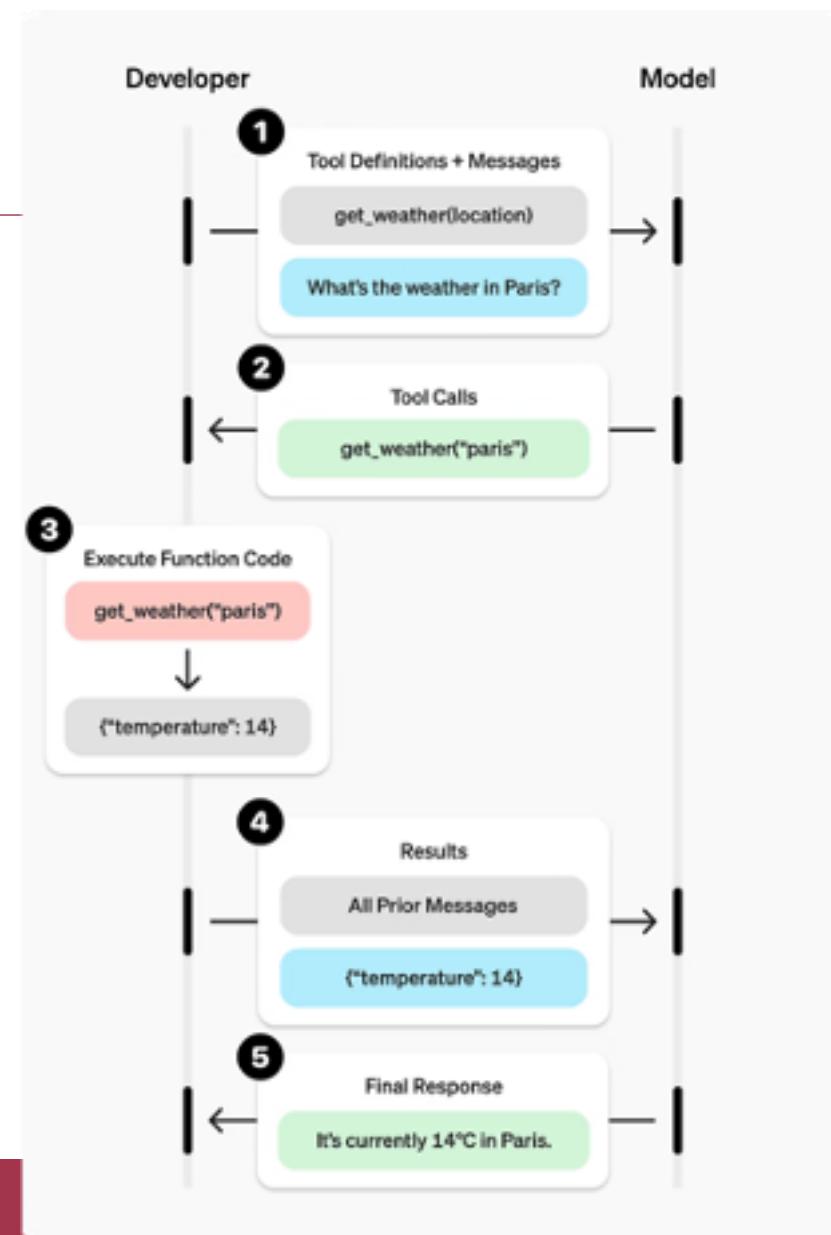
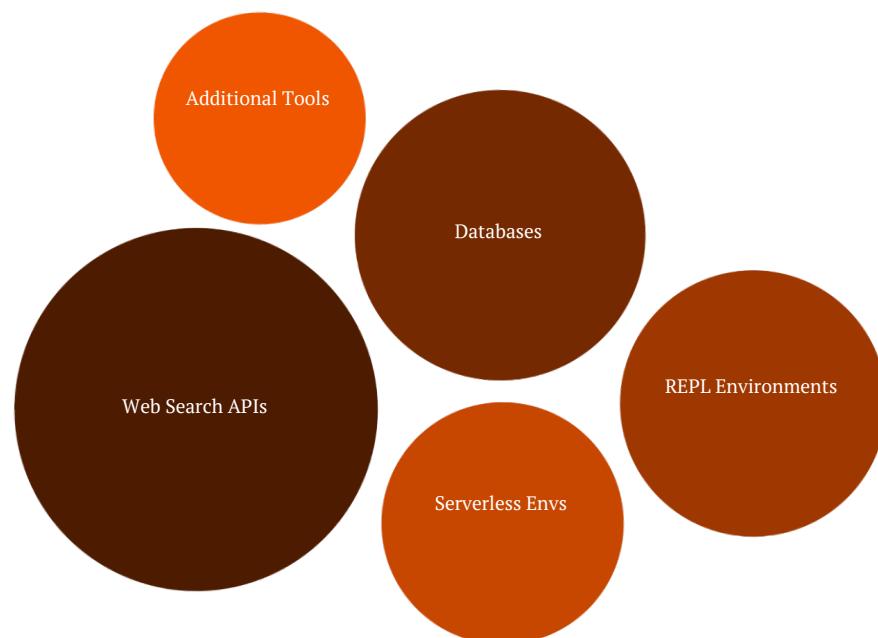
<https://martinfowler.com/articles/engineering-practices-llm.html>

¿Qué sucede si un LLM supiera de antemano que posee herramientas disponibles para realizar determinadas tareas?

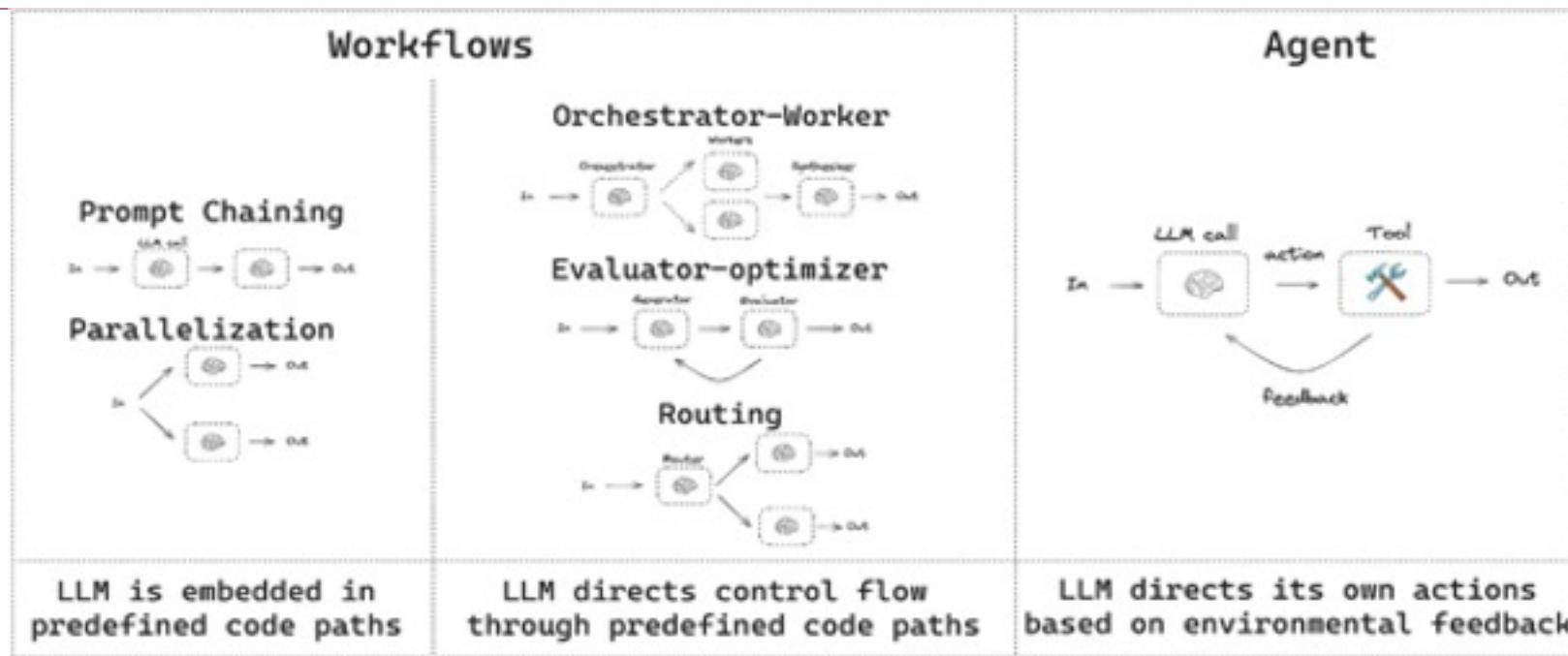


Tools / Function Calls

Dar a los modelos acceso a nuevas funcionalidades y datos que pueden usar para seguir instrucciones y responder a las indicaciones.



