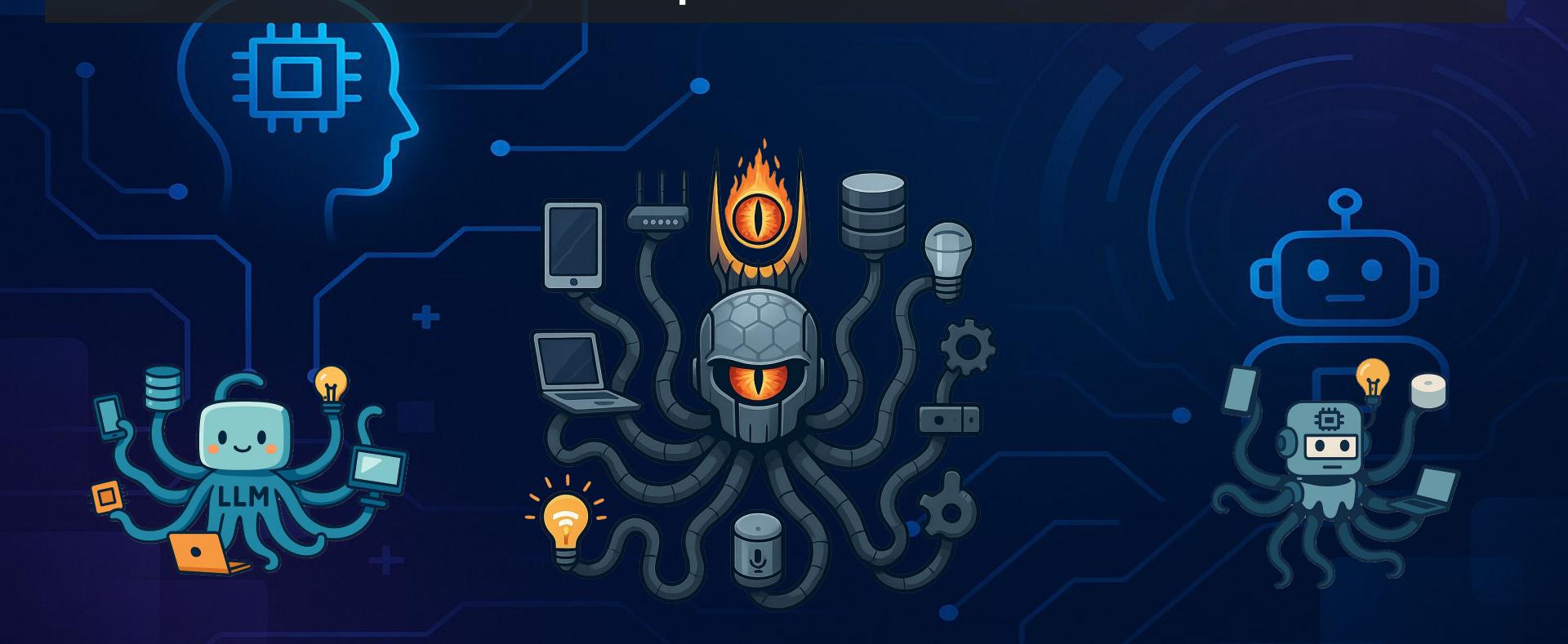


# MCP: Un Protocolo para Gobernarlos a Todos.



Domina el MCP y Desata el Poder de tus Modelos



# MCP: Un Protocolo para Gobernarlos a Todos.



<https://www.linkedin.com/company/centic/>



<https://www.linkedin.com/in/juanjofp/>



<https://www.linkedin.com/in/pacobravolozano/>



Cofinanciado por  
la Unión Europea



Fondos Europeos



Región de Murcia



Región de Murcia  
RITSI

**CENTinEI**  
OBSERVATORIO TECNOLÓGICO INTELIGENTE

# MCP: Un Protocolo para Gobernarlos a Todos.



¿Y si tus agentes de IA pudieran interactuar con el mundo real tan fácilmente como pedir un café?

En esta charla descubrirás el **Model Context Protocol (MCP)**, el nuevo estándar que está revolucionando cómo los modelos de lenguaje se conectan con tus herramientas, tus datos y tu imaginación.

