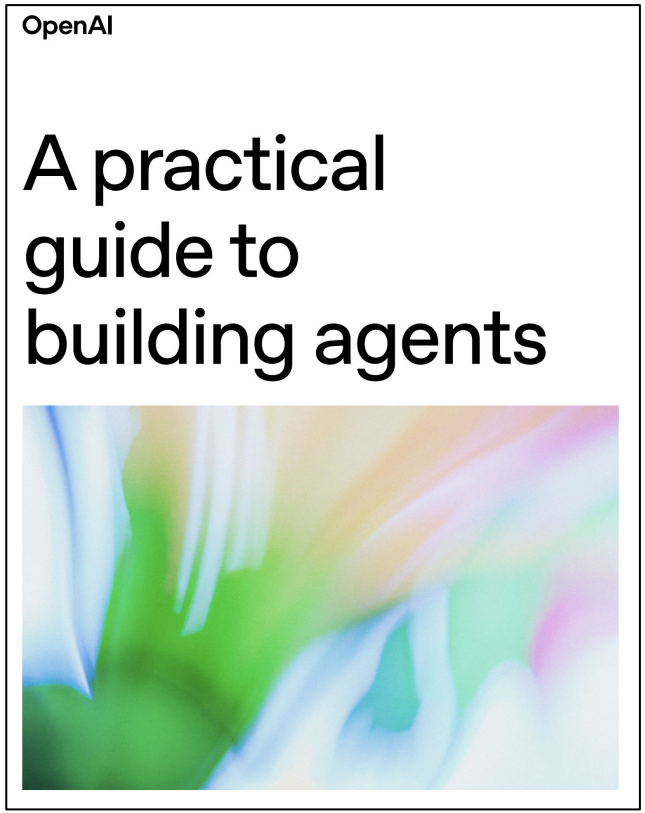
# Curso Agentes

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

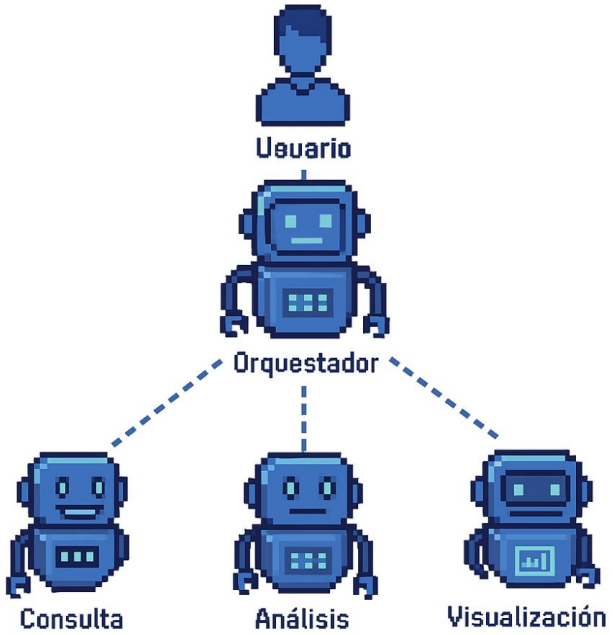
- Python-first: características nativas de Python en lugar de introducir nuevas capas de abstracción.
- Agent Loop: loop integrado, con llamadas a herramientas, envío de resultados al LLM ...
- Sessions: gestión automática del historial de la conversación (sin tener que hacerlo manualmente)
- Multi-Agentes: 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/1IobqWUOTWLI-IoRvMHke4f9d-eqBtk7H?usp=sharing

