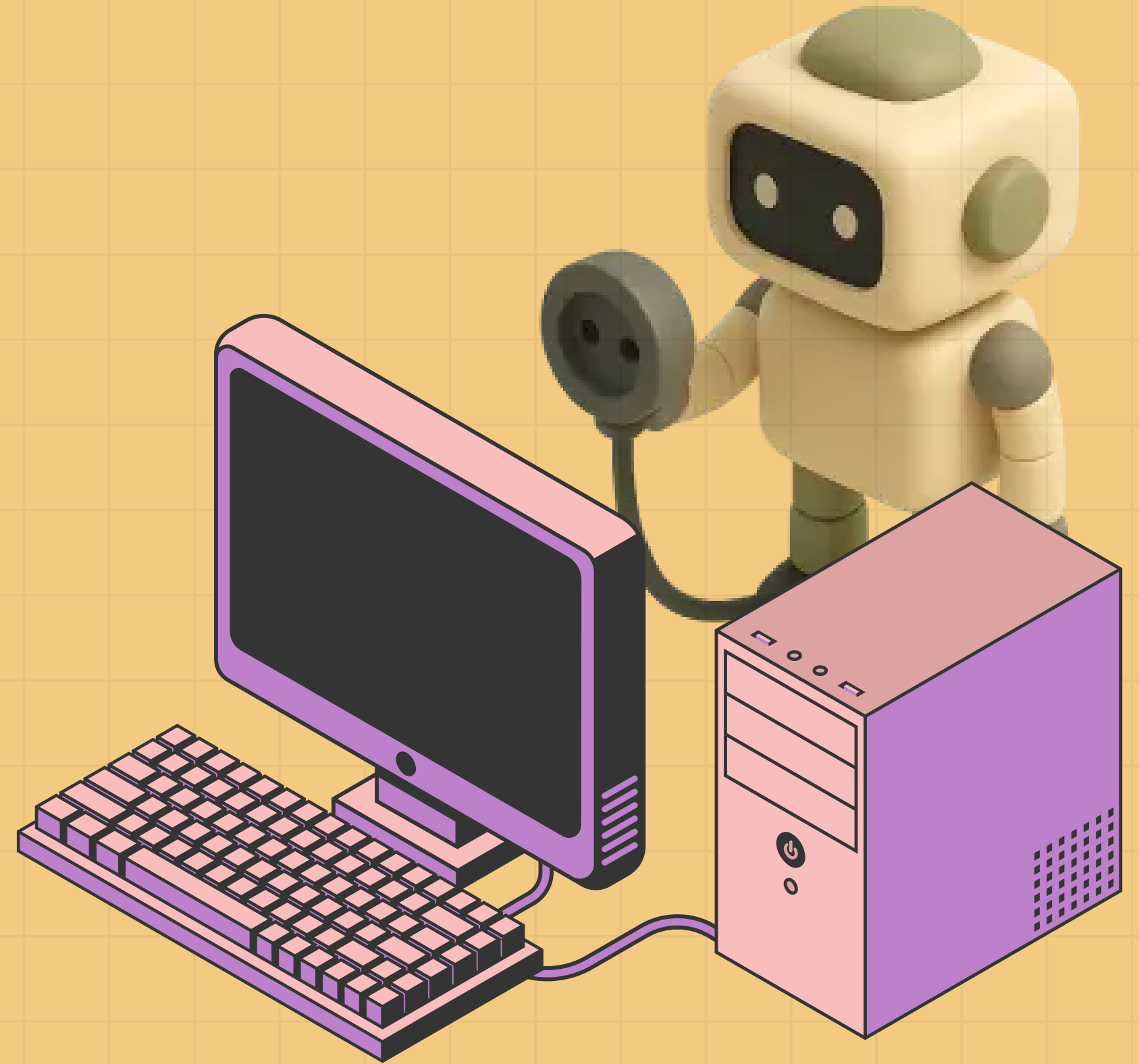


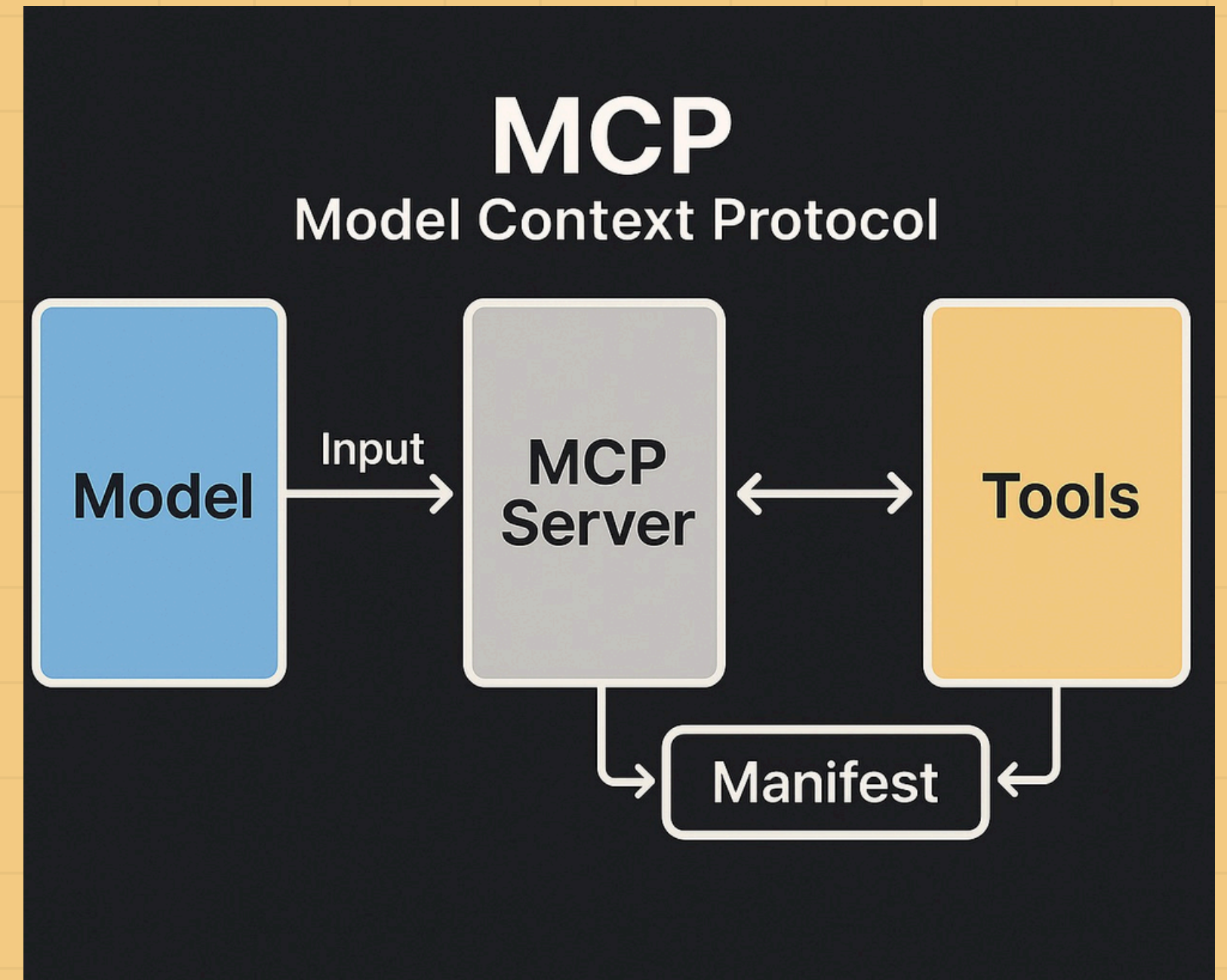
CREA TU PRIMER SERVIDOR MCP CON PYTHON Y FASTMCP

