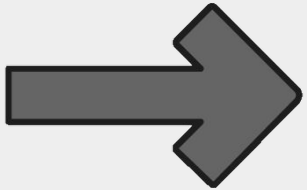
Google Developer Group
Universidad Autónoma de Madrid

Despliega tus agentes de IA con Google ADK

Google Developer Group on Campus UAM



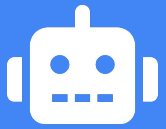
¿Qué es un agente de IA?

LLM

Razonamiento puro, pero aislado.

"Cerebro en un frasco"

Pensar



Agente

Razonamiento + Herramientas + Memoria.

Ejecuta en el mundo real

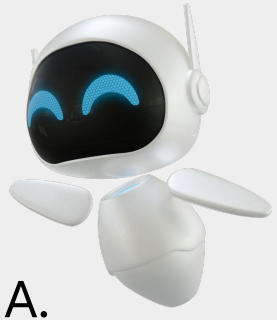
Actuar

¿Qué es un agente de IA?

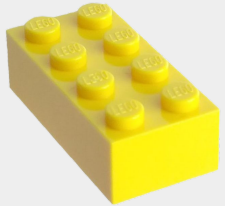
+

“SOLO ES UNA CAPA DE SOFTWARE POR ENCIMA DE UN MECANISMO QUE SIMULA EL RAZONAMIENTO”

¿Qué es ADK?



Es un **Framework** que permite desarrollar agentes de IA.
Plataforma que simplifica el código para construir agentes de IA



El código es el prompt y sus herramientas.

Puedes crear sistemas multiagénticos y coordinarlos como piezas de Lego.

Define el comportamiento con lógica Python y objetos estructurados (también Ts, Go o Java).

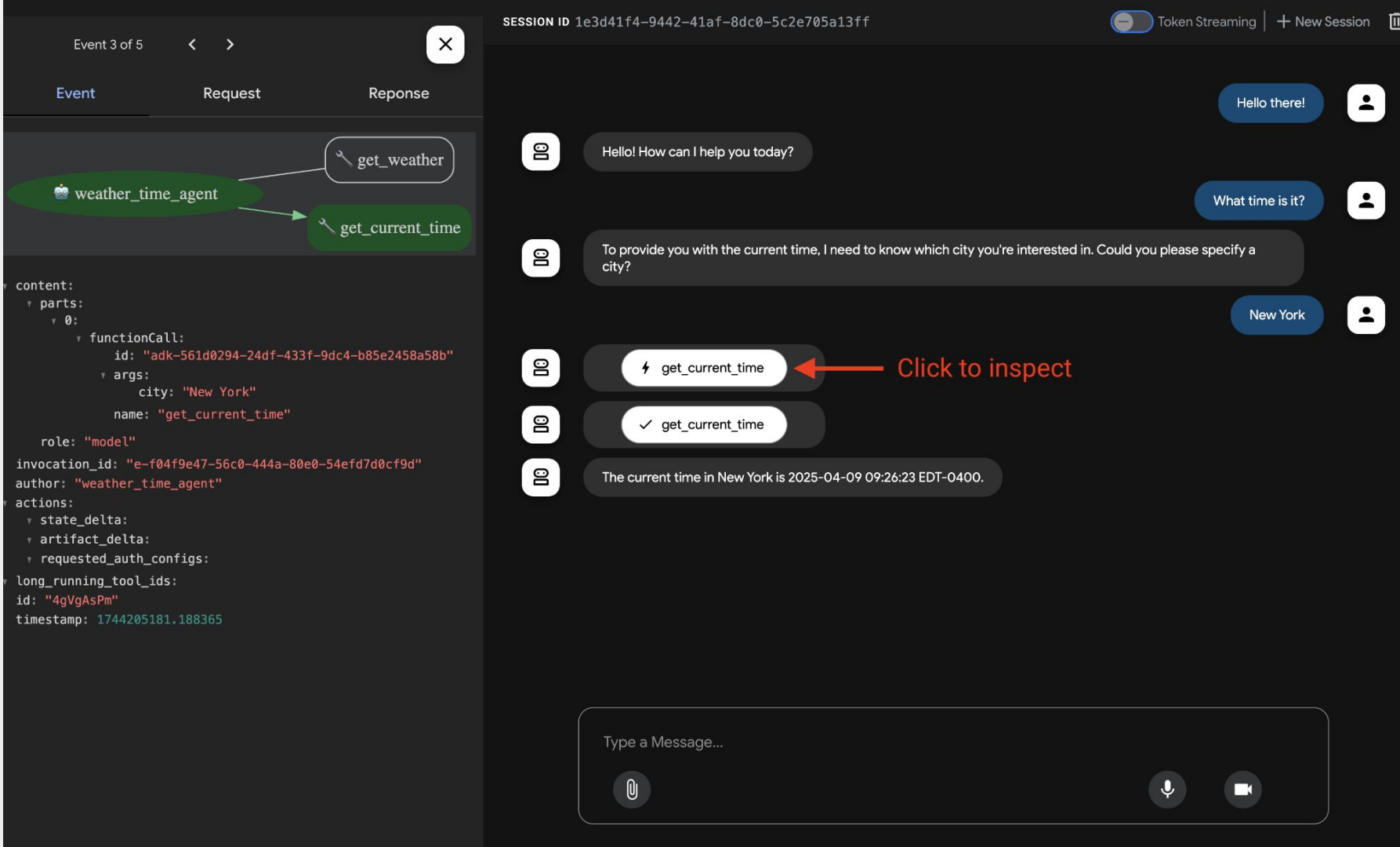
¿Cómo?