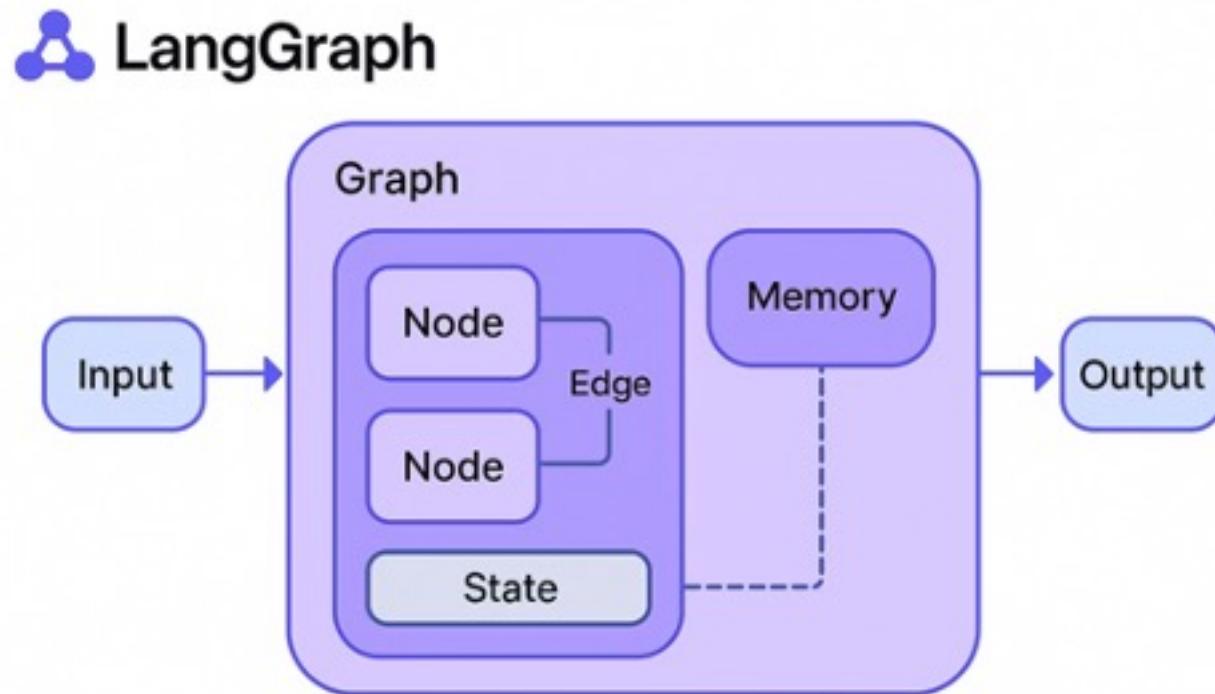
Workflows y Agentes



Los workflows son sistemas donde los LLMs y tools se orquestan a través de “caminos de Código” predefinidos.

Los agentes, por otro lado, son sistemas donde los LLMs dinámicamente dirigen sus procesos y uso de tools, manteniendo control sobre cómo ejecutan las tareas.

Workflows en Langgraph



Conceptos de Langgraph



Prebuilt

Componentes predefinidos para acelerar el desarrollo



Add Tools

Integración de APIs externas y utilidades, incluyendo MCP



Add Memory

Persistencia de contextos entre diferentes interacciones



Human-in-the-Loop

Aprobar o refinar salidas del modelo



Customize State

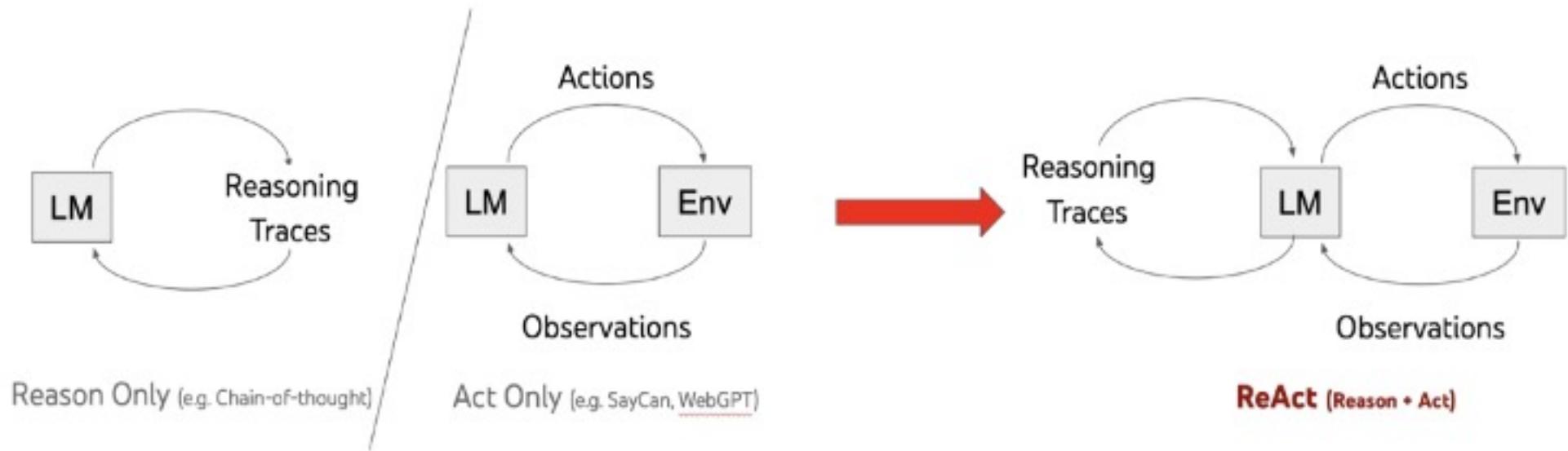
Variables internas para utilizar en el flujo