Explicaremos cómo MCP se convierte en el **Anillo Único** que lo conecta todo: desde editores como VS Code, Claude Desktop o Cursor IDE, hasta frameworks de agentes como OpenAI, Claude, Agentic, ADK o LangChain.

⚙️ Verás en directo una **demonstración de uso real de MCPs existentes**, donde un modelo se convierte en algo más que un chatbot y comienza a tomar acción en tu entorno.

🛠️ Y después, presencia la magia de construir tu **propio MCP desde cero**, uniendo modelos con APIs, dispositivos o cualquier sistema que imagines.

Esta no es solo otra charla sobre IA. Es el mapa hacia una nueva generación de agentes autónomos.

🌟 **Trae tus preguntas, tu curiosidad y prepárate para enchufarte al futuro.**

# Motivación



## 💡 Fragmentación de APIs

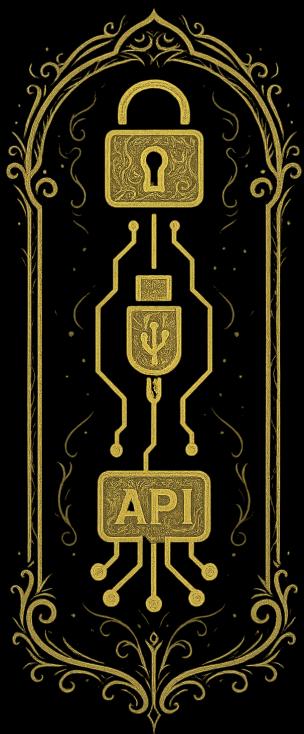
Cada herramienta o servicio expone su propia API con su propio formato, documentación y formas de autenticación.

## 🔗 Necesidad de interoperabilidad "USB-C" para agentes

Igual que el USB-C unificó cables y cargadores, MCP busca unificar cómo los modelos interactúan con el mundo.

## 🛡️ Seguridad y control de permisos

Los agentes necesitan actuar, pero también deben hacerlo con transparencia, trazabilidad y control granular.



# ¿Qué es MCP?



## Protocolo para conectar modelos con el mundo real

Define cómo los modelos descubren y usan recursos externos de forma segura, estructurada y auditible.

## Un “manifest” lo describe todo

Cada servidor MCP expone un manifest.json con sus recursos, acciones disponibles, permisos requeridos y versión.

## Agnóstico al modelo, simple para integradores

No requiere que el modelo sepa nada de APIs: solo interpreta el manifest. Cualquier herramienta que soporte MCP puede conectarse con cualquier servicio MCP.

## Seguridad integrada y control de permisos

El protocolo soporta diferentes tipos de autenticación, scopes y validación de acciones antes de ejecutarlas.



