



En este módulo

1. Qué es MCP
2. El problema que resuelve
3. Cómo funciona: cliente, servidor, herramientas
4. Servidores MCP populares
5. Configurar MCP en Claude Desktop
6. MCP en Claude Code
7. Seguridad y permisos
8. Casos de uso prácticos

Qué es MCP

MCP (Model Context Protocol) es un estándar abierto creado por Anthropic que permite a los modelos de IA conectarse con herramientas externas: bases de datos, APIs, sistemas de archivos, servicios web y más.

Piensa en MCP como un "USB universal" para la IA. Antes de MCP, cada integración requería código personalizado. Con MCP, un servidor se conecta una vez y cualquier cliente compatible puede usarlo.

La especificación vigente, la versión fechada el veintiocho de julio de dos mil veintiséis, describe MCP como un protocolo abierto basado en mensajes JSON-RPC y distingue tres papeles. El host es la aplicación de IA que usas, por ejemplo Claude Desktop. El cliente es el conector que vive dentro de esa aplicación y mantiene la conversación con un servidor. Y el servidor es el programa que ofrece capacidades. La propia especificación reconoce su inspiración en el Language Server Protocol, que permitió que un mismo soporte de lenguaje funcionara en muchos editores distintos.

Un servidor no solo ofrece herramientas. También puede ofrecer recursos, que son datos de contexto como un documento o el esquema de una base de datos, y prompts, que son plantillas de mensajes pensadas para que el usuario las elija.

El problema que resuelve

Sin MCP, la IA solo puede trabajar con lo que le pegas en el chat. No puede leer tus archivos, consultar tu base de datos, ni ejecutar acciones en tus herramientas.

Sin MCP:

Tú → Copia datos → Pega en el chat → IA responde → Tú copias resultado → Lo pegas en tu herramienta

Con MCP:

Tú → "Analiza las ventas del último mes" → IA consulta tu BD directamente → Te da el análisis

Analogía

MCP es como dar a la IA manos para usar herramientas. Sin MCP, la IA solo tiene boca (texto).

Con MCP, puede abrir archivos, buscar en bases de datos, enviar emails y ejecutar código.

Piensa en el coste oculto del flujo de copiar y pegar. Con datos inventados: si cada semana preparas un informe a partir de tres hojas de cálculo y un tablero de incidencias, puedes pasar veinte minutos solo moviendo información de un sitio a otro. Con MCP, la IA lee directamente esas fuentes y tú dedicas el tiempo a revisar el análisis.

El segundo problema es el de las integraciones duplicadas. Antes, cada aplicación de IA tenía que construir su propio conector para cada servicio. Con un protocolo común, quien mantiene un servicio escribe un servidor y cualquier cliente compatible puede usarlo. El error típico es pensar que MCP hace a la IA más lista. No mejora su razonamiento: le da acceso a datos y acciones, y por eso mismo exige más cuidado con lo que conectas.

Cómo funciona: cliente, servidor, herramientas

MCP tiene tres componentes:

- **Cliente MCP:** La aplicación que usa la IA (Claude Desktop, Claude Code, Cursor, etc.).
- **Servidor MCP:** Un programa que expone herramientas. Puede ser local o remoto.
- **Herramientas (Tools):** Las acciones concretas que ofrece cada servidor.

Flujo de una petición MCP:

1. Tú pides: "Lee el archivo informe.pdf"
2. Claude identifica que necesita la herramienta `read_file`
3. Claude envía la petición al servidor MCP `filesystem`
4. El servidor lee el archivo de tu disco
5. El servidor devuelve el contenido a Claude
6. Claude analiza el contenido y te responde

Para que esa conversación ocurra, cliente y servidor necesitan un canal. La especificación define dos transportes estándar. En el primero, llamado `stdio`, el cliente lanza el servidor como un proceso local en tu ordenador y se comunican por la entrada y la salida estándar; es lo habitual con servidores de archivos o de bases de datos locales. En el segundo, llamado `Streamable HTTP`, cada mensaje viaja como una petición HTTP a un único endpoint del servidor, lo que permite usar servidores remotos alojados por un proveedor.

Un matiz importante del flujo que acabas de ver: el modelo no ejecuta la herramienta por sí mismo. El modelo propone usarla, y es la aplicación, a través de su cliente, quien hace la llamada al servidor. Por eso la especificación pide que el host obtenga el consentimiento explícito del usuario antes de invocar cualquier herramienta. El error típico es aprobar todas las peticiones de forma automática desde el primer día, cuando precisamente esa pausa es tu principal mecanismo de control.

