



# INTRODUCCIÓN A MODEL CONTEXT PROTOCOL (MCP) CON C#

Con Javier Suárez



# Javier Suárez Ruiz

## Software Engineer at Microsoft

- Email:  
[javiersuarezruiz@hotmail.com](mailto:javiersuarezruiz@hotmail.com)
- X: @jsuarezruiz







r/ClaudeAI • 4 mo. ago  
ceremy

## Anthropic's Model Context Protocol (MCP) people think

News: Official Anthropic news and announcements



Steven Sinofsky @stevesi · Mar 30

if you don't subscribe to [@benedictevans](#) newsletter you  
Here's what he had to say this week on **MCP**.



Santiago @svpino · Mar 10

**MCP** is not just another dumb abstraction. So far, I like it a lot, and I think  
the hype is well-deserved.

Here is an explanation of two of the most exciting features of **MCP**:



Angie Jones @techgirl1908 · Mar 30

Excellent writeup on how **MCP** future-proofs API integrations ~ [@nilslice](#)



Jaana Dogan ヤナ ドガン @rakyll · Feb 14

**MCP** tutorials are great. There are no tutorials really.

"Copy these resources to Claude, and start asking some questions like



Cory Zue @czue · Apr 2

There are only two types of people in this world: (1) People who think **MCP**  
is dumb/hype and (2) people who have actually used **MCP**.

I just went from camp 1 to camp 2 and holy fuck.

# MCP

## THE FUTURE?

LLMs + MCP

# MCP

## The Different

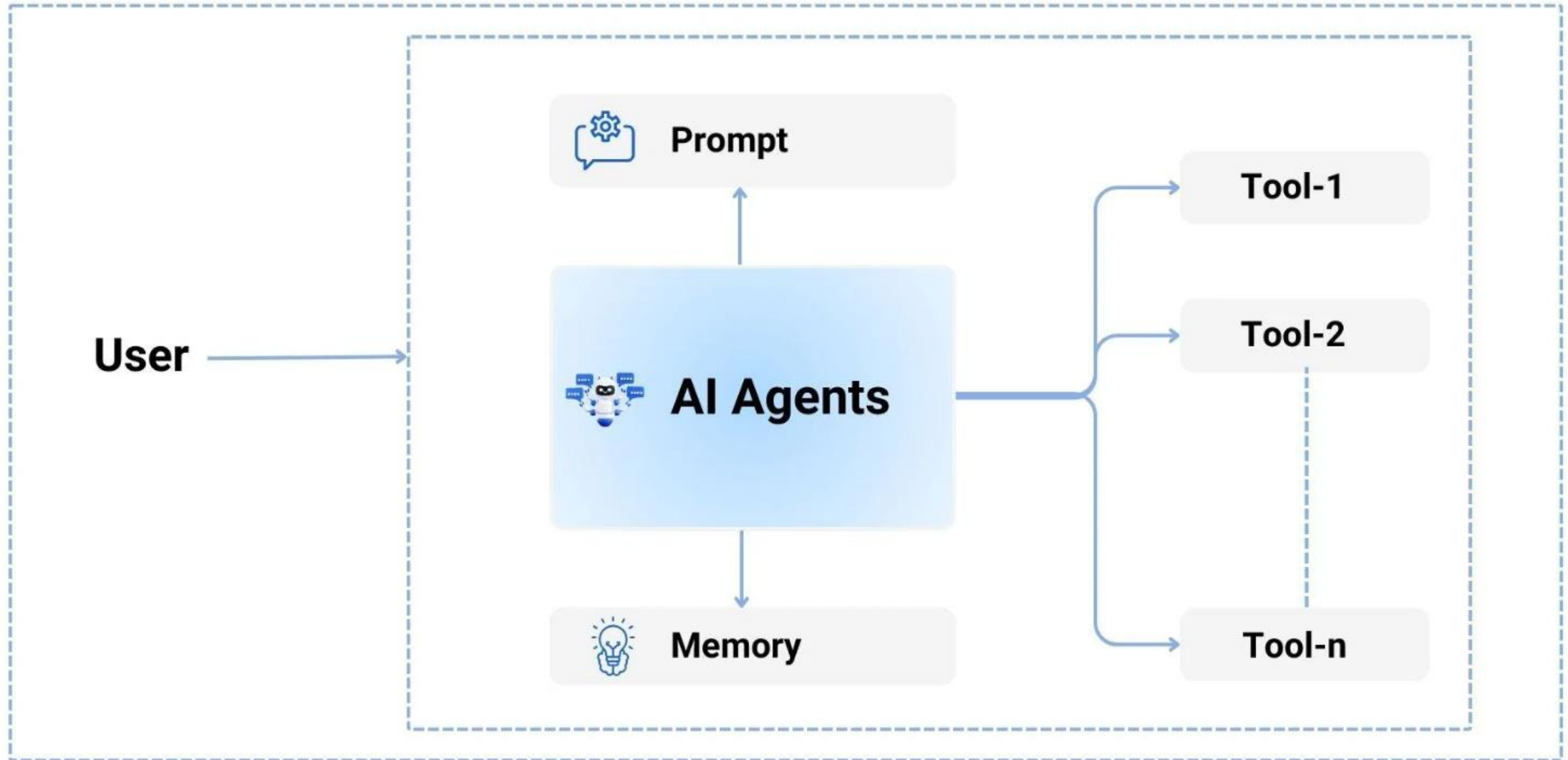
MCP: The Differential for modern APIs and Systems

# La agenda

1. **¿Por qué?** – La necesidad que resuelve MCP
2. **¿Qué es?** – Conociendo todo lo necesario de MCP
3. **¿Cómo?** Usando MCPs y desarrollando nuestro propio cliente, servidor, etc. Además, aprenderemos a usar herramientas como MCP Inspector y veremos casos prácticos.

Por qué?  
La necesidad de  
MCP

# Aplicación típica de IA con un único agente



# Desafíos de las aplicaciones actuales de IA

- GitHub ofrece funcionalidad para gestionar código, incidencias, Pull Requests y más.
- Su objetivo es ampliar estas capacidades mediante la integración con diversos IDE basados en IA, como VS Code, Cursor, Windsurf, Zed, Cline y otros.
- Para lograrlo, deben desarrollar integraciones de GitHub específicas para cada IDE, implementándolas paso a paso para garantizar la compatibilidad.