# MCP Y Agentes

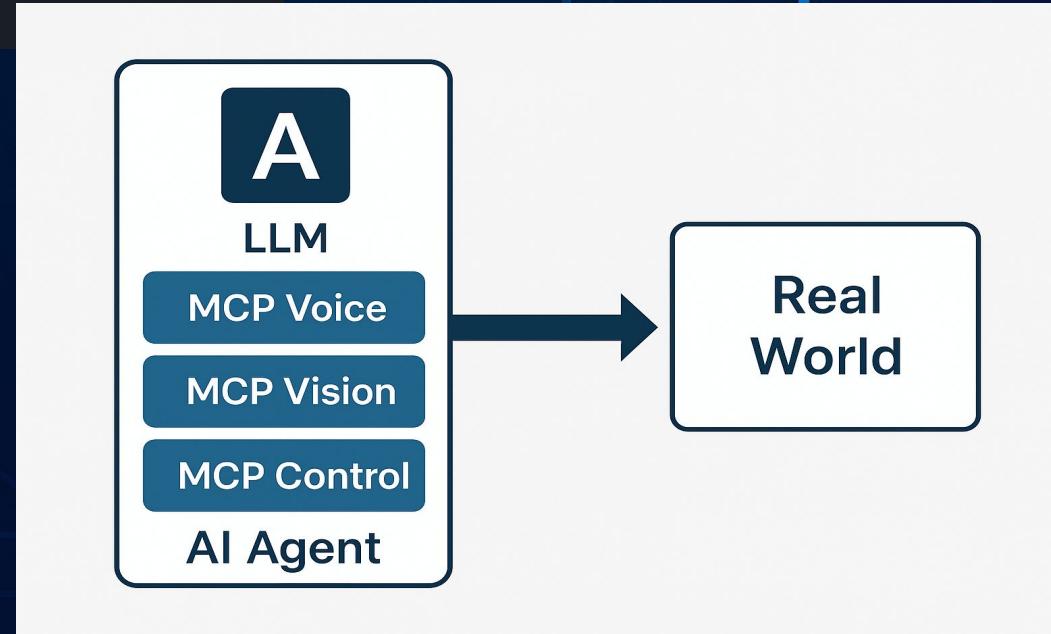


## 🧠 Agentes: entidades que razonan y actúan

Deciden qué hacer en base a objetivos, contexto y herramientas disponibles.

## 📡 MCP: interfaz universal para herramientas externas

Actúa como capa de abstracción que permite descubrir y usar herramientas de forma dinámica y segura.



# MCP Y Agentes

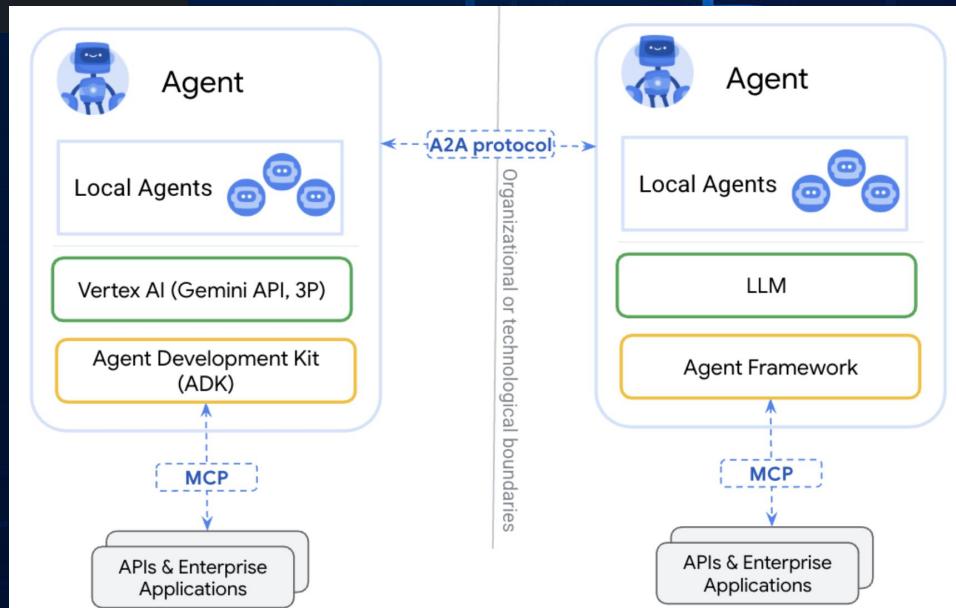


## Frameworks de agentes + MCP

Los frameworks (como LangChain, Agentic, OpenAI Agents o AutoGen) usan MCP como fuente estandarizada de *tools* y *resources*.

## Ejecución delegada a través de MCP

El agente decide qué acción tomar y la ejecuta invocando al MCP.



