Karlsruher Institut für Technologie

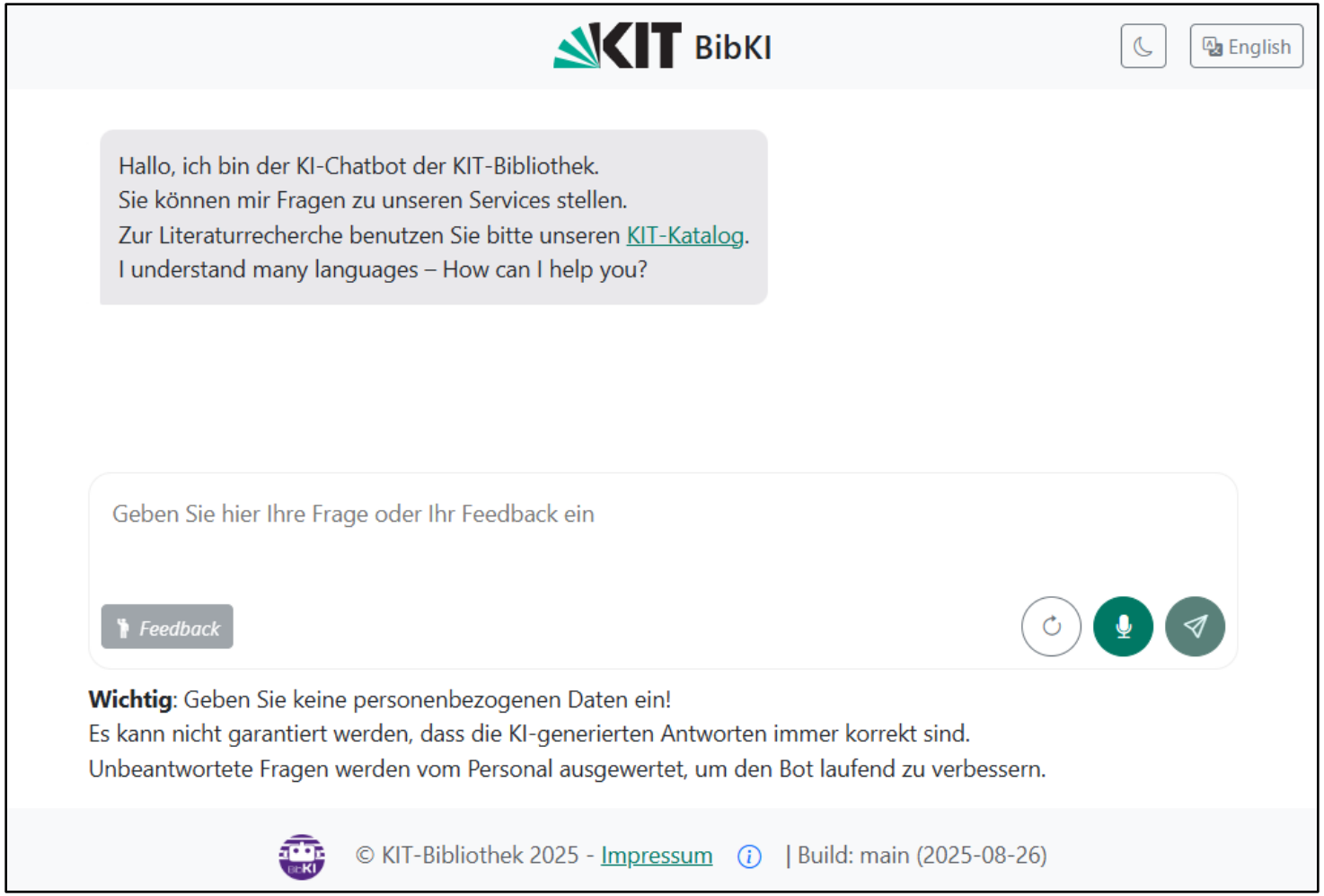
KI-Austausch Tool Calls & MCP in KI-Chatbots