# OpenAI Agents SDK: Sistemas Multi-Agente

Problems Output Debug Console Terminal Ports zsh

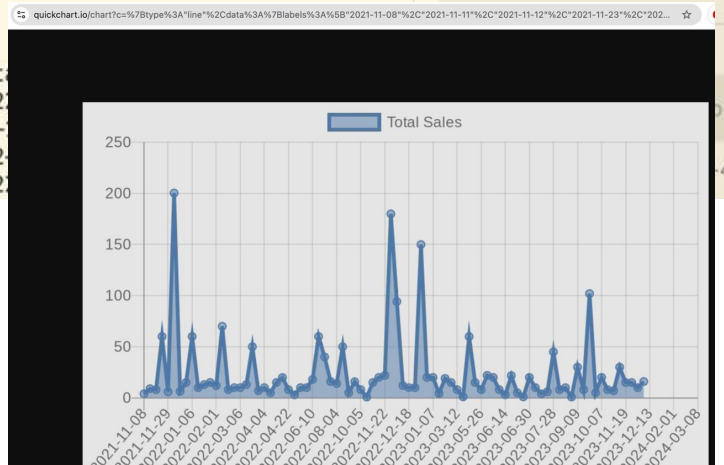
```
[TRACE] Ejecutando consulta SQL en DuckDB...
[TRACE] Consulta ejecutada. Filas resultantes: 93



Salida final de la consulta 2:
Aquí tienes la visualización que muestra la evolución de las ventas a lo largo del tiempo para la tienda con *
*Store_Number** igual a **2420**.

### Gráfico de Evolución de Ventas

![Evolución de Ventas](https://quickchart.io/chart?c=%7Btype%3A%22line%22%2Cdat
8%22%2C%222021-11-11%22%2C%222021-11-12%22%2C%222021-11-23%22%2C%222021-11-29%2
12-26%22%2C%222022-01-03%22%2C%222022-01-06%22%2C%222022-01-13%22%2C%222022-01-
022-02-01%22%2C%222022-02-23%22%2C%222022-02-25%22%2C%222022-03-05%22%2C%222022
%222022-03-16%22%2C%222022-03-19%22%2C%222022-04-04%22%2C%222022-04-07%22%2C%22
```

en tu entorno virtual.

¿Quieres que ejecute el comando
ti?

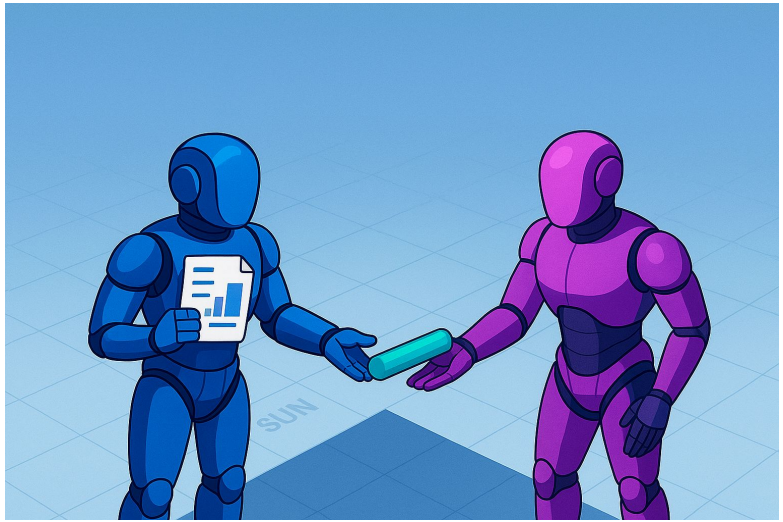
mention code

4.1 (promo)




Cuidado con los agentes... intentarán resolver tu query como sea (aunque no estén conectados... y queremos conectarlos entre sí)

# OpenAI Agents SDK: Sistemas Multi-Agente




Patrón Handoff

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:

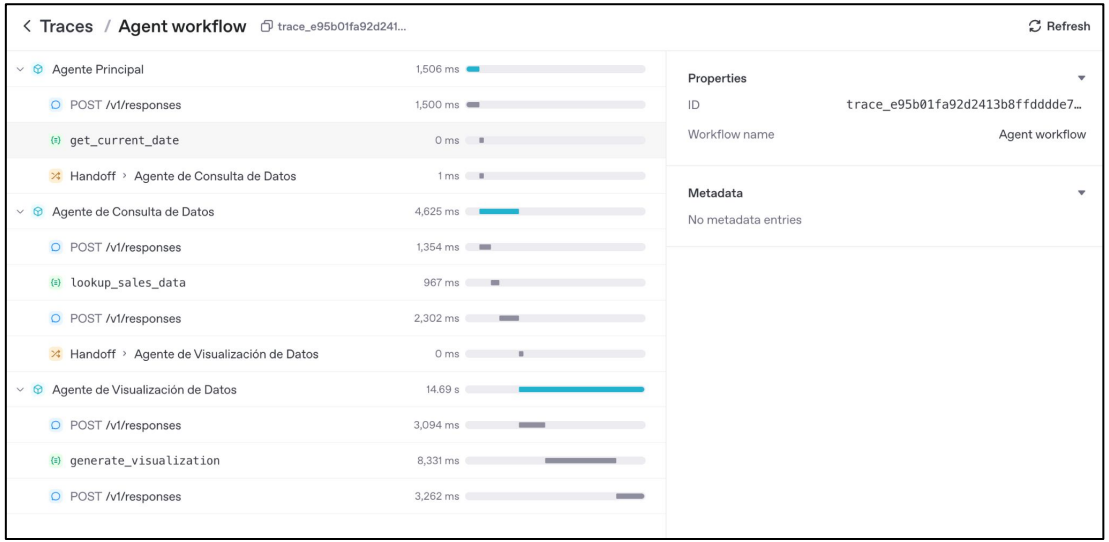
**Notebook:**
https://colab.research.google.com/drive/1v_A_7o8MgHDbUp20UU1dz7dH1Uqxd86d?usp=sharing

