

# Curso Agentes

2ª Ed. (Día 3)

# OpenAI Agents SDK

<https://openai.github.io/openai-agents-python/>



<https://cdn.openai.com/business-guides-and-resources/a-practical-guide-to-building-agents.pdf>

# OpenAI Agents SDK: Principales Características

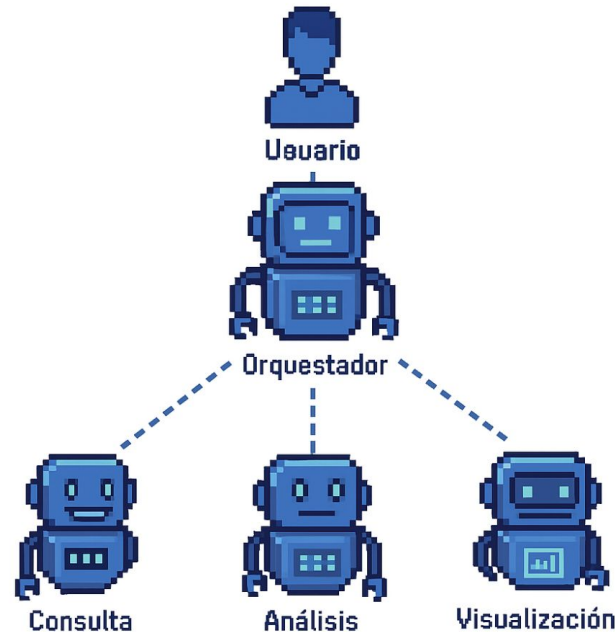
- **Python-first**: características nativas de Python en lugar de introducir nuevas capas de abstracción.
- **Agent Loop**: loop integrado, con llamadas a herramientas, envío de resultados al LLM ...
- **Sessions**: gestión automática del historial de la conversación (sin tener que hacerlo manualmente)
- **Multi-Agentes**: Patrón orquestador y handoff (coordinación y delegación entre múltiples agentes).
- **Guardrails**: Validación de entrada/salida
- **Function Tools**: Funciones Python como herramientas
- **Trazabilidad**: Visualización, depuración y monitorización de los pasos del agente.

# OpenAI Agents SDK:

Ejemplos básicos de prueba de los principales componentes: [openai-agents-1.ipynb](#)

# OpenAI Agents SDK: Sistemas Multi-Agente

## ✓ Orquestador-Subagente



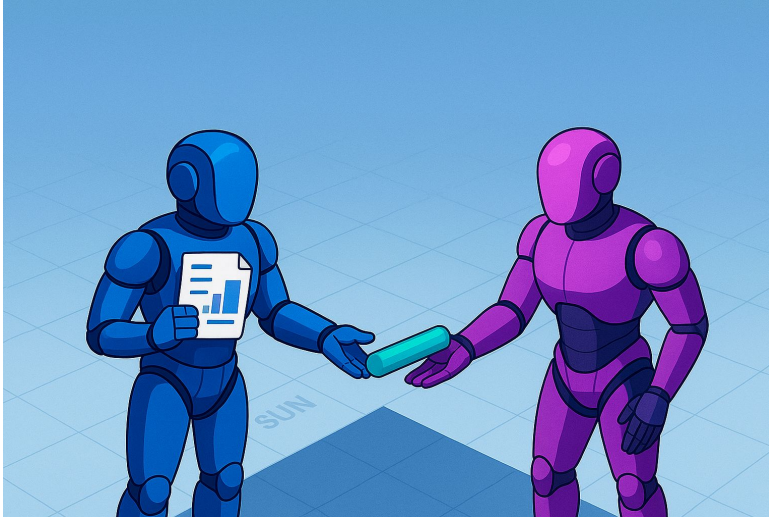
## Patrón Orquestador

<https://colab.research.google.com/drive/1lobqWUOTWLI-IoRvMHke4f9d-eqBtk7H?usp=sharing>

# OpenAI Agents SDK: Sistemas Multi-Agente



# OpenAI Agents SDK: Sistemas Multi-Agente



Patrón Handoff

[https://colab.research.google.com/drive/1TSmfJU10a07RJQZw7uNEyXZdJQrQW\\_EL?usp=sharing](https://colab.research.google.com/drive/1TSmfJU10a07RJQZw7uNEyXZdJQrQW_EL?usp=sharing)

**Ejemplo básico:**

<https://colab.research.google.com/drive/1ncP-SxA9ZZYxk1R0pk8x9GbeCOwbhAhE?usp=sharing>