POR : MATIAS PALOMINO LUNA



¿QUÉ ES MCP?

- MCP (Model Context Protocol) = estándar abierto de Anthropic.
- Es un “idioma común” que permite que un LLM se conecte con:
 - Clientes (Claude Desktop, Open WebUI, etc.).
 - Servidores.
 - Herramientas (funciones, APIs, bases de datos).
- Resumen: el LLM deja de solo hablar → ahora puede actuar.



PROBLEMA QUE RESUELVE (ANTES VS AHORA)



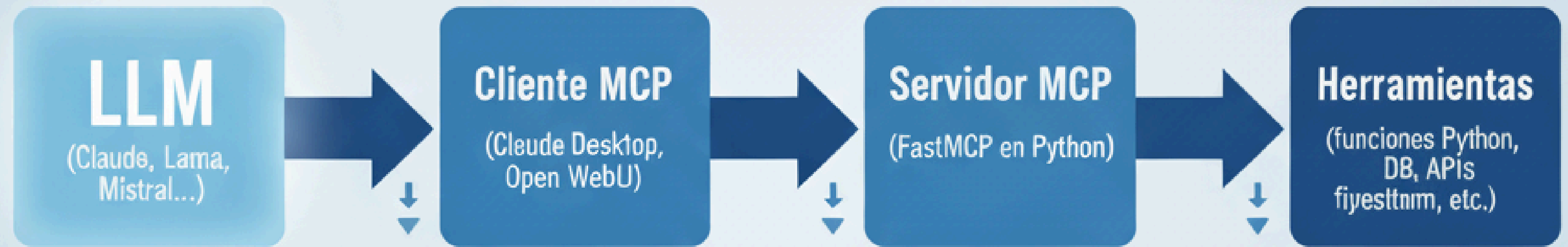
ANTES MCP:

- Cada integración era un “hack casero”.
- Mucho código duplicado.
- Difícil de mantener o portar entre clientes.

CON MCP:

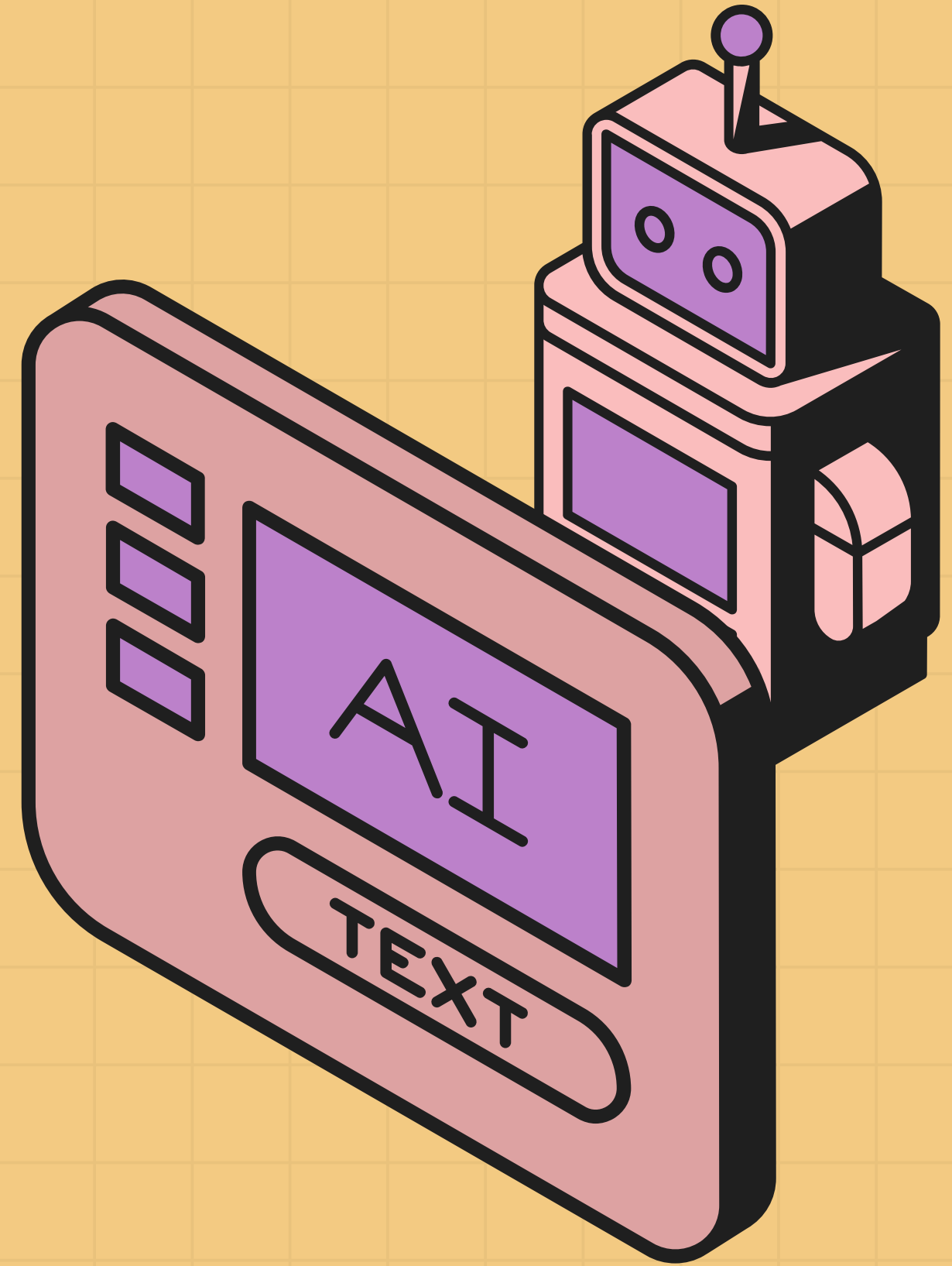
- Un solo estándar.
- Portabilidad → lo que haces en Claude, sirve en otros clientes MCP.
- Ecosistema creciente: Filesystem MCP, DB MCP, Browser MCP...