Time Travel

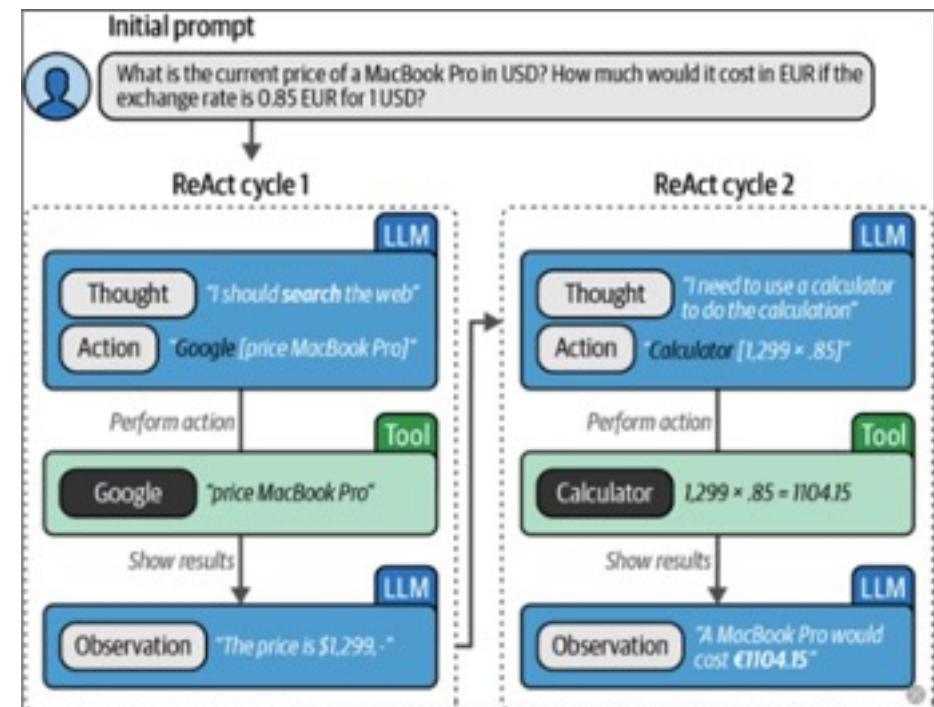
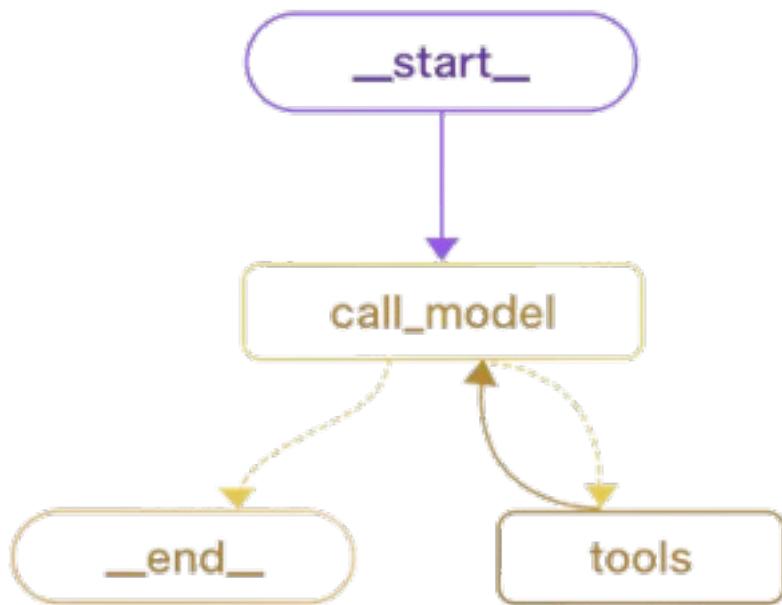
Inspeccionar y re-ejecutar estados pasados

