

***AraCom***



# Model Context Protocol

**Tools einfach intelligent?**

Robert Jeutter

12.06.2025

***AraCom***

# Wir sind AraCom

## We release a better version of the world – EVERYDAY.

Wir geben uns nicht mit dem Standard zufrieden — Das sollten Sie auch nicht. Unser Team bestehend aus 250 hochambitionierten IT-Experten zeigt Ihnen, wie mit technischem Know-How und viel Herzblut ihr IT-Projekt erfolgreich umgesetzt wird. Gemeinsam entwickeln wir innovative Software für ihren entscheidenden Vorsprung seit 1998!



**Gersthofen**  
München  
Stuttgart  
Bamberg



250  
**IT-Experten**



**Regional**  
IT Made in  
Germany



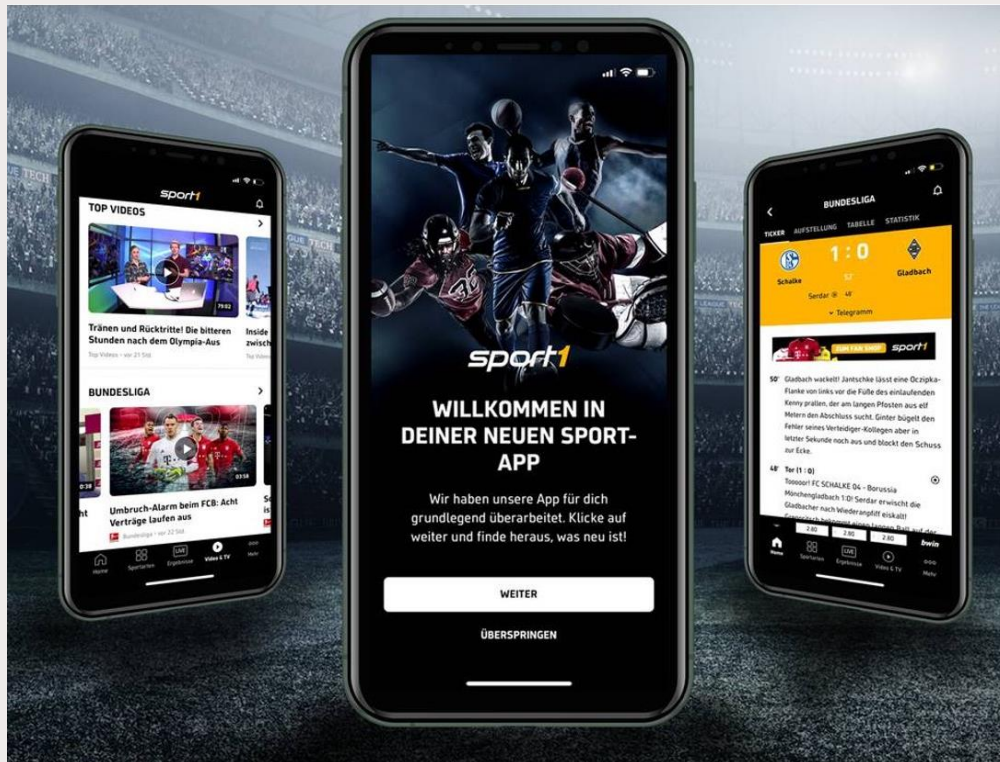
**Erfahrung**  
in allen  
Branchen



Entwicklung in allen  
aktuellen &  
zukunftssträchtigen  
**Technologien**

# Sport1 App

Webseite und App mit Livesportereignissen in Microservice-Infrastruktur



Beliebt:CHECK24 DoppelpassHandballVolleyballBBLNewsletterDAZN

sport1

HOMEFUSSBALLMOTORSPORTEISHOCKEYUS-SPORTDARTSMEHR

📅

LIVE

📺

TV

▶

VIDEOS

🛒

SHOP

Aktuelles Programm

Sie sind hier: Home > Live

HEUTE LIVE

<

Mo14. Okt.

Di15. Okt.

Mi16. Okt.

Heute17. Okt.

Fr18. Okt.

Sa19. Okt.

So20. Okt.

Mo21. Okt.

Di22. Okt.

Mi23. Okt.

Do24. Okt.

Fr25. Okt.

Sa26. Okt.

So27. Okt.