# Puntos críticos para los desarrolladores

Para desarrolladores de herramientas:

- Si GitHub planea integrarse con las 100 principales aplicaciones de IA, debe crear e implementar integraciones individualmente para cada aplicación, atendiendo a sus requisitos y características específicos.

Para desarrolladores de aplicaciones de IA:

- Si Rider busca integrarse con la herramienta VS Code GitHub Copilot de GitHub, no podrá reutilizarla directamente. En su lugar, deberá desarrollar una nueva integración adaptada a su plataforma desde cero.



# La demanda de estandarización

Necesidad de un protocolo universal para optimizar las integraciones de IA:

- Reducir la fragmentación
- Promover la interoperabilidad

# DEMO:

## Usando GitHub Copilot y alcanzando límites

# ¿Qué? Conceptos básicos de MCP

# ¿Qué es el Model Context Protocol (MCP)?

**MCP** es un protocolo abierto que permite una integración perfecta entre las aplicaciones **LLM** y sus herramientas y **fuentes de datos**.

## APIs

Estandariza cómo las **aplicaciones web** interactúan con el **backend**:

- Servidores
- Base de datos
- Servicios

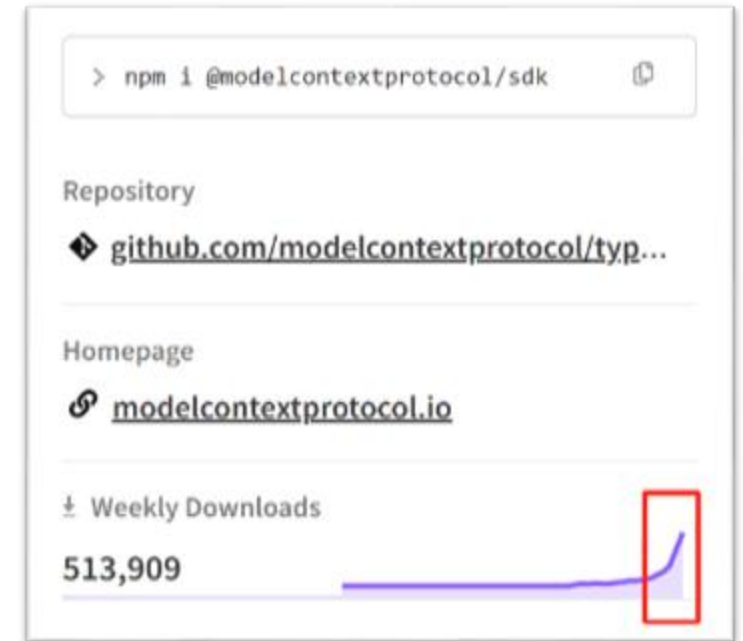
## MCP

Estandariza cómo las aplicaciones de IA interactúan con los **sistemas externos**:

- Prompts
- Tools
- Data & resources
- Sampling

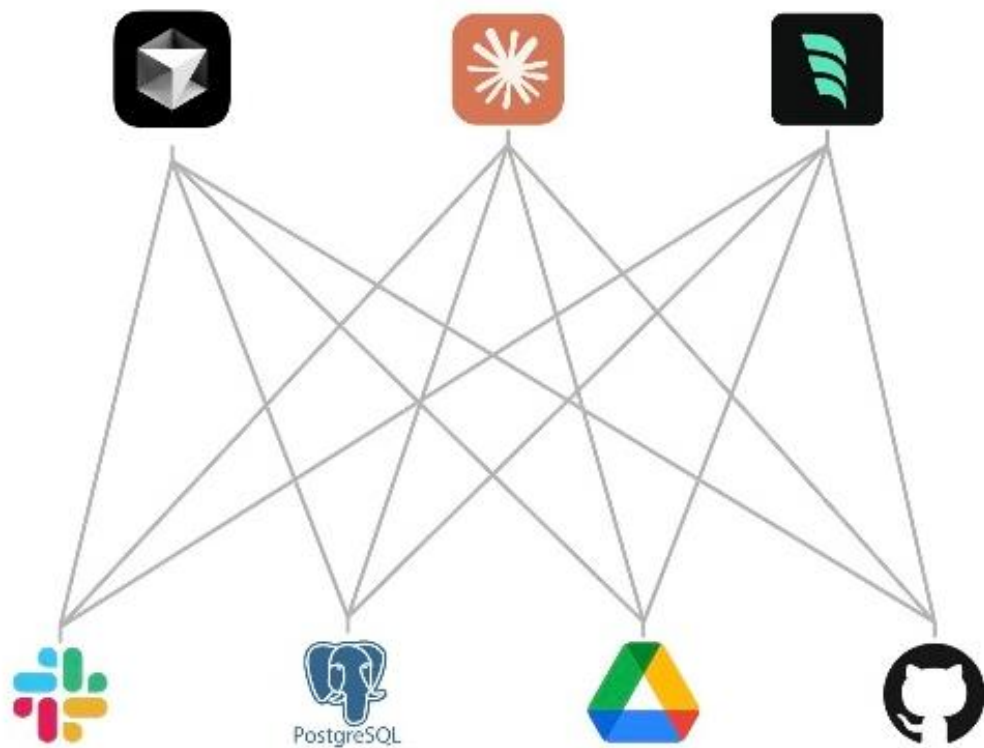
# Timeline & Trend

- Noviembre de 2024: Anthropic anunció MCP y Zed lo soporta.
- Diciembre de 2024: Cline añade soporte.
- Enero de 2025: Cursor añade soporte.
- Febrero de 2025: Windsurf añade soporte
- Marzo de 2025: VS Code añade soporte