Uwe Dierolf

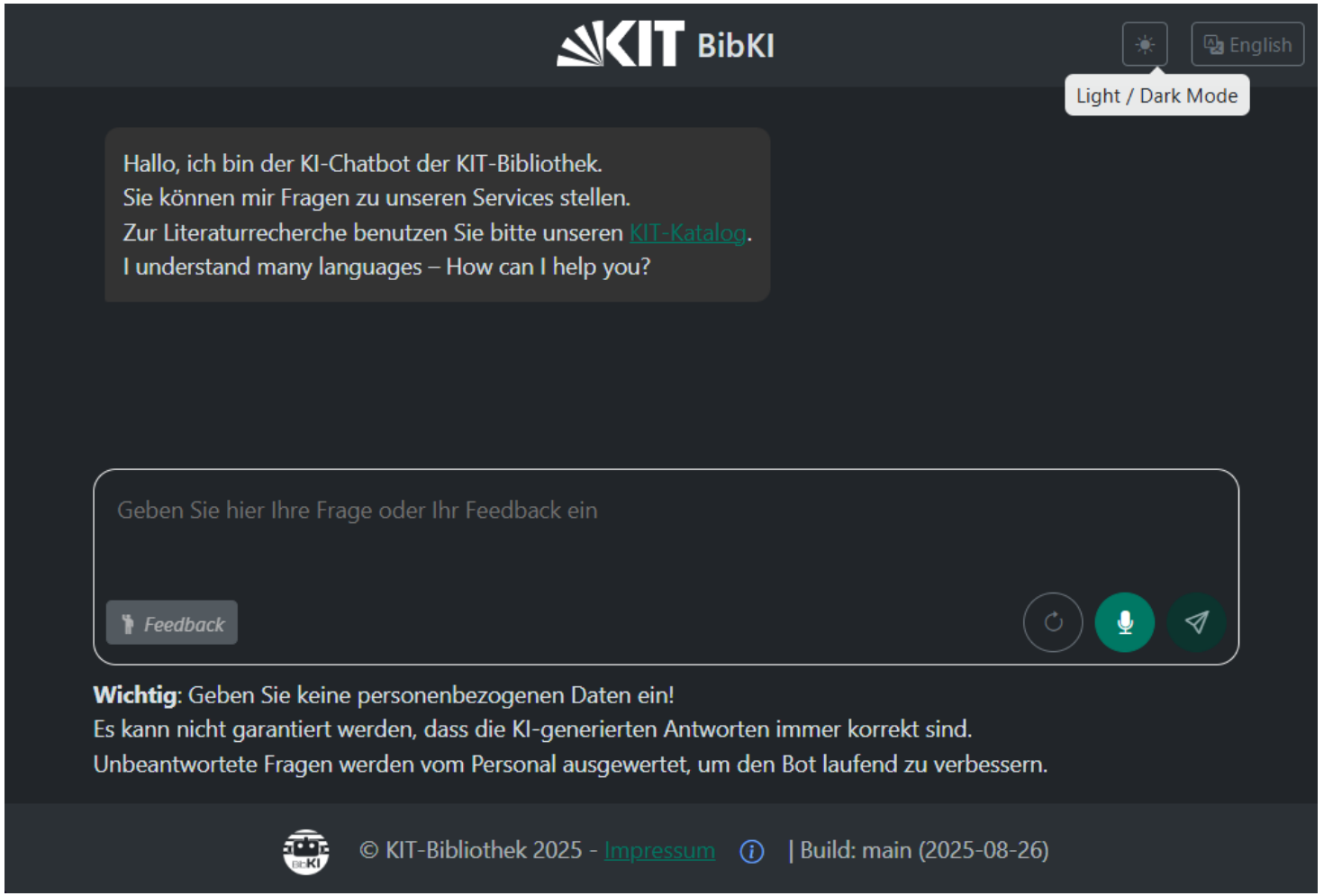
Übersicht

- Motivation
- BibKI und RAG – kurze Vorstellung
- Wozu noch Tool Calls? Genügt RAG nicht?
- Tool Calls in BibKI dem KI-Service-Chatbot der KIT-Bibliothek

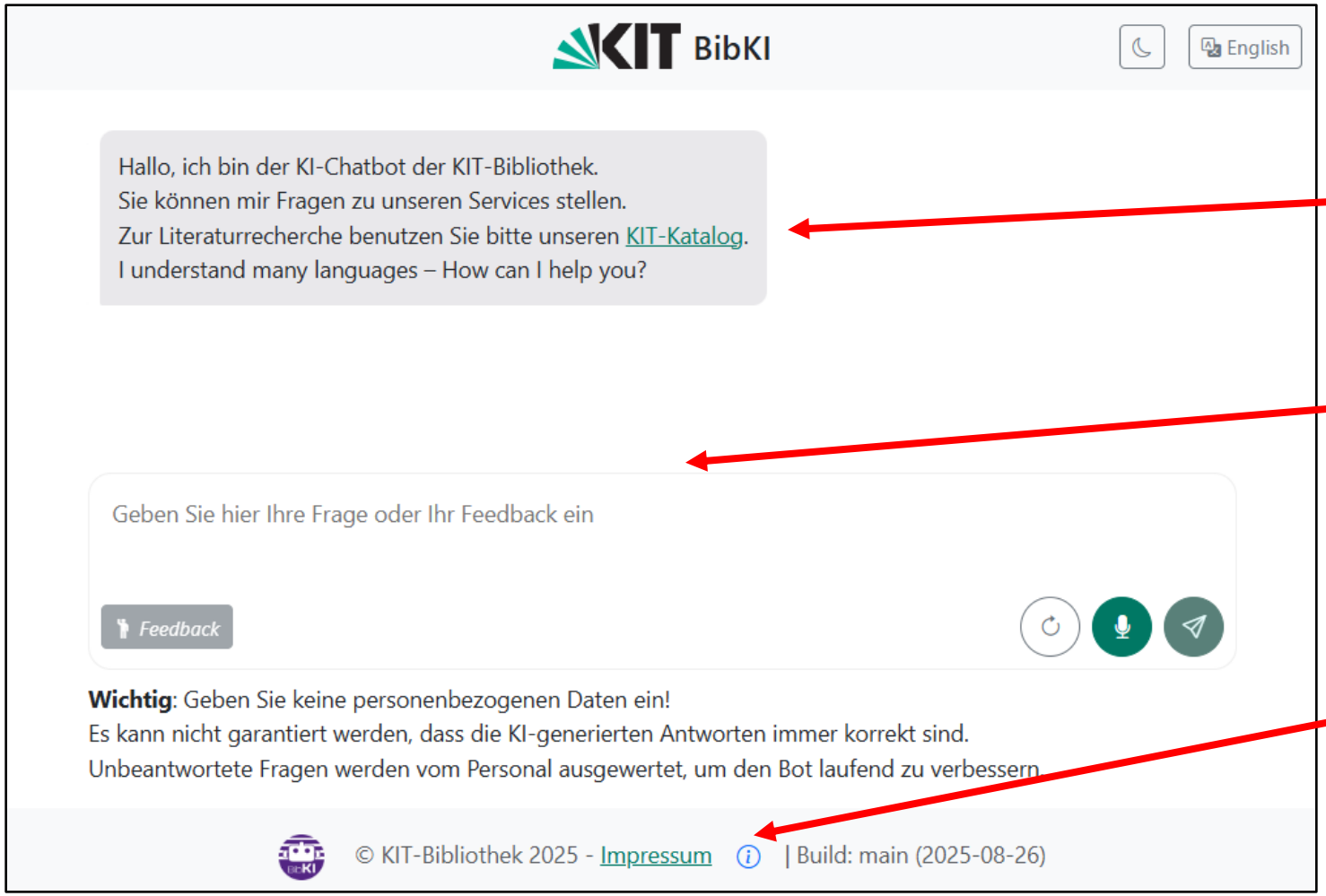
BibKI – der KI-Service-Chatbot – Light Mode



BibKI – der KI-Service-Chatbot – Dark Mode



BibKI – der KI-Service-Chatbot – Light Mode