# Arquitectura



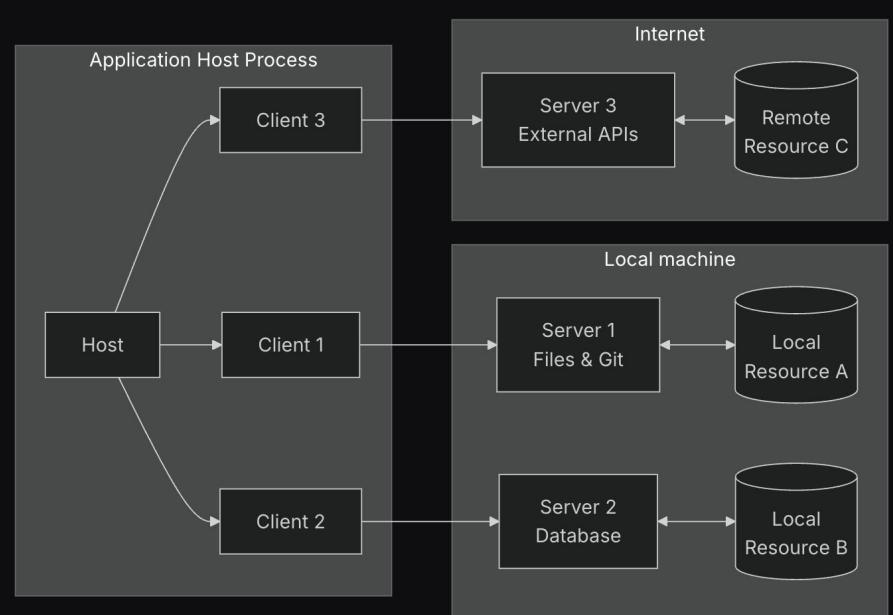
**MCP Hosts:** Programas como Claude Desktop, IDEs o herramientas de IA que quieren acceder a datos a través del protocolo MCP.

**MCP Clients:** Clientes del protocolo que mantienen conexiones 1:1 con los servidores MCP.

**MCP Servers:** Programas ligeros que exponen capacidades específicas mediante el protocolo estandarizado Model Context Protocol.

**Fuentes de datos locales:** Sistemas de ficheros, bases de datos y servicios de tu ordenador a los que los servidores MCP pueden acceder de forma segura.

**Servicios remotos:** Sistemas externos disponibles en internet (por ejemplo, mediante APIs como la de Google Maps, Drive, Telegram, Notion...) a los que los servidores MCP pueden conectarse.



# Comunicación



El protocolo utiliza mensajes JSON-RPC 2.0 para establecer la comunicación entre:

- **Hosts:** Aplicaciones con LLMs que inicián las conexiones
- **Creadores:** Conectores dentro de la aplicación host
- **Servidores:** Servicios que proporcionan contexto y capacidades

## Requests

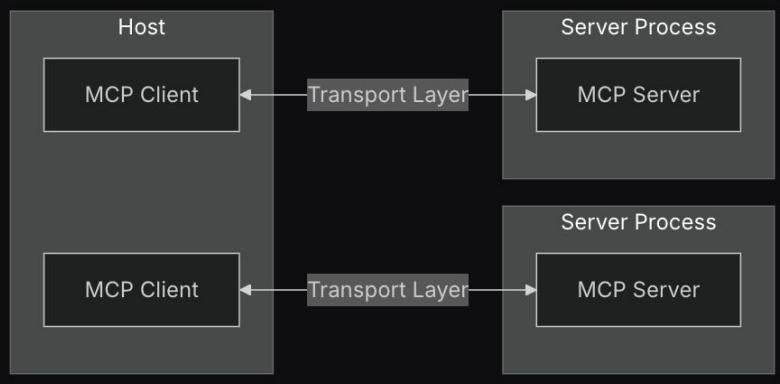
```
{  
  jsonrpc: "2.0",  
  id: number | string,  
  method: string,  
  params?: object  
}
```

## Responses

```
{  
  jsonrpc: "2.0",  
  id: number | string,  
  result?: object,  
  error?: {  
    code: number,  
    message: string,  
    data?: unknown  
  }  
}
```

## Notifications

```
{  
  jsonrpc: "2.0",  
  method: string,  
  params?: object  
}
```



## Client

## Server

```
initialize request  
initialize response  
initialized notification
```

Connection ready for use

## Client

## Server



# Demo implementación

# Implementación - SDKs



**Ecosistema de Desarrollo:** El protocolo ha fomentado un entorno con abundantes recursos para los desarrolladores:

- **SDKs Oficiales:** Compatibles, optimizados, con ejemplos prácticos.
- **SDKs de la Comunidad:** Multilenguaje, con utilidades avanzadas y abstracciones útiles.