# Trazabilidad de agentes con LangFuse

1. Date de alta en LangFuse
2. Obtén un API Key
3. Prueba la integración básica con OpenAI API:  
<https://github.com/juananpe/openai-langfuse>
4. Prueba la integración con OpenAI Agents SDK:

**Notebook:**


[https://colab.research.google.com/drive/1v\\_A\\_7o8MgHDbUp20UU1dz7dH1Uqxd86d?usp=sharing](https://colab.research.google.com/drive/1v_A_7o8MgHDbUp20UU1dz7dH1Uqxd86d?usp=sharing)

Más info: <https://langfuse.com/integrations/frameworks/openai-agents>



# OpenAI Agents SDK: Sistemas Multi-Agente

## Traces

 Enter id to view details

 Workflow Search...

 Group Search...

Workflow	Flow	Handoffs	Tools	Execution time	Created
● Agent workflow	Agente Principal ▶ Agente de Consulta de Datos ▶ Agente de Visualizac	2	3	20.82s	May 26, 2025, 10:57 PM
● Agent workflow	Agente Principal ▶ Agente de Consulta de Datos ▶ Agente de Visualizac	2	1	8.27s	May 26, 2025, 10:57 PM
● Agent workflow	Agente Principal ▶ Agente de Consulta de Datos ▶ Agente de Visualizac	2	2	27.15s	May 26, 2025, 10:52 PM

### Ejercicio AVANZADO:

Modifica el script de los agentes con patrón handoff para que se envíen entre ellos las tareas que no sepan resolver directamente

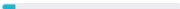
< Traces / Agent workflow

trace\_e95b01fa92d241...

Refresh


Agente Principal

1,506 ms




POST /v1/responses

1,500 ms




get\_current\_date

0 ms




Handoff ▶ Agente de Consulta de Datos

1 ms




Agente de Consulta de Datos

4,625 ms




POST /v1/responses

1,354 ms




lookup\_sales\_data

967 ms




POST /v1/responses

2,302 ms




Handoff ▶ Agente de Visualización de Datos

0 ms



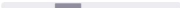
Agente de Visualización de Datos

14.69 s




POST /v1/responses

3,094 ms




generate\_visualization

8,331 ms



POST /v1/responses

3,262 ms



Properties

ID

trace\_e95b01fa92d2413b8ffddde7...

Workflow name

Agent workflow

Metadata

No metadata entries

# OpenAI Agents SDK: Soporte MCP



## OpenAI Agents SDK

Intro

Quickstart

Examples

Documentation

Agents

Running agents

Results

Streaming

Tools

Model context protocol (MCP)

Handoffs

Tracing

Context management

Guardrails

Orchestrating multiple agents

Models

Models

Using any model via LiteLLM

Configuring the SDK

Agent Visualization

Voice agents

Quickstart

Pipelines and workflows

Tracing

API Reference

## Model context protocol (MCP)

The [Model context protocol](#) (aka MCP) is a way to provide tools and context to the LLM. From the MCP docs:

MCP is an open protocol that standardizes how applications provide context to LLMs. Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect your devices to various peripherals and accessories, MCP provides a standardized way to connect AI models to different data sources and tools.

The Agents SDK has support for MCP. This enables you to use a wide range of MCP servers to provide tools to your Agents.

## MCP servers

Currently, the MCP spec defines two kinds of servers, based on the transport mechanism they use:

1. **stdio** servers run as a subprocess of your application. You can think of them as running "locally".
2. **HTTP over SSE** servers run remotely. You connect to them via a URL.
3. **Streamable HTTP** servers run remotely using the Streamable HTTP transport defined in the MCP spec.

You can use the `MCPServerStdio`, `MCPServerSse`, and `MCPServerStreamableHttp` classes to connect to these servers.

For example, this is how you'd use the [official MCP filesystem server](#).

