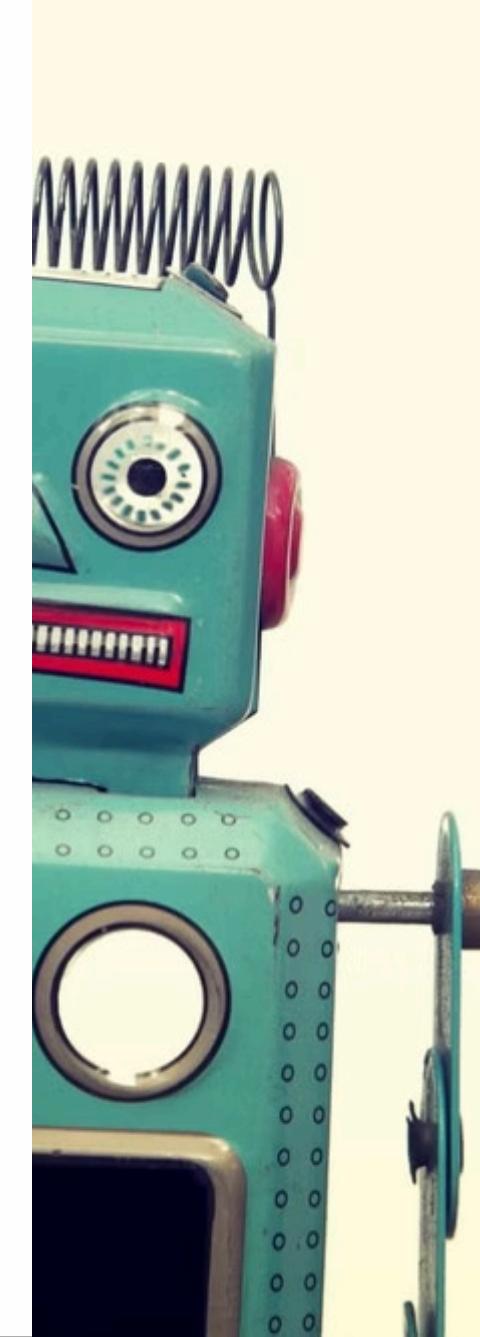


Agentes LLM

 by Ivan Ruiz Rube



Contenidos

- Introducción
- RAG modular
- Uso de herramientas
- Colaboración

Introducción

Evolución de la Interacción Persona-Ordenador

1

Interfaces de Línea de Comandos

Interacción básica mediante texto, requiriendo conocimientos técnicos específicos.

2

Interfaces Gráficas de Usuario

Mejora en la accesibilidad con elementos visuales intuitivos.

3

Interfaces Táctiles

Interacción a través de pantallas táctiles, revolucionando la experiencia móvil.

4

Asistentes Virtuales

Introducción de la interacción por voz y procesamiento del lenguaje natural.

5

Realidad Aumentada y Realidad Virtual

Integración de elementos virtuales en el mundo real para experiencias inmersivas.

6

Chatbots basados en IA generativa

Asistentes, copilotos y agentes que permiten conversación natural y ejecución de tareas complejas.



Chatbots Copilots Agents

CHATBOTS



CHATBOTS

COPILOTS



AGENTS





Asistentes (chatbots)

Definición

Software diseñado para imitar conversaciones humanas, principalmente a través de texto. Tiene la motivación de ayudar al usuario para alguna tarea determinada.

Chatbots pre-LLM

Se basan en respuestas predefinidas y tienen una capacidad limitada para manejar consultas complejas o ambiguas.

Chatbots basados en LLM

Con la IA generativa, los chatbots pueden comprender mejor las preguntas del usuario y ofrecer respuestas más precisas y naturales.

Copilotos

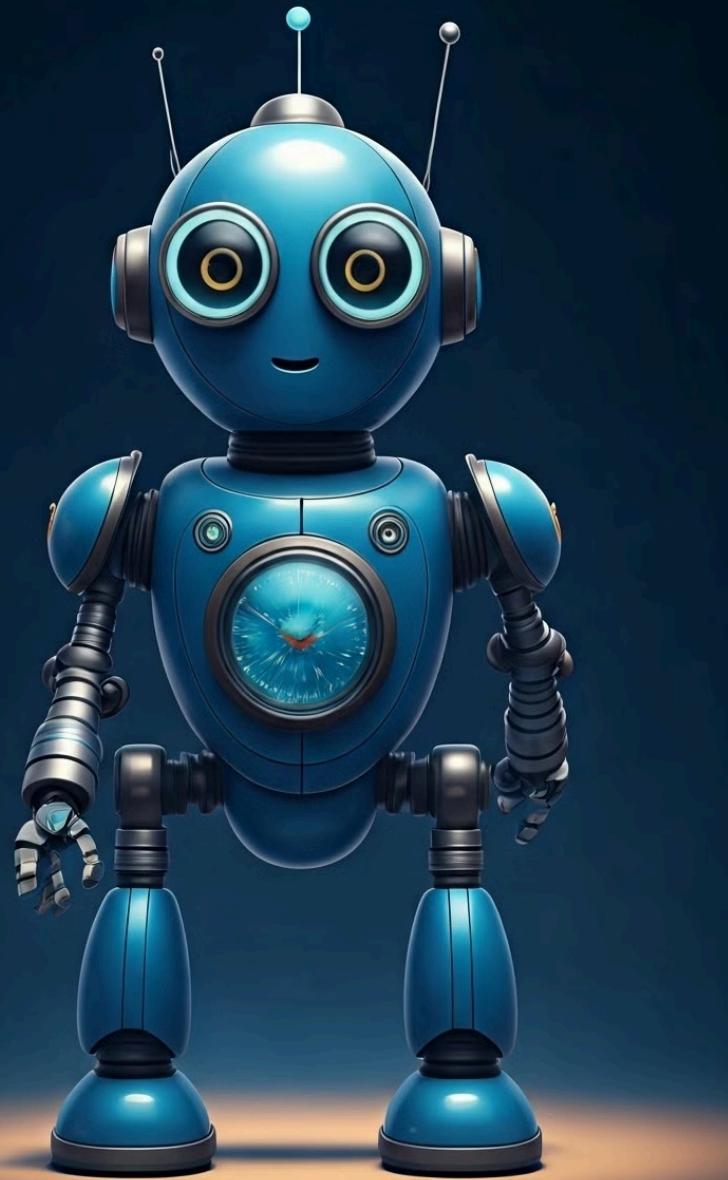
Características

- Trabajo conjunto con el usuario en tiempo real
- Recomendaciones personalizadas basadas en el contexto
- Evolución constante a través del aprendizaje automático
- Integración fluida en las herramientas que se utilizan diariamente

Beneficios

- Optimización del tiempo y recursos
- Minimización de fallos y errores
- Aprendizaje acelerado y efectivo





Agentes



Autonomía

Capaces de tomar decisiones independientes basadas en objetivos y contexto.



Multifuncionalidad

Integrarán diversas herramientas y APIs para realizar tareas complejas.



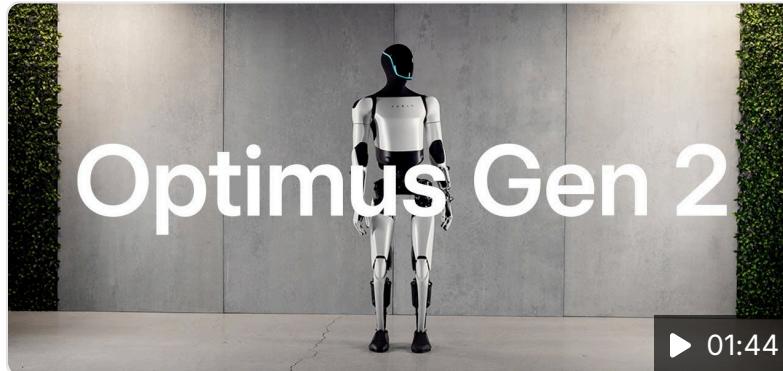
Colaboración

Interactúan con otros agentes y sistemas para resolver problemas de forma coordinada.

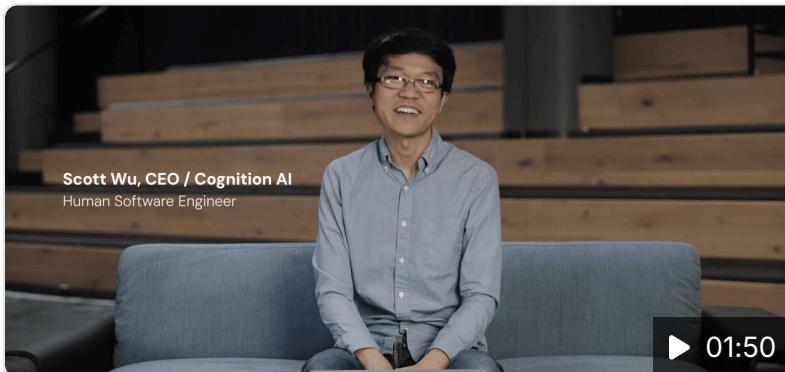


Adaptabilidad

Se ajustan a cambios en el entorno y mejoran con el tiempo.