>

<

Fußball

Motorsport

American Football

Eishockey

Basketball


Volleyball

Tennis



Baseball

Rad

>

 **Bundesliga**  
8. Spieltag


Eintracht Frankfurt

 20:30 Uhr 



Bayer 04 Leverkusen

>

[Spielplan](#) > [Tabelle](#) >

 **2. Bundesliga**  
10. Spieltag



SpVgg Greuther Fürth

 18:30 Uhr 

SG Dynamo Dresden

>


FC Erzgebirge Aue

 18:30 Uhr 



1. FC Nürnberg

>

[Spielplan](#) > [Tabelle](#) >

 **3. Liga**  
12. Spieltag

MSV Duisburg

 19:00 Uhr 

1. FC Kaiserslautern

>

[Spielplan](#) > [Tabelle](#) >

# Model Context Protocol

## Tools einfach intelligent?

---

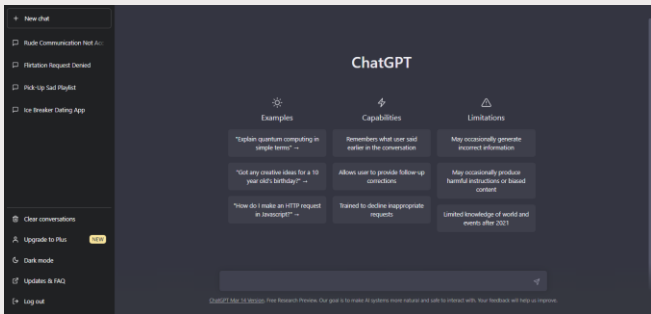
Robert Jeutter

12.06.2025

# Die bisherigen LLM Werkzeuge

## ChatGPT/Claude/...

Chat-Interaktion im Vordergrund  
Wenige/Eingeschränkte Tools



chatgpt.com

## LLM API

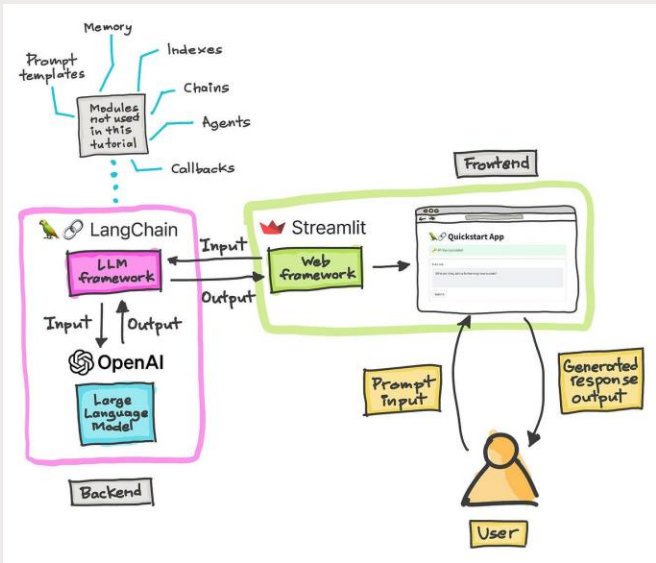
Hohe API-Kosten bei externen Aufrufen

		most advanced	most powerful	powerful	fast	fastest
tokens	~words	gpt-3.5-turbo	davinci price	curie price	babbage price	ada price
1,000	750	\$0.002	\$0.02	\$0.002	\$0.0005	\$0.0004
2,500	1,875	\$0.005	\$0.05	\$0.005	\$0.0013	\$0.0010
5,000	3,750	\$0.010	\$0.10	\$0.010	\$0.0025	\$0.0020
10,000	7,500	\$0.020	\$0.20	\$0.020	\$0.0050	\$0.0040
25,000	18,750	\$0.050	\$0.50	\$0.050	\$0.0125	\$0.0100
50,000	37,500	\$0.100	\$1.00	\$0.100	\$0.0250	\$0.0200
100,000	75,000	\$0.200	\$2.00	\$0.200	\$0.0500	\$0.0400
250,000	187,500	\$0.500	\$5.00	\$0.500	\$0.1250	\$0.1000
500,000	375,000	\$1.000	\$10.00	\$1.000	\$0.2500	\$0.2000
1,000,000	750,000	\$2.000	\$20.00	\$2.000	\$0.5000	\$0.4000
1,250,000	937,500	\$2.500	\$25.00	\$2.500	\$0.6250	\$0.5000
1,500,000	1,125,000	\$3.000	\$30.00	\$3.000	\$0.7500	\$0.6000
1,750,000	1,312,500	\$3.500	\$35.00	\$3.500	\$0.8750	\$0.7000
2,000,000	1,500,000	\$4.000	\$40.00	\$4.000	\$1.0000	\$0.8000
10,000,000	7,500,000	\$20.000	\$200.00	\$20.000	\$5.0000	\$4.0000
		ChatGPT Plus				