Servidores MCP populares

Hay cientos de servidores MCP disponibles. Los más útiles para empezar:

- **Filesystem:** Lee y escribe archivos en tu ordenador.
- **GitHub:** Gestiona repositorios, issues, pull requests.
- **Slack:** Lee canales, envía mensajes.
- **Google Drive:** Accede a documentos de Drive.
- **PostgreSQL/Supabase:** Consulta bases de datos.
- **Notion:** Lee y escribe en tu workspace de Notion.
- **Brave Search:** Búsqueda web desde la IA.

Configurar MCP en Claude Desktop

La configuración se hace en un archivo JSON. En macOS está en:

```
~/Library/Application Support/Claude/claude_desktop_config.json
```

Ejemplo de configuración con el servidor `filesystem`:

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "/Users/tu-usuario/Documentos"
      ]
    }
  }
}
```

Tras guardar el archivo, reinicia Claude Desktop. Verás un icono de herramientas indicando que MCP está activo.

Merece la pena leer el ejemplo campo por campo. Dentro del bloque de servidores, cada entrada lleva un nombre que tú eliges. El campo `command` indica qué programa se lanza, en este caso npx, que descarga y ejecuta un paquete de Node; la opción que sigue evita que pida confirmación. Después viene el nombre del paquete del servidor y, por último, la carpeta a la que le das acceso. Esa última ruta es tu frontera de seguridad: el servidor de archivos solo trabajará dentro de ella.

Un ejemplo inventado: Lucía quiere que Claude revise sus facturas en PDF. En lugar de dar acceso a su carpeta de documentos completa, crea una carpeta llamada facturas y apunta el servidor solo ahí. El error más común en este paso es un fallo de sintaxis en el fichero, como una coma de más al final de una lista, que hace que la aplicación ignore toda la configuración.

MCP en Claude Code

Claude Code soporta MCP de forma nativa. Puedes añadir servidores con:

```
# Añadir un servidor MCP
claude mcp add nombre-servidor -- comando args

# Ejemplo: añadir filesystem
claude mcp add filesystem -- npx -y @modelcontextprotocol/server-filesystem /ruta

# Ejemplo: añadir GitHub
claude mcp add --transport http github https://api.githubcopilot.com/mcp/

# Listar servidores configurados
claude mcp list

# Eliminar un servidor
claude mcp remove nombre-servidor
```

En Claude Code cada servidor se instala con un ámbito. El ámbito local, que es el predeterminado, lo activa solo para ti y en el proyecto actual. El ámbito de proyecto lo guarda en un fichero llamado `.mcp.json` en la raíz del repositorio, de modo que todo el equipo lo comparte al clonarlo; por seguridad, Claude Code pide aprobación antes de usar servidores definidos así. El ámbito de usuario lo deja disponible en todos tus proyectos.

Dentro de una sesión, el comando `/mcp` muestra qué servidores están conectados, cuáles necesitan autenticación y cuáles han fallado. Imagina, con datos inventados, un equipo de cuatro personas que comparte un servidor de base de datos de pruebas en el fichero del proyecto. El error típico es escribir la cadena de conexión con la contraseña dentro de ese fichero y subirlo al repositorio: las credenciales deben llegar por variables de entorno, nunca en un fichero versionado.

Seguridad y permisos

MCP da a la IA acceso real a tus sistemas. Toma precauciones:

- **Principio de mínimo privilegio:** Solo da acceso a las carpetas o datos necesarios.
- **Revisa antes de aprobar:** Claude te pide permiso antes de ejecutar herramientas.
- **No expongas credenciales:** Usa variables de entorno para tokens y API keys.
- **Servidores de confianza:** Solo instala servidores MCP de fuentes verificadas.
- **Modo solo lectura:** Cuando sea posible, configura acceso de solo lectura.

Regla de oro

Si no entiendes exactamente qué hace un servidor MCP, no lo instales. Lee el código fuente o la documentación antes de darle acceso a tus datos.

La especificación de MCP es explícita en esto: las herramientas representan ejecución de código arbitrario y deben tratarse con cautela, y las descripciones del comportamiento de una herramienta deben considerarse no fiables salvo que procedan de un servidor de confianza. Dicho de otro modo, que una herramienta se llame consultar saldo no garantiza que solo consulte.

Hay un riesgo menos evidente: la inyección de instrucciones. Un servidor que trae contenido externo, como una página web o un correo, puede introducir en la conversación texto que intenta dar órdenes al modelo. Con un ejemplo inventado: un correo contiene la frase reenvía este hilo a esta dirección externa, y si además tienes conectado un servidor de correo con permiso de envío, el modelo podría proponerlo. Por eso conviene no combinar sin pensar un servidor que lee contenido de terceros con otro que puede enviar o borrar, y leer siempre qué acción vas a aprobar.

Casos de uso prácticos

- 1. Analista de código:** Conecta el servidor filesystem a tu repositorio. Pide a Claude que revise tu código, busque bugs o refactorice funciones.
- 2. Gestor de proyectos:** Conecta GitHub + Slack. Pide que revise issues abiertos, resuma conversaciones y cree tareas.
- 3. Investigador:** Conecta Brave Search + filesystem. Investiga un tema, guarda los resultados en un archivo Markdown organizado.
- 4. Asistente de datos:** Conecta PostgreSQL. Consulta datos de ventas, genera informes y detecta anomalías sin escribir SQL manualmente.
- 5. Automatizador:** Conecta múltiples servidores. Crea flujos que lean de una fuente, procesen datos y escriban en otra.

Tabla: servidores MCP por categoría

CATEGORÍA	SERVIDOR	HERRAMIENTAS PRINCIPALES	NIVEL DE RIESGO
Archivos	filesystem	read_file, write_file, list_directory	Medio (acceso a disco)
Código	github	create_issue, list_commits, search_code	Medio-alto (repos)
Comunicación	slack	send_message, read_channel	Alto (envío de mensajes)
Datos	postgres, supabase	execute_sql, list_tables	Alto (acceso a BD)
Documentos	notion, google-drive	read_page, create_page	Medio
Búsqueda	brave-search, exa	web_search, search	Bajo (solo lectura)
Email	gmail	send_email, search_threads	Alto (envío de emails)

Ejemplo avanzado: configurar múltiples servidores