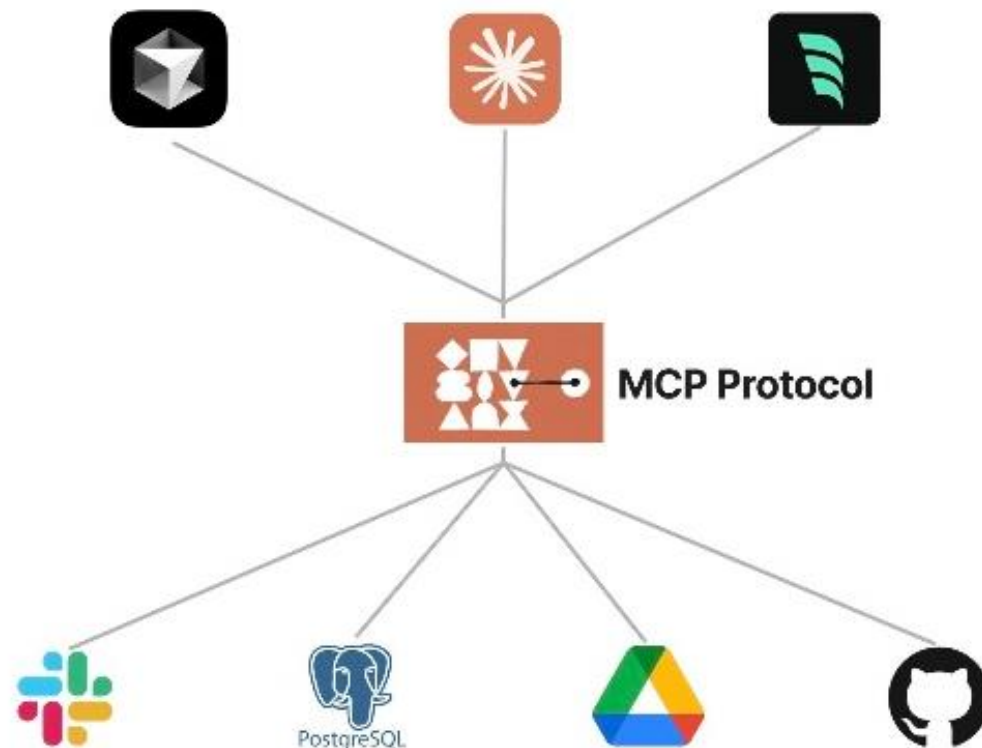




## Without MCP



## With MCP



# Con MCP: Desarrollo de IA estandarizado

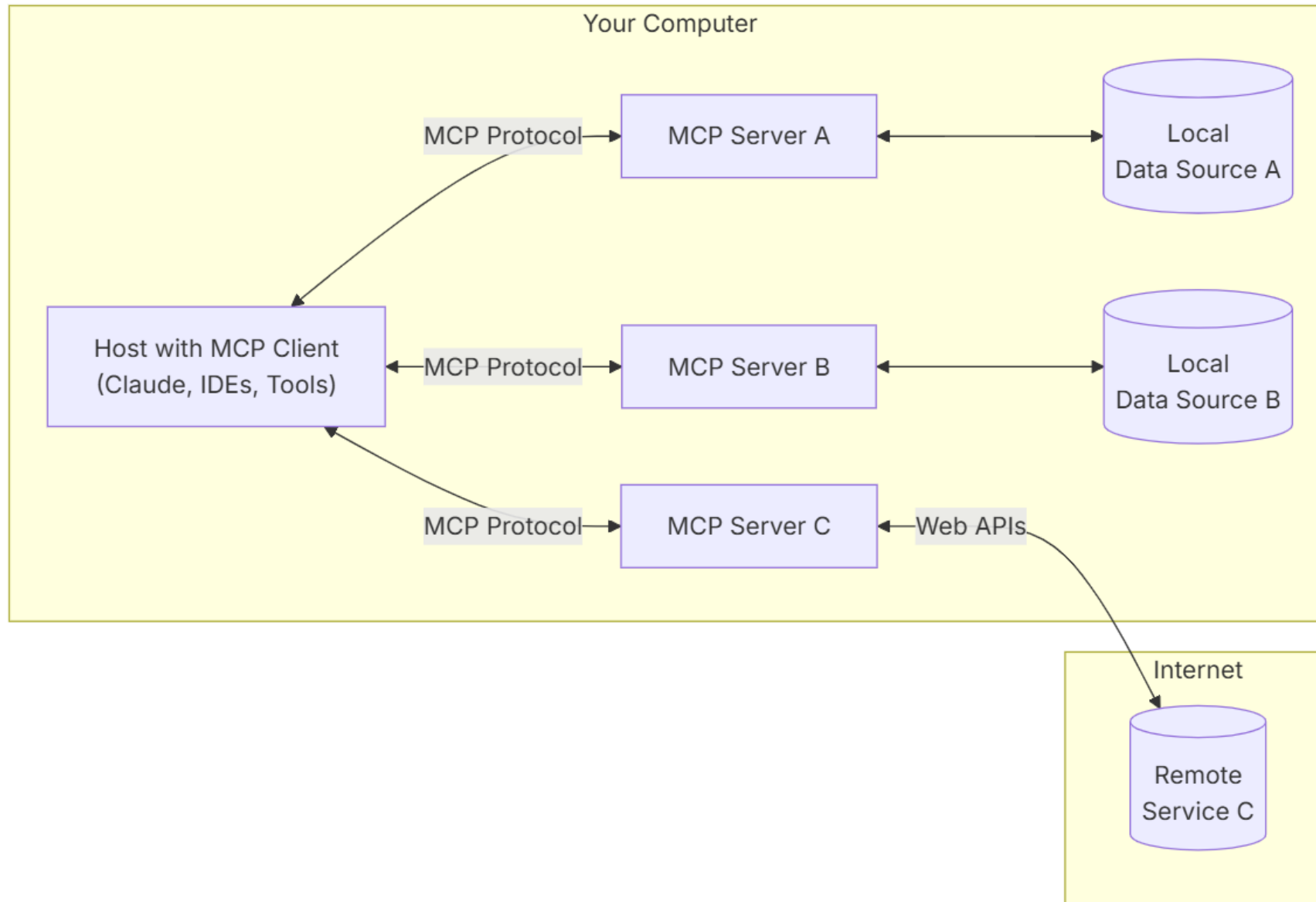
Para desarrolladores de aplicaciones de IA