tech-mags.com

## LLM Integration

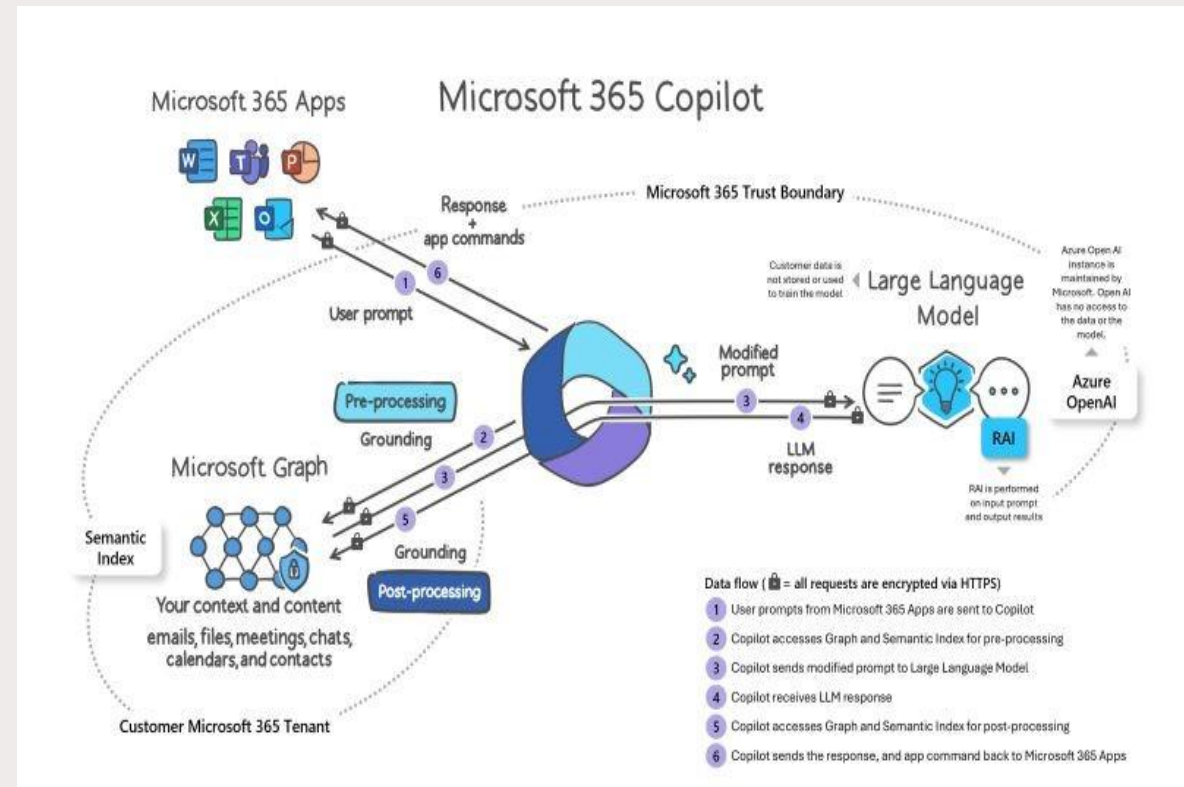
Umständliche Integration eigener Services/Dienste/Daten



miro.Medium.com

# Beispiel Microsoft Copilot

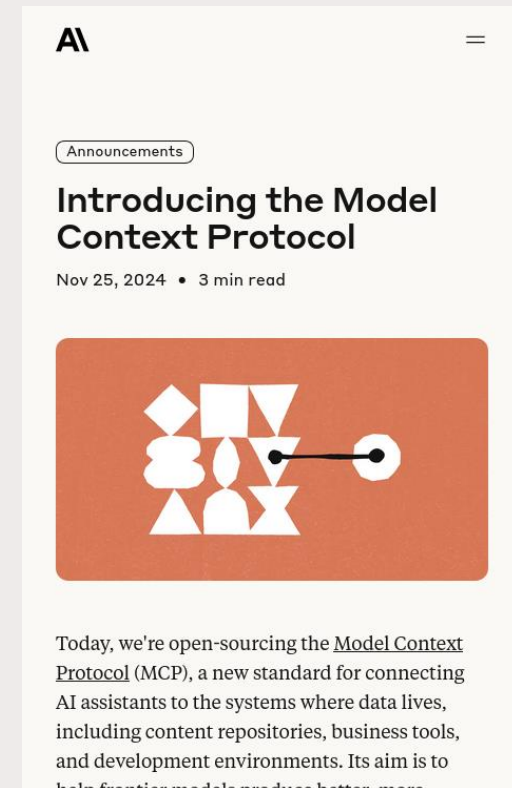
- Verfügbar in Office, Windows, GitHub, ...
- Integration oft monolithisch und proprietär
- Hoher Entwicklungsaufwand
- für IT-Riesen vorbehalten