# Implementación - estructura - instanciar server MCP



```
3
4 import { McpServer, ResourceTemplate } from '@modelcontextprotocol/sdk/server/mcp.js';
5 import { StdioServerTransport } from '@modelcontextprotocol/sdk/server/stdio.js';
6 import { z } from 'zod';
7 import { ReadResourceResult } from '@modelcontextprotocol/sdk/types';
8
9 const server = new McpServer({
10   name: 'CharlaMCP',
11   version: '1.0'
12 });
13
```

# Implementación - estructura - tools



```
14 // Ejemplo de una tool que suma dos numeros
15
16 server.tool(
17   'add',
18   'add two number',
19   { a: z.number(), b: z.number() },
20   async ({ a, b }) => ({
21     content: [{ type: 'text', text: `${a + b}` }]
22   })
23 );
```

# Implementación - estructura - Resource



```
25 // Ejemplo recurso dinámico
26
27 server.resource(
28   'greeting',
29   new ResourceTemplate('greeting://{{name}}', { list: undefined }),
30   async (uri, { name }) => {
31     contents: [
32       {
33         uri: uri.href,
34         text: `Hello, ${name}!`
35       }
36     ]
37   }
38 );
```

# Implementación - estructura - Prompt



```
44 // ~~~~~ Ejemplo de PROMPT
45
46 server.prompt('resume', {}, () => {
47   messages: [
48     {
49       role: 'user',
50       content: {
51         type: 'text',
52         text: `Make a resume in markdown from previous messages and save it as pdf`
53       }
54     }
55   ]
56 });


```

# Implementación - estructura - definir el transporte



```
57  
58 const transport = new StdioServerTransport();  
59 server.connect(transport).catch(console.error);  
60
```

```
43   · const transport = new StreamableHTTPServerTransport({  
44   ·   · sessionIdGenerator: config.sessionIdGenerator,  
45   ·   · enableJsonResponse: config.enableJsonResponse ?? false,  
46   ·   · eventStore: config.eventStore  
47   · });  
48  
49   · await mcpServer.connect(transport);  
50
```

# Depuración



MCP Inspector v0.13.0

Transport Type

STDIO

Command

npm

Arguments

run dev

> Environment Variables

Server Entry

Servers File

> Configuration

Restart

Resources Prompts Tools Ping Sampling Roots Auth

Tools

List Tools

Clear

add

add two number

add

add two number

a

5

b

10

Run Tool

Tool Result: Success

"15"

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

PLAYWRIGHT

typescript-mcp-sample npx @modelcontextprotocol/inspector npm run dev

# Buenas prácticas



## CONEXIÓN

Gestiona correctamente el ciclo de vida de la conexión

Libera los recursos al cerrar la conexión

Implementa un manejo de errores adecuado

Implementa lógica de reconexión cuando sea necesario

## ERRORES

Envuelve siempre las llamadas a herramientas en bloques try-catch

Cierra las conexiones cuando hayas terminado

Proporciona mensajes de error significativos

Gestiona de forma elegante los problemas de conexión

## SEGURIDAD

Sé cuidadoso con los permisos de las herramientas

Cierra las conexiones cuando hayas terminado

Usa AsyncExitStack para una limpieza adecuada

Aplica medidas de seguridad adecuadas

## VALIDACIONES

Valida los mensajes antes de enviarlos

Supervisa el estado de la conexión

Usa tiempos de espera (timeouts) apropiados

Valida las respuestas del servidor

# Retos de Seguridad



- 💡 **Envenenamiento del modelo:** un servidor MCP malicioso puede responder con datos manipulados para alterar el comportamiento del LLM.
  - 💡 **Escalada de permisos:** si el manifest no está bien validado, un servidor podría declarar acciones peligrosas como seguras.
  - 🔒 **Acceso a recursos privados:** un LLM podría invocar acciones sin el conocimiento del usuario si el host no exige confirmación.
  - 🔍 **Filtración de datos:** un MCP podría registrar y filtrar argumentos o respuestas si no está aislado adecuadamente.
  - 📅 **Logs falsificados:** un servidor MCP podría devolver respuestas erróneas sin trazabilidad clara si no se audita correctamente.





