Clase 153 — MCP (Model Context Protocol): herramientas y datos para LLMs

Parte: 2 — Deep Learning · Fuente: MCP spec (Anthropic 2024). Duración estimada: 80 min.

Clase 153 — MCP (Model Context Protocol): herramientas y datos para LLMs

Parte: 2 — Deep Learning · Fuente: MCP spec (Anthropic 2024). Duración estimada: 80 min.

Objetivo

Aprender MCP (Model Context Protocol) — estándar abierto publicado por Anthropic en noviembre 2024 que define cómo un LLM se conecta a herramientas externas (filesystems, databases, APIs, search engines, etc.). Antes de MCP, cada framework (LangChain, LlamaIndex, OpenAI plugins) tenía API propia. MCP unifica → portabilidad entre LLMs y clients.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Explicar la arquitectura MCP: client, server, resources, tools, prompts.
- Conectar MCP servers existentes a Claude Desktop, Cursor, Zed.
- Escribir un MCP server propio en Python (con fastmcp).
- Diferenciar MCP de tool use clásico de OpenAI / LangChain.
- Usar MCP servers populares: filesystem, postgres, git, slack, brave-search.

Temas

- Cliente (LLM app) vs Server (provee tools/resources/prompts).
- Transport: stdio (local) y SSE (network).
- Resources: read-only data (archivos, DB rows).
- Tools: funciones invocables (search, write).
- Prompts: templates reusables.
- Discovery: el client descubre dinámicamente qué hay disponible.

Definiciones y características

- MCP server: programa que expone resources/tools/prompts via JSON-RPC.
- MCP client: typically un LLM app (Claude Desktop, Cursor, IDE plugins).
- Resource URI: e.g., file:///path/to/x, postgres://db/table.
- Tool schema: JSON Schema describing inputs/outputs.
- fastmcp: librería Python para escribir servers MCP rápido.

Dataset / recursos

- MCP servers oficiales.
- Librerías: mcp (Python SDK), fastmcp.

Ejercicios

1. Conectar server existente: instalar mcp-server-filesystem en Claude Desktop. Hacer queries sobre archivos del file system.
2. Server propio: con fastmcp, exponer un tool search_docs(query) sobre un corpus local.
3. Resource: exponer archivos .md como resources lectura.
4. Prompt template: definir summarize(file_uri) como prompt reusable.
5. Multi-server: conectar 2-3 servers simultáneos a Claude Desktop; usar conjuntamente.

Homework verificable

MCP server propio para RAG sobre documentación:

1. fastmcp con tool search(query) que usa el RAG de clase 129.
2. Resource list_documents() que lista archivos indexados.
3. Conectar a Claude Desktop; usarlo en una conversación.
4. Comparar UX vs hacer RAG manual en código.

Criterio de aceptación: el LLM cliente puede llamar al search tool naturalmente y trae resultados relevantes.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Server no aparece en client	Config mal en ~/.config/claude/.... Fix: v
Tool schema rejected	JSON Schema inválido. Fix: validar con un
stdio server no inicia	Path al executable mal. Fix: paths absolut
Server lento	Llamadas síncronas. Fix: usar async def to
Cliente no encuentra resource	URI mal formado. Fix: schema correcto.

Preguntas frecuentes

¿MCP vs LangChain tools?

LangChain tools son Python objects acoplados al framework. MCP es un protocol — funciona con cualquier LLM client que lo soporte. Más portable.

¿OpenAI plugins / GPTs?

Más cerrados, especificos a OpenAI. MCP open + multi-vendor.

Servers existentes utiles?

filesystem, git, postgres, sqlite, slack, brave-search, fetch (HTTP), memory, github, gdrive — muchos en el repo oficial.

Seguridad?

Cada server corre con permisos del usuario. Cuidado con qué exponés. Auditar antes de instalar de terceros.

¿OpenAI agrega MCP?

Anthropic open-sourceó el spec; otros vendors lo están adoptando (Microsoft Copilot Studio, etc.). Está volviéndose estándar.

Referencias

- MCP spec.
- MCP servers official repo.
- fastmcp.
- Anthropic blog (Nov 2024), Introducing the Model Context Protocol.

Siguiente clase

Clase 154 — Agentes: tool use, ReAct, multi-agent

Apéndice: notebook (primer bloque)

Implementamos un mini servidor y cliente MCP en Python: JSON-RPC sobre stdin/stdout (simulado in-process). Sin dependencias externas.

```
import json, re, uuid
import numpy as np
np.random.seed(42)
```

Archivos complementarios

- notebook.ipynb