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

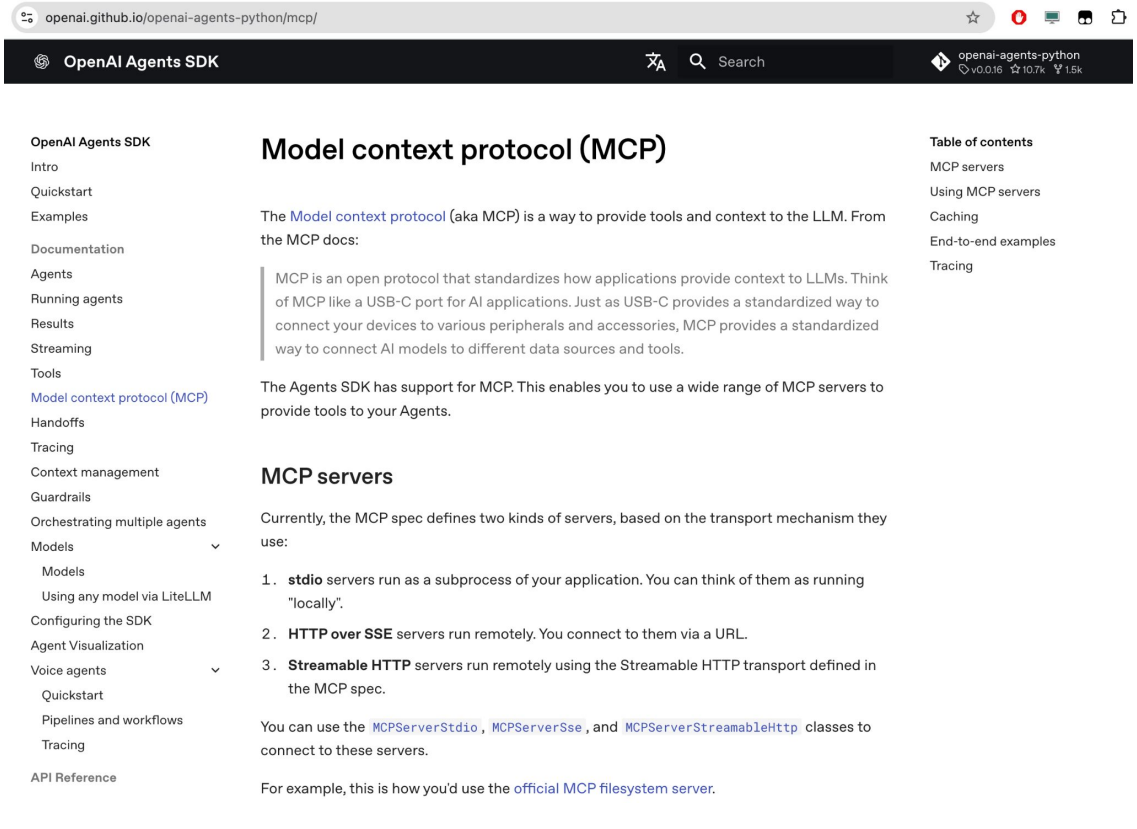
# OpenAI Agents SDK: Sistemas Multi-Agente


Ejercicio AVANZADO:

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



# OpenAI Agents SDK: Soporte MCP




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

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

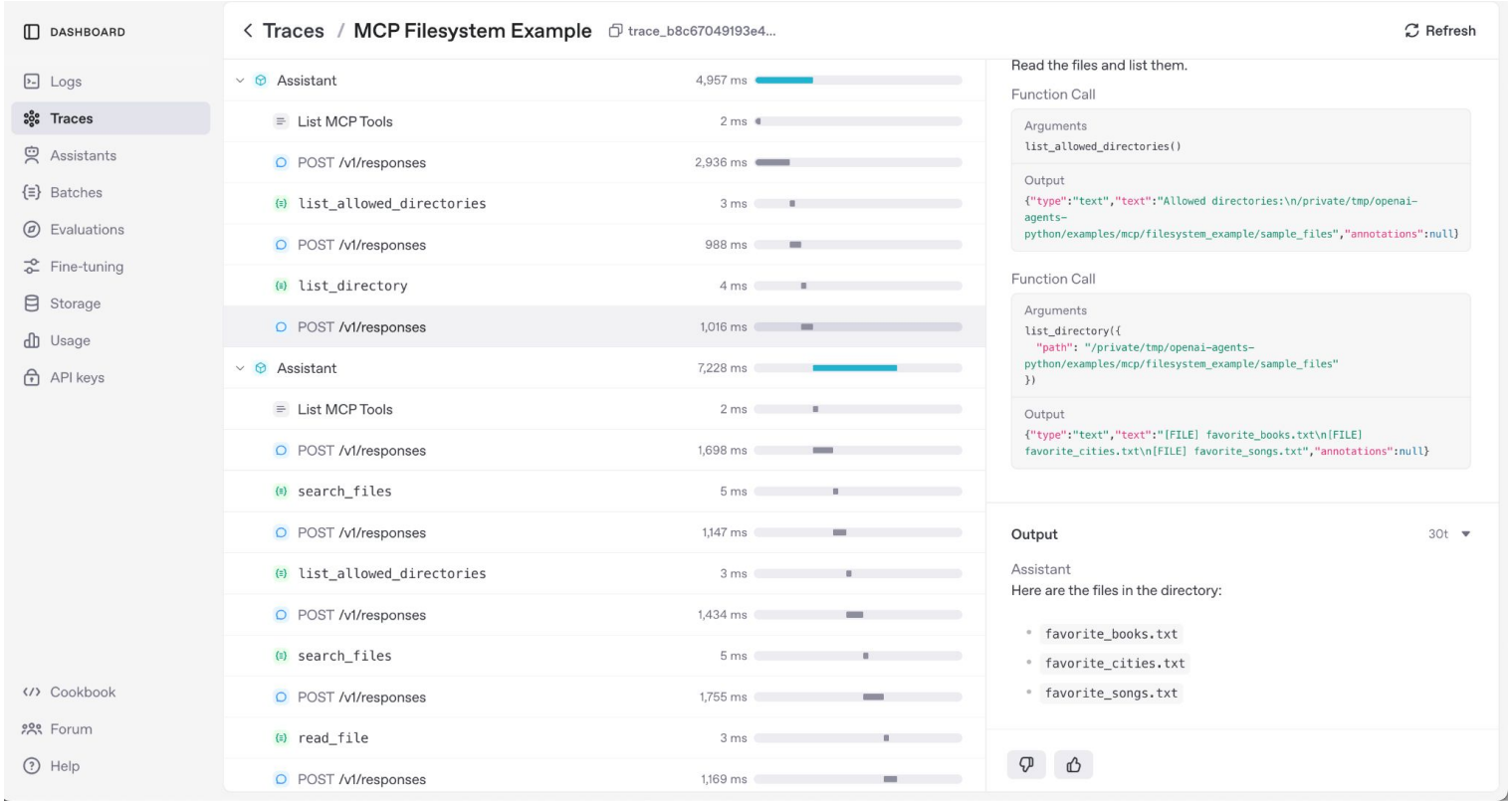
1. Clona openai-agents-python
2. uv venv
3. source .venv/bin/activate
4. uv sync
5. cd examples/mcp/filesystem_example
6. macos/linux:
export OPENAI_API_KEY=sk-xxxxx
windows:
$env:OPENAI_API_KEY="sk-xxxxx"
7. python main.py

# OpenAI Agents SDK: Soporte MCP

## Ejemplo de ejecución:

```
(openai-agents-python) mcp/filesystem_example ᛘ 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 ᛘ main ±
```

# OpenAI Agents SDK: Soporte MCP




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