 YouTube**Optimus - Gen 2 | Tesla**

New bot in town! Optimus Gen 2 features Tesla-designed actuators and sensors, faster and more capable hands, faster walking, lower total weight,...

 YouTube

Introducing Devin, the first AI software engineer

Meet Devin, the world's first fully autonomous AI software engineer. Devin is a tireless, skilled teammate, equally ready to build alongside you or independent...

Características de los Agentes LLM



Conversación en Streaming

Respuestas fluidas en tiempo real, mejorando la interacción.



Información de contexto

Acceso a datos internos y externos, vía RAG, para respuestas más completas.



Memoria Contextual

Recuerda conversaciones anteriores para respuestas más precisas.



Integración de Herramientas

Usa APIs y servicios externos para ampliar sus capacidades.



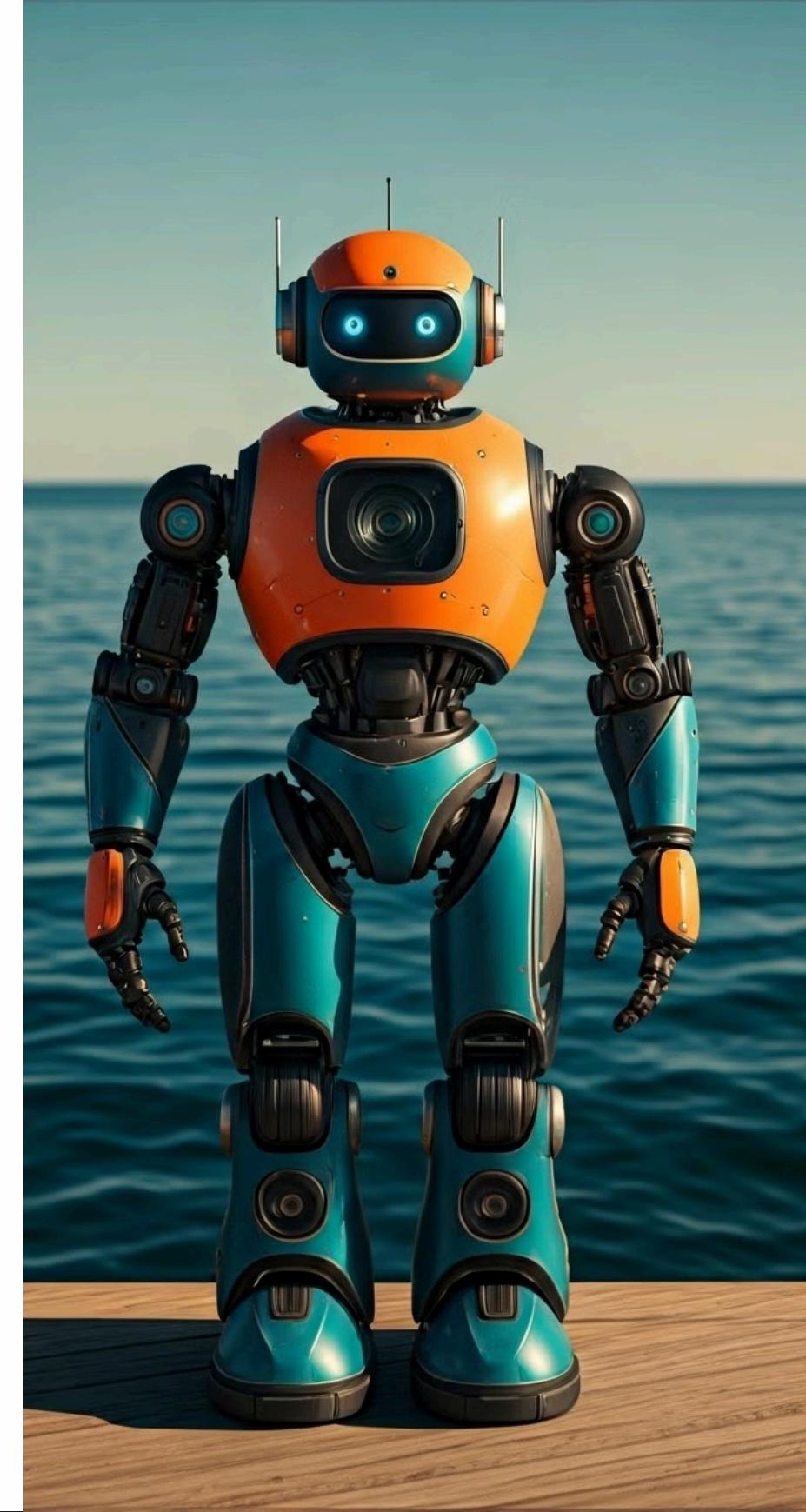
Colaboración entre Agentes

Trabajo en equipo con otros agentes para resolver problemas complejos.



Adaptabilidad

Se adapta a cambios y toma la iniciativa.



RAG modular

Limitaciones del RAG naive/avanzado



Pensamiento complejo

Los problemas complejos con frecuencia requieren un razonamiento multi-paso. Un solo paso de recuperación de información a veces no basta.



Procesos humanos

Los humanos, al escribir, cometemos errores, retrocedemos y corregimos para alcanzar la calidad. Los LLM también sufren esta limitación.

Proceso (humano) de elaboración de un texto escrito

1

Planificar un esquema

Definir la estructura del texto para asegurar la coherencia y claridad.

2

Investigación

Buscar información relevante en Internet para complementar y enriquecer el contenido.

3

Primer borrador

Redactar un primer borrador para organizar las ideas y el contenido principal.

4

Revisión del borrador

Identificar argumentos débiles y eliminar información superflua.

5

Reescritura

Reescribir el borrador para corregir los puntos débiles detectados.

6

Iterar

Repetir los pasos anteriores para mejorar la calidad del texto.



RAG modular



Modularidad

Esta técnica permite agregar o reemplazar componentes de RAG según las necesidades específicas de la tarea.



Adaptabilidad

La modularidad de RAG permite adaptar el sistema a diversas tareas y contextos, ofreciendo una mayor flexibilidad y control.



Patrones RAG

Además de los patrones *Naive* y *Avanzado*, han surgido otros patrones que permiten aumentar y refinar el contexto con el fin de manejar tareas más complejas.

Multi-step augmentation: iterative

Query: "Quiero información sobre los programas de máster ofrecidos en la ESI"

Retrieve: El sistema busca y recupera una lista de másteres disponibles.

Generate: "Máster en Ciberseguridad, Máster de Investigación en Ingeniería."

Judge: La respuesta es incompleta, ya que faltan detalles como la duración y requisitos de acceso.

Query: "¿Cuáles es la duración y los requisitos de admisión en el Máster en Ciberseguridad?"

Retrieve: El sistema busca y recupera información de este máster

Generate: "El Máster en Ciberseguridad consta de 60 créditos y se necesita el Grado en Ingeniería Informática"

Judge: La respuesta es incompleta, ya que me falta información del Máster de Investigación en Ingeniería.

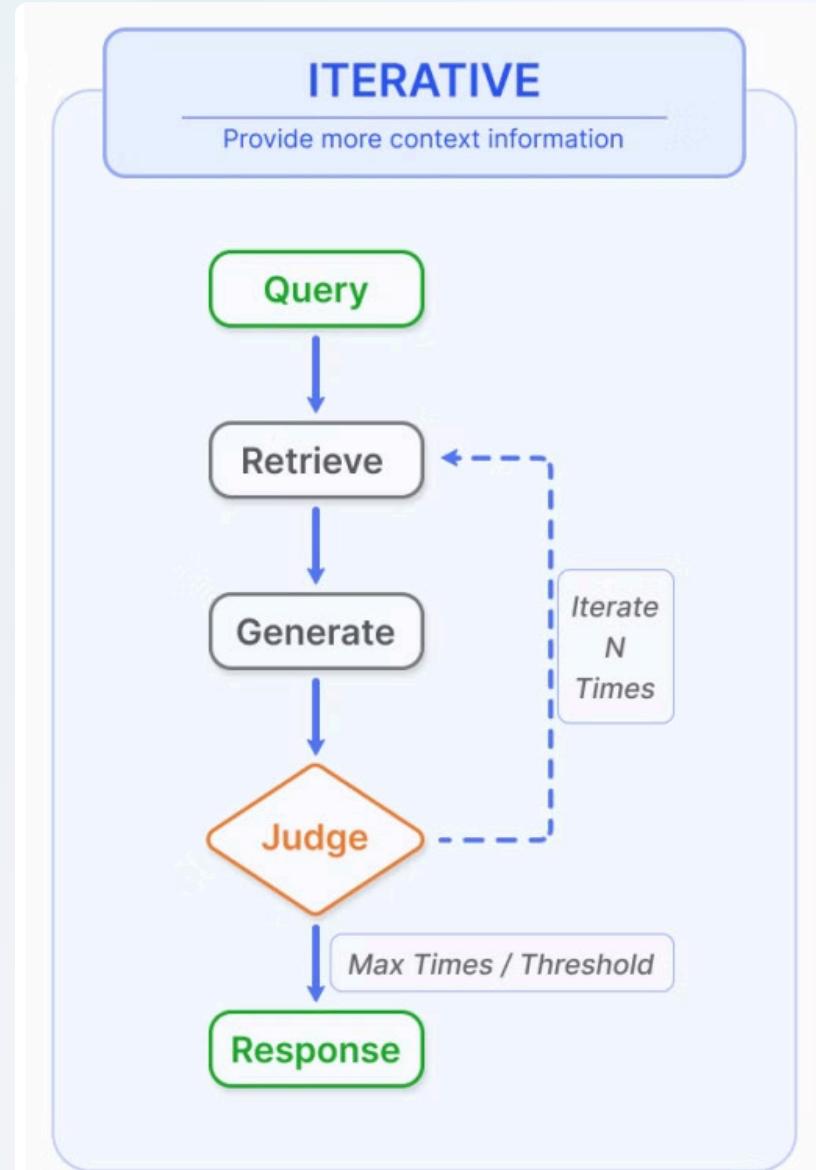
Query: "¿Cuáles es la duración y los requisitos de admisión en el Máster de Investigación en Ingeniería?"