Richard Bowes @ Intapp

# Anthropic stellt MCP vor (Nov 2024)

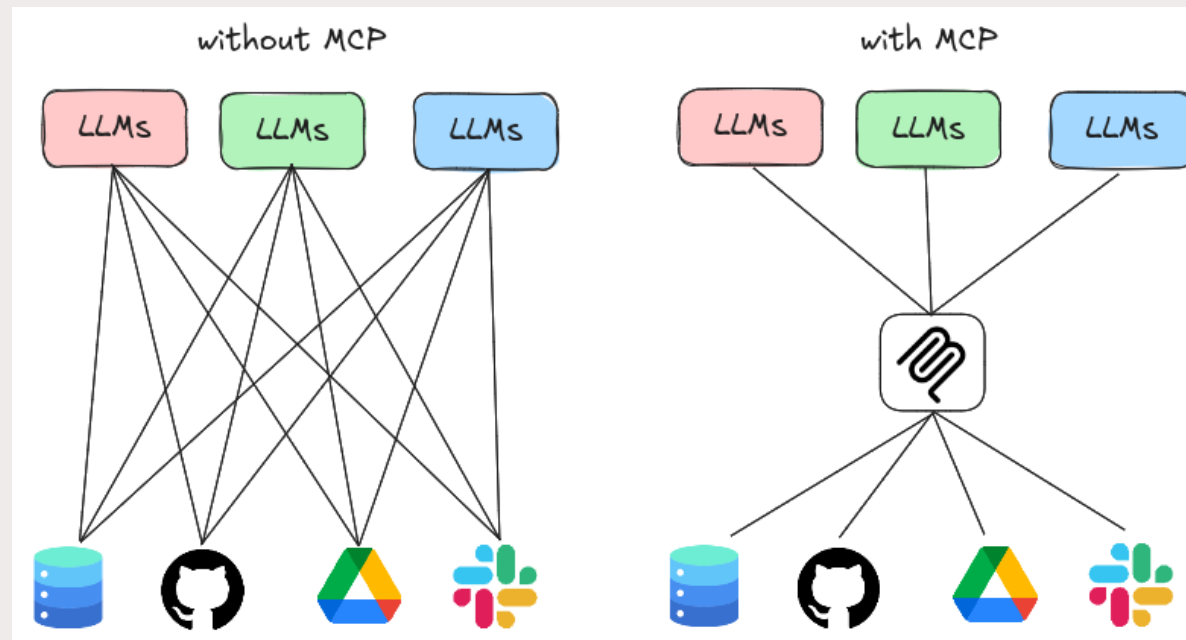
- Offenes Protokoll zur Tool-/Datenanbindung für LLMs
- Vision: viele Tools über eine einheitliche Schnittstelle
  - "... wie USB-C nur für Daten an LLMs statt Geräte an Rechner"
- Standardisierung der Tool-/Daten-Integration
- Anbieterunabhängig (Claude, lokale LLMs, OpenAI, ...)
- Schutz von Daten und Kontrolle durch Nutzer



anthropic.com

# MCP als LLM Hub

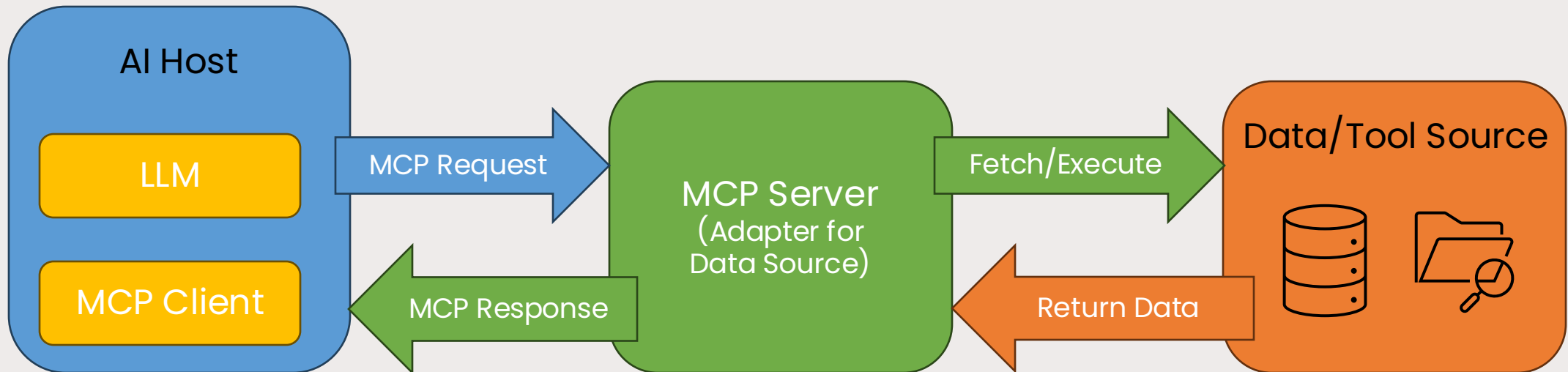
- MCP als Hub: Tool einmalig implementieren, viele konsumieren
- $M \times N$  vs.  $M + N$  Integrationen



Hoang Dinh Nguyen



# MCP-Architektur im Überblick



# Rollen: Host, Client, Server



## Host

User Interface &  
LLM-Anwendung

Beispiele

- Eigenes LLM (Ollama)
- Claude Desktop
- ...

## Client

- Vermittlungsmodul
- hält 1:1 Verbindung zu Server
- Abstraktion für JSON-RPC-Kommunikation

## Server

Liefert Tools, Resources,  
Prompts

- Exekutierbare Anwendung
- Kontext
- weitere Funktionalität

Ressourcen können Lokal  
und Remote sein

# MCP-Nachrichtenformat (JSON-RPC 2.0)

## Request

```
{
  "jsonrpc": "2.0",
  "id": X,
  "method": "...",
  "params": {...}
}
```

## Response

```
{
  "jsonrpc": "2.0",
  "id": X,
  "result": {...}
  ODER
  "error": {code, message}
}
```

## Notification

```
{
  "jsonrpc": "2.0",
  "method": "...",
  "params": {...}
}
```

- Die ID verknüpft Request und Response
- Notifications brauchen keine ID, werden als Einbahnkommunikation genutzt

# Transport-Schicht: stdio vs. HTTP-SSE



## Stdio

- Kommunikation über Standard Input/Output (lokaler Prozess)
- Einfach, kein Netzwerk-Stack nötig
- der Host startet den Server-Prozess lokal