- Conecta tu aplicación a cualquier servidor MCP sin esfuerzo adicional

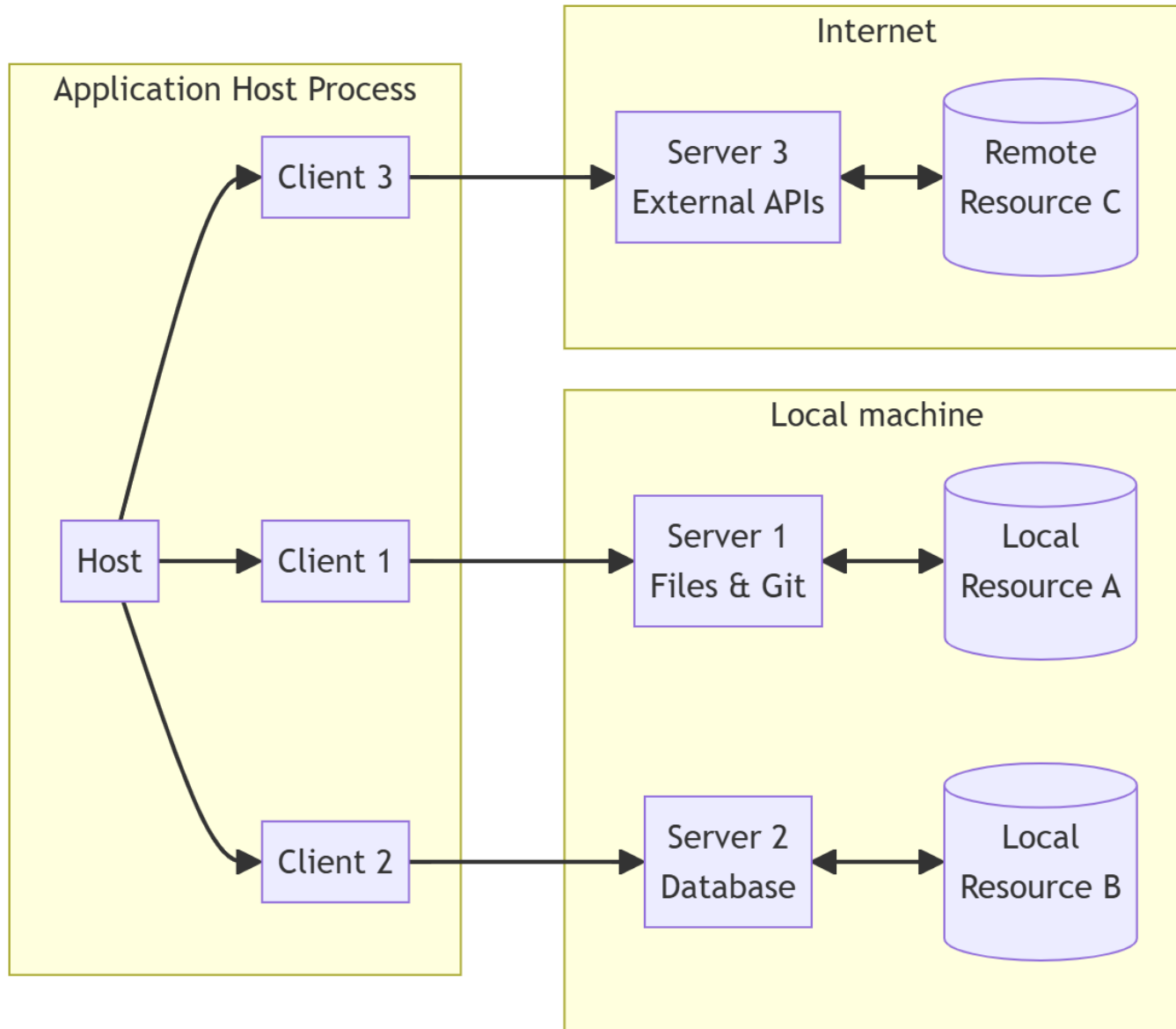
Para desarrolladores de herramientas o API

- Construye un servidor MCP una vez y observa su adopción en todas partes

# Arquitectura



# Componentes



- **Hosts MCP:** Programas como Claude Desktop, IDEs o herramientas de IA que desean acceder a datos a través de MCP.
- **Cientes MCP:** Clientes que mantienen conexiones 1:1 con servidores.
- **Servidores MCP:** Programas que exponen capacidades específicas a través del Protocolo de Contexto de Modelo estandarizado.

# Transports

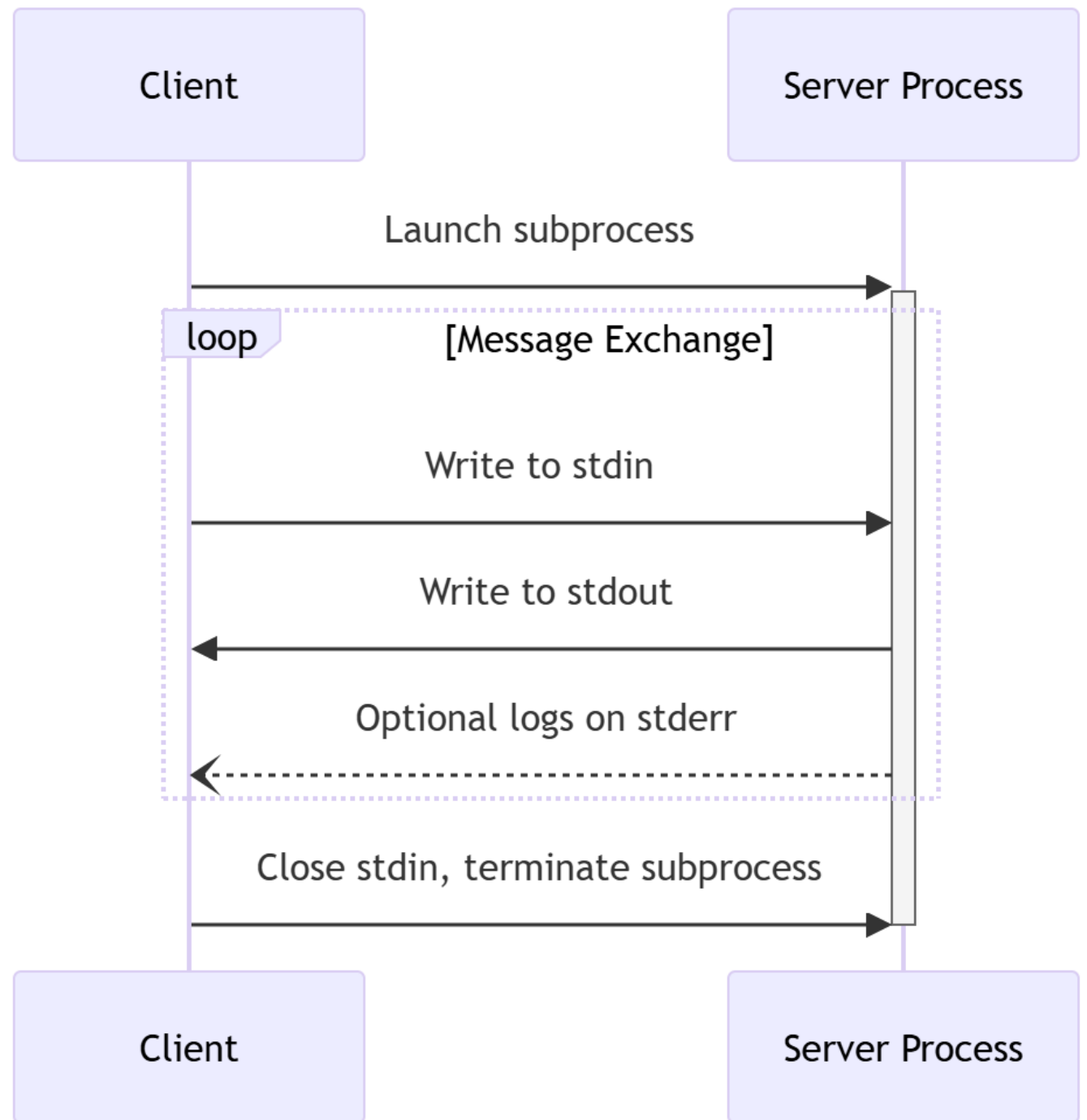
Dos mecanismos de transporte estándar para la comunicación cliente-servidor:

- **stdio**, comunicación mediante entrada y salida estándar
- HTTP con eventos enviados por el servidor (**SSE**)



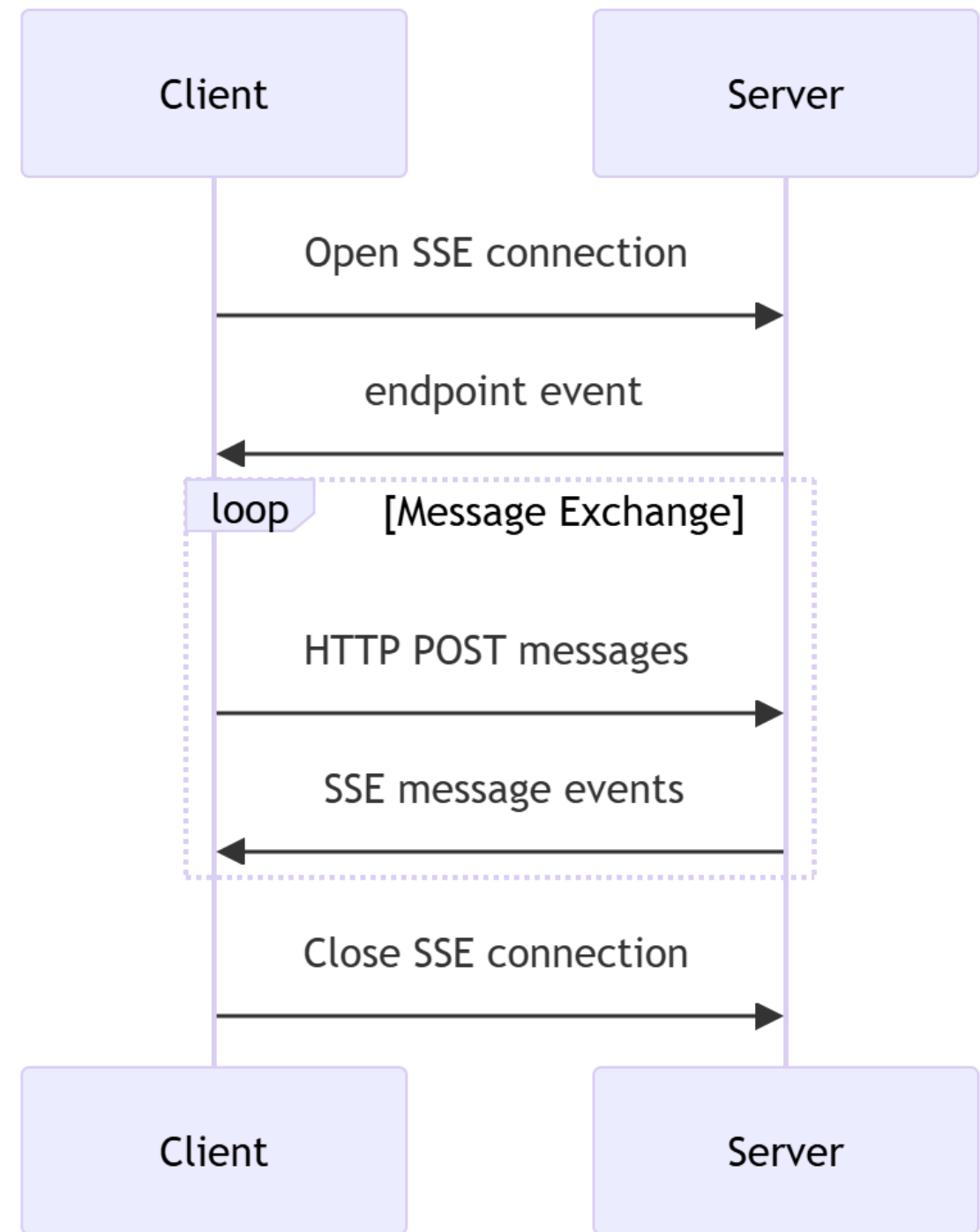
# stdio

- El servidor solo podía ejecutarse localmente.
- El cliente inicia el servidor MCP como subprocesso.
- El servidor recibe mensajes JSON-RPC en su entrada estándar (stdin) y escribe las respuestas en su salida estándar (stdout).

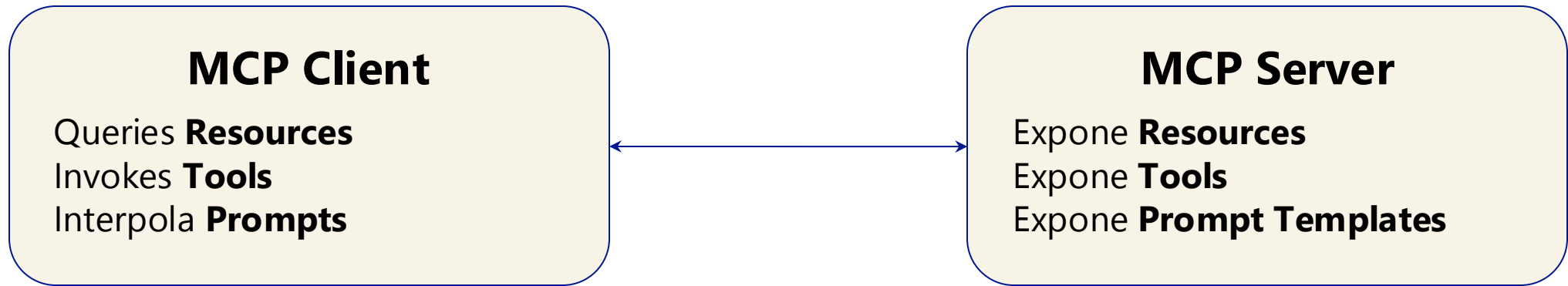


# HTTP con SSE

- El servidor puede ejecutarse tanto local como remotamente.
- El servidor DEBE proporcionar dos puntos finales:
  - Un punto final SSE para que los clientes establezcan una conexión y reciban mensajes del servidor.
  - Un punto final HTTP POST para que los clientes envíen mensajes al servidor.



