




Día 1

Planificación y material a usar en el curso

Presentación del curso y del profesor:  Curso Agentes - 2ª Ed. - Presentación (Día 1)

Calendario y guía docente:  Curso Agentes IA / 2. Edición / Programa

Día 1

Diapos:  Curso Agentes IA - 2ª Ed. - Día 1 - Introducción a MCP

* Presentación de los profesores

* Presentación del curso

(Ver recurso Agenda, más abajo)

Comentar Proyecto

* Qué es un agente.

<https://youtu.be/7Dcc1Ag50bA>

* Vamos a usar agentes / Intro a MCP

* Qué es MCP (vídeos en sección Recursos, más abajo)

* Configurar Claude Desktop

* Instalando MCP Servers:

* Instalar y Testear MCP **filesystem**

* **Ejercicios**:

- instalar y probar **desktop commander** MCP

(mostrar tabla con duración de ejercicios)

(3 posibles ejercicios)

- instalar y probar **playwright** MCP

- **Avanzado**: Telegram MCP Server

(**ejercicios**: ehu.eus/bilatu, google docs, telegram...)

* Programación y depuración de servidores MCP

- Ejemplo: búsqueda, clasificación y descarga de imágenes Google Images

Vídeos en sección recursos, más abajo

- **Ejercicio**: programar servidor MCP para búsquedas en Google Scholar

Número de ejercicios para el día 1: 10

Recursos día 1

Intro a MCP

1. <https://youtu.be/u-M109U8aVc>
2. https://youtu.be/VyTh2m11e_4
3. <https://youtu.be/j67VzqRMG3E>
4. <https://youtu.be/2eWbaajlOnQ>

MCP Avanzado / Playwright / Chrome+Remote CDP

https://youtu.be/J_T99KC1rol

Solución a búsqueda comparativa de precios DeepSeek vs OpenAI con MCP:

<https://youtu.be/FEGKAQJuN6w>

Implementando un servidor MCP:

1. <https://youtu.be/UEZABGkibh0>
2. <https://youtu.be/z08J43j94WQ>
3. <https://youtu.be/8GLAec3jtQw>
4. <https://youtu.be/dNYLFMMXRAY>
5. <https://youtu.be/pouYs8ljGZQ>

Repositorio GitHub con ejemplo de Servidor MCP a implementar en el curso:

<https://github.com/juananpe/google-image-search-mcp-python>

Día 2

Día 2

Back to the basics

* Antes de los agentes...

- Diapos (RUDIMENTOS):
 - ▢ Curso Agentes - Rudimentos - Back To the Basics
 - <https://github.com/juananpe/stateful-openai-chatbots>
 - Conocimientos básicos (Completions API, Responses API)
 - COLAB: [🔗 backtothebasics-openai.ipynb](#)

Diapos:

▢ Curso Agentes (2ª Ed.) - Día 2 - Agentes / Programando MCP / LLM + Function Calling

Function Calling

- * Programando un agente (ReAct) con soporte de Function Calling
- * Presentación de OpenAI Function Calling
- * Presentación de ejemplo de Tool Use with Claude
 - * Ejemplo de [chatbot](#) para consulta de papers en Arxiv

Profundizando en MCP

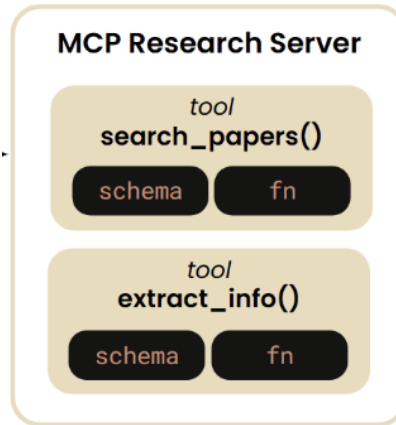
- * Protocolo de Transporte en MCP (stdio, sse, streamable http)

Avanzado

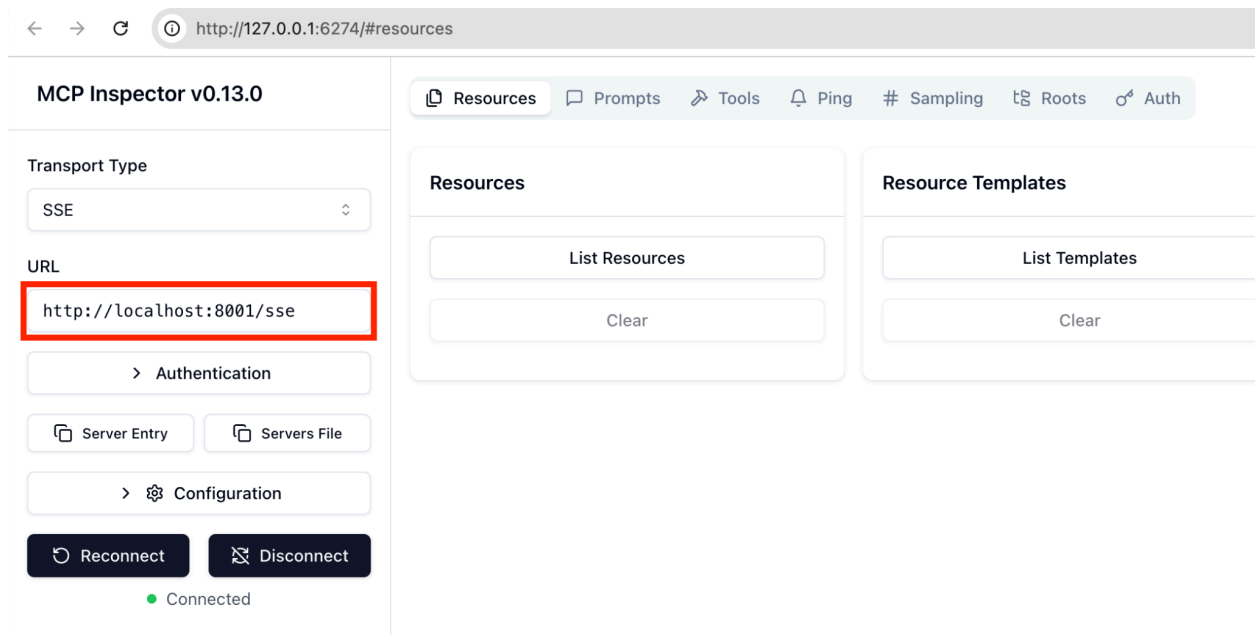
- * Creación de cliente MCP en Python (Chatbot MCP Client) para Research Server
- * Cliente MCP en Python (Chatbot MCP Client) para CUALQUIER server
- * Ejemplo de Server MCP Download Google Images
- * Ejemplo de Server MCP arxiv research server
- * Prompt templates (plantillas de prompt) y Resources (recursos) en MCP