## HTTP + SSE

- Server-Sent Events (Server-Client), HTTP POST (Client-Server)
- Geeignet für entfernte MCP-Server
- der Server bindet an eine URL, der Client stellt eine SSE-Verbindung her
- Heartbeat und Reconnect-Strategien notwendig
- Sicherheit: SSE über HTTPS, CORS-Check, Origin-Validierung

# Konfiguration

- Die Konfiguration steuert, welche MCP-Server der Host beim Start lädt
- Tipp: Nur notwendige Server eintragen, um LLM-Entscheidungen zu fokussieren

```
/* Example config.json */
```

```
{
```

```
  "mcpServers": {
```

```
    "sqlite": {
```

```
      "command": "sqlite3",
```

```
      "args": [
```

```
    ]
```

```
  }
```

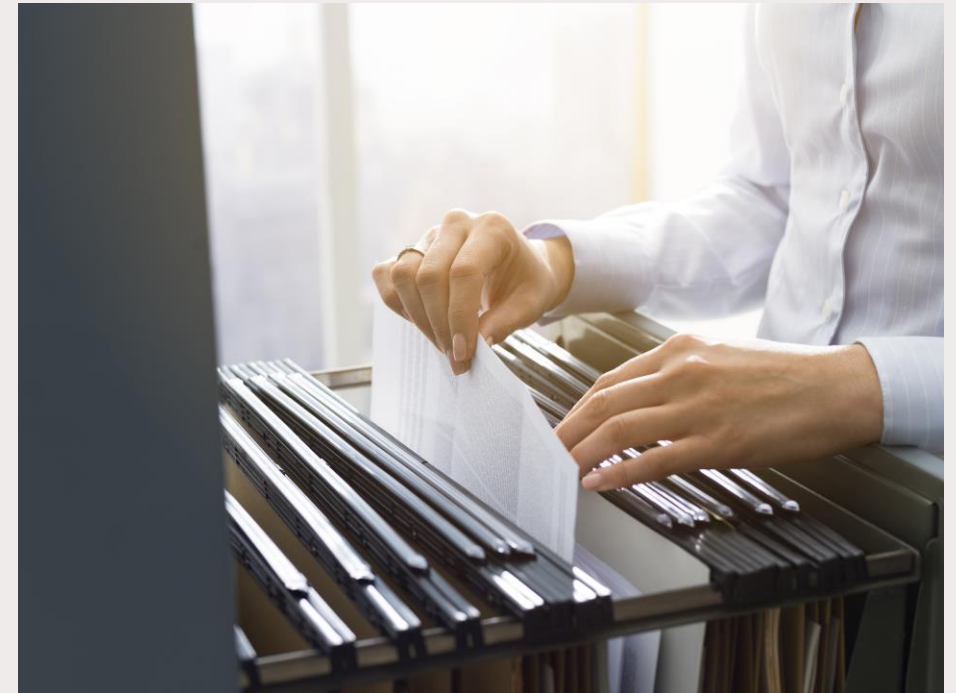
```
}
```

Argument-Liste für den Server-Start

ausführbares Programm

# Komponente: Roots (URIs)

- Client definiert Fokusbereiche als URI-Liste
  - o `file:///home/user/projekt/`
  - o <https://api.example.com/v1>
- Beispiel: Filesystem-Server operiert nur im Projektverzeichnis, nicht im gesamten Home-Verzeichnis
- Server kann Roots ignorieren!
- Nutzen: Verhindert ungewollte Zugriffe auf andere Pfade



# Komponente: Resources (Datenkontext)



- **Text**-Ressourcen: UTF-8 kodierte Texte (z.B. Quelldateien, Logs)
- **Binary**-Ressourcen: Base64-kodierte Daten (z.B. Bilder, PDFs)
- API:
  - `<resources/list>` → Metadaten aller verfügbaren Ressourcen
  - `<resources/read>` → Inhalte per URI abfragen
  - `<resources/subscribe>` → Notifications bei Änderung
- Hinweis: Ressourcen müssen vom Nutzer ausgewählt werden, damit LLM sie sehen kann (Privacy)

# Komponente: Prompts (Workflows)



Wiederverwendbare Prompt-Vorlagen mit Argumenten

Discovery: <**prompts/list**>

```
"prompts": [{ "name": "analyze-code", "description": "Code analysieren",  
              "arguments": [{"name": "language", "required": true}] } ]
```

Verwendung: <**prompts/get**> mit Argumenten (Parameter), Server liefert fertige Nachrichten:

```
{ "messages": [ { "role": "user", "content": { "type": "text", "text": "Analysiere folgenden Code: ..." } } ] }
```

Vorteil: Konsistente Interaktionen, weniger Tippaufwand



# Komponente: Tools (Funktionen)

Tools führen Aktionen aus

Definition:

**name:** eindeutiger Bezeichner

**description:** Kurzbeschreibung

**inputSchema:** JSON-Schema für Parameter

**annotations:** Hinweise

Discovery: <tools/**list**>

Aufruf: <tools/**call**> mit Parameter-Objekt

Beispiel-Tool-Definition "calculate\_sum":

```
{
  "name": "calculate_sum",
  "description": "Addiere zwei Zahlen",
  "inputSchema": {
    "type": "object",
    "properties": {
      "a": { "type": "number" },
      "b": { "type": "number" }
    },
    "required": [ "a", "b" ]
  },
  "annotations": { ... }
}
```