Esquema ReAct



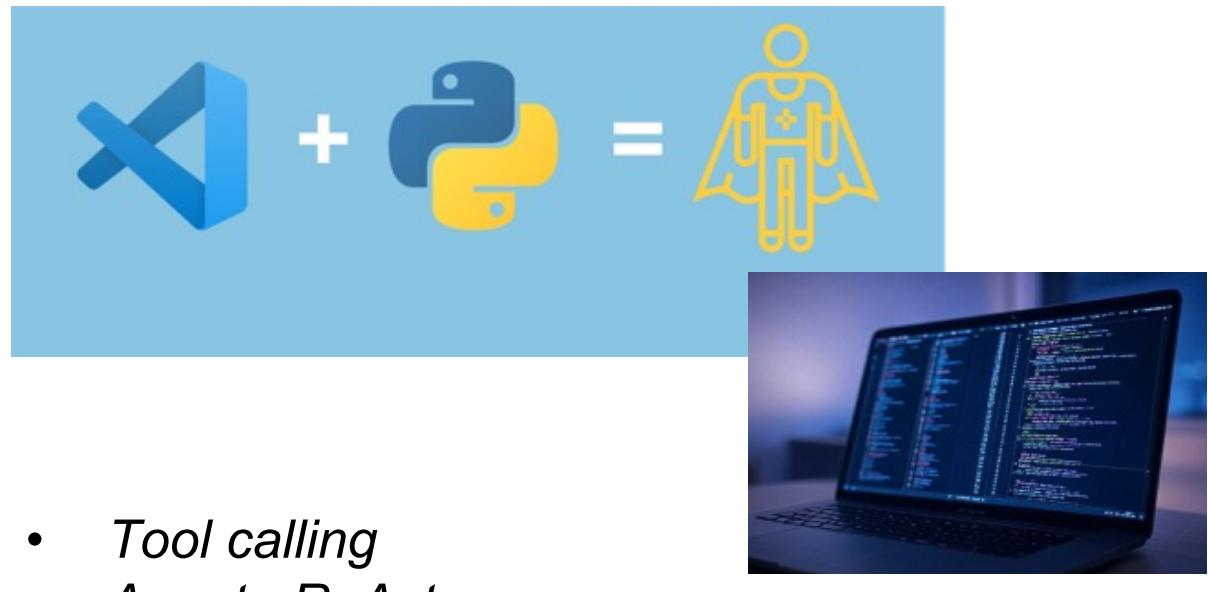
<https://react-lm.github.io/>

ReAct en Langgraph



<https://github.com/langchain-ai/react-agent>

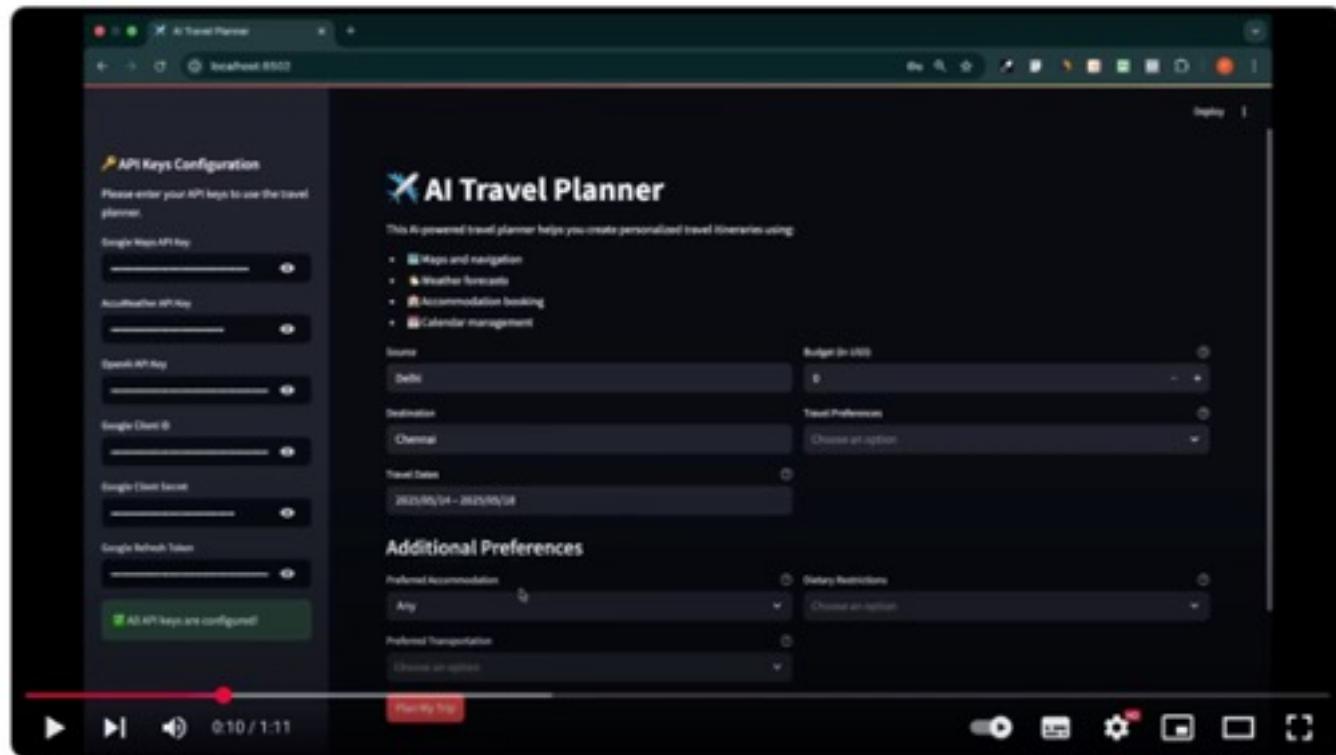
Notebooks



Pausa

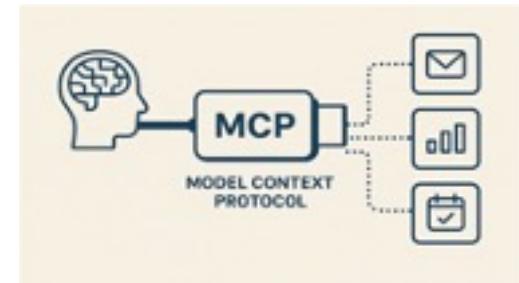


Ejemplo de Agente ReAct: Travel Planner

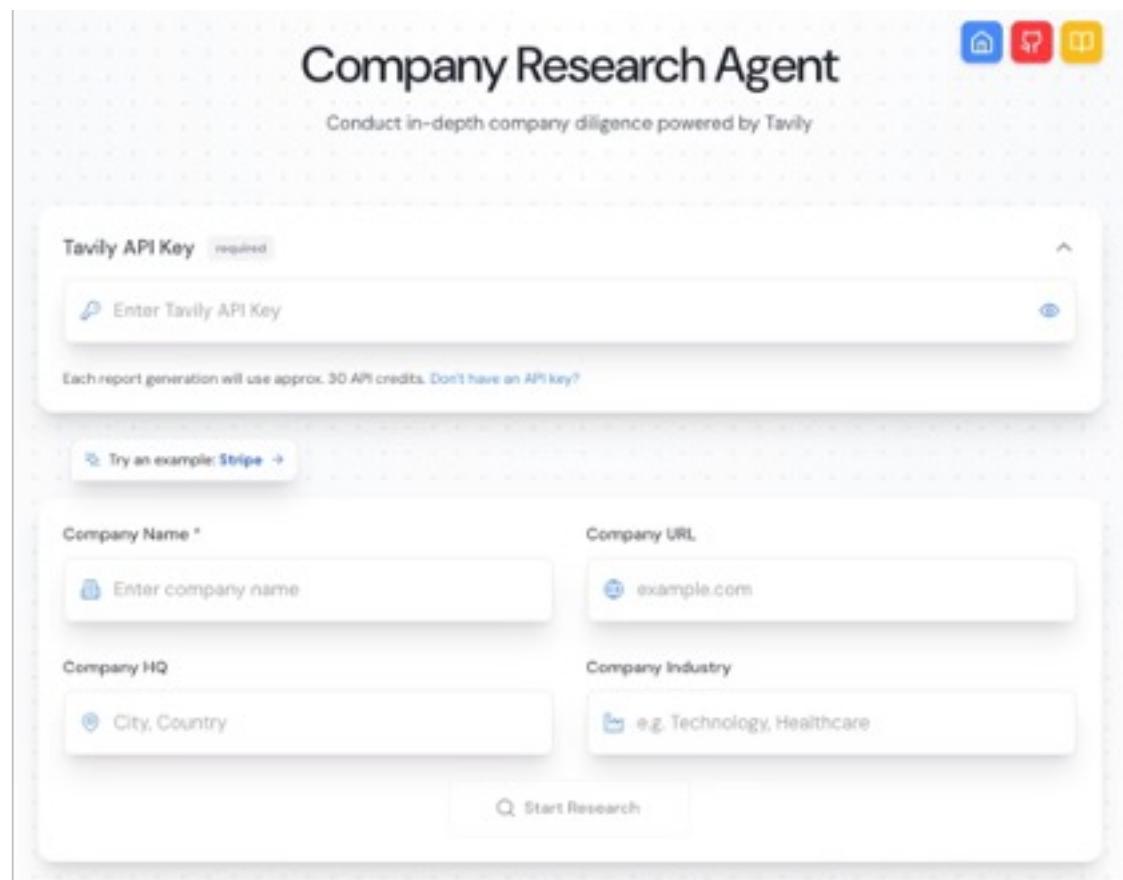


Build an AI Travel Planning Agent with MCP

<https://www.youtube.com/watch?app=desktop&v=WikMj1xKF7M>



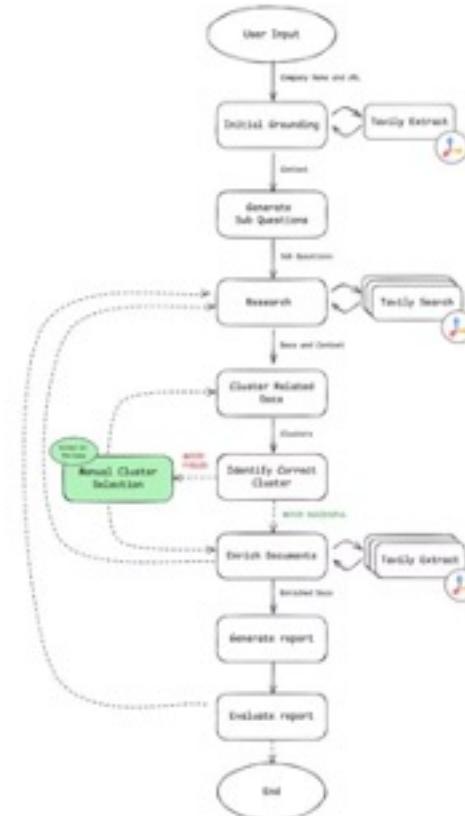
Ejemplo de Workflow: Company Researcher



The screenshot shows the 'Company Research Agent' interface. At the top, there are three colored icons: blue, red, and yellow. Below the title, a subtext reads 'Conduct in-depth company diligence powered by Tavily'. The main input field is labeled 'Tavily API Key' with a 'required' indicator. A placeholder text 'Enter Tavily API Key' is present, along with a 'Try an example: Stripe' button. A note below states: 'Each report generation will use approx. 30 API credits. Don't have an API key?'. Below the API key field are four input fields: 'Company Name' (placeholder 'Enter company name'), 'Company URL' (placeholder 'example.com'), 'Company HQ' (placeholder 'City, Country'), and 'Company Industry' (placeholder 'e.g. Technology, Healthcare'). A 'Start Research' button is located at the bottom right.