# Demo de uso

# Integración: Chats de usuario



Los Chatbot con soporte MCP funcionan como hosts y son capaces de crear clientes que consumen servidores MCP, de terceros o propios, simplemente con configurarlos, haciendo uso de un LLM seleccionado:

- Claude Desktop (panel MCP)
- ChatGPT (beta)
- Chats Opensource

{} claude\_desktop\_config.json ×

Users > paco.bravo > Library > Application Support > Claude > {} claude\_desktop\_config.json > ...

```
1  [
2    "mcpServers": {
3      "randomnumber": {
4        "type": "stdio",
5        "command": "/Users/paco.bravo/Downloads/mcp/servers_random_number",
6        "args": []
7      },
8      "filesystem": {
9        "command": "docker",
10       "args": [
11         "run",
12         "-i",
13         "--rm",
14         "--mount", "type=bind,src=/Users/paco.bravo/Downloads/mcp,dst=/projects/mcp",
15         "mcp/filesystem",
16         "/projects"
17       ]
18     }
19   }
```

c genera un fuchero txt donde el nombre sea un numero aleatorio que generes en este momento

Pensando sobre generando archivo de texto con nombre de número aleatorio  
random\_number\_one  
Ahora que tengo el número aleatorio (4), voy a crear un archivo txt con ese nombre en el directorio de trabajo  
write\_file  
Generé archivo con nombre de número aleatorio exitosamente.

¡Listo! He creado un archivo TXT con un nombre aleatorio.

## Detalles:

- Número aleatorio generado: 4
- Archivo creado: [4.txt](#)
- Ubicación: [/projects/mcp/4.txt](#)

El archivo contiene información sobre su creación automática. ¿Te gustaría ver el contenido del archivo o hacer algo más con él?



Reintentar

Claude puede cometer errores. Por favor, verifique las respuestas.

# Integración: Editores (IDE)



VS Code interface showing a file tree and code editor:

```
{} mcp.json X @ main.rs 1, U  
.vscode > {} mcp.json > {} servers > {} file-system > [ ] args  
1 {  
2   "servers": {  
3     "rust-stdio-sample": {  
4       "type": "stdio",  
5       "command": "/Users/juanjo/Projects/IA/rust-stdio-sample",  
6       "args": []  
7     },  
8     "file-system": {  
9       "command": "npx",  
10      "args": [  
11        "-y",  
12        "@modelcontextprotocol/server-fs",  
13        "/Users/juanjo/Downloads"  
14      ]  
15    },  
16  },  
17 }
```

File Explorer sidebar:

- File
- Search
- Problems (6)
- Run
- Terminal
- Icons (1)
- Help

Code Editor sidebar:

- File
- Edit
- View
- Insert
- Format
- Refactor
- Tools
- Output
- Terminal
- Help

Bottom right corner: Ask auto...

A modal dialog titled "Select tools that are available to chat" is open, listing various tools and their descriptions. A "39 Selected" button and an "OK" button are at the top right of the modal.

Tool	Description
increment	Increment the counter by 1
say_hello	Say hello to the client
sum	Calculate the sum of two numbers
MCP Server: file-system	from rust-mcp/vscode/mcp.json
create_directory	Create a new directory or ensure a directory exists. Can create multiple nested directories.
directory_tree	Get a recursive tree view of files and directories as a JSON structure. Each entry includes the path and file type.
edit_file	Make line-based edits to a text file. Each edit replaces exact line sequences with new ones.
get_file_info	Retrieve detailed metadata about a file or directory. Returns comprehensive information like size, type, and last modified date.
list_allowed_directories	Returns the list of directories that this server is allowed to access.
list_directory	Get a detailed listing of all files and directories in a specified path. Results can be paginated.
move_file	Move or rename files and directories. Can move files between directories and rename them.
read_file	Read the complete contents of a file from the file system. Handles various text encodings.
read_multiple_files	Read the contents of multiple files simultaneously. This is more efficient than reading each file individually.
search_files	Recursively search for files and directories matching a pattern. Searches through entire file trees.
write_file	Create a new file or completely overwrite an existing file with new content. Use with caution.