# Komponente: Tools (Funktionen)



Annotationen:

**readOnlyHint** → kein Datenbank-Update

**destructiveHint** → Vorsicht! löscht/ändert Daten

**openWorldHint** → kommuniziert extern (z.B. Web-API)

**idempotentHint** → wiederholbar ohne Seiteneffekt

Beispiel-Tool-Call "calculate\_sum":

```
{
  "jsonrpc": "2.0",
  "id": 5,
  "method": "tools/call",
  "params": {
    "name": "calculate_sum",
    "arguments": {"a": 5, "b": 7}
  }
}
```

# Sicherheitsaspekte bei Tools



- **Input-Validierung:** JSON-Schema prüfen, Sanitizing (Pfad- und SQL-Injection verhindern)
- **Zugriffsrechte:** Sandbox, minimale Privilegien (z.B. nur Lesezugriff auf notwendige Pfade)
- **Output-Validation:** Tool-Ausgaben prüfen, damit kein böartiger Prompt Injection stattfindet
- **User-Bestätigung:** Vor Ausführen kritischer Tools (z.B. Lösch-Operationen) explizit bestätigen lassen

# Komponente: Sampling (Server-initiierte LLM-Abfragen)



- Server sendet **«sampling/createMessage»** an Client
- Client validiert Prompt (Nutzer-Review), führt LLM-Sampling durch
- Client sendet generierte Completion per **«sampling/return»** zurück
- Erlaubt Server, LLM als Subroutine zu nutzen (z.B. Zusammenfassung, Klassifikation)

# Zusammenfassung Architektur

## Client-Server Rollen

**Transportschicht:** stdio / HTTP-SSE

Konzepte:

**Roots:** Fokus-URIs

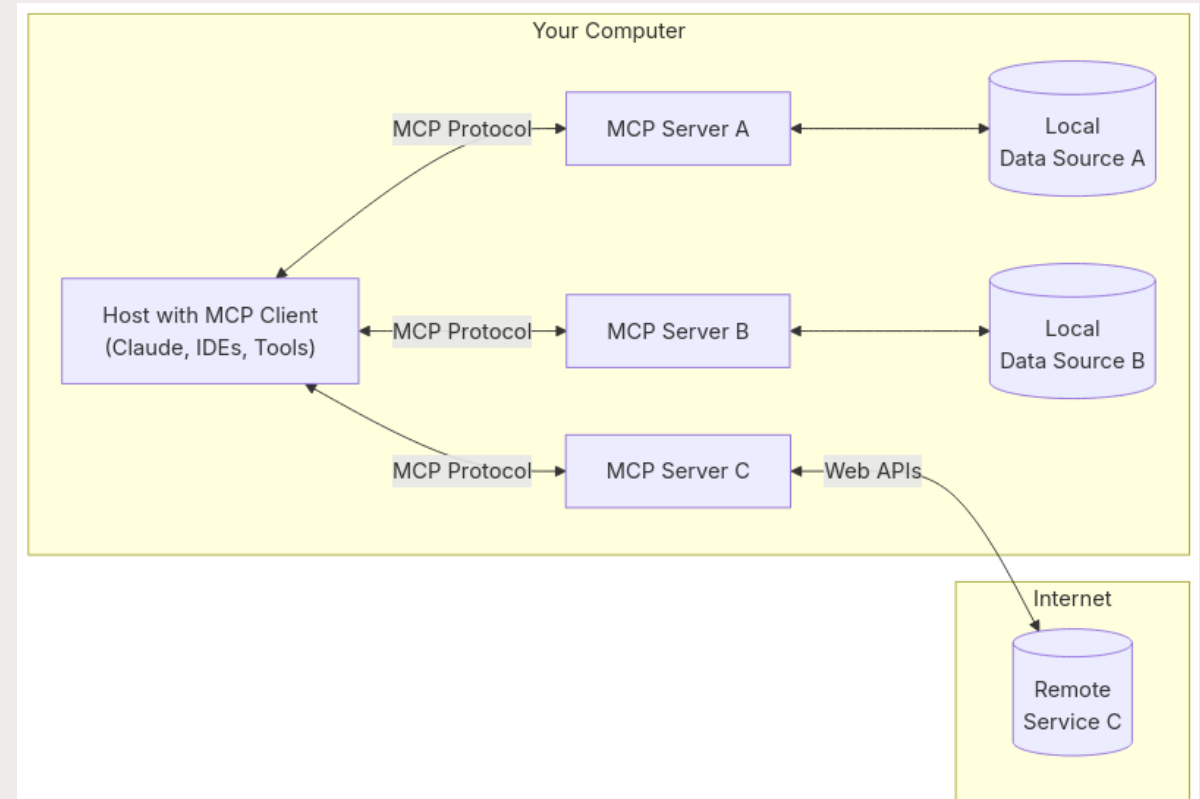
**Resources:** Lesbare Datensätze

**Prompts:** Vordefinierte Templates

**Tools:** Exekutive Funktionen

**Sampling:** Server-initiierte LLM-Nutzung

**TLDR:** MCP entkoppelt LLM und Tools/Daten

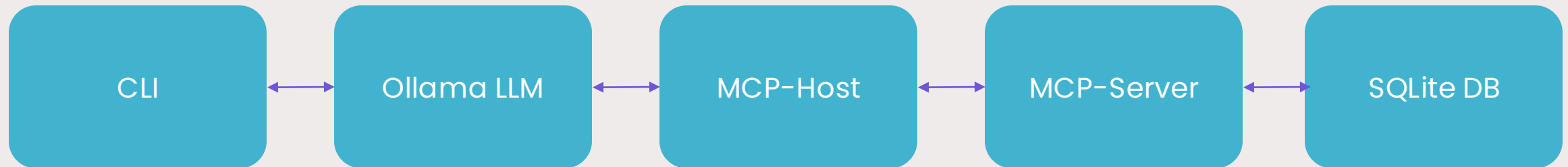


modelcontextprotocol.io


# Beispiel-Szenario

**Ziel:** Lokales LLM nutzt MCP, um SQL-Datenbank zu befragen

1. SQLite-Datenbank mit Car\_Database.db
2. Ollama stellt LLM (z.B. Qwen-2.5) lokal bereit
3. SQLite-MCP-Server akzeptiert SQL-Queries und liefert Ergebnisse als JSON/Text
4. Bedienung über CLI Tool



## Beispiel: <config.json>



```
{
  "mcpServers": {
    "sqlite": {
      "command": "uvx",
      "args": [ "mcp-server-sqlite", "--db-path", "/pfad/Car_Database.db" ]
    }
  }
}
```

**Tipp:** Nur das notwendige Minimum konfigurieren, damit LLM nicht zu viele Tools sieht.

# Beispiel: SQLite-MCP-Server



Pseudocode des SQLite-Servers

```
server.setRequestHandler(ListToolsRequestSchema, async () => {  
  return {  
    tools: [{  
      name: "execute_sql",  
      description: "SQL-Query ausführen",  
      inputSchema: {  
        type: "object",  
        properties: { sql: { type: "string" } },  
        required: ["sql"]  
      }  
    }  
  ]  
}
```

...