ARQUITECTURA BÁSICA



CASOS DE USO REALES

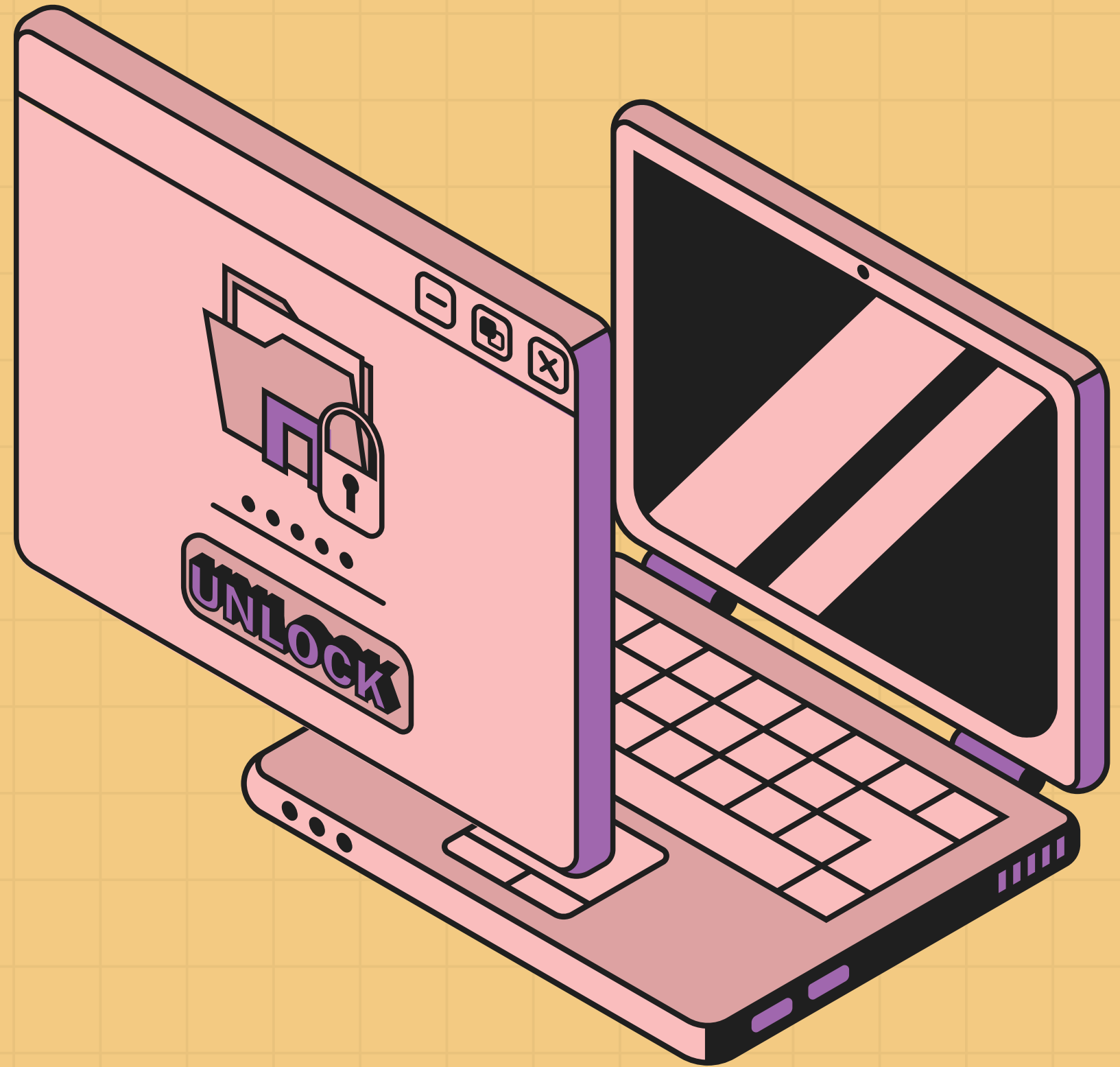
- Automatización → scraping, logs, workflows.
- Datos → consultas en CSV/SQL, análisis en tiempo real.
- Ciberseguridad → monitorizar servidores, bloquear IPs.
- Productividad → integraciones con Slack, Notion, Google Drive.
- Creatividad → conectar LLM con APIs externas (clima, noticias, redes sociales).