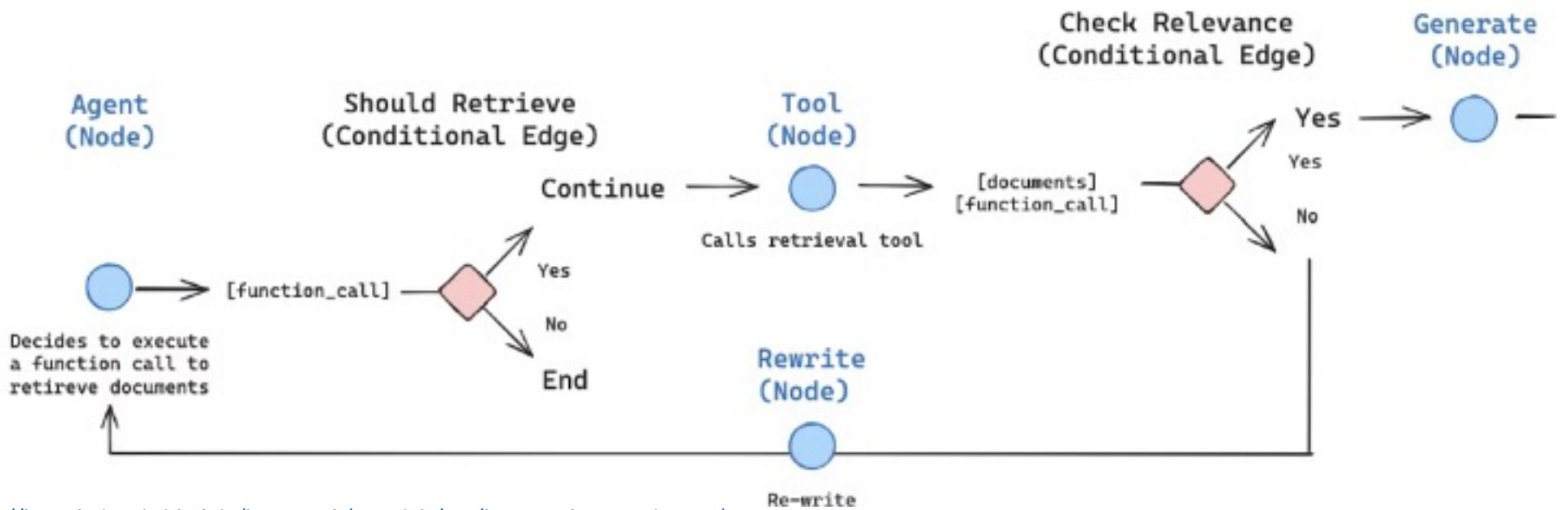
<https://companyresearcher.tavily.com/>

Ejemplo de Workflow: Company Researcher



: <https://github.com/guy-hartstein/company-research-agent>

RAG agéntico (tipo workflow)



[1] https://langchain-ai.github.io/langgraph/tutorials/rag/langgraph_agentic_rag/