# Características del servidor



¿Cómo?  
Usando e  
implementando MCP

# Usando MCP

1. Instala la aplicación compatible con las integraciones de MCP:

<https://modelcontextprotocol.io/clients>

2. Instala/configura el servidor MCP:

<https://modelcontextprotocol.io/examples>



# Aplicaciones populares que admiten MCP

- Claude Desktop
- Cline
- VS Code

<https://modelcontextprotocol.io/clients>

# Servidores MCP populares

- <https://mcp.so/servers>
- <https://glama.ai/mcp/servers>
- <https://smithery.ai/>
- <https://www.pulsemcp.com/servers>


4431 MCP Servers

## Find Awesome MCP Servers and Clients

The largest collection of MCP Servers.


Search with keywords

Featured MCP Servers

 **Amap Maps**  
by amap


高德地图 MCP Server

# amap # maps

 **Playwright**  
by microsoft


A Model Context Protocol (MCP) server that provides browser automation capabilities using...

# playwright # browser-automation

 **Baidu Map**  
by baidu-maps


百度地图核心API现已全面兼容MCP协议，是国内首家兼容MCP协议的地图服务商。

# baidu-map # location-services

 **Tavily MCP Server**  
by tavily-ai


Compatible with Cline, Cursor, Claude Desktop, and any other MCP Clients! Tavily MCP is also...

# tavily-mcp # mcp-server

 **Blender**  
by ahujasid


BlenderMCP connects Blender to Claude AI through the Model Context Protocol (MCP), allowing...

# blender # 3d-modeling

 **Perplexity Ask MC...**  
by ppl-ai


A Model Context Protocol Server connector for Perplexity API, to enable web search without leavin...

# mathgpt # math-solver

 **AgentQL MCP...**  
by tinyfish-io

Model Context Protocol server that integrates AgentQL's data extraction capabilities.

# agent # web

 **Figma MCP Server**  
by GLips

MCP server to provide Figma layout information to AI coding agents like Cursor

# typescript # ai

View All

Smithery

Docs + Add Server

Search or prompt for servers...

Code Runner MCP Server

@formulahendry/mcp-server-code-runner

Overview Tools API

Deployments Settings

Edit Configuration

run-code

Run code snippet and return the result.

code \*  
print(1+2)

languageId \*  
python

Run

Results

3

MCP Server Playground

Calling MCP Server Tools online

MCP Servers

fetch  
amap-maps  
playwright-mcp  
baidu-map  
tavily-mcp  
aws-kb-retrieval-...  
time  
sequentialthinking  
perplexity  
agentql-mcp

Playwright

A Model Context Protocol (MCP) server that provides browser automation capabilities using Playwright. This server enables LLMs to interact with web pages through structured accessibility snapshots, bypassing the need for screenshots or visually-tuned models.

Tools

browser\_navigate  
browser\_go\_back  
browser\_go\_forward  
browser\_snapshot  
browser\_click  
browser\_hover

Connect Server with SSE URL

Claude Cursor Windsurf Cline ChatWise

```
{
  "mcpServers": {
    "@microsoft/playwright-mcp": {
      "url": "https://router.mcp.so/sse/ph3zv1m8pd3fxw"
    }
  }
}
```

# VS Code soporta MCP

The screenshot displays the Visual Studio Code (VS Code) interface with the following components:

- Editor:** The `settings.json` file is open, showing the configuration for the MCP (Model Context Protocol) server. The `my-mcp-server-github` configuration is highlighted with a red box.
- Settings:** The `settings.json` file is open, showing the configuration for the MCP (Model Context Protocol) server. The `my-mcp-server-github` configuration is highlighted with a red box.
- Tools List:** A list of tools available to the chat is shown, including `Test Failure`, `Terminal Selection`, `Terminal Last Command`, `mcp-server-time`, `get_current_time`, `convert_time`, `mcp-server-everything`, `echo`, `add`, `printEnv`, `longRunningOperation`, `sampleLLM`, `getTinyImage`, `annotatedMessage`, `my-mcp-server-github`, `get_user`, `get_issues`, and `authorize_github`. The `my-mcp-server-github` configuration is highlighted with a red box.
- GitHub Copilot:** The GitHub Copilot interface is shown on the right, displaying the `authorize_github` tool. The `authorize_github` tool is highlighted with a red box.
- Terminal:** The terminal shows the output of the `my-mcp-server-github` server, indicating that it is running.

```
20 "spectral.rulesetFile": "https://raw.githubusercontent.com/azure/az...
21 "security.workspace.trust.untrustedFiles": "open",
22 "fx-extension.enableMicrosoftKiota": true,
23 "github.copilot.selectedCompletionModel": "gpt-4o-copilot",
24 "github.copilot.nextEditSuggestions.enabled": true,
25 "mcp": {
26   "inputs": [],
27   "servers": {
28     "mcp-server-time": {
29       "command": "python",
30       "args": [
31         "-m",
32         "mcp_server_time",
33         "--local-timezone=America/Los_Angeles"
34       ],
35       "env": {}
36     },
37     "mcp-server-everything": {
38       "type": "sse",
39       "url": "http://localhost:3001/sse"
40     },
41     "my-mcp-server-github": {
42       "type": "sse",
43       "url": "https://mcp-demo-dev.azure-api.net/github/sse"
44     }
45   }
46 }
47 }
48 }
```

Tools available to chat (23 Selected):

- Test Failure: Includes information about the last unit test failure.
- Terminal Selection: The active terminal's selection
- Terminal Last Command: The active terminal's last run command
- mcp-server-time: MCP - Global in Code - Insiders (Running)
- get\_current\_time: Get current time in a specific timezones
- convert\_time: Convert time between timezones
- mcp-server-everything: MCP - Global in Code - Insiders (Running)
- echo: Echoes back the input
- add: Adds two numbers
- printEnv: Prints all environment variables, helpful for debugging MCP server configuration
- longRunningOperation: Demonstrates a long running operation with progress updates
- sampleLLM: Samples from an LLM using MCP's sampling feature
- getTinyImage: Returns the MCP\_TINY\_IMAGE
- annotatedMessage: Demonstrates how annotations can be used to provide metadata about content
- my-mcp-server-github: MCP - Global in Code - Insiders (Running)
- get\_user: Get user associated with GitHub access token. Returns: GitHub user information
- get\_issues: Get all issues for the specified repository for the authenticated user. Args: username
- authorize\_github: Validate Credential Manager connection exists and is connected. Args: id...