SETUP DEL ENTORNO

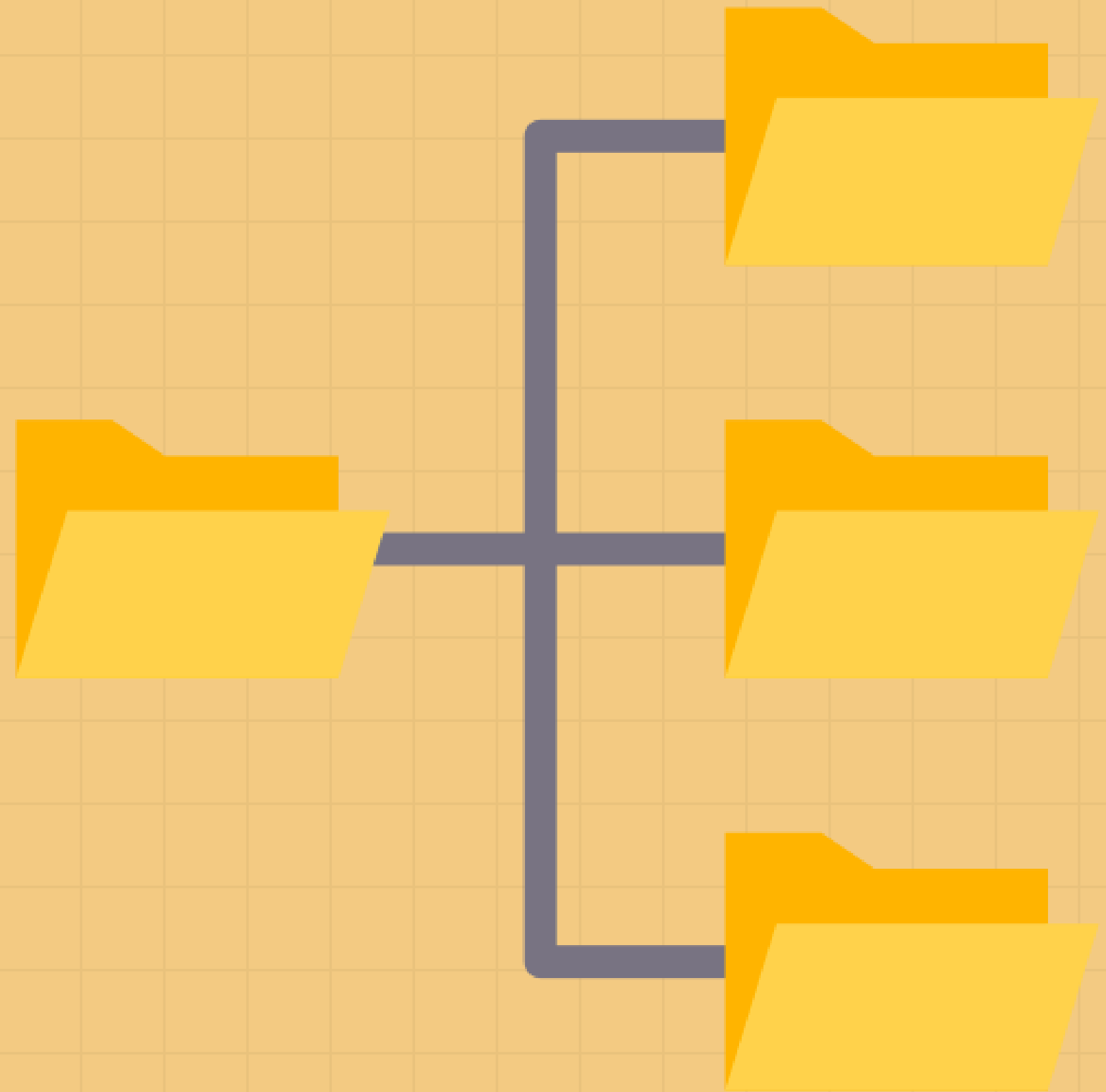
Requisitos técnicos

- Python 3.10+
- pip
- FastMCP



PROYECTO LIMPIO

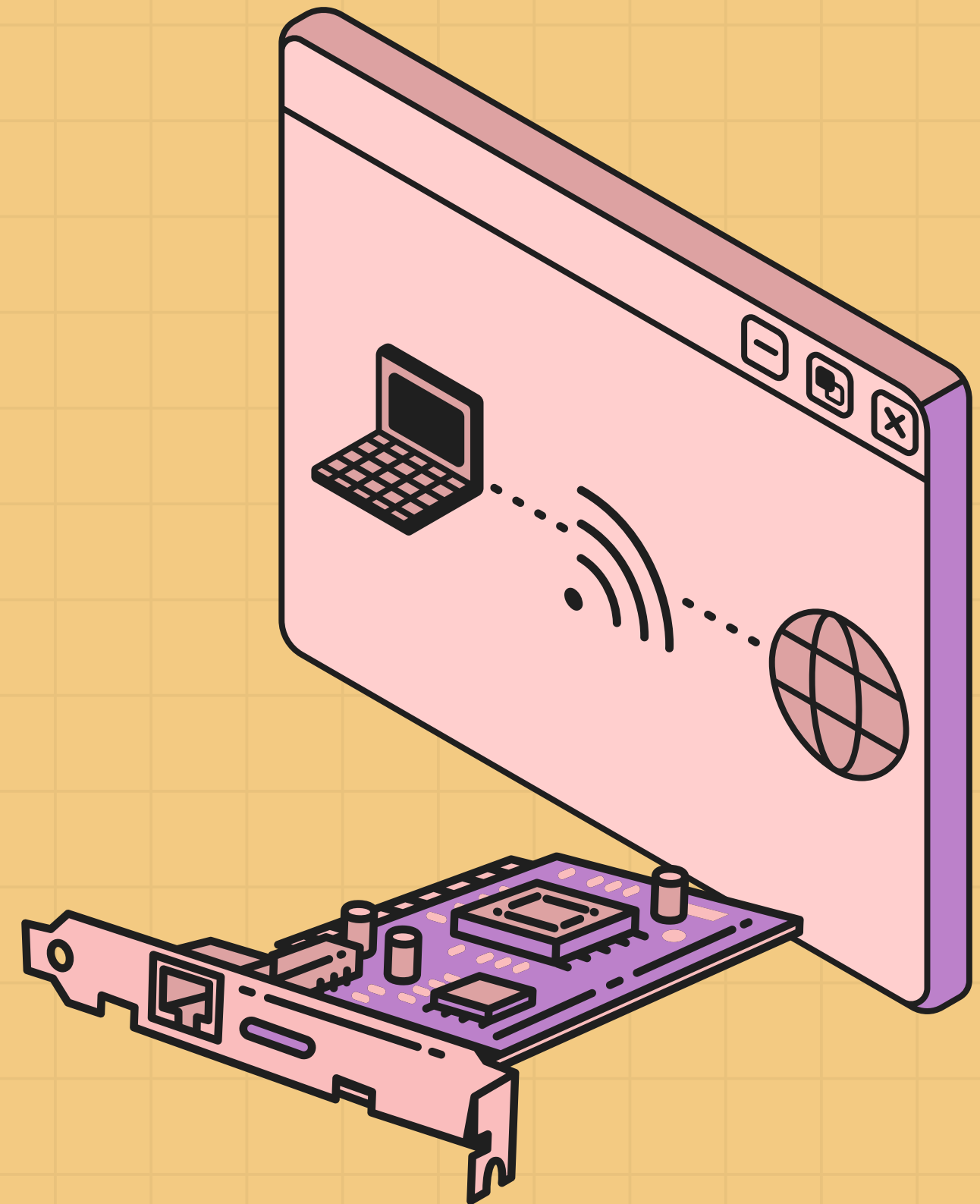
 proyecto_mcp/
| server.py
| requirements.txt
| venv/



INSTALACIÓN Y VERIFICACIÓN RÁPIDA

pip install fastmcp python server.py

- Paso 1: instalar la librería.
- Paso 2: ejecutar un server.py de prueba (lo veremos en la próxima sección).
- Si ves logs tipo "MCP Server running on port..." → 🎉 ya tienes tu servidor listo.