1. Instala las librerías en Python y crea tu agente

```
pip install google-adk
adk create my_agent
```

2. Lanza tu agente

```
adk run my_agent     # o adk web
```


Event 3 of 5
Event
Request
Reponse
get_weather
weather_time_agent
get_current_time
content:
parts:
0:
functionCall:
id: "adk-561d0294-24df-433f-9dc4-b85e2458a58b"
args:
city: "New York"
name: "get_current_time"
role: "model"
invocation_id: "e-f04f9e47-56c0-444a-80e0-54efd7d0cf9d"
author: "weather_time_agent"
actions:
state_delta:
artifact_delta:
requested_auth_configs:
long_running_tool_ids:
id: "4gVgAsPm"
timestamp: 1744205181.188365
SESSION ID 1e3d41f4-9442-41af-8dc0-5c2e705a13ff
Token Streaming
New Session
Hello there!
Hello! How can I help you today?
What time is it?
To provide you with the current time, I need to know which city you're interested in. Could you please specify a city?
New York
get_current_time
Click to inspect
get_current_time
The current time in New York is 2025-04-09 09:26:23 EDT-0400.
Type a Message...

El Agente en acción: conceptos

Tools: cómo actuar

Las "manos" del agente. Funciones Python puras que ejecutan acciones.

- Consultas a Bases de Datos
- Búsquedas en tiempo real
- Cálculos matemáticos o uso de modelos

MCP: cómo pedir

Model Context Protocol: El estándar universal de conexión.

- Interoperabilidad Total
- Sin reescribir código
- Conexión "Plug & Play"

```
@tool
def mi_funcion(data): return "Acción ejecutada con éxito"
```

El Agente en acción: conceptos

Tools: cómo actuar

MCP: cómo pedir

¡Trabajan juntas! El modelo pide y el software ejecuta

Diseño de un agente

Aislamiento

Cada agente opera en su propio ámbito, reduciendo el ruido y las alucinaciones.

Modularidad

Añade o quita agentes especialistas sin afectar la lógica del orquestador central.

Interoperabilidad

Uso de protocolos estándar (MCP) para conectar herramientas de forma universal.

Construyendo el Agente

agent_definition.py

```python
@tool
def get_news(topic: str):
    # Lógica de búsqueda real
    return search_api.run(topic)

Class NewsAgent(Agent):
    name = "Investigador"
    model = "gemini-1.5-pro"
    tools = [get_news]
    system_prompt = "Eres un experto..."
```

Clase Agent

Define el modelo, el nombre y las herramientas disponibles en un solo objeto.

System Prompt

Instrucciones de rol claras y concisas que guían el razonamiento del agente.