GitHub Copilot

junhan\_microsoft

login to my github account

GitHub Copilot

✓ Ran `authorize_github`

Please visit the following URL to authorize the connection to your GitHub account: [Authorize GitHub](#)

Once you have authorized, please let me know so we can proceed.

junhan\_microsoft

get user info for my github account

GitHub Copilot

Run `get_user` from `my-mcp-server-github` (MCP server)

Get user associated with GitHub access token.

Returns: GitHub user information

Input:

`{}`

⚠ MCP servers or malicious conversation content may attempt to misuse 'Code - Insiders' through the installed tools. Please carefully review any requested actions.

Continue Cancel

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE ... Filter

2025-03-18 12:21:23.562 [info] Starting server my-mcp-server-github

2025-03-18 12:21:23.562 [info] Connection state: Starting

2025-03-18 12:21:24.335 [info] Connection state: Running

main\* Launchpad 0 0 0

Ln 26, Col 22 Spaces: 4 UTF-8 CRLF {} JSON with Comments kiota 1.24.1

# Modos de GitHub Copilot en VS Code

## Modo Ask (Preguntar)

- Función: Asistencia conversacional para dudas de desarrollo
- Características:
  - Responde preguntas sobre código en lenguaje natural
  - Genera explicaciones y documentación

## Modo Edit (Editar)

- Función: Transformación directa de código mediante instrucciones
- Características:
  - Modifica código existente según tus indicaciones
  - Corrige errores y refactoriza automáticamente

## Modo Agent (Agente)

- Función: Completa tareas complejas de forma autónoma
- Características:
  - Ejecuta flujos de trabajo de múltiples pasos
  - Resuelve problemas en varios archivos

# Instalar/configurar el servidor MCP en VS Code

## npx for VS Code

Configuration in `settings.json`:

```
{
  "mcp": {
    "inputs": [],
    "servers": {
      "mcp-server-code-runner": {
        "command": "npx",
        "args": [
          "-y",
          "mcp-server-code-runner"
        ],
      },
    },
  },
}
```

## Docker

Use VS Code as example. Configuration in `settings.json`:

```
{
  "mcp": {
    "inputs": [],
    "servers": {
      "mcp-server-code-runner": {
        "command": "docker",
        "args": [
          "run",
          "--rm",
          "-i",
          "formulahendry/mcp-server-code-runner"
        ],
      },
    },
  },
}
```

```
code --add-mcp '{"name":"mcp-server-code-runner","command":"npx","args":["-y", "mcp-server-code-runner"]}'
```



DEMO:  
Usando GitHub  
Copilot de nuevo  
usando un MCP

# Implementando MCP

- SDKs para crear clientes MCP y servidores MCP
  - [TypeScript SDK](#)
  - [Python SDK](#)
  - [Java SDK](#)
  - [Kotlin SDK](#)
  - [C# SDK](#)
- Tooling
  - [MCP Inspector](#): Herramienta de desarrollo interactiva para probar y depurar MCP Server
  - [generator-mcp](#): Yeoman Generator para MCP Server

# Crear un MCP Server

## Prerrequisitos:

- VS Code
- .NET
- Node.js

## Pasos:

1. Crear una aplicación de consola .NET.
2. Agregar el paquete NuGet ModelContextProtocol  
<https://www.nuget.org/packages/ModelContextProtocol>
3. Implementar la lógica de la herramienta para el servidor MCP
4. Depurar/probar el servidor MCP en el Inspector MCP
5. Ejecutar el servidor MCP en el modo Agente de VS Code

# DEMO:

## Creando un servidor MCP

# Iniciando nuestro servidor

Hay que actualizar la clase Program.cs con algunos andamiajes básicos para crear el servidor MCP, configurar el transporte del servidor estándar e indicarle al servidor que busque herramientas (o API disponibles) en el ensamblaje en ejecución.

```
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using ModelContextProtocol.Server;
using System.ComponentModel;

var builder = Host.CreateEmptyApplicationBuilder(settings: null);
builder.Services
    .AddMcpServer()
    .WithStdioServerTransport()
    .WithToolsFromAssembly();

await builder.Build().RunAsync();
```

# Tools

Las herramientas permiten que la IA haga más que hablar: le permiten actuar.

- Ejemplos: Concertar una reunión, enviar un correo electrónico, generar un informe.
- **Propósito:** Actuar en su nombre.
- En conjunto, los prompts, los resources y las tools le dan a MCP su potencial real.

# Definiendo una tool

En nuestro código de inicio, **WithToolsFromAssembly** escaneará el ensamblado en busca de clases con el atributo **McpServerToolType** y registrará todos los métodos con el atributo **McpServerTool**. Observa que **McpServerTool** tiene una descripción que se enviará a cualquier cliente que se conecte al servidor. Esta descripción ayuda al cliente a determinar qué herramienta llamar.

```
[McpServerToolType]
public static class EchoTool
{
    [McpServerTool, Description("Echoes the message back to the client.")]
    public static string Echo(string message) => $"Hello from C#: {message}";

    [McpServerTool, Description("Echoes in reverse the message sent by the client.")]
    public static string ReverseEcho(string message) => new
string(message.Reverse().ToArray());
}
```

# Buenas prácticas al crear Tools

- Proporciona nombres y descripciones claros y descriptivos.
- Incluye ejemplos en las descripciones de las herramientas para demostrar cómo el modelo debería usarlos.
- Implementa la validación de errores adecuados.
- Usa informes de progreso para operaciones largas.
- Mantenga las operaciones de la herramienta enfocadas y atómicas.



# MCP Inspector

`npx @modelcontextprotocol/inspector dotnet run`

The screenshot displays the MCP Inspector v0.7.0 interface. On the left sidebar, the 'Transport Type' is set to 'STDIO', the 'Command' is 'node', and the 'Arguments' field contains 'rs/src/everything/dist/index.js'. Below this, there are buttons for 'Environment Variables', 'Configuration', and a 'Connect' button which is currently disabled. A 'Connected' status indicator is shown below the 'Connect' button. The 'Logging Level' is set to 'debug'. At the bottom of the sidebar, there is a 'Light' theme selector and some utility icons.