HERRAMIENTAS PRÁCTICAS

Nivel Básico 🍼

- Hola Mundo 🙌
- Calculadora ➕ inputs/outputs

```
from fastmcp import FastMCP

# Crear instancia del servidor
mcp = FastMCP("DemoMCP")

@mcp.tool
✓ def hola(nombre: str) -> str:
    """Devuelve un saludo personalizado."""
    return f"Hola {nombre}, ¡tu servidor MCP funciona!"

@mcp.tool
✓ def suma(a: int, b: int) -> int:
    """Suma dos números."""
    return a + b

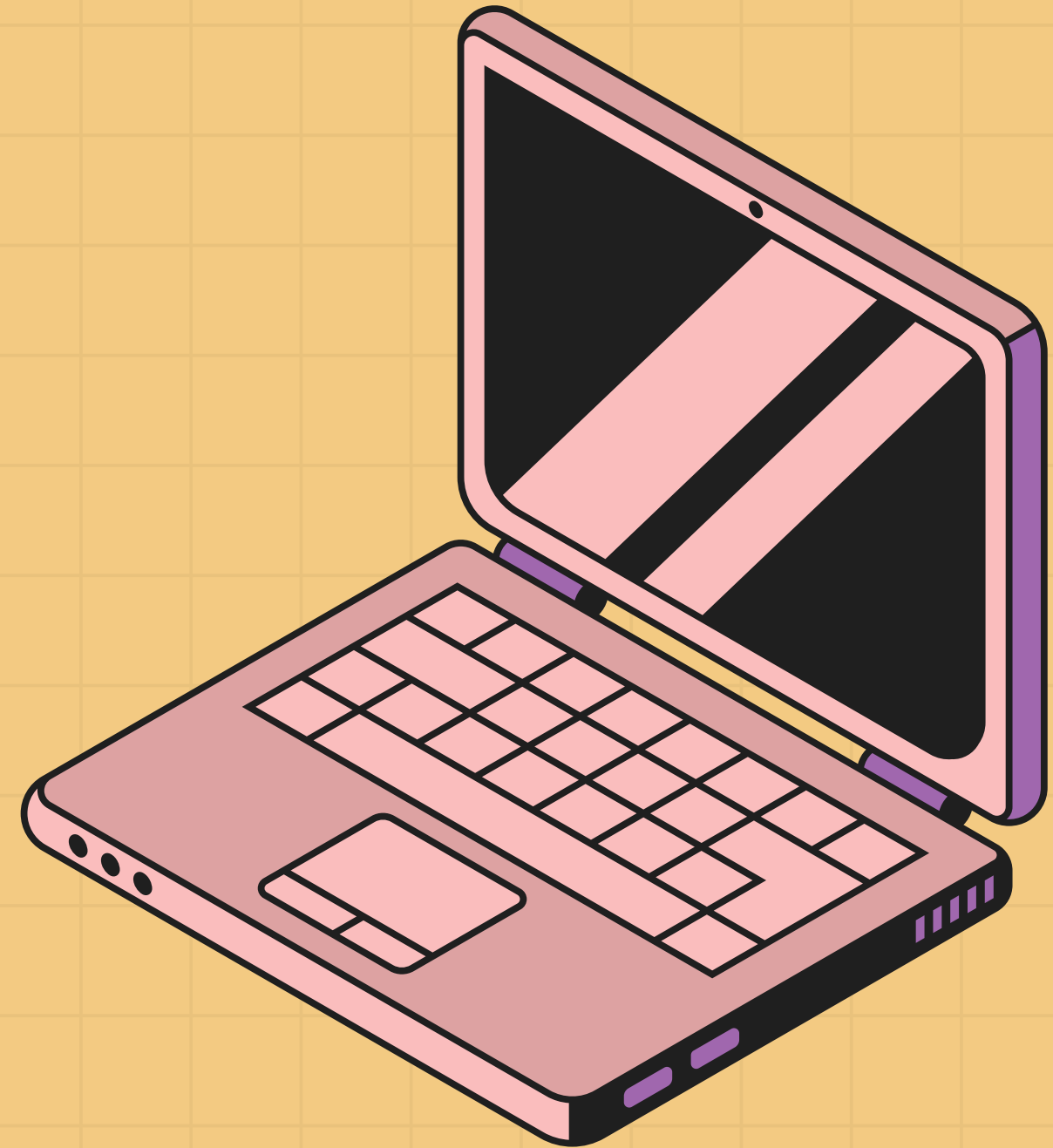
✓ if __name__ == "__main__":
    mcp.run()
```

CONCEPTOS CLAVE

- `FastMCP("Nombre")` → servidor
- `@mcp.tool` → función → herramienta
- **Tipos + docstring** → schema
- `mcp.run()` → expone todo

Tip de diseño de herramientas:

- Inputs explícitos (tipa todos los parámetros).
- Mensajes de error legibles (facilitan el razonamiento del LLM).
- Limitar salidas (ej. `max_bytes`) para evitar respuestas gigantes.



LOGS Y COMPROBACIÓN INICIAL

```
# (Opcional) activar venv  
# Windows: .\.venv\Scripts\activate  
# macOS/Linux: source .venv/bin/activate
```

```
pip install fastmcp python server.py
```

```
return pong

@mcp.tool
def read_file(path: str, max_bytes: int = 2000) -> str:
    """
    Lee un archivo de texto y devuelve hasta N bytes.
    Útil para probar integración con filesystem de forma segura.
    """
    p = Path(path).expanduser().resolve()
    if not p.exists():
        return f"[ERROR] No existe: {p}"
    if not p.is_file():
        return f"[ERROR] No es un archivo: {p}"
    data = p.read_bytes()[:max_bytes]
    try:
        return data.decode("utf-8", errors="replace")
    except Exception as e:
        return f"[ERROR] No pudo decodificar UTF-8: {e}"

if __name__ == "__main__":
    # Arranca el servidor y expone las herramientas definidas con @mcp.tool
    mcp.run()
```

FastMCP 2.0

 **Server name:** DemoMCP

 **Transport:** STDIO

 **FastMCP version:** 2.12.3

 **MCP SDK version:** 1.14.1

 **Docs:** <https://gofastmcp.com>

 **Deploy:** <https://fastmcp.cloud>

PRIMER SERVIDOR MCP

Obtener el Clima

- Definir un servidor con FastMCP.
- Crear una tool (clima) que recibe como parámetro una ciudad.
- Consultar el servicio gratuito wttr.in para obtener el clima.
- Devolver una respuesta simple y fácil de interpretar.

```
import requests
from fastmcp import FastMCP

mcp = FastMCP("WeatherMCP")

@mcp.tool
def clima(ciudad: str) -> str:
    """
    Devuelve el clima actual en una ciudad (formato breve).
    Ejemplo: clima("Madrid") -> "Madrid: ☀️ +25°C"
    """
    try:
        url = f"http://wttr.in/{ciudad}?format=3"
        resp = requests.get(url, timeout=5)
        return resp.text.strip()
    except Exception as e:
        return f"[ERROR] No se pudo obtener el clima: {e}"

if __name__ == "__main__":
    mcp.run()
```

MCP INSPECTOR

(EL POSTMAN DE MCP)

- Herramienta oficial MCP (Anthropic).
- UI para probar servidores MCP.
- Inspecciona inputs/outputs en vivo.
- Como Postman/Swagger, pero para MCP.