Retrieve: El sistema busca y recupera información de este máster

Generate: El de Investigación en Ingeniería consta de 60 créditos y se necesita algún Grado en Ingeniería

Judge: La respuesta está ya completa.

Response: "A continuación le indico los másteres disponibles junto con la información que dispongo de ellos..."



Multi-step augmentation: recursive

Query: "¿Qué proyectos de investigación está llevando a cabo la UCA en el área de energías renovables?"

Retrieve: El sistema busca y recupera información general indicando que la UCA tiene un proyecto de investigación sobre energía eólica marina.

Generate: "La Universidad de Cádiz está llevando a cabo un proyecto de investigación sobre energía eólica marina."

Judge: La respuesta es incompleta, ya que falta información detallada sobre este proyecto específico.

Query: "¿Cuál es el objetivo principal del proyecto de energía eólica marina de la Universidad de Cádiz?"

Retrieve: El sistema busca y encuentra que el objetivo principal del proyecto.

Generate: "El proyecto de energía eólica marina tiene como objetivo principal estudiar la viabilidad de instalar parques eólicos en la costa de Cádiz."

Judge: Se detecta que aún falta más detalle sobre los resultados del proyecto.

Query: "¿Cuáles son los resultados obtenidos hasta ahora en el proyecto de energía eólica marina?"

Retrieve: El sistema encuentra los resultados preliminares del proyecto

Generate: "Los resultados preliminares muestran un alto potencial para la instalación de parques eólicos en ciertas áreas de la costa de Cádiz."

Judge: La respuesta ahora es completa y ofrece información detallada sobre el proyecto.

Response: "La Universidad de Cádiz está llevando a cabo un proyecto de investigación en energía eólica marina. El objetivo es... Los resultados preliminares indican..."

RECURSIVE

Break down complex problems step by step

Query

Retrieve

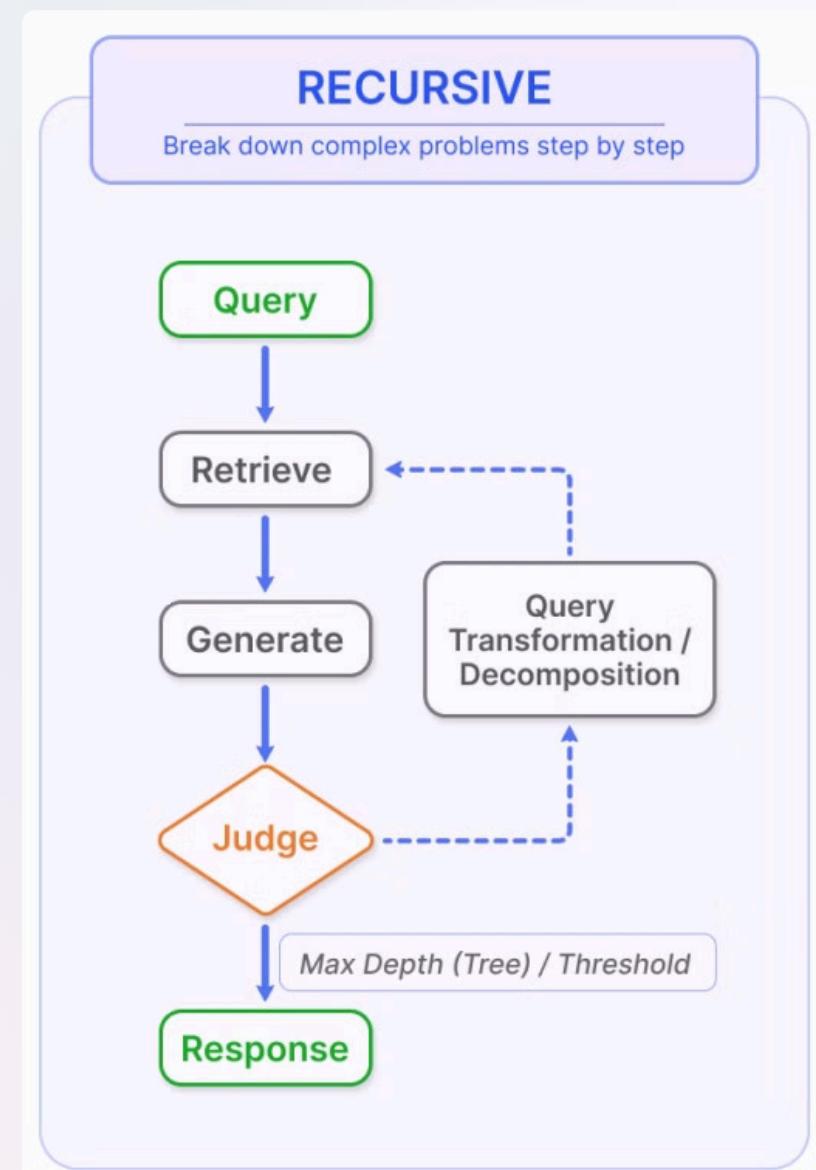
Generate

Judge

Response

Query Transformation / Decomposition

Max Depth (Tree) / Threshold



Multi-step augmentation: adaptive

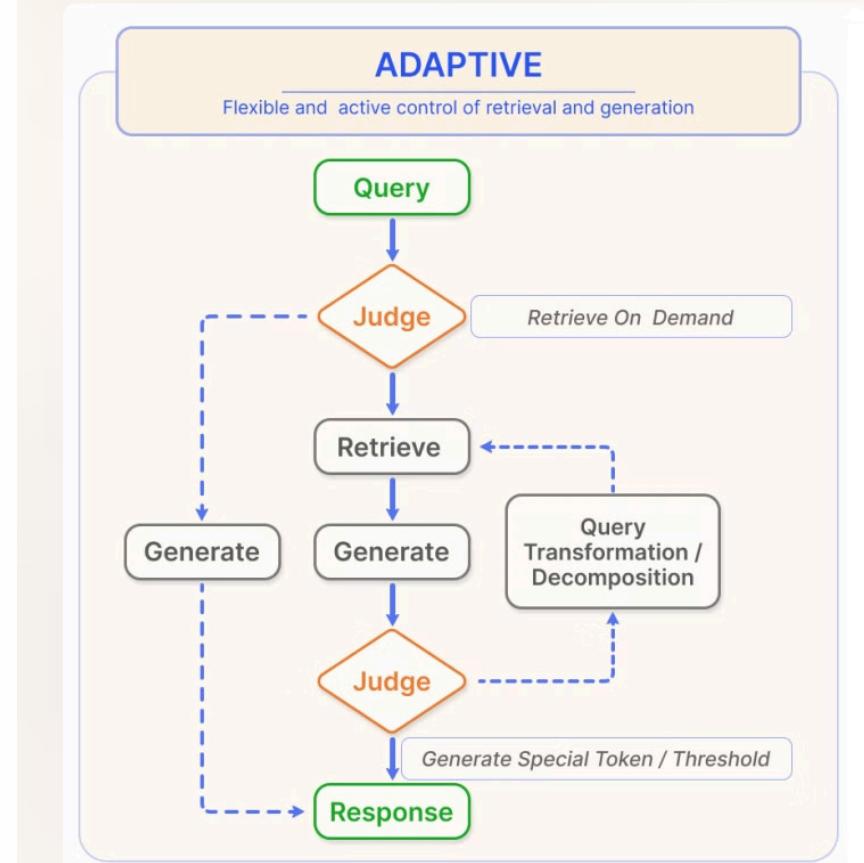
Query: "¿Qué proyectos de investigación está llevando a cabo la UCA en el área de energías renovables?"

Retrieve: El sistema busca y recupera información general indicando que la UCA tiene un proyecto de investigación sobre energía eólica marina.

Generate: "La Universidad de Cádiz está llevando a cabo un proyecto de investigación sobre energía eólica marina."

Judge: El modelo se pregunta si es necesario obtener más detalles para responder adecuadamente. Dado que la pregunta original es general, decide que tiene suficiente información y no realiza más búsquedas. Sin embargo, al reconocer que el usuario podría desear más detalles sobre estos proyectos, el sistema se prepara para ofrecer información adicional si el usuario lo solicita.

Response: "La Universidad de Cádiz está llevando a un proyecto de investigación en el área de energías renovables, concretamente sobre energía eólica marina. ¿Te gustaría saber más detalles sobre alguno de estos proyectos?"