[2] <https://www.ibm.com/think/topics/agentic-rag>

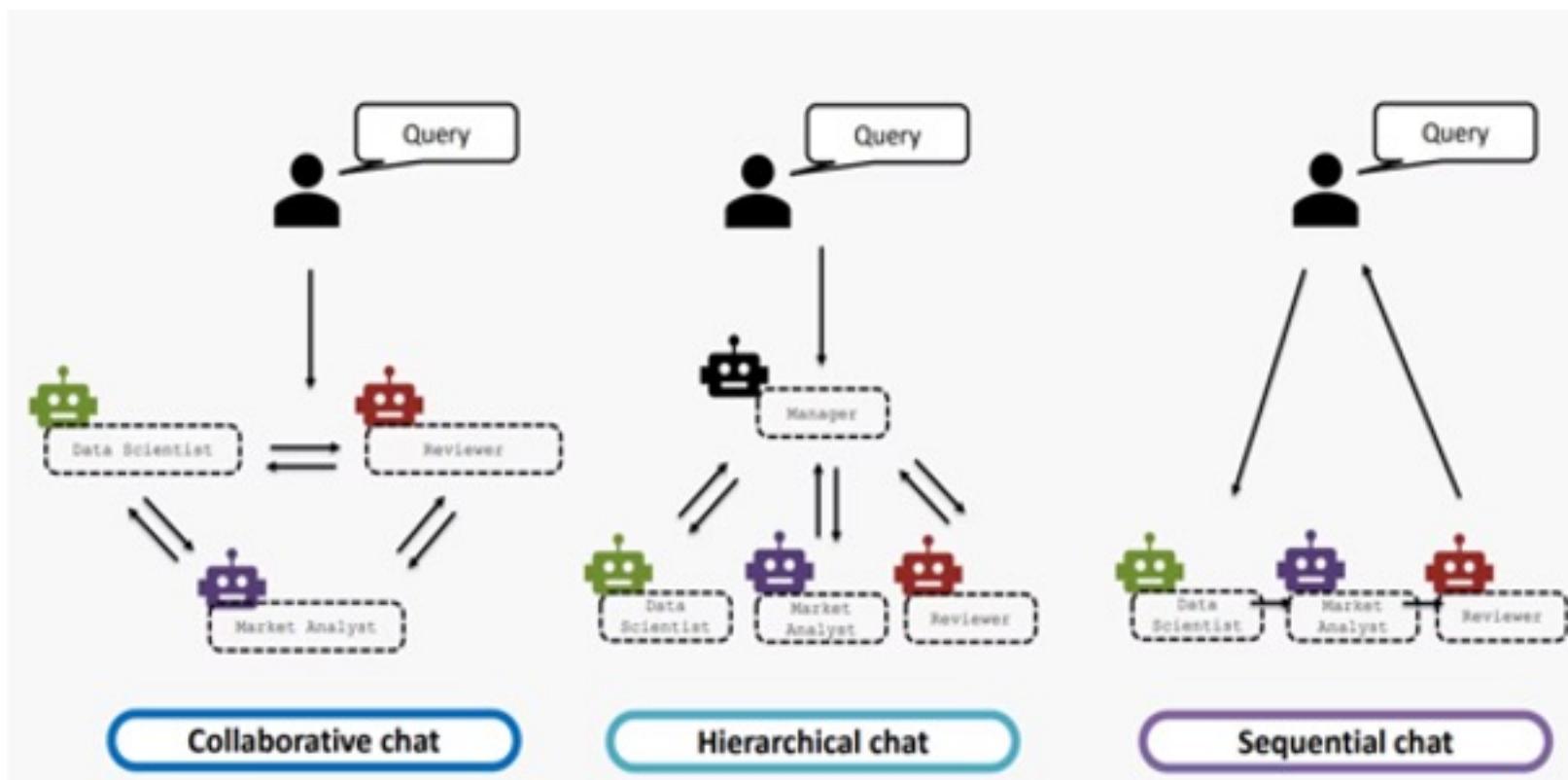
Notebooks



- *RAG agentico*
- *Reflection*
- *Human in the loop*

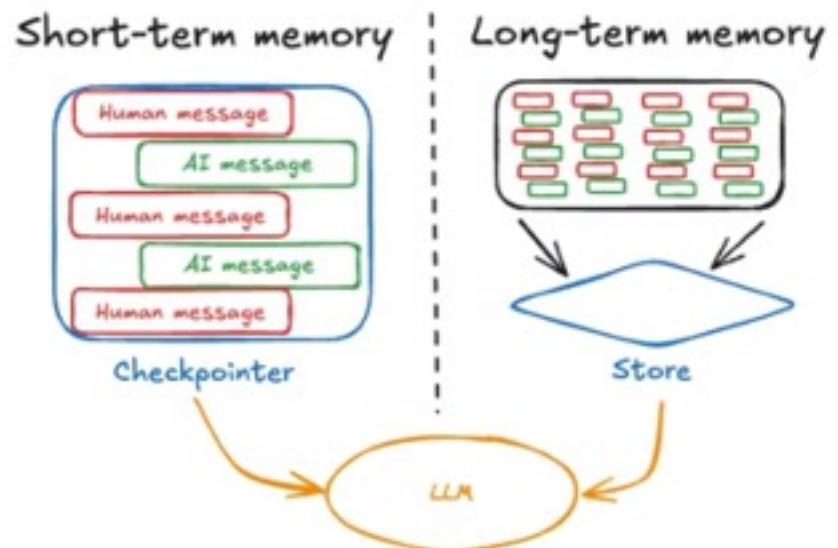


Coordinación entre Agentes



Gestión de memoria

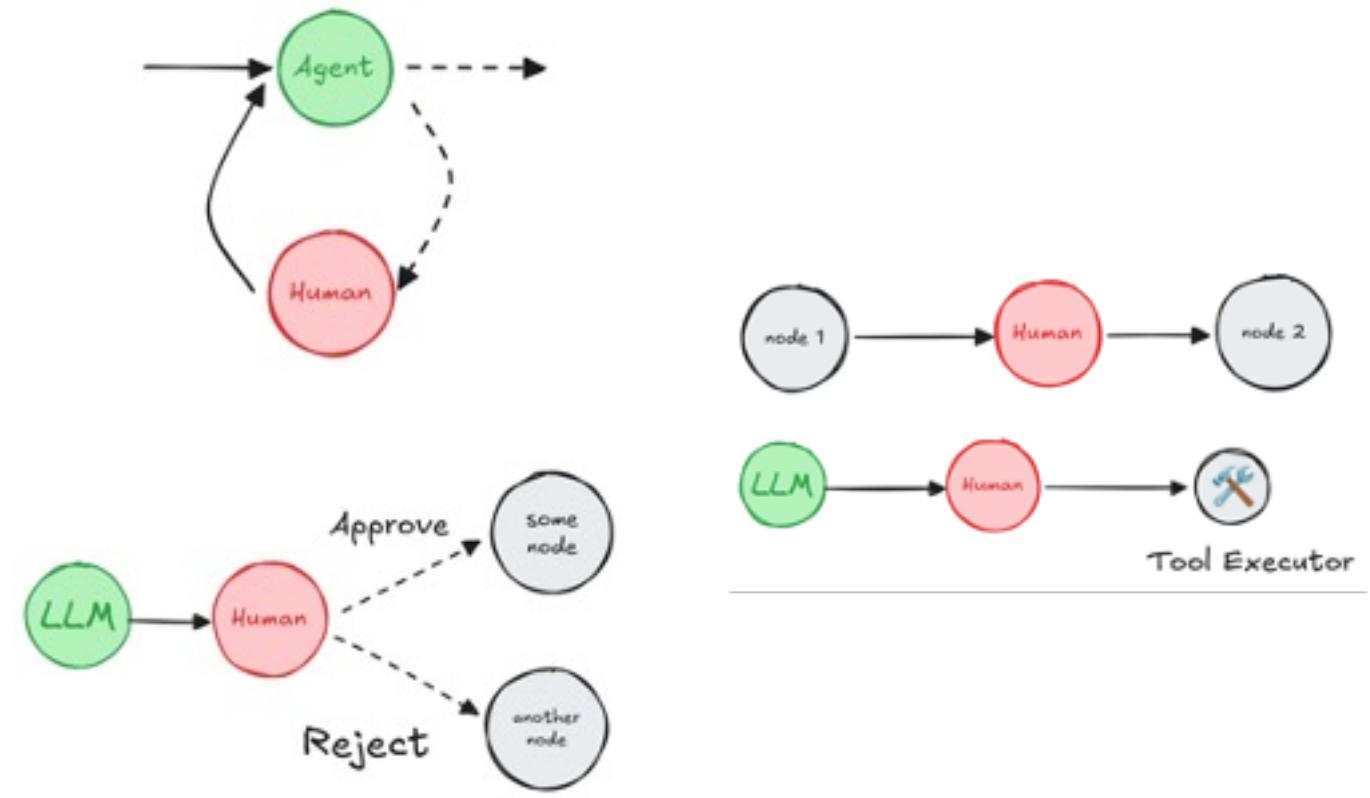
- Es un mecanismo para permitirle a los agentes retener y utilizar información en distintas partes de un flujo o aplicación para la resolución de problemas
 - Memoria de corto plazo (dentro de la misma interacción o sesión)
 - Memoria de largo plazo (entre interacciones separadas o sesiones)
 - En Langgraph, se maneja con los conceptos de state, checkpointer y store
- En una memoria de largo plazo, a menudo se puede hacer búsqueda semántica
 - Algunas implementaciones soportan también nociones de temporalidad y grafos de relaciones entre eventos



Human in the Loop

- En un flujo o agente suele ser necesaria la intervención humana
 - Para autorizar ejecución de ciertas acciones
 - Para dar feedback
 - Para realizar moderación (por ej., consideraciones éticas)
 - Cuando la "confianza" del LLM en cierta respuesta es baja
 - Para asignar etiquetas

LangGraph provee un mecanismo de *interrupción* que luego puede continuarse



Patrones constructivos



Reflection

Permite a la IA evaluar y mejorar su propio desempeño analizando sus resultados, decisiones y procesos de razonamiento.



Planning

Capacita a la IA para desglosar tareas complejas en pasos manejables, desarrollando enfoques estructurados para la resolución de problemas.



Tool Use

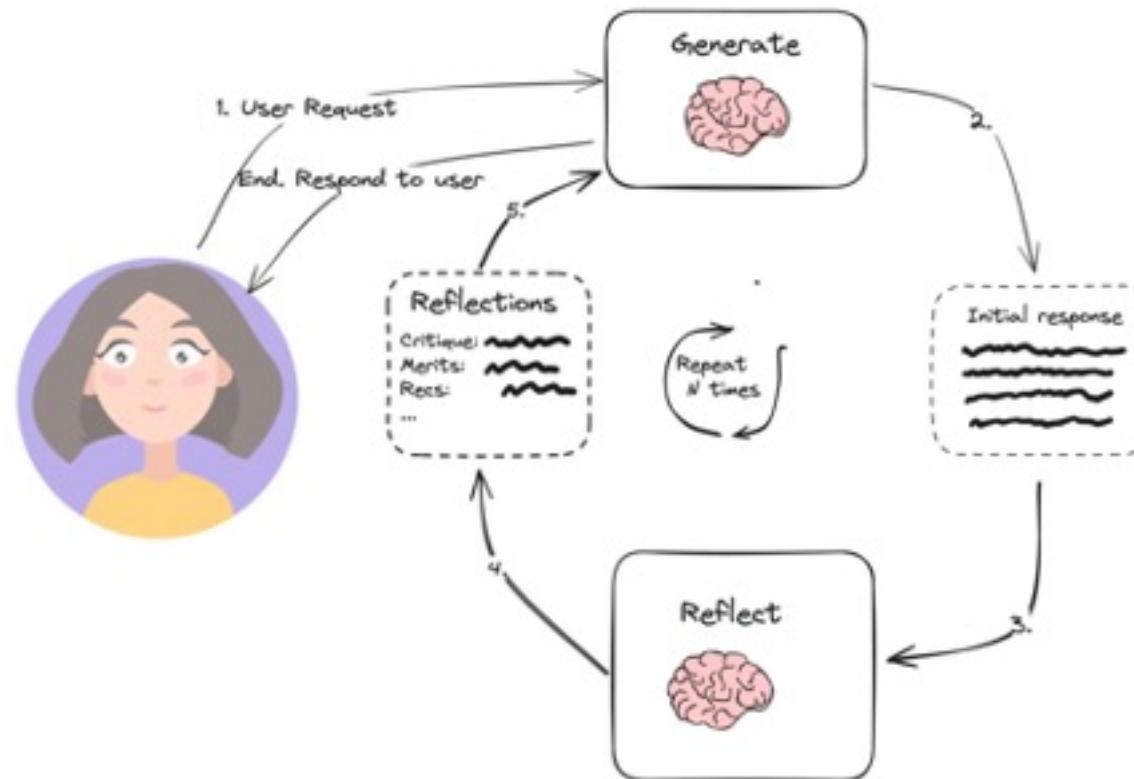
Permite a la IA aprovechar herramientas y recursos externos, expandiendo sus capacidades más allá del procesamiento del lenguaje.