The main panel is divided into three sections. The top section, titled 'Tools', contains a 'List Tools' button and a 'Clear' button. Below these, a list of tools is shown: 'echo' (Echoes back the input), 'add' (Adds two numbers), 'printEnv' (Prints all environment variables, helpful for debugging MCP server configuration), 'longRunningOperation' (Demonstrates a long running operation with progress updates), and 'sampleLLM' (Samples from an LLM using MCP's sampling feature). The bottom section, titled 'History', shows a list of recent tool calls: '2. tools/call' and '1. tools/list'. The right section, titled 'echo', shows the details of the selected tool. It includes a description 'Echoes back the input', a 'message' input field containing 'ping', a 'Run Tool' button, and a 'Tool Result: Success' message displaying '"Echo: ping"'. Below this, the 'Server Notifications' section shows a list of notifications: '4. notifications/message', '3. notifications/message', '2. notifications/message', and '1. notifications/message'.

<https://github.com/modelcontextprotocol/inspector>

DEMO:  
Creando un cliente  
MCP y usando MCP  
Inspector

# Prompts

Los Prompts guían la respuesta de la IA y permiten a los usuarios activar tareas específicas con clics simples.

- **Propósito:** Mantiene las conversaciones enfocadas.
- **Control del usuario:** Tú eliges las indicaciones; la IA las sigue.
- **Personalización:** Adapta fácilmente las indicaciones a diferentes necesidades.

# Definiendo un Prompt

En nuestro código de inicio, **WithPromptsFromAssembly** escaneará el ensamblado en busca de clases con el atributo **McpServerPromptType** y registrará todos los métodos con el atributo **McpServerPrompt**.

```
[McpServerPrompt(Name = "SevillaDotNetBasicPrompt"), Description("A simple prompt  
without arguments about SevillaDotNet.")]  
public static string GetBasicPrompt() => "Provide a detailed summary of SevillaDotNet,  
the .NET user group based in Seville, Spain.";
```

# Buenas prácticas al crear Prompts

- Utiliza nombres claros y descriptivos.
- Proporciona descripciones detalladas de las indicaciones y los argumentos.
- Implementa la gestión de errores.
- Documenta los formatos de argumentos esperados.
- Prueba los Prompts con varias entradas.

# DEMO:

## Usando y analizando el código de servidores MCP

<https://github.com/jsuarezruiz/mobile-dev-mcp-server>

<https://github.com/jsuarezruiz/maui-graphics-mcp-server>

# Recursos

Esta es la base de conocimientos de tu IA. Documentos, archivos, conjuntos de datos: todo lo que el sistema necesita para comprender tu entorno.

- **Propósito:** Proporciona contexto a la IA.
- **Flexibilidad:** Muestra los recursos en listas, herramientas de búsqueda o visualización automática.
- **Actualizaciones en tiempo real:** La IA se mantiene actualizada a medida que cambian los recursos.

# Publicar un MCP Server

.NET facilita la creación de imágenes de contenedor para cualquier aplicación .NET. Solo hay que añadir la configuración necesaria al archivo del proyecto:

```
<PropertyGroup>
  <EnableSdkContainerSupport>true</EnableSdkContainerSupport>
  <ContainerRepository>jsuarezruiz/mobile-dev-mcp-server</ContainerRepository>
  <ContainerFamily>alpine</ContainerFamily>
  <RuntimeIdentifiers>linux-x64;linux-arm64</RuntimeIdentifiers>
</PropertyGroup>
```

Si queremos tomar estas imágenes y subirlas, podemos hacerlo desde CLI, pasando el registro del contenedor específico al que se enviarán:

```
dotnet publish /t:PublishContainer -p ContainerRegistry=docker.io
```



# Publicar un MCP Server

Podemos configurar el MCP en VS Code u otras herramientas de esta manera:

```
{
  "inputs": [],
  "servers": {
    "mobile-dev-mcp-server": {
      "command": "docker",
      "args": [
        "run",
        "-i",
        "--rm",
        "jsuarezruiz/mobile-dev-mcp-server"
      ],
      "env": {}
    }
  }
}
```

# Beneficios de MCP

- **Conversaciones más inteligentes:** MCP hace que la IA se sienta más humana. Recuerda tu última conversación y lo que te importa. Esto se traduce en respuestas más relevantes.
- **Integración con apps:** MCP se comunica con tu calendario, correos electrónicos y herramientas de gestión de proyectos, manteniendo todo sincronizado. Se acabó el saltar de una app a otra.
- **Mejor trabajo en equipo entre agentes de IA:** ¿Tienes varios agentes de IA haciendo diferentes cosas? MCP les ayuda a compartir información y colaborar sin interferir entre sí.
- **Multitarea sin esfuerzo:** Pregunta sobre el tiempo mientras programas una reunión. No hay problema. MCP mantiene ambas tareas bajo control y retoma justo donde la dejaste.

# Combinando MCPs

Una de las mayores ventajas de MCP reside en su capacidad para encadenar múltiples servidores sin problemas, lo que permite flujos de trabajo complejos.

Supongamos que quieres que tu IA organice una quedada presencial de SevillaDotNet. Con servidores MCP encadenados, podría:

- Obtener datos del calendario Google de las comunidades (servidor MCP de Calendario).
- Usar el servidor MCP del tiempo para encontrar fechas ideales con cielos despejados.
- Consultar opciones de viaje con mejores condiciones mediante un servidor de trenes o mapas.
- Después, resumir el plan y publicarlo en un canal de Telegram.
- Se gestiona todo el proceso con un lenguaje sencillo, sin necesidad de codificación ni configuraciones complejas. Es un gran paso hacia una IA capaz de gestionar tareas complejas con mínima ayuda.

# Recursos

<https://github.com/jsuarezruiz/sevilladotnet-mcp-materials>

# Recursos

- Documentación oficial: <https://modelcontextprotocol.io/>
- Especificación de los detalles del protocolo: <https://spec.modelcontextprotocol.io/>
- GitHub: <https://github.com/modelcontextprotocol>
- MCP Servers: <https://github.com/modelcontextprotocol/servers>
- CSharp SDK: <https://github.com/modelcontextprotocol/csharp-sdk>

Preguntas & Respuestas

¿Preguntas?

P & R

**¡Gracias a todos!**