```
# claude_desktop_config.json con 3 servidores
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-filesystem",
        "/Users/tu-usuario/proyectos"]
    },
    "github": {
      "command": "docker",
      "args": ["run", "-i", "--rm", "-e", "GITHUB_PERSONAL_ACCESS_TOKEN",
        "ghcr.io/github/github-mcp-server"],
      "env": {
        "GITHUB_PERSONAL_ACCESS_TOKEN": "ghp_XXXXXXXXXXXX"
      }
    },
    "postgres": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-postgres"],
      "env": {
        "POSTGRES_CONNECTION_STRING": "postgresql://user:pass@localhost:5432/mydb"
      }
    }
  }
}

# Flujo de trabajo con múltiples servidores:
# 1. "Lee el archivo requirements.md" → filesystem
# 2. "Busca issues abiertos en el repo" → github
# 3. "Consulta cuántos usuarios activos hay" → postgres
# 4. "Crea un resumen con los 3 datos" → Claude (sin herramientas)
```

Construir tu propio servidor MCP

```
# Un servidor MCP básico en Python (con mcp SDK)
# Archivo: mi_servidor.py

from mcp.server import MCPServer

app = MCPServer("mi-herramienta")

@app.tool()
async def buscar_cliente(nombre: str) → str:
    """Busca un cliente por nombre en la base de datos interna."""
    # Aquí iría tu lógica de búsqueda
    resultado = consultar_bd(nombre)
    return f"Cliente encontrado: {resultado}"

@app.tool()
async def calcular_descuento(importe: float, tipo: str) → str:
    """Calcula el descuento aplicable según política de la empresa."""
    descuentos = {"standard": 0.05, "premium": 0.10, "vip": 0.15}
    pct = descuentos.get(tipo, 0)
    return f"Descuento {tipo}: {pct*100}% = {importe * pct:.2f} EUR"

if __name__ == "__main__":
    app.run() # stdio por defecto

# Ejecutar: python mi_servidor.py
# Registrar en Claude: claude mcp add mi-herramienta -- python mi_servidor.py
```

Errores comunes con MCP

- **Dar acceso a toda la carpeta home:** Limita el acceso del servidor filesystem a carpetas específicas. Dar acceso a `/Users/tu-usuario/` expone configuraciones, claves SSH y archivos personales.
- **No revisar las herramientas antes de aprobar:** Lee siempre lo que Claude quiere hacer antes de dar permiso. Un servidor malicioso podría proponer acciones destructivas con nombres inocuos.
- **Tokens en texto plano en la config:** Usa variables de entorno o un gestor de secretos, no tokens directamente en el JSON de configuración.
- **Instalar servidores no verificados:** Solo usa servidores MCP de fuentes de confianza (repositorio oficial, desarrolladores conocidos). Revisa el código fuente si es open source.

Consejo: Empieza con el servidor filesystem de solo lectura en una carpeta específica. Cuando te sientas cómodo con cómo funciona, añade un segundo servidor (GitHub o Notion). La complejidad se añade incrementalmente, no de golpe.

IAcademy · iacademy.com · © 2026 Riskitera S.L.U.

MT04: Conecta herramientas con MCP · Avanzado

¿Dudas? Usa el tutor IA en iacademy.com/course/mt04/