Orquestación

ADK gestiona automáticamente los turnos y la invocación de herramientas.

Orquestación de agentes

El Equipo Experto: Especialización Multi-Agente.
Ejemplo para Newsletter del GDG

Orquestador

Analiza la intención, planifica y delega.

Coordinación

Investigador

Experto en datos, llama a tools de búsqueda.

Especialización

Redactor

Experto en redactar artículos interesantes y llamativos

Especialización

Orquestación: código

multi_agent_system.py

```
# Definición de expertos
investigador = Agent(name="Searcher", tools=[search])
redactor = Agent(name="Writer", tools=[refine])

# El Orquestador central
manager = ManagerAgent(
    agents=[investigador, redactor],
    strategy="collaborative",
    system_prompt="Coordina la investigación..."
)

# Ejecución compartiendo contexto
response = manager.run("Analiza el mercado de IA")
```

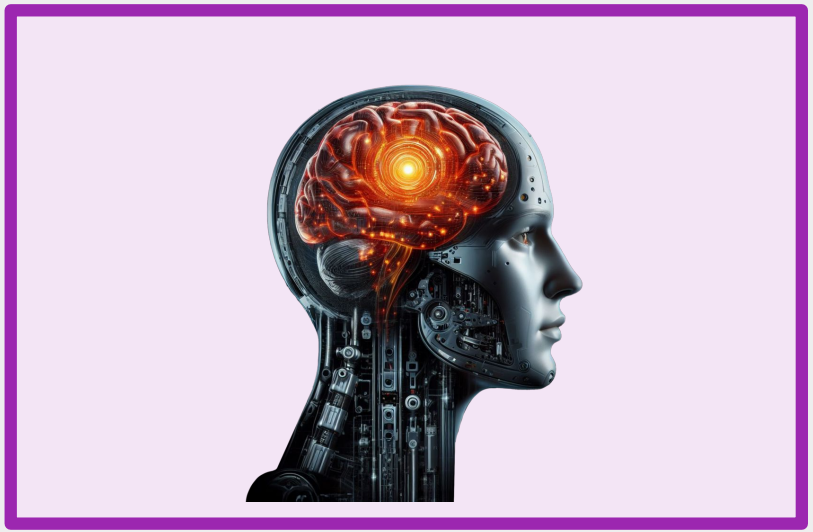
Memoria compartida

ADK gestiona automáticamente el paso de mensajes y estado entre agentes.

Estrategias

Define cómo colaboran: secuencial, jerárquica o colaborativa libre.

El agente que recuerda: memoria



Short-term Memory

Historial de conversación para coherencia inmediata y referencias contextuales.

Long-term Memory

Bases de datos vectoriales para conocimiento persistente y aprendizaje entre sesiones (e.g: RAG)

Context Window

Gestión inteligente de tokens para maximizar la precisión y reducir costes.

Escalabilidad > Complejidad

Monitorizando el Pensamiento

Trazas de Ejecución

Visualiza qué herramientas llamó cada agente y el razonamiento detrás de cada acción.

Gestión de Estado

Persistencia de sesiones para que el agente mantenga el hilo conductor entre llamadas API.

Evaluación Continua

Detección automática de alucinaciones, bucles infinitos o respuestas fuera de tono.

Logs de Decisión

Registro detallado de por qué el orquestador delegó una tarea a un experto específico.

Observabilidad = Confianza en producción

Despliegue: de tu laptop a la nube

Empaquetado

Contenedores inmutables para portabilidad total entre entornos.

Dependencias

Aislamiento completo de librerías y versiones de Python.

Serverless

Escalado automático a cero cuando no hay peticiones activas.

Alta Disp.

Gestión nativa de tráfico y versiones (Blue/Green).

API REST

Expón tu agente ADK como un servicio consumible por cualquier app.

Seguridad

Gestión de secretos y permisos mediante Google IAM.

Escalabilidad > Complejidad

¡Regístrate!

Nuestra Web

Comunidad

Código fuente

Accede al repositorio del Taller

https://github.com/GDG-UAM/workshop-googleadk

Muchas gracias y bienvenidos a la comunidad.

Google Developer Group
Universidad Autónoma de Madrid