# Integración: Frameworks



## OpenAI Agents

```
# Configurar agente
agent = MCPOpenAIAgent(
    openai_api_key="tu-api-key-aqui",
    mcp_server_command=["python", "servidor.py"]
)
```

## Claude SDK

```
class ClaudeAgentMCP:
    """Agente Claude con configuración MCP"""

    def __init__(self, api_key: str, mcp_configs: List[MCPServerConfig]):
        self.client = anthropic.Anthropic(api_key=api_key)
        self.mcp_configs = mcp_configs
        self.mcp_processes = {}
```

## Google ADK

```
# Configuración MCP
mcp_config = MCPConfig(
    server_command=mcp_server_command,
    server_name="mcp-server",
    capabilities=["tools", "resources"],
    timeout=30
)

# Crear agente
agent = GeminiMCPAgent(gemini_config, mcp_config)
```

## Agentic Framework

```
# Crear agente con MCP
agent = OpenAIAGentMCP(
    agent_name="Asistente MCP",
    instructions="Eres un asistente que usa herramientas MCP para cálculos y clima",
    mcp_configs=mcp_servers
)

# Inicializar (configura MCP automáticamente)
await agent.initialize_agent()
```

## LangChain

```
# Configurar LLM
llm = ChatOpenAI(
    temperature=0.7,
    model_name="gpt-4",
    openai_api_key="tu-openai-key"
)

# Crear agente con MCP
agent = LangChainMCPAgent(llm, mcp_servers)
```

# Ventajas clave



# Enlaces de interés



## Manuscritos Fundamentales (Introducción a MCP)

- Introducción oficial a MCP  
👉 <https://modelcontextprotocol.io/introduction>
- Video explicativo (YouTube)  
👉 <https://www.youtube.com/watch?v=kQmXtrmQ5Zg>

## Crónicas de la Guerra Digital (Seguridad y Vulnerabilidades)

- Software Supply Chain Attack Surface – Martin Fowler  
👉 <https://martinfowler.com/articles/exploring-gen-ai/software-supply-chain-attack-surface.html>
- MCP GitHub Vulnerability – Invariant Labs  
👉 <https://invariantlabs.ai/blog/mcp-github-vulnerability>
- Análisis técnico en YouTube  
👉 <https://www.youtube.com/watch?v=wnHczxwukYY>



# Enlaces de interés

## 💡 Herramientas del Herrero Elfo (SDKs y Protocolos)

- SDK en TypeScript  
👉 <https://github.com/modelcontextprotocol/typescript-sdk>
- SDK en Rust  
👉 <https://github.com/modelcontextprotocol/rust-sdk>
- Especificación JSON-RPC  
👉 <https://www.jsonrpc.org/specification>

## 🏰 Portales de Sabiduría Antigua (A2A & ADK de Google)

- Google A2A  
👉 <https://google.github.io/A2A/>
- Google ADK Docs  
👉 <https://google.github.io/adk-docs/>

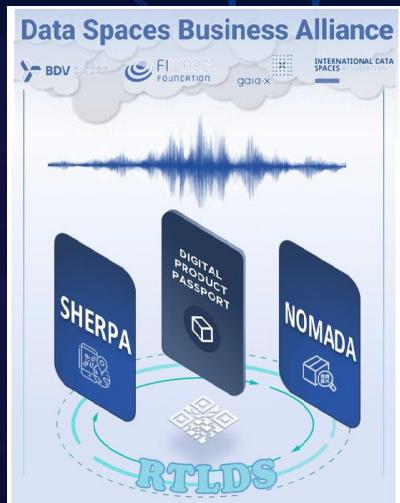


# Gracias por venir y estamos a vuestra disposición



<https://centic.es/pacmania/>

Plataforma para la automatización de la  
Ciberseguridad basada en IA



<https://centic.es/rtlds/>

Sistemas de localización Indoor para  
Espaces de Datos (Real Time Location  
for Data Spaces)