Uso de herramientas

LLM con Funciones

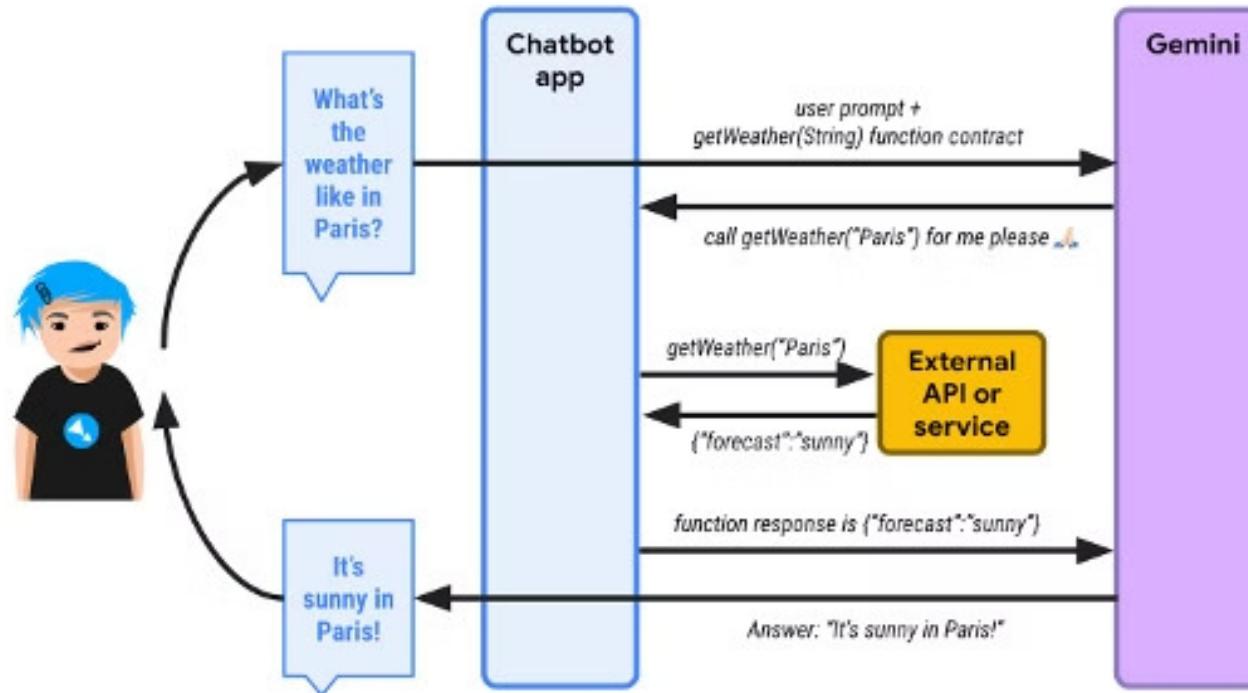
Los modelos de lenguaje grandes (LLM) no solo generan texto, sino que también pueden ejecutar acciones, conocidas como "plugins" o "tools".

Sin embargo, no todos los modelos de lenguaje grandes (LLM) actualmente soportan esta característica.

Cuando se invoca a un LLM con estas capacidades, el LLM no llama directamente a la herramienta, sino que en su respuesta indica la intención de llamarla, junto con los argumentos necesarios.



Flujo



Exponer funciones al LLM

```
# This is the function that we want the model to be able to call
def get_delivery_date(order_id: str) -> datetime:
    # Connect to the database
    conn = sqlite3.connect('ecommerce.db')
    cursor = conn.cursor()
    # ...

# your function definition should be described using JSON Schema.
{
    "name": "get_delivery_date",
    "description": "Get the delivery date for a customer's order. Call this whenever you need to know the delivery date, for example when a customer asks 'Where is my package?'",
    "parameters": {
        "type": "object",
        "properties": {
            "order_id": {
                "type": "string",
                "description": "The customer's order ID."
            },
            "required": ["order_id"],
            "additionalProperties": false,
        }
    }
}
```

Berkeley Function Calling Leaderboard

Este benchmark permite hacer un análisis para evaluar el rendimiento de los LLM en cuanto a su capacidad de invocar funciones externas.

leaderboard consists of real-world data and will be updated periodically. For more information on the evaluation dataset and methodology, please refer to our [blog post](#) and [code release](#).

Rank	Overall Acc	Model	Cost (\$)	Average Latency (s)	ASR Summary	exec Summary	Relevance	Organization	License
1	90	Claude-3.5-Sonnet-20240620 (Prompt)	2.2	1.37	91.31	89.5	85.42	Anthropic	Proprietary
2	88	GPT-4-0125-Preview (Prompt)	5.25	1.97	91.22	88.1	70.42	OpenAI	Proprietary
3	87.65	Claude-3-Opus-20240229 (Prompt)	10.84	4.61	88.93	86.16	80.42	Anthropic	Proprietary
4	86.35	Gemini-1.5-Pro-Preview-0514 (FC)	0.86	1.94	87.92	83.32	89.58	Google	Proprietary
5	85.88	Gemini-1.5-Pro-Preview-0409 (FC)	0.86	1.95	87.62	82.88	88.75	Google	Proprietary
6	85.88	GPT-4-1106-Preview (FC)	5.07	5.97	88.62	81.73	80.42	OpenAI	Proprietary
7	85.59	GPT-4-turbo-2024-04-09 (Prompt)	5.25	2.57	90.01	86.04	62.5	OpenAI	Proprietary
8	84.71	Gorilla-OpenFunctions-v2 (FC)	0.31	0.05	89.38	81.55	61.25	Gorilla LLM	Apache 2.0
9	84.65	GPT-4-0125-Preview (FC)	4.83	4.72	87.17	84.76	82.92	OpenAI	Proprietary
									Meta Llama 3

 gorilla.cs.berkeley.edu

Berkeley Function Calling Leaderboard V2 (aka Berkeley Tool Callin...)

Explore The Berkeley Function Calling Leaderboard (also called The Berkeley Tool Calling Leaderboard) to see the LLM's ability to call functions (aka tools)...

Tipos de Acciones



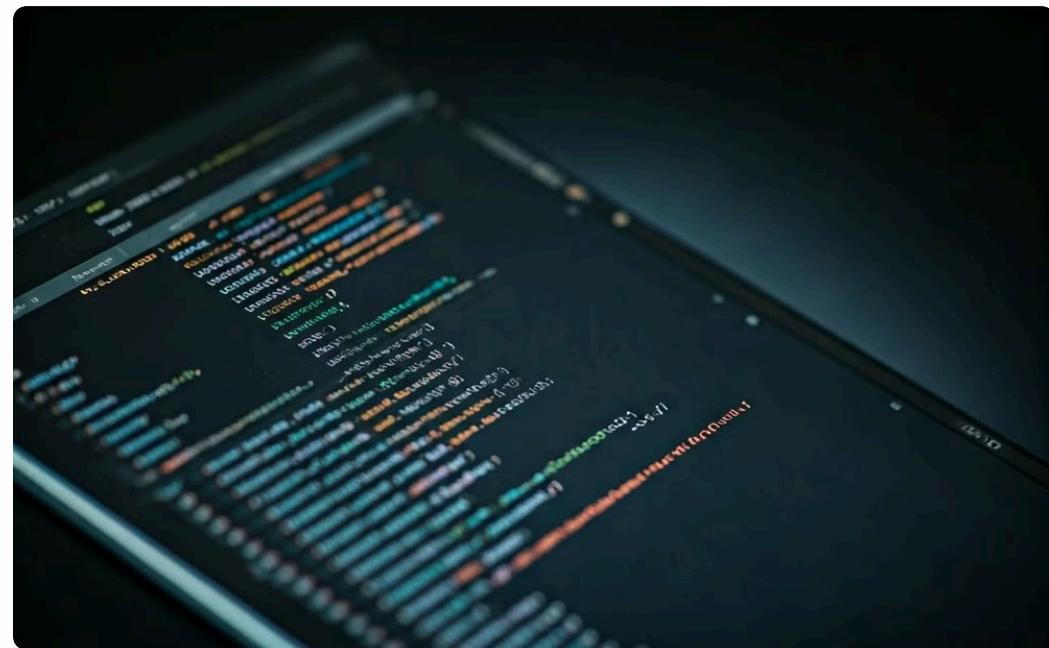
Invoker a funciones locales

Los LLM pueden interactuar con funciones definidas dentro del mismo entorno.