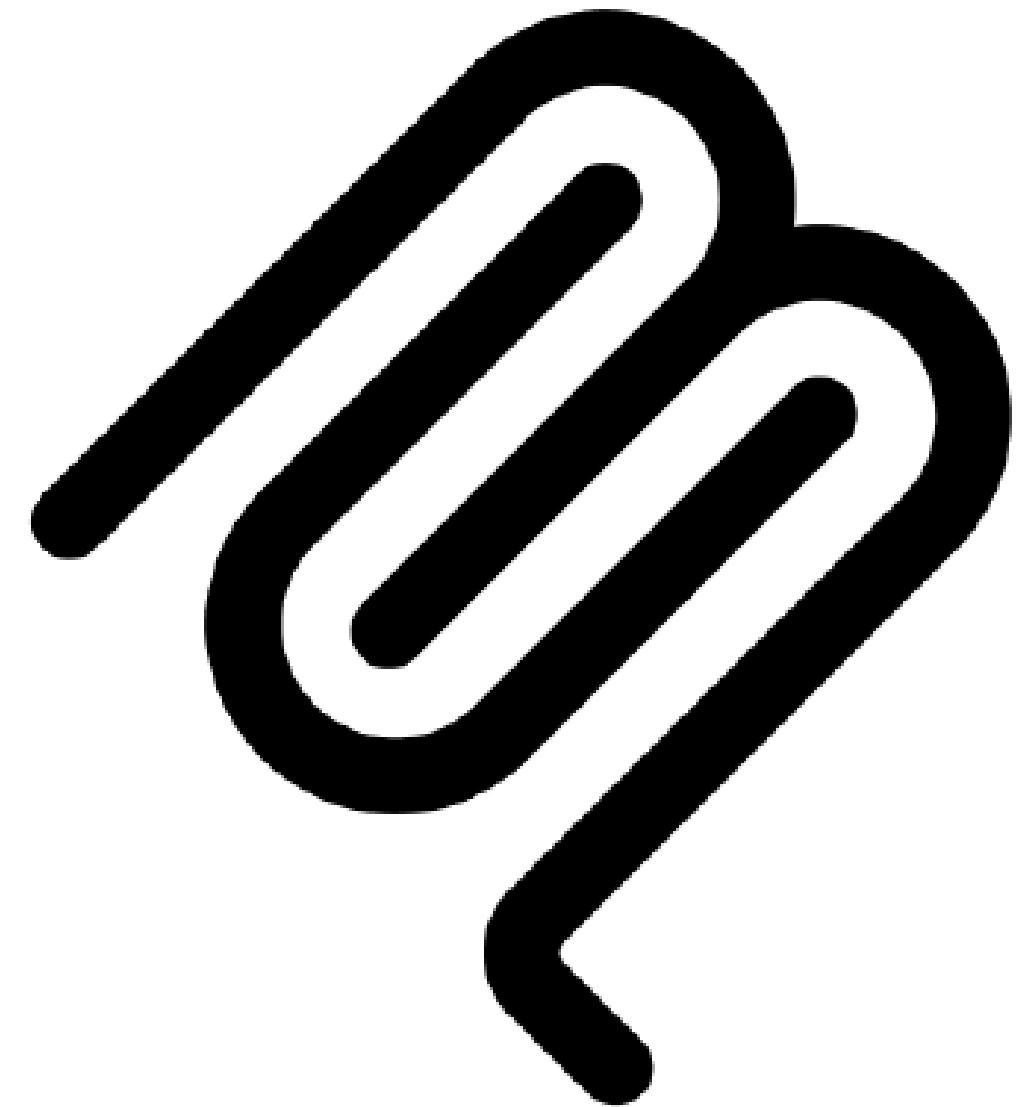
MCP INSPECTOR

(EL POSTMAN DE MCP)

Instalación

`npm install -g @modelcontextprotocol/inspector`

`npx @modelcontextprotocol/inspector python server.py`



MCP INSPECTOR

(EL POSTMAN DE MCP)

Cómo usarlo

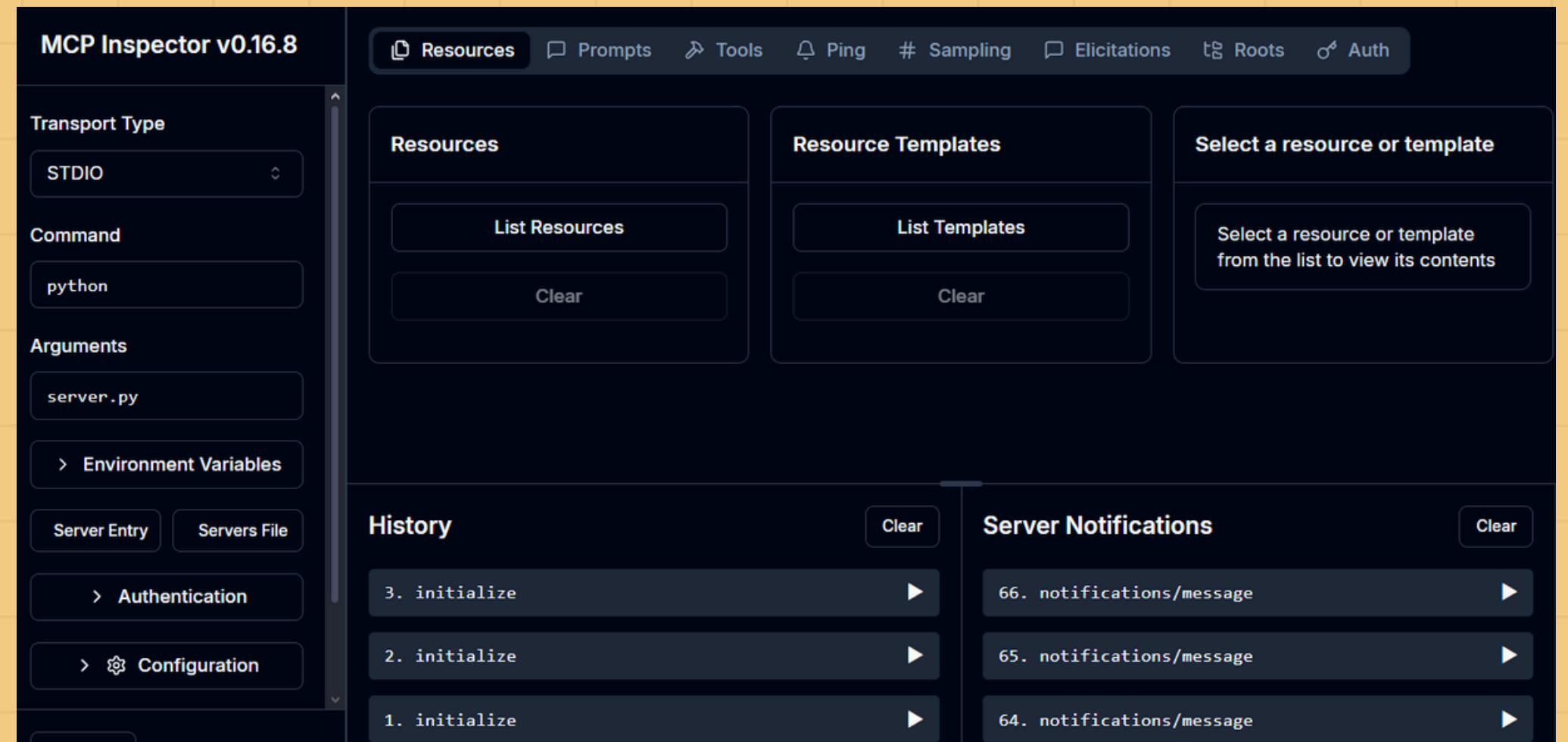
Levantar servidor:

```
python server.py
```