Multi-Agent Collaboration

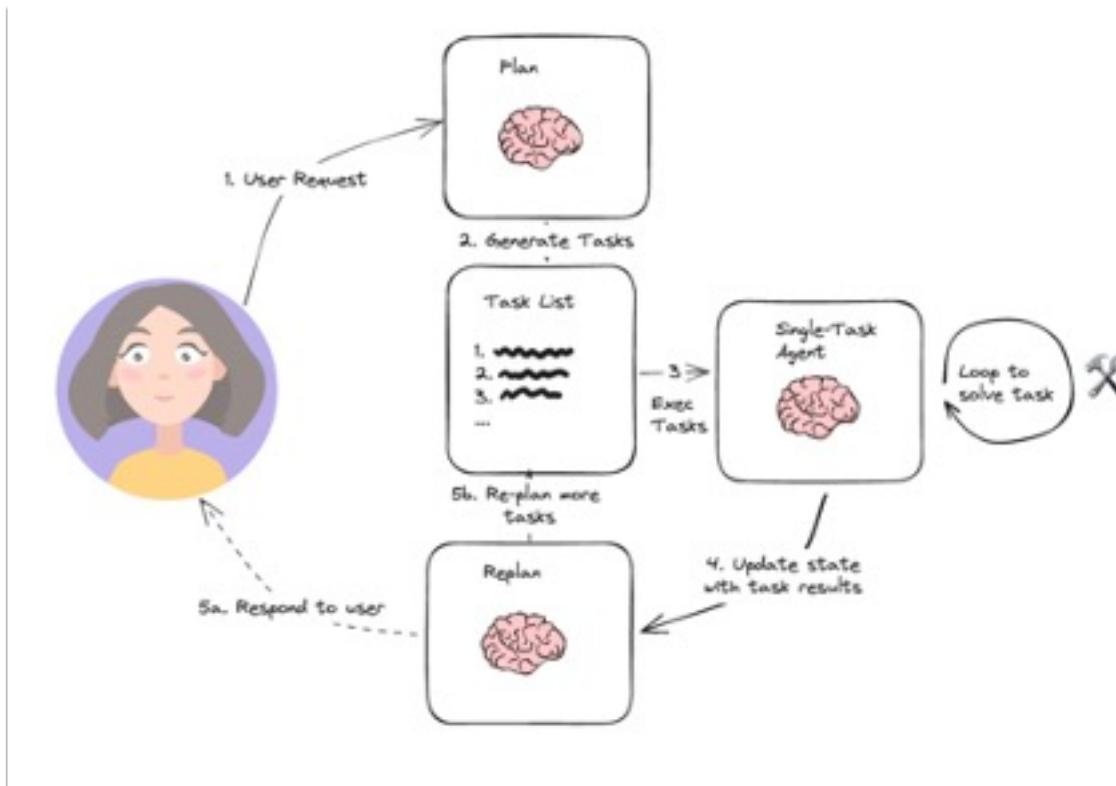
Facilita que múltiples agentes de IA trabajen juntos, cada uno con roles especializados, para abordar problemas complejos de manera más efectiva.

Reflexión



<https://langchain-ai.github.io/langgraph/tutorials/reflection/reflection/>

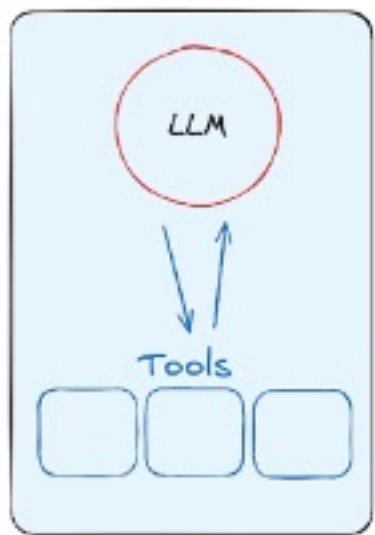
Planning (Plan & Execute)



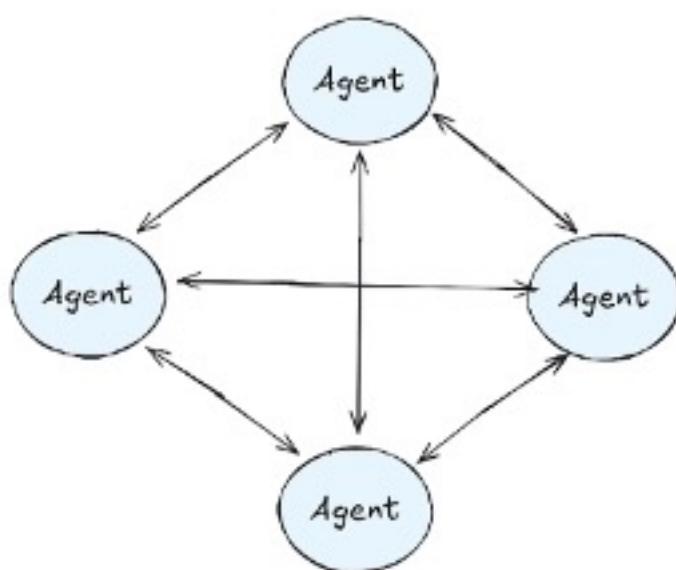
<https://langchain-ai.github.io/langgraph/tutorials/plan-and-execute/plan-and-execute/>