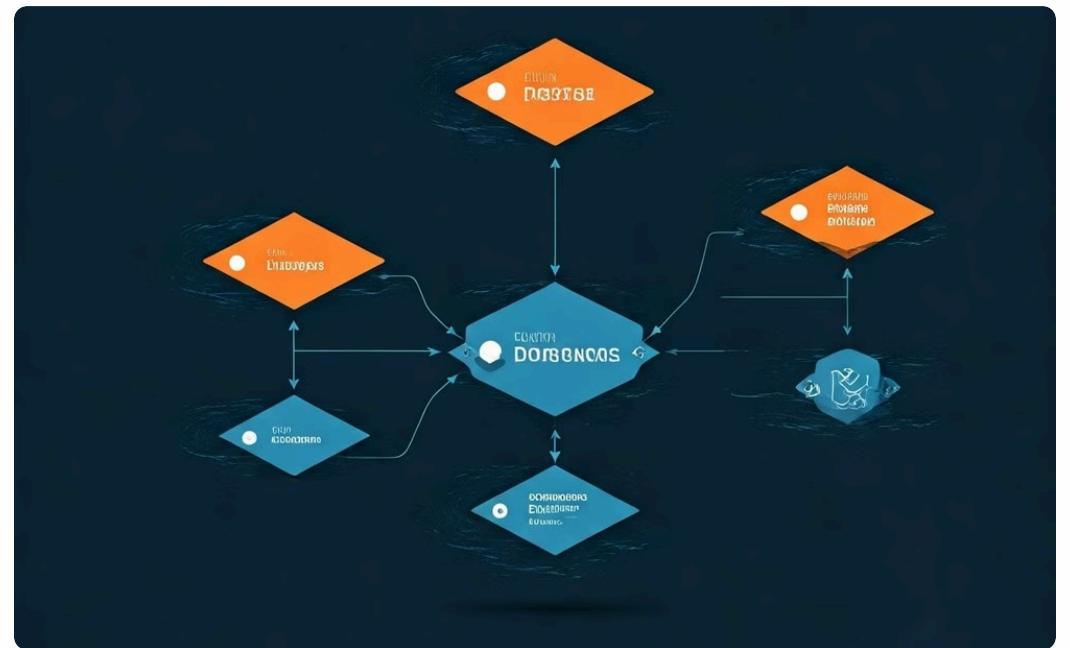
Invoker a procedimientos remotos

Los LLM pueden ejecutar tareas en servidores externos, accediendo a recursos remotos.



Generar y ejecutar/interpretar código

Los LLM pueden generar código y ejecutar/interpretar programas en diferentes lenguajes de programación.



Automatizar procesos

Los LLM pueden automatizar procesos complejos, combinando diferentes acciones para completar tareas específicas.

Ejemplo de llamada a procedimiento local

```
query = "What is 3 * 12? Also, what is 11 + 49?"  
  
llm_with_tools.invoke(query).tool_calls  
  
[{'name': 'multiply',  
 'args': {'a': 3, 'b': 12},  
 'id': 'call_1fyhJAbJHuKQe6n0PacubGsL',  
 'type': 'tool_call'},  
 {'name': 'add',  
 'args': {'a': 11, 'b': 49},  
 'id': 'call_fc2jVkJzwuPWyU7kS9qn1hyG',  
 'type': 'tool_call'}]
```

```
# The function name, type hints, and docstring are all part of the tool  
# schema that's passed to the model. Defining good, descriptive schemas  
# is an extension of prompt engineering and is an important part of  
# getting models to perform well.  
def add(a: int, b: int) -> int:  
    """Add two integers.  
  
    Args:  
        a: First integer  
        b: Second integer  
    ....  
    return a + b  
  
def multiply(a: int, b: int) -> int:  
    """Multiply two integers.  
  
    Args:  
        a: First integer  
        b: Second integer  
    ....  
    return a * b
```

https://python.langchain.com/docs/how_to/tool_calling/

https://python.langchain.com/docs/how_to/custom_tools/

Invocar a procedimientos remotos

Los LLM pueden acceder a información y ejecutar tareas en el mundo real a través de la invocación de procedimientos remotos.

Estas invocaciones se realizan mediante APIs, como la API de Moodle, que permite obtener información sobre las actividades del campus virtual.

El LLM envía una solicitud a la API, procesa los datos recibidos y proporciona una respuesta al usuario.



Open API
Specification

Swagger

Invocar a procedimientos remotos

1 **Evaluar la consulta**

Se analiza si la consulta del usuario se puede resolver mediante la API documentada.

2 **Generar un plan de llamadas API**

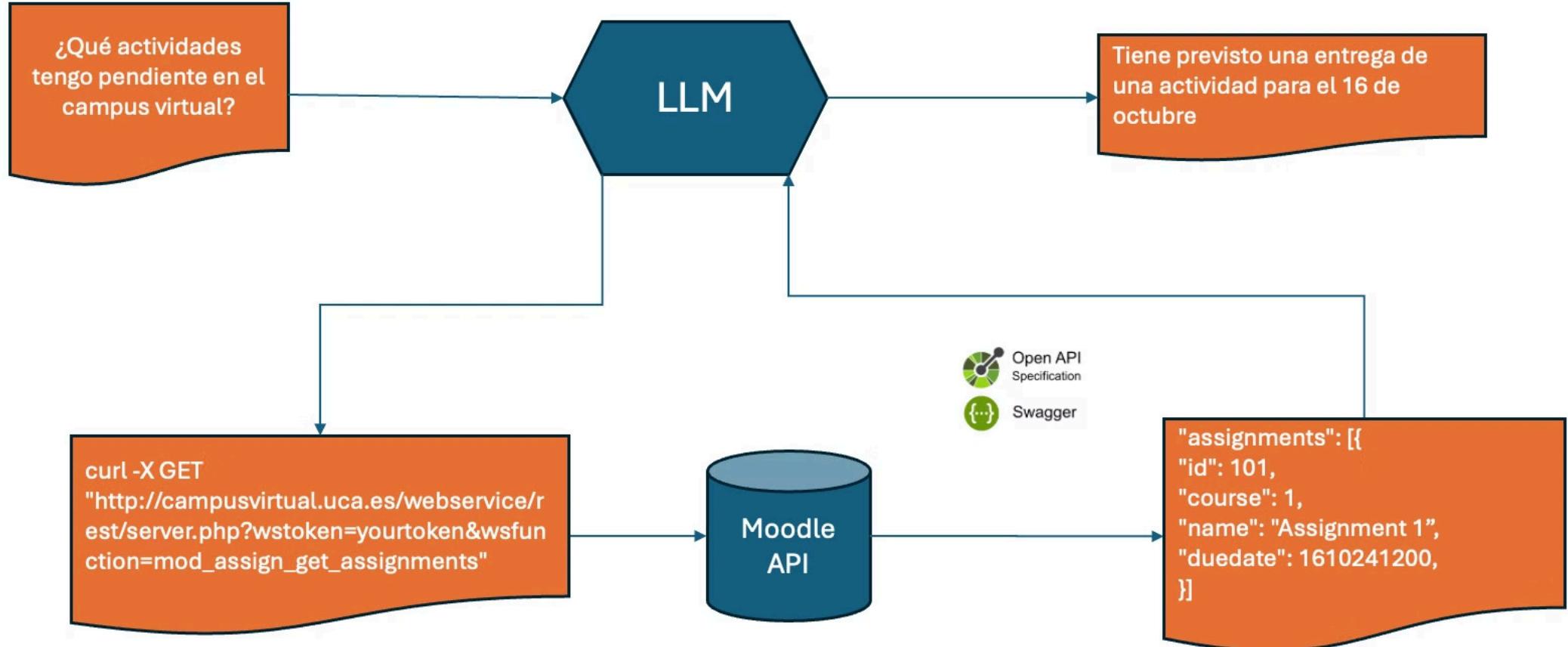
Si la consulta se puede resolver, se crea un plan que describe cada llamada API y su objetivo.

3 **Gestionar acciones sensibles**

Si el plan incluye una llamada DELETE, se solicita primero la autorización al usuario.



Flujo de invocación a procedimientos remotos



OpenAPI tool system prompt

You should:

- 1) evaluate whether the user query can be solved by the API documented below. If no, say why.
- 2) if yes, generate a plan of API calls and say what they are doing step by step.
- 3) If the plan includes a DELETE call, you should always return an ask from the User for authorization first unless the User has specifically asked to delete something.

You should only use API endpoints documented below ("Endpoints you can use:").

You can only use the DELETE tool if the User has specifically asked to delete something. Otherwise, you should return a request authorization from the User first.

Some user queries can be resolved in a single API call, but some will require several API calls.

The plan will be passed to an API controller that can format it into web requests and return the responses.

Here are some examples:

Fake endpoints for examples:

GET /user to get information about the current user

GET /products/search search across products

POST /users/{{id}}/cart to add products to a user's cart

PATCH /users/{{id}}/cart to update a user's cart

PUT /users/{{id}}/coupon to apply idempotent coupon to a user's cart

DELETE /users/{{id}}/cart to delete a user's cart

Ejemplo de llamada a procedimiento remoto

```
spotify_agent = planner.create_openapi_agent(  
    spotify_api_spec,  
    requests_wrapper,  
    llm,  
    allow_dangerous_requests=ALLOW_DANGEROUS_REQUEST,  
)  
user_query = (  
    "make me a playlist with the first song from kind of blue. call it machine  
blues."  
)  
spotify_agent.invoke(user_query)
```

<https://python.langchain.com/docs/integrations/tools/openapi/>

Generar e interpretar código



Ejecución de código

Los LLM pueden generar código en varios lenguajes de programación, como Python, Java, y JavaScript.