Ejercicios:

- 1) Migrar el código del Agente del ejemplo 3 (ver recursos, más abajo) para que en lugar de usar Function Calling mediante el Chat Completions API use el Responses API.
- 2) Convertir las funciones del chatbot para consulta de papers en Arxiv en herramientas MCP usando FastMCP (guarda el resultado en un fichero research_server.py) Pruébalas con MCP Inspector.



- 3) Configura Claude Desktop para que haga uso de Research Server MCP
 - a) Comprueba que tienes acceso a las tools, al prompt template y a los resources
 - i) Claude Desktop [no soporta](#) aún dynamic resources (25/05/2025)
- 4) Configura VSCode para que haga uso de Research Server MCP
- 5) Descarga remote research server (MCP). Ejecútalo en local. Usa MCP Inspector para probarlo vía SSE (o Streamable HTTP)



- 6) Despliega tu Research Server en una URL pública
 - a) Por ejemplo, usando [render.com](#) (deberás seguir estas instrucciones: <https://gist.github.com/juananpe/a9f13d7d17eb7202e1f3cc3ce4ef400e#file-read-me-md>)
- 7) Programa un servidor MCP con la tool `download_video` que dado como parámetro una URL de un vídeo de YouTube, lo baje usando el API de YouTube, y lo deje en la carpeta vídeos. El servidor debe ofrecer también un resource para listar los vídeos que ya hayamos bajado.

- 8) Programa un servidor MCP que tome como parámetro una consulta (query) en lenguaje natural y un número X y lance una búsqueda en Google Scholar para esa query, devolviendo como resultado las X primeras referencias bibliográficas obtenidas.
- 9) Modifica Research Server (Streamable HTTP MCP server programado en NodeJS) para que la autenticación del usuario sea real (ahora se acepta a cualquier usuario sin requerir identificación).

Recursos

Programando un agente desde 0 (SIN soporte de function calling) (vídeo)

1. <https://www.youtube.com/watch?v=1X8UiRWvXSQ>

Function Calling (vídeo tutorial)

1. <https://youtu.be/hrTal2sWXp4>
2. <https://youtu.be/Nk-EiZjmnLo>
3. <https://youtu.be/oxpFeTEf5SI>

Jupyter Notebooks (Colab)

🔗 ejemplo1.ipynb Programando un agente ReAct (Reason+Act) desde cero (sin function calling)

🔗 ejemplo2.ipynb Uso básico de function calling (completions API)

🔗 ejemplo3.ipynb Agente básico con function calling (completions API)

Function calling (tool use) with Anthropic Claude (Chatbot Arxiv)

🔗 function_calling_anthropic_SDK.ipynb

Back to the basics: (Completions API, Responses API)

<https://github.com/juananpe/stateful-openai-chatbots>

Cliente MCP Client en Python (Chatbot MCP client) (específico)

<https://gist.github.com/juananpe/30ca35c64cda2e9391f8ef2a50562c7e>

Cliente MCP Client en Python (Chatbot MCP client) (general)

<https://gist.github.com/juananpe/588b0967cd6f1ed3385e56f81ed87896>

Research Server MCP (instalar en local y probar) (stdio)

https://gist.github.com/juananpe/a9f13d7d17eb7202e1f3cc3ce4ef400e/raw/666cee9fa8d894297d67cb184b687ca04cb6d7a2/research_server.py

Remote Research Server MCP (instalar en local y probar) (sse)

https://gist.github.com/juananpe/a9f13d7d17eb7202e1f3cc3ce4ef400e/raw/666cee9fa8d894297d67cb184b687ca04cb6d7a2/remote_research_server.py

Documentación **OpenAI** Function Calling:

<https://platform.openai.com/docs/guides/function-calling?api-mode=chat>

Documentación **Tool use with Anthropic** Claude:

<https://docs.anthropic.com/en/docs/agents-and-tools/tool-use/overview#how-to-implement-tool-use>

Avanzado

Prueba el streamable http server con autenticación OAuth y dynamic client registration (DCR) del zoo: <https://zoo.ikasten.io> (Login: johndoe, Password: pass)

Analiza el código: Branch: DCR

Repo: <https://github.com/juananpe/zoo-animal-mcp-server/tree/dcr>

Repositorios

<https://github.com/juananpe/research-server/>

<https://github.com/juananpe/google-image-search-mcp-python>

<https://github.com/juananpe/zoo-animal-mcp-server/>

Día 3

Día 4

Día 5

Día 6

Propuesta de Proyecto