Abrir Inspector:

```
mcp-inspector python server.py
```

```
npx @modelcontextprotocol/inspector python server.py
```



COMPARATIVA VISUAL

Mundo REST (FastAPI)	Mundo MCP (FastMCP)
Swagger UI	MCP Inspector
Endpoints HTTP	Herramientas MCP
Requests / Responses	Inputs / Outputs
Docs automáticas	Schemas automáticos

SEGUNDO SERVIDOR MCP

Precio Bitcoin

- Definir un servidor con FastMCP.
- Crear una tool que interactúa con servicios externos.
- Manejar errores probando varias fuentes de datos.

```
import requests
from datetime import datetime
from fastmcp import FastMCP

mcp = FastMCP("CryptoMCP")

def _get(url: str, timeout: int = 5):
    return requests.get(url, timeout=timeout, headers={"User-Agent": "CryptoMCP/1.0"})

@mcp.tool
def precio_bitcoin() -> str:
    """
    Devuelve el precio actual de BTC en USD.
    Intenta varias fuentes y devuelve 'fuente: precio (ISO8601)'.
    """
    # 1) CoinDesk
    try:
        r = _get("https://api.coindesk.com/v1/bpi/currentprice.json")
        d = r.json()
        price = d["bpi"]["USD"].get("rate_float") or float(d["bpi"]["USD"]["rate"].replace(",", ""))
        return f"CoinDesk: {price:.2f} USD ({datetime.utcnow().isoformat()}Z)"
    except Exception:
        pass

    # 2) CoinGecko
    try:
        r = _get("https://api.coingecko.com/api/v3/simple/price?ids=bitcoin&vs_currencies=usd")
        price = float(r.json()["bitcoin"]["usd"])
        return f"CoinGecko: {price:.2f} USD ({datetime.utcnow().isoformat()}Z)"
    except Exception:
        pass

    # 3) Blockchain.info
    try:
        r = _get("https://blockchain.info/ticker")
        price = float(r.json()["USD"]["last"])
        return f"Blockchain.info: {price:.2f} USD ({datetime.utcnow().isoformat()}Z)"
    except Exception as e:
        return f"[ERROR] No se pudo obtener el precio desde ninguna fuente: {e}"

if __name__ == "__main__":
    mcp.run()
```

TERCER SERVIDOR MCP

Diagnóstico de Red

- Definir un servidor con FastMCP.
- Crear una tool (conexion_red) que diagnostica el estado de la red.
- Verificar si el DNS resuelve dominios y si hay salida HTTPS disponible.
- Devolver un reporte simple con resultados y posibles errores.

```
import requests, socket
from fastmcp import FastMCP

mcp = FastMCP("NetCheckMCP")

@mcp.tool
def conexion_red() -> dict:
    """
    Diagnóstico rápido de red: DNS y salida HTTPS.
    - dns_ok: True si se resuelve api.coindesk.com
    - https_ok: True si se puede acceder a https://example.com
    - detalle: info extra de IP, status o errores
    """
    out = {"dns_ok": False, "https_ok": False, "detalle": {}}

    # Comprobación DNS
    try:
        ip = socket.gethostbyname("api.coindesk.com")
        out["dns_ok"] = True
        out["detalle"]["coindesk_ip"] = ip
    except Exception as e:
        out["detalle"]["dns_error"] = str(e)

    # Comprobación HTTPS
    try:
        r = requests.get("https://example.com", timeout=5)
        out["https_ok"] = (200 <= r.status_code < 400)
        out["detalle"]["https_status"] = r.status_code
    except Exception as e:
        out["detalle"]["https_error"] = str(e)

    return out

if __name__ == "__main__":
    mcp.run()
```

CONEXIÓN CON CLIENTES MCP

Añadir a Claude Desktop

```
{
  "mcpServers": {
    "CryptoMCP": {
      "command": "E:\\Anaconda3\\envs\\Agentes\\python.exe",
      "args": [
        "C:\\Users\\jmati\\Desktop\\Taller - MCP\\bitcoin.py"
      ],
      "cwd": "C:\\Users\\jmati\\Desktop\\Taller - MCP"
    },
    "NetCheckMCP": {
      "command": "E:\\Anaconda3\\envs\\Agentes\\python.exe",
      "args": [
        "C:\\Users\\jmati\\Desktop\\Taller - MCP\\red.py"
      ],
      "cwd": "C:\\Users\\jmati\\Desktop\\Taller - MCP"
    }
  }
}
```

- mcpServers → lista de servidores MCP que Claude podrá usar.
- CryptoMCP → nombre del servidor (lo verás en Claude).
- command → programa que se ejecuta (aquí, Python de tu entorno).
- args → parámetros a ese programa → la ruta a tu server.py.
- cwd → carpeta donde se ejecuta el script (Working Directory).

OTROS CLIENTES MCP YA EXISTENTES

- Filesystem MCP → navegar archivos locales.
- DB MCP → consultas SQL.
- Browser MCP → navegar y scrapear web.
- Slack / Notion MCP → productividad.
- Custom MCPs → hechos por la comunidad.