Agente + Tools versus Multi-agente

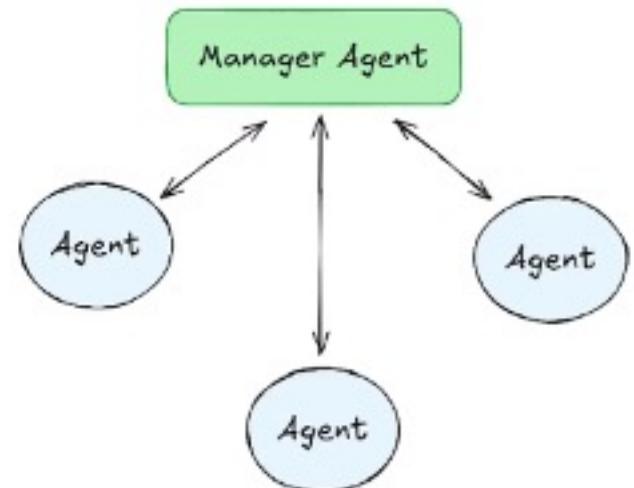
Single Agent Architecture



Network Architecture

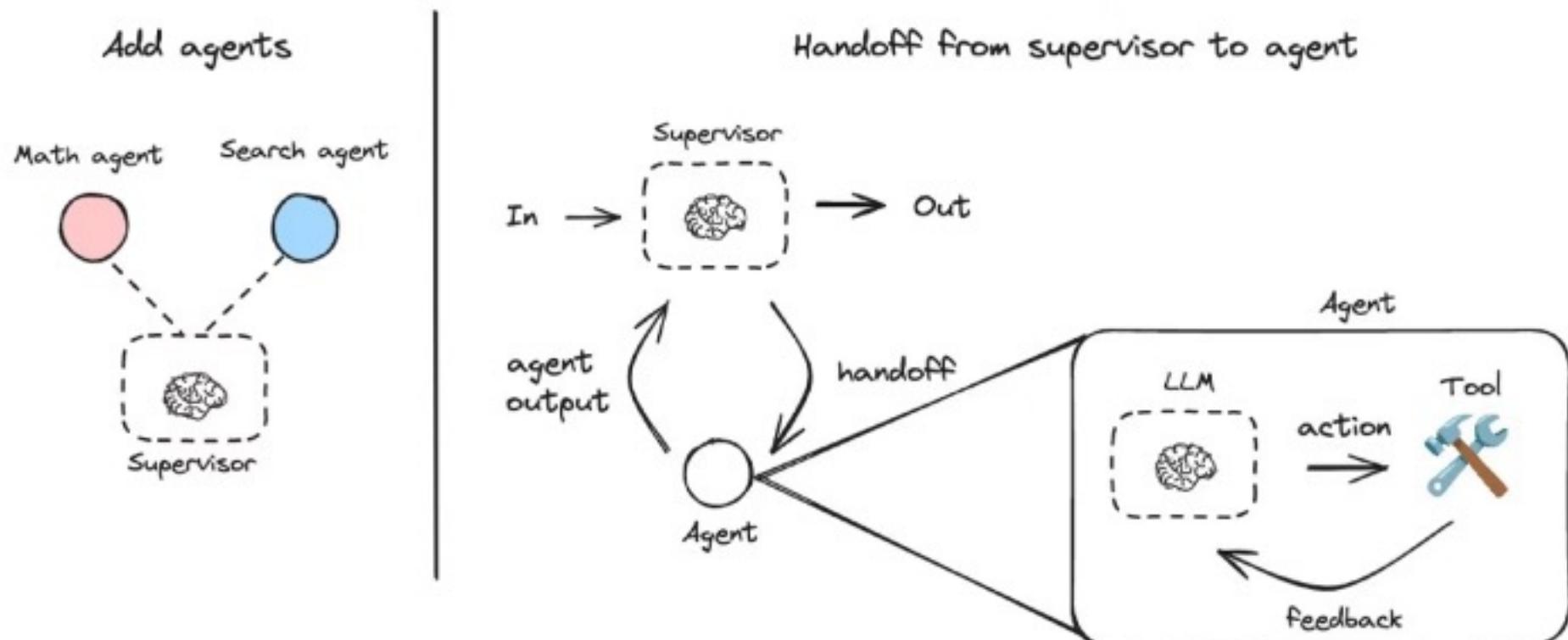


Supervisor Architecture



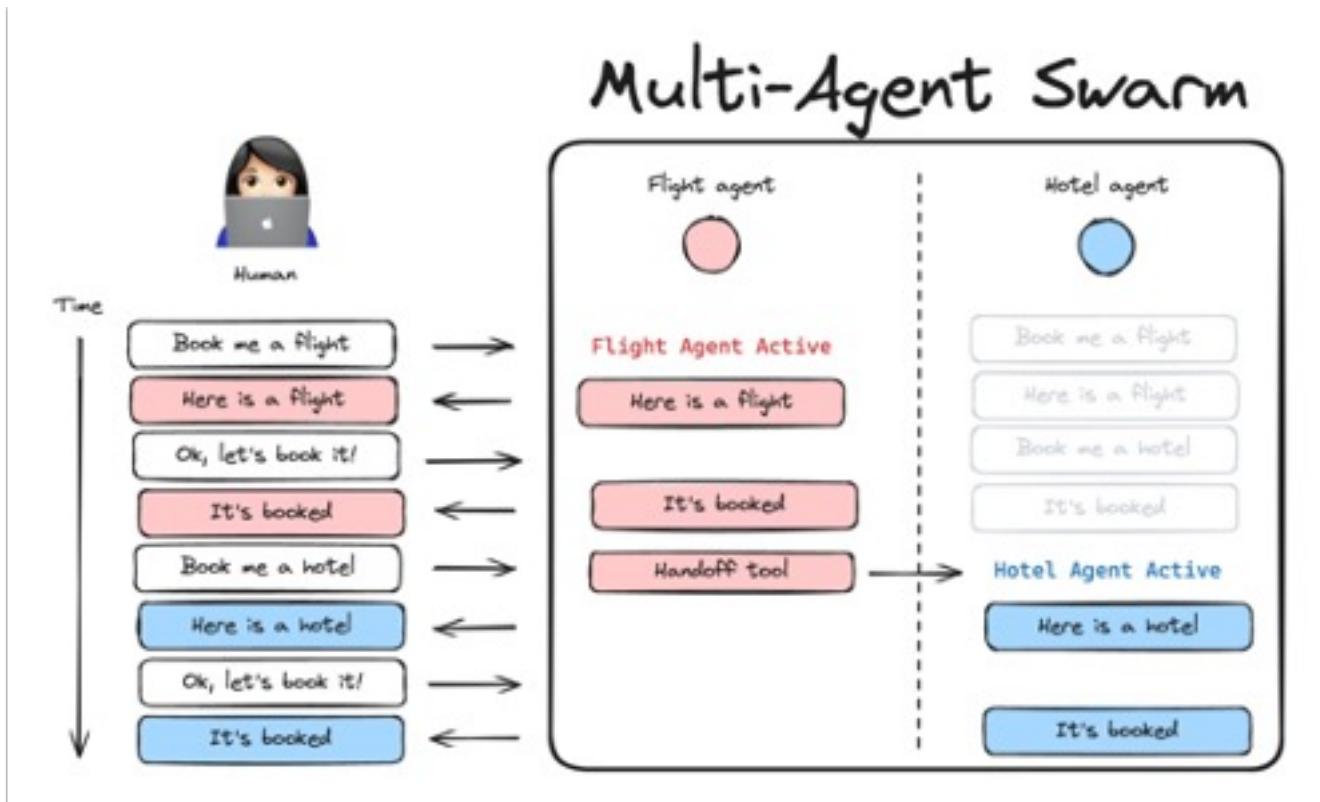
<https://blog.langchain.com/benchmarking-multi-agent-architectures/>

Supervisor (con handoffs)



<https://github.com/langchain-ai/langgraph-supervisor-py>

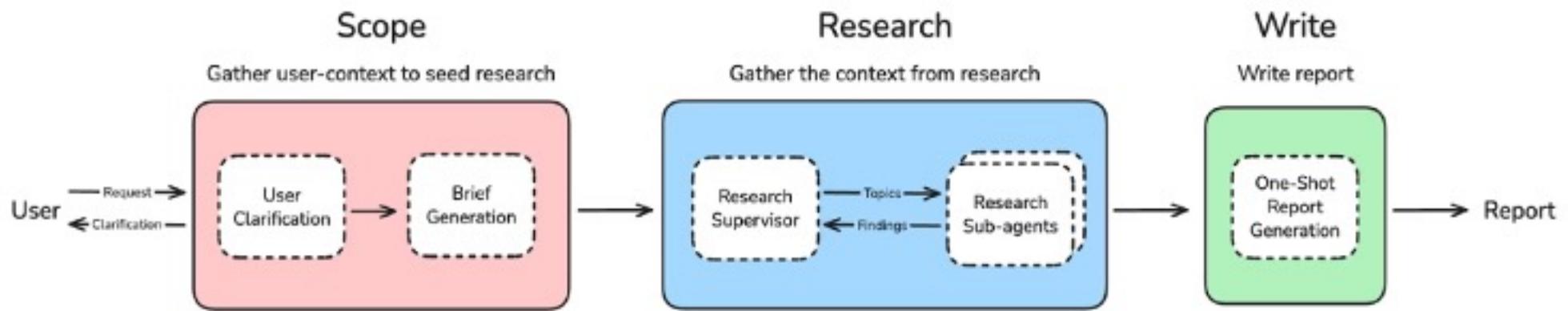
Swarm (con handoffs)



<https://github.com/langchain-ai/langgraph-swarm-py>

Aplicación de Supervisor: Deep Research

Búsqueda desde diferentes perspectivas, partiendo de un tema y buscando derivaciones



[1] <https://blog.langchain.com/deep-agents/>

[2] <https://blog.langchain.com/open-deep-research/>

Aspectos de Ingeniería de Software

- Agentes y tools como microservicios
- Observabilidad (de todo lo que se pueda)
- Resiliencia (por ej., si el servicio de LLM falla o se satura)
- Performance
 - Minimizar el número de llamadas al LLM
 - Acotar la cantidad de tokens enviados (costos)
 - Mantener una cierta calidad en la salida
 - Tener poca variabilidad en la salida
- Decidir cuál es el framework "correcto" de agentes a utilizar
- LLMOps
- Diseño de UX cuando hay LLMs, agentes, workflows involucrados



Intercambio



Andres Diaz Pace

andres.diazpace@isistan.unicen.edu.ar