Interpretación de código

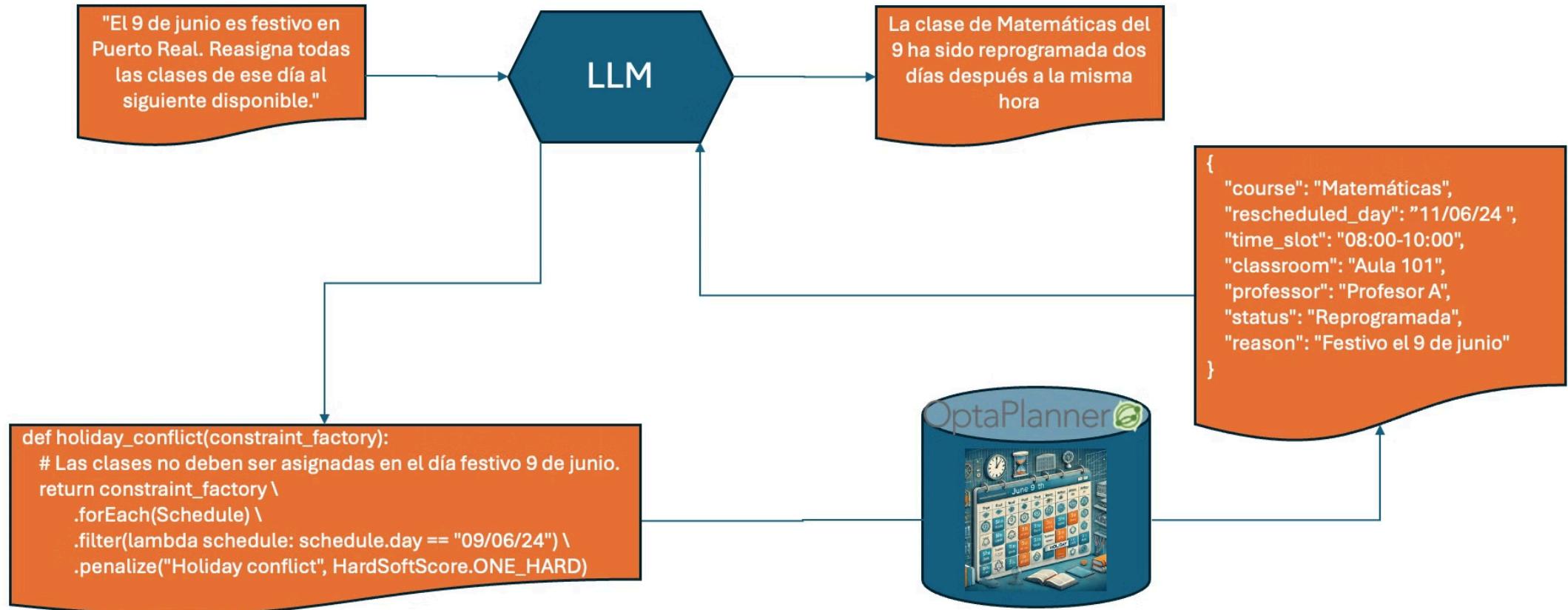
Pueden interpretar y ejecutar código en tiempo real en algún entorno protegido, ofreciendo una forma más dinámica de trabajar con aplicaciones.



Interacción con la lógica de negocio

Pueden interactuar con sistemas existentes mediante código, automatizando tareas y simplificando procesos complejos.

Flujo



Ejemplo de generación e interpretación de código

```
python_repl = PythonREPL()
```

```
python_repl.run("print(1+1)")
```

Python REPL can execute arbitrary code. Use with caution.

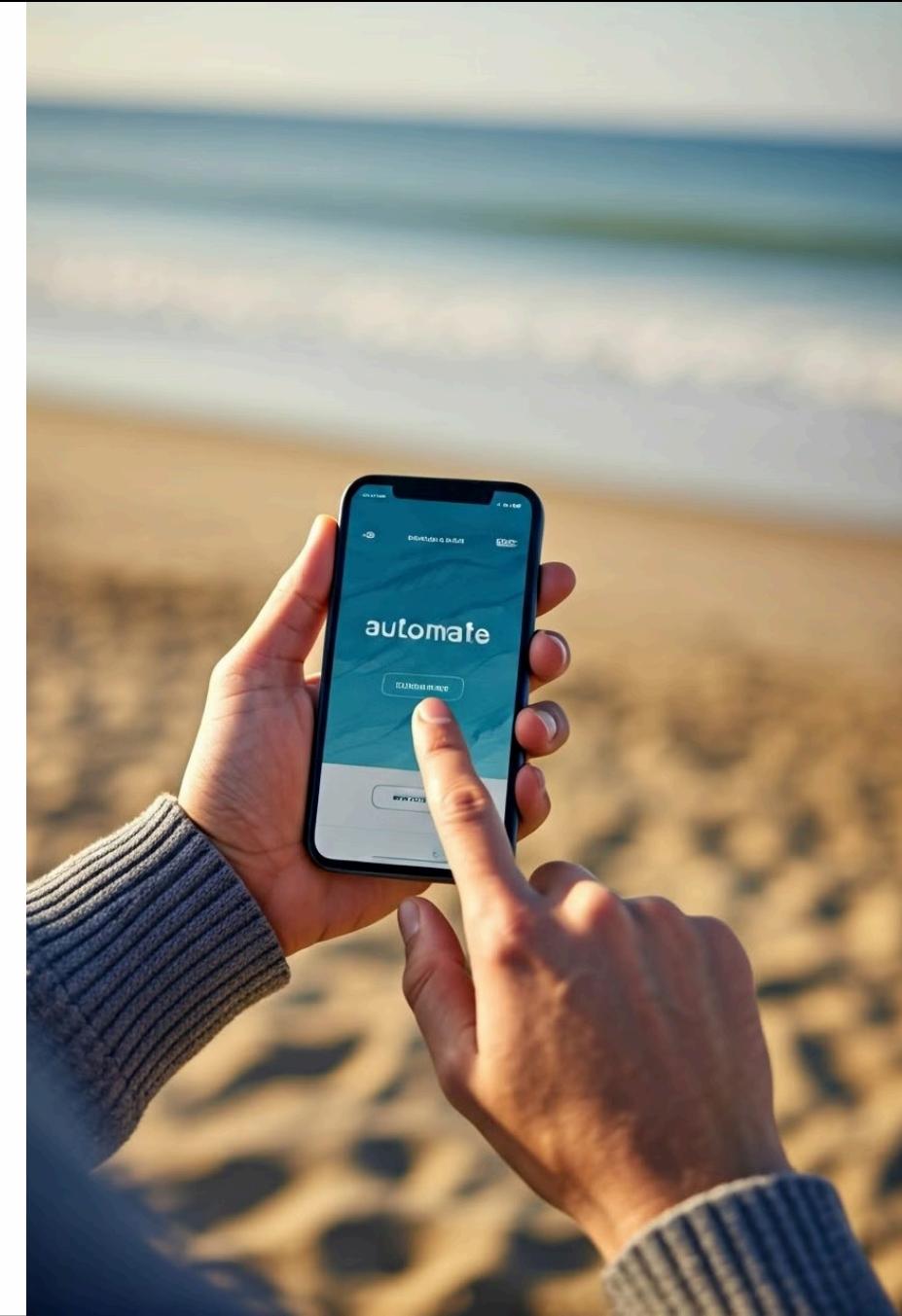
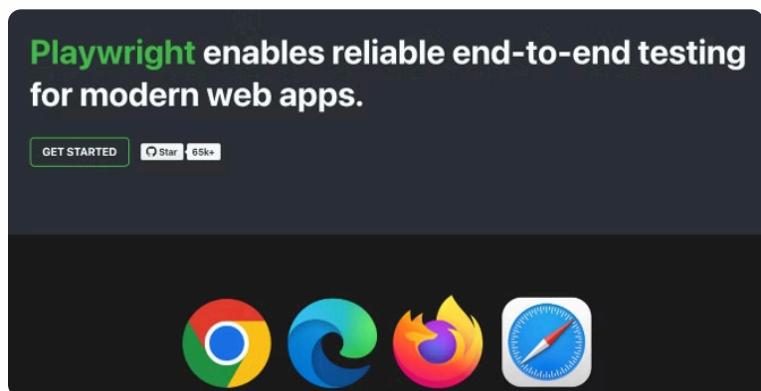
```
'2\n'
```

```
# You can create the tool to pass to an agent
repl_tool = Tool(
    name="python_repl",
    description="A Python shell. Use this to execute python commands. Input
should be a valid python command. If you want to see the output of a value, you
should print it out with 'print(...)'.",
    func=python_repl.run,
)
```