## Table of contents

MCP servers

Using MCP servers

Caching

End-to-end examples

Tracing

**Ejercicio:** ¿sabrías explicar qué es lo que hace este script?

[https://github.com/openai/openai-agents-python/blob/main/examples/mcp/filesystem\\_example/main.py](https://github.com/openai/openai-agents-python/blob/main/examples/mcp/filesystem_example/main.py)

**Ejercicio:** ¿sabrías probarlo? (sigue las siguientes indicaciones)

1. Clona openai-agents-python
2. `uv venv`
3. `source .venv/bin/activate`
4. `uv sync`
5. `cd examples/mcp/filesystem_example`
6. macos/linux:  
`export OPENAI_API_KEY=sk-xxxxx`
- windows:  
`$env:OPENAI_API_KEY="sk-xxxxx"`
7. `python main.py`

# OpenAI Agents SDK: Soporte MCP

## Ejemplo de ejecución:

```
(openai-agents-python) mcp/filesystem_example  main ± python main.py
Secure MCP Filesystem Server running on stdio
Allowed directories: [
  '/private/tmp/openai-agents-python/examples/mcp/filesystem_example/sample_files'
]
View trace: https://platform.openai.com/traces/trace?trace\_id=trace\_b8c67049193e450faf02223ccd179576

Running: Read the files and list them.
Here are the files in the directory:

- `favorite_books.txt`
- `favorite_cities.txt`
- `favorite_songs.txt`

Running: What is my #1 favorite book?
Your #1 favorite book is "To Kill a Mockingbird" by Harper Lee.

Running: Look at my favorite songs. Suggest one new song that I might like.
Based on your favorites, you might like "Livin' on a Prayer" by Bon Jovi. It's a classic rock anthem with a similar vibe to some of the songs you enjoy.
(openai-agents-python) mcp/filesystem_example  main ±
```

# OpenAI Agents SDK: Soporte MCP

The screenshot displays the OpenAI Agents SDK dashboard with a sidebar on the left containing navigation links: DASHBOARD, Logs, Traces (selected), Assistants, Batches, Evaluations, Fine-tuning, Storage, Usage, and API keys. The main content area is titled 'Traces / MCP Filesystem Example' and shows a list of traces for a specific trace ID 'trace\_b8c67049193e4...'. The traces are organized into two sections, each starting with an 'Assistant' trace. The first section shows a sequence of tool calls: 'List MCP Tools' (2 ms), 'POST /v1/responses' (2,936 ms), 'list\_allowed\_directories' (3 ms), 'POST /v1/responses' (988 ms), 'list\_directory' (4 ms), and 'POST /v1/responses' (1,016 ms). The second section shows: 'List MCP Tools' (2 ms), 'POST /v1/responses' (1,698 ms), 'search\_files' (5 ms), 'POST /v1/responses' (1,147 ms), 'list\_allowed\_directories' (3 ms), 'POST /v1/responses' (1,434 ms), 'search\_files' (5 ms), 'POST /v1/responses' (1,755 ms), 'read\_file' (3 ms), and 'POST /v1/responses' (1,169 ms). To the right of the trace list, there are two 'Function Call' panels. The first panel shows the arguments for 'list\_allowed\_directories()' and its output, which is a JSON object listing allowed directories. The second panel shows the arguments for 'list\_directory()' with a specific path and its output, which is a JSON object listing files in the directory. Below these panels is an 'Output' section showing the assistant's response: 'Here are the files in the directory:' followed by a list of files: 'favorite\_books.txt', 'favorite\_cities.txt', and 'favorite\_songs.txt'.

**Ejercicio:** incluye `cache_tools_list=True` en el constructor de `MCPServerStdio` y compara las trazas de ejecución (latencia)

# OpenAI Agents SDK: Soporte MCP

En <https://github.com/juananpe/openai-agents/> encontrarás el código de una aplicación web básica: frontend con un simple formulario y backend en Flask que recoge los datos del formulario y los guarda en datos.json.

**Ejercicio:** crear un Agente capaz de interactuar con cualquier web y sistema de archivos.

Crea un agente utilizando MCPServerStdio para conectar dos servidores MCP:

- Filesystem MCP: Para que el agente pueda leer archivos locales.

- Playwright MCP: Para que el agente pueda navegar y controlar un navegador web.

Ejecuta el agente en modo interactivo (REPL) y pídele en lenguaje natural que:

- Lea el archivo personas.json.

- Entre en la web local (<http://localhost:5000>).

- Rellene y envíe el formulario para cada persona encontrada en el JSON.

**Cuidado!** El agente NO debe tener ninguna función definida en código (`@function_tool`) para rellenar el formulario. Debe utilizar las herramientas proporcionadas por los servidores MCP.

Observad cómo el LLM utiliza las herramientas genéricas de navegación (*navigate*, *click*, *fill*) para deducir la estructura del formulario y completar la tarea sin haber sido programado explícitamente para esa web.

# OpenAI Agents SDK: Uso de cualquier modelo vía LiteLLM

Crea el siguiente agente y usa `openrouter/anthropic/claude-sonnet-4.5` como LLM:

<https://github.com/juananpe/open>