# Beispiel: SQLite-MCP-Server



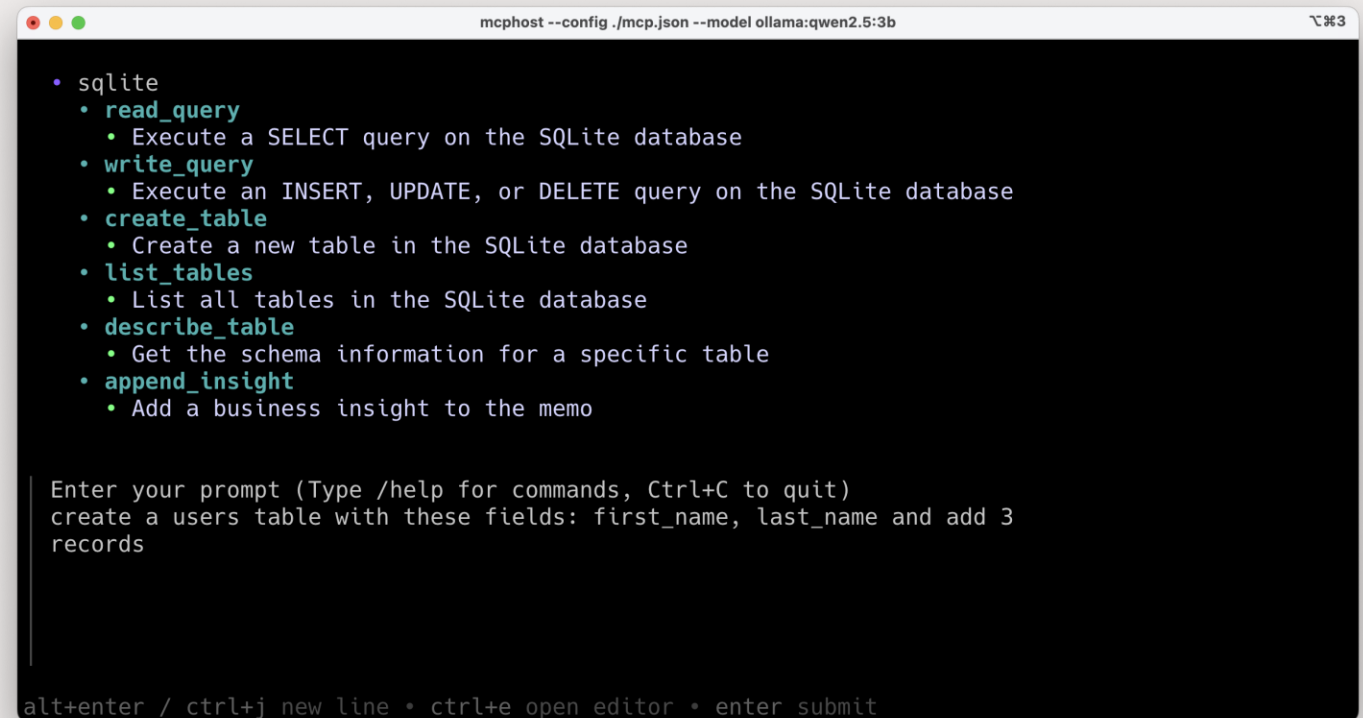
Pseudocode des SQLite-Servers

...

```
server.setRequestHandler(CallToolRequestSchema, async (request) => {  
  if (request.params.name === "execute\_sql") {  
    const { sql } = request.params.arguments;  
    const result = await db.execute(sql);  
    return {  
      content: [{ type: "text", text: formatTable(result) }]  
    };  
  }  
  throw new Error("Tool nicht gefunden");  
});
```

# Beispiel: MCP Host

mcphost \  
---model ollama:qwen2.5:3b \  
--config ./config.json \  
--system-prompt ./systemprompt.json



```
mcphost --config ./mcp.json --model ollama:qwen2.5:3b

• sqlite
  • read_query
    • Execute a SELECT query on the SQLite database
  • write_query
    • Execute an INSERT, UPDATE, or DELETE query on the SQLite database
  • create_table
    • Create a new table in the SQLite database
  • list_tables
    • List all tables in the SQLite database
  • describe_table
    • Get the schema information for a specific table
  • append_insight
    • Add a business insight to the memo

Enter your prompt (Type /help for commands, Ctrl+C to quit)
create a users table with these fields: first_name, last_name and add 3
records

alt+enter / ctrl+j new line • ctrl+e open editor • enter submit
```

[github.com/mark3labs/mcphost](https://github.com/mark3labs/mcphost)

# Beispiel: Interaction Flow



1. Nutzer-Prompt: „Wie viele Einträge aus Baujahr 2015?“
2. LLM generiert Tool-Call

```
... "method": "tools/call",  
  "params": {  
    "name": "execute_sql",  
    "arguments": { "sql": "SELECT COUNT(*) FROM Cars WHERE year=2015" }  
  }
```
3. Client leitet Request an MCP-Server weiter.
4. Server führt Query auf DB aus, liefert z.B.: `{"content": [{"type": "text", "text": "42"}]}`
5. LLM baut finale Antwort: „42 Autos aus 2015.“

# Vorteile von MCP



**Standardisierung:** Einheitliche Schnittstelle zu vielen Tools.

**Datenkontrolle:** Client/Nutzer behält Kontrolle über Zugriff.

**Flexibilität:** Wechsel zwischen LLM-Anbietern (Claude, Ollama, OpenAI).

**Skalierbarkeit:**  $M+N$  Integrationen statt  $M \times N$ .

**Community-Ecosystem:** Schnell wachsende Zahl an MCP-Servern.

# Nachteile & Risiken



**Security-Lücken:** Keine eingebaute Authentifizierung, potenziell unsichere Tools.

**Komplexität:** Einrichtung (config, Host, mehrere Prozesse).

**Reifegrad:** Spezifikation jung, Breaking Changes möglich.

**Modell-Limitationen:** Kontextfenster, Multi-Step-Logik, Prompt-Injection-Risiken.

**Host-Abhängigkeit:** Noch wenige Hosts (Claude, Ollama, Cursor, ...).

# Sicherheitsempfehlungen



**TLS/HTTPS:** für SSE-Transports.

**Auth & Access Control:** eigene Layer (API-Keys, JWT, OAuth).

**Sandboxing:** Tools isoliert ausführen (Container, minimaler User).

**Output-Validation:** Tool-Antworten prüfen und filtern.

**Logging & Auditing:** Alle MCP-Interaktionen protokollieren.

# Zukunftsansichten



standardisiertes  
Logging/Metrik-  
API

Batches &  
Streaming  
(HTTP/2, gRPC)

weitere LLM-  
Anbieter (OpenAI  
GPT-4, Claude 4, ...)

Auth-Spec &  
Fine-Grained  
Access Control

und vieles mehr...

# Takeaways MCP



1. Schafft klare Trennung: LLM – Tools/Daten – Ausführung.
2. Ermöglicht Entwickler-Workflows, bei denen KI direkt in Systemlandschaften eingebunden ist.
3. Aktuell frühe Phase, Sorgfalt bei Sicherheit und Setup erforderlich.
4. Große Chancen für vielseitige KI-Assistenz-Szenarien.



# Quellen & weiterführend



[Modelcontextprotocol.io](https://modelcontextprotocol.io)

[Anthropic.com](https://anthropic.com)

[Github.com/mark3labs/mcp-host](https://github.com/mark3labs/mcp-host)

[Aracom.de](https://aracom.de)

A grayscale photograph of a modern, multi-story office building with a parking lot in the foreground. The building has a large 'A' logo on its facade and 'AraCom' signage on the upper right. The parking lot has several empty spaces marked with white lines. The overall scene is slightly blurred, giving it a professional yet approachable feel.

**Zeit für eure Fragen**

A small, horizontal, rounded rectangular logo with a blue-to-purple gradient.

***AraCom***