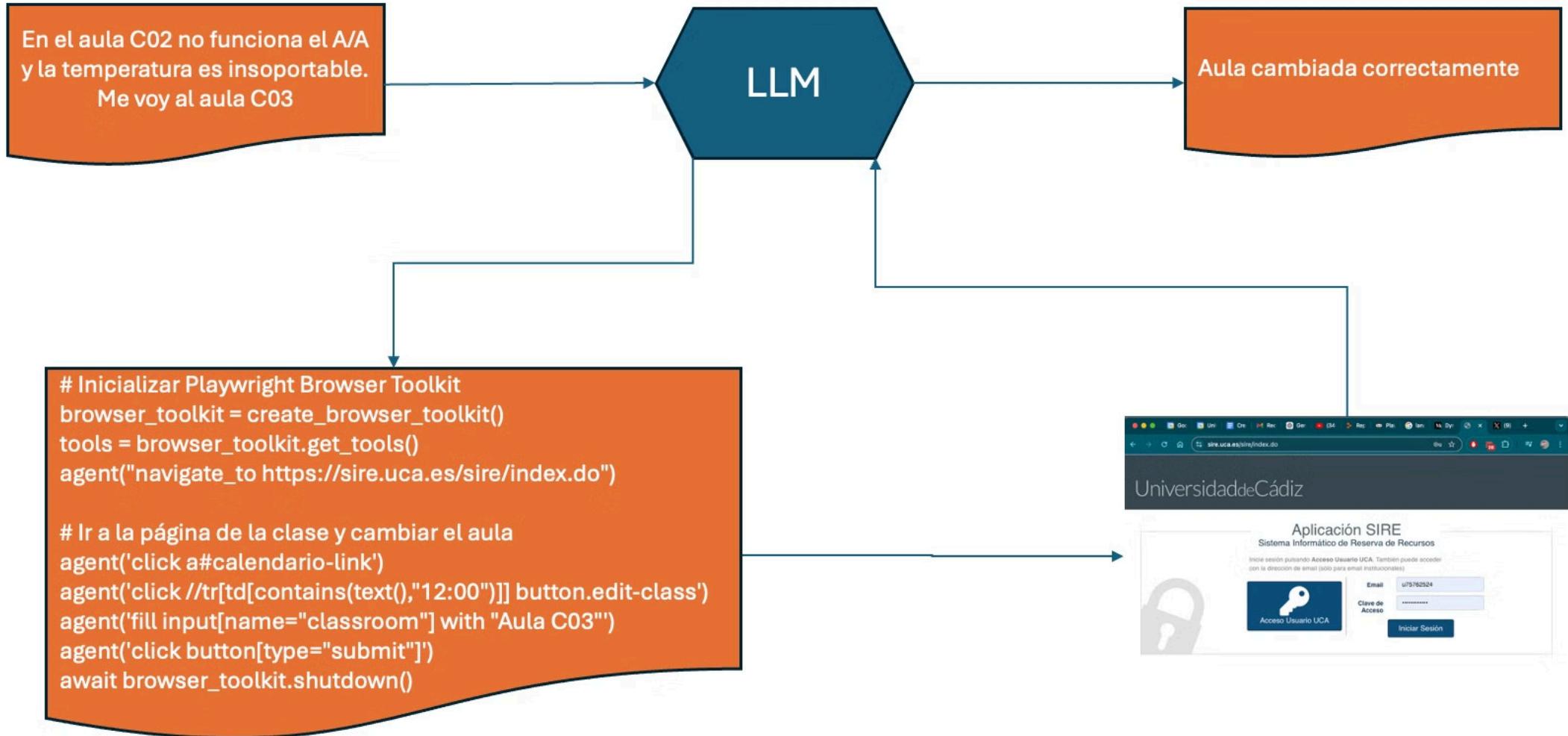
<https://python.langchain.com/docs/integrations/tools/#code-interpreter>

Automatizar procesos

Los LLM son capaces de desencadenar la ejecución de scripts de automatización de tareas. Bibliotecas como Playwright posibilitan la interacción con un navegador web para la recuperación y envío de datos a través de su interfaz.



Flujo



Ejemplo de automatización en el browser

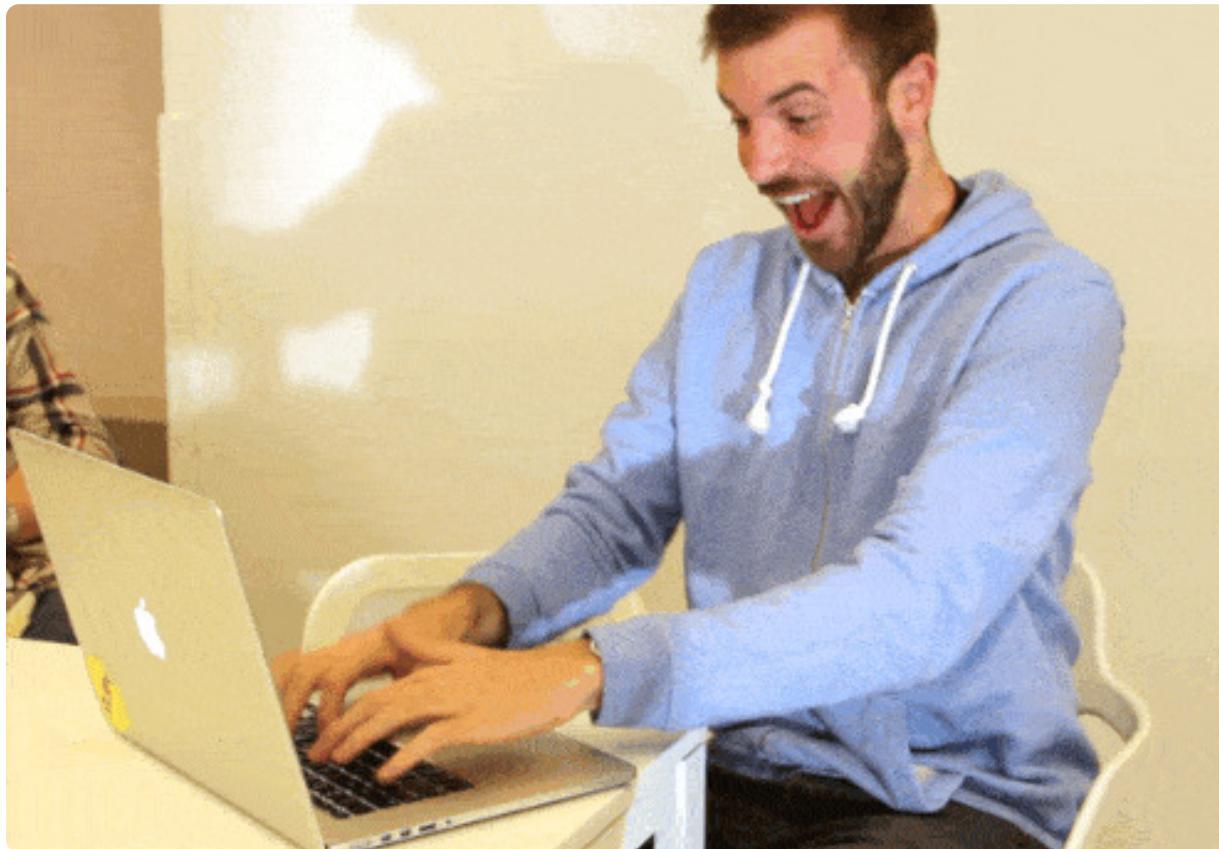
```
async_browser = create_async_playwright_browser()
toolkit = PlaywrightBrowserToolkit.from_browser(async_browser=async_browser)
tools = toolkit.get_tools()
tools
```

```
[ClickTool(async_browser=<Browser type=<BrowserType name=chromium executable_path=
  NavigateTool(async_browser=<Browser type=<BrowserType name=chromium executable_path=
  NavigateBackTool(async_browser=<Browser type=<BrowserType name=chromium executable_path=
  ExtractTextTool(async_browser=<Browser type=<BrowserType name=chromium executable_path=
  ExtractHyperlinksTool(async_browser=<Browser type=<BrowserType name=chromium executable_path=
  GetElementsTool(async_browser=<Browser type=<BrowserType name=chromium executable_path=
  CurrentWebPageTool(async_browser=<Browser type=<BrowserType name=chromium executable_path=
```

```
result = await agent_chain.arun("What are the headers on langchain.com?")
print(result)
```

<https://python.langchain.com/docs/integrations/tools/playwright/>

Ej 1. Creación de un copilot





Colaboración entre Agentes

Consideraciones al trabajar con Agentes Complejos



Escenarios Múltiples

Intentar cubrir todos los casos en un solo prompt suele generar resultados subóptimos. Es mejor segmentar y especializar.

Información Adicional

No siempre es necesario proporcionar datos extra al LLM en cada iteración. Puede aumentar costos y reducir eficiencia.

Acceso a Herramientas

Limitar el acceso del LLM a herramientas específicas mejora la seguridad y eficiencia del sistema.

Equilibrio de Creatividad

La temperatura del LLM debe ajustarse según la tarea: baja para precisión, alta para creatividad.

Divide y vencerás

Habilidades Diversas

Cada agente se especializa en una tarea específica, lo que optimiza su rendimiento mediante prompts personalizados.

Diferentes LLM

Se selecciona el modelo de lenguaje más adecuado para cada tarea, lo que mejora la eficiencia global del sistema.

Acceso Controlado

Se asignan estratégicamente las herramientas y los recuperadores de información a cada agente, lo que aumenta la seguridad y el rendimiento.



Encadenamiento de Agentes



Ejecución Secuencial

Los agentes operan en serie, pasando resultados de uno a otro para tareas complejas.



Ejecución Condicional

Se implementan lógicas de decisión para activar agentes específicos según las circunstancias.



Ejecución Iterativa

Los bucles permiten refinar resultados o procesar grandes volúmenes de datos eficientemente.



Control del Desarrollador

El programador mantiene el control del flujo, integrando LLMs como componentes modulares.



Agentes Autónomos

Autonomía Total

Los LLMs dirigen la aplicación, tomando decisiones complejas sin intervención humana constante.