Github MCP

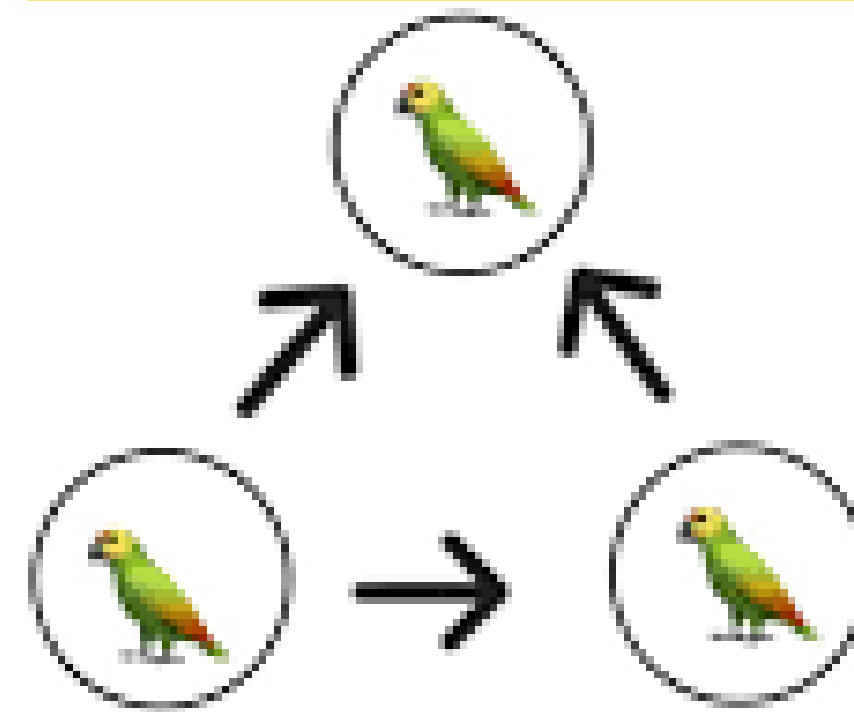
MCP.SO

EXPANSIONES POSIBLES

Integración con LangChain

LangChain puede usar MCP como tool provider → tus funciones Python quedan disponibles como herramientas de agente.

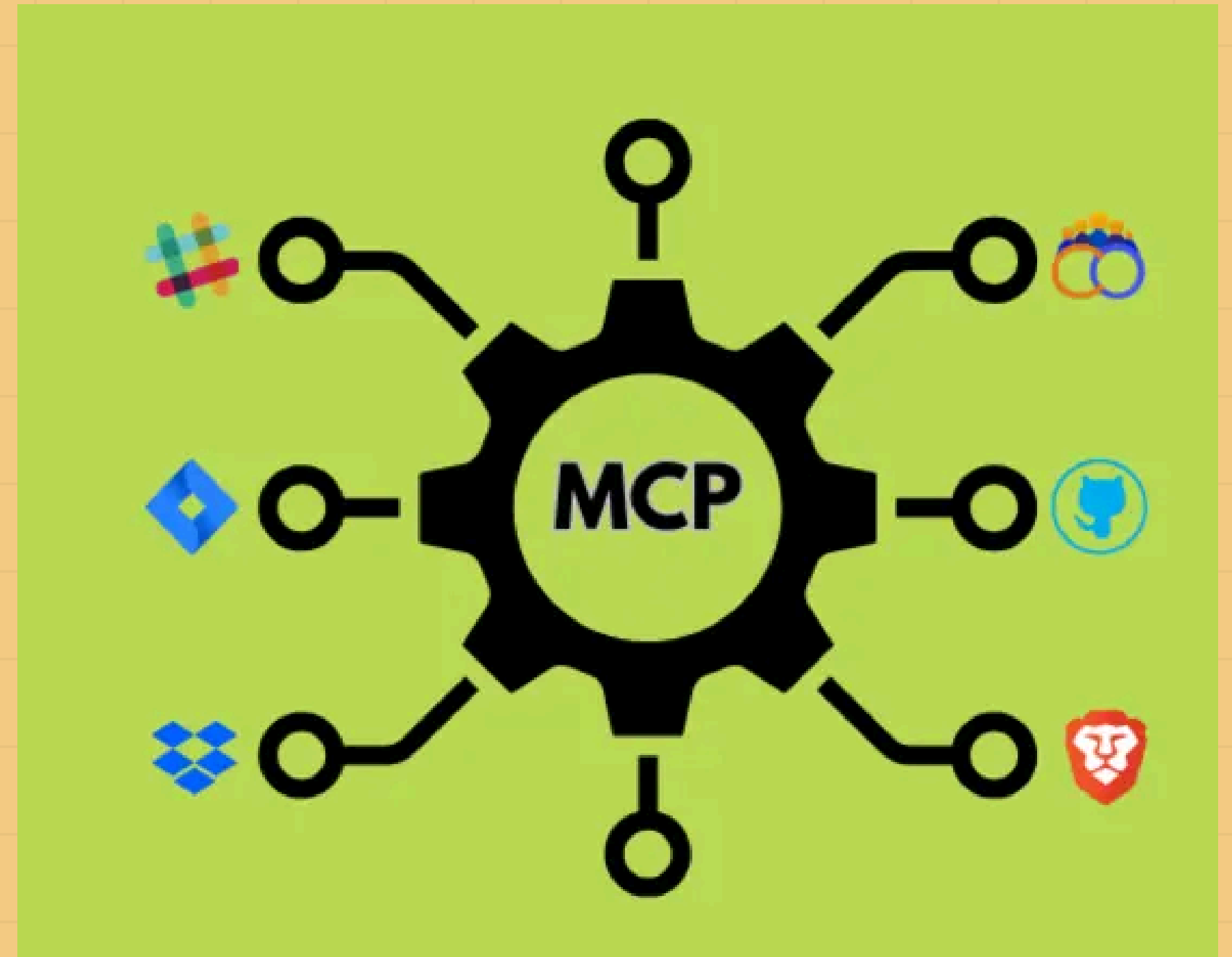
LANGCHAIN MCP-RAG



OLLAMA

RESUMEN

- Qué es MCP
- Setup básico
- Herramientas prácticas
- MCP Inspector
- Conexión con clientes
- Ecosistema MCP (nube y más)



¡CONECTEMOS!

Si te gustó este taller o quieres seguir aprendiendo sobre IA, seguridad y agentes inteligentes...

Suelo compartir:
Proyectos de IA aplicados al mundo real
Reflexiones sobre tecnología y ciberseguridad

🧠 ¡Gracias por participar y que viva el código libre!

Sígueme en LinkedIn:



Matias Palomino Luna

-

GitHub : [jmatias2411](#)