Jerarquía de Agentes

Los agentes pueden actuar como herramientas para otros, creando sistemas altamente sofisticados.

Sistemas Avanzados

Incluyen bucles, condiciones y persistencia para tareas complejas y aprendizaje continuo.

Human-in-the-loop

Se mantiene la opción de supervisión humana para aprobar o cancelar acciones críticas.

Workflows

Están surgiendo herramientas y frameworks, como LangGraph, que permiten modelar los agentes mediante diagramas de flujo

Arquitectura Reflexión (self-reflection)

Análisis de Razonamiento

El LLM examina sus propias trazas de razonamiento, para detectar errores o inconsistencias.

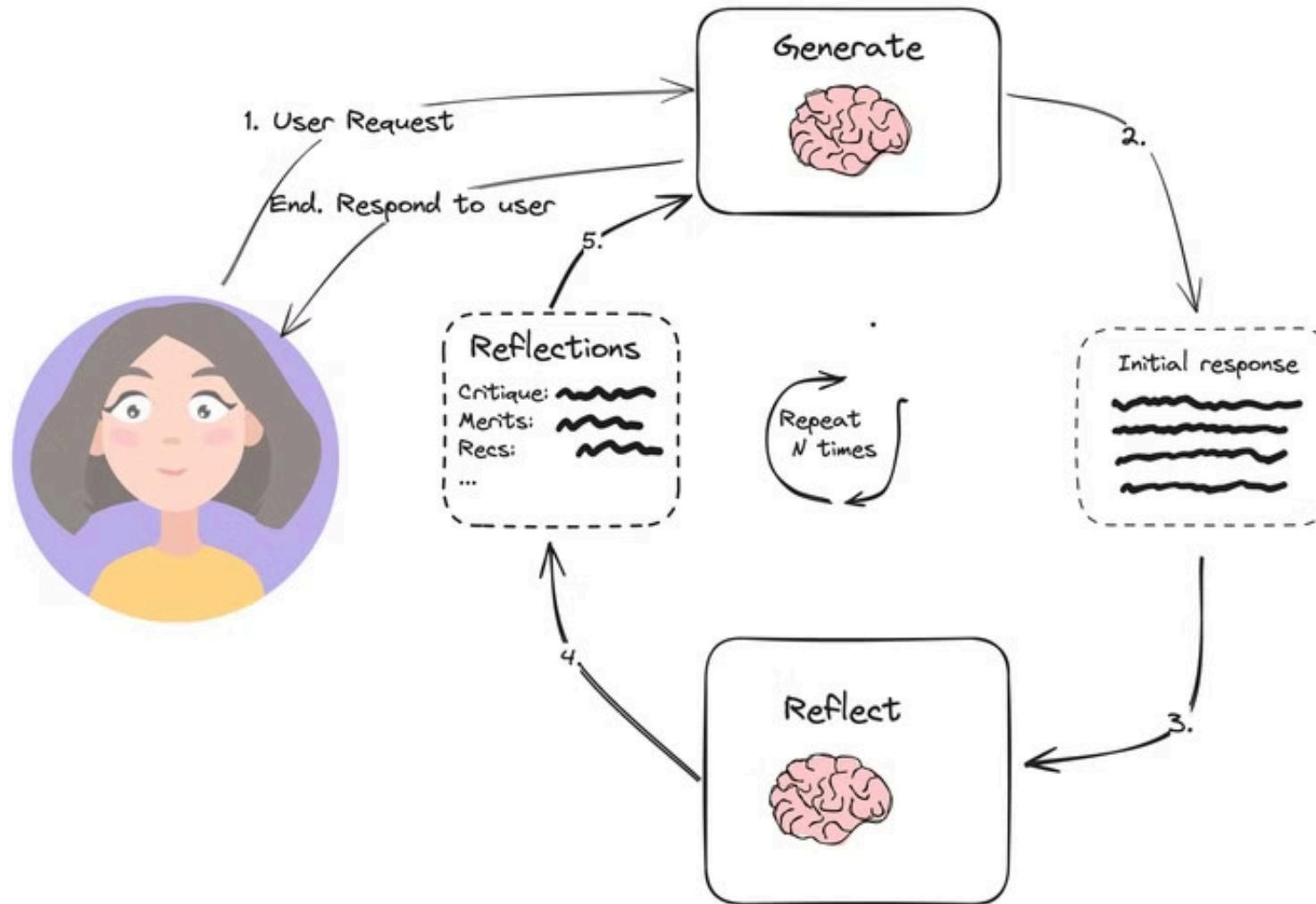
Ajuste de Estrategia

Basándose en la reflexión, el agente modifica su enfoque para futuras acciones.

Mejora Continua

Este ciclo permite una evolución constante del agente, aumentando su eficacia.

Basic Reflection



Arquitectura Plan-and-Execute

Planner

Responsable de enviar un prompt al LLM para generar un plan basado en multiples pasos planes

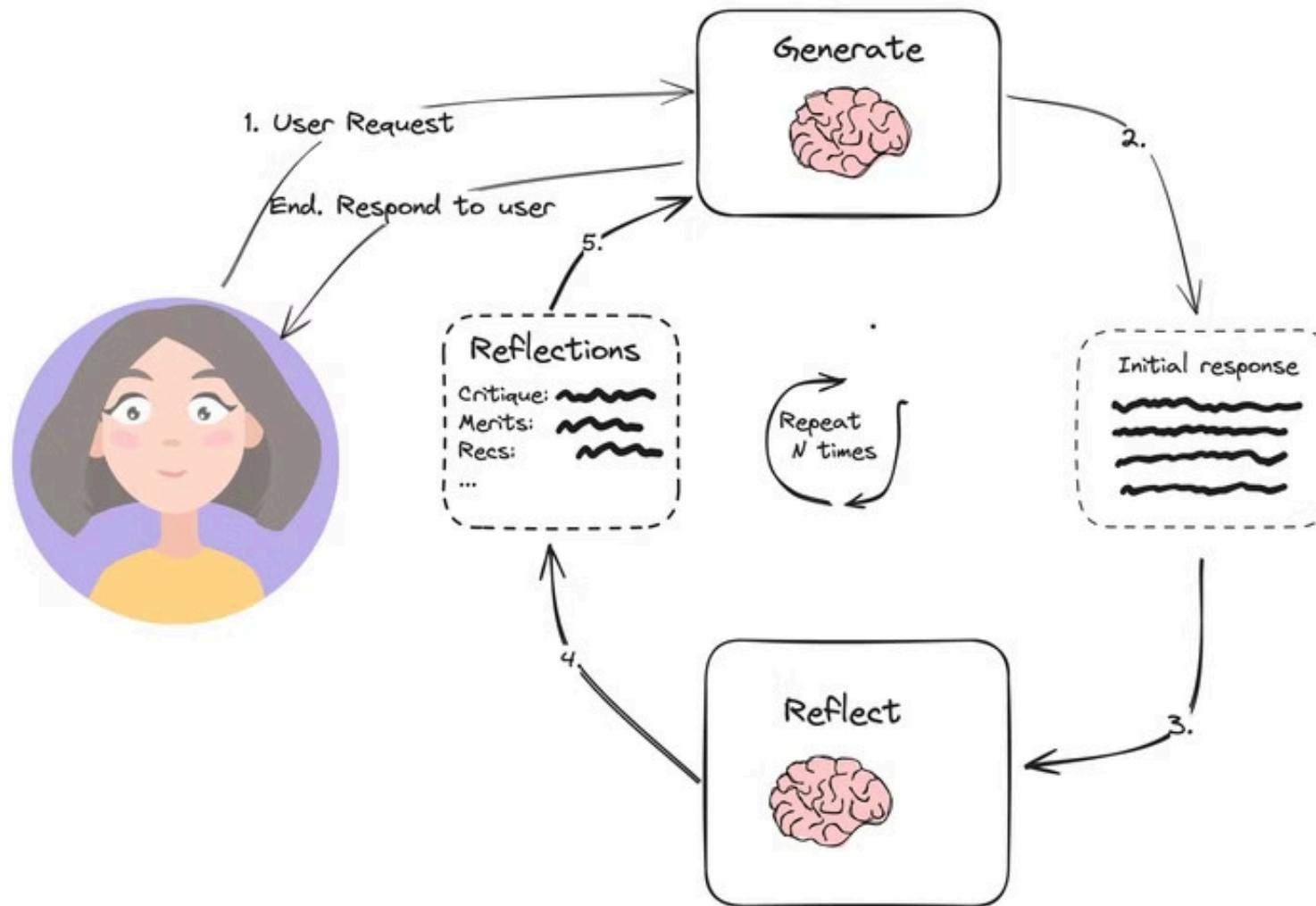
Executor

Ejecuta cada paso del plan utilizando herramientas específicas para realizar la acción.

Agentes planificadores

Existen otras arquitecturas con propósitos similares: ReAct, ReWOO, LLM Compiler

Basic Reflection



Resumen

Esta presentación ha explorado el panorama de los agentes LLM, desde sus orígenes hasta la actualidad.

Hemos profundizado en la autonomía creciente de los agentes, examinando arquitecturas como la reflexión y la planificación-ejecución.

Se han destacado las herramientas y plataformas que habilitan el desarrollo y la aplicación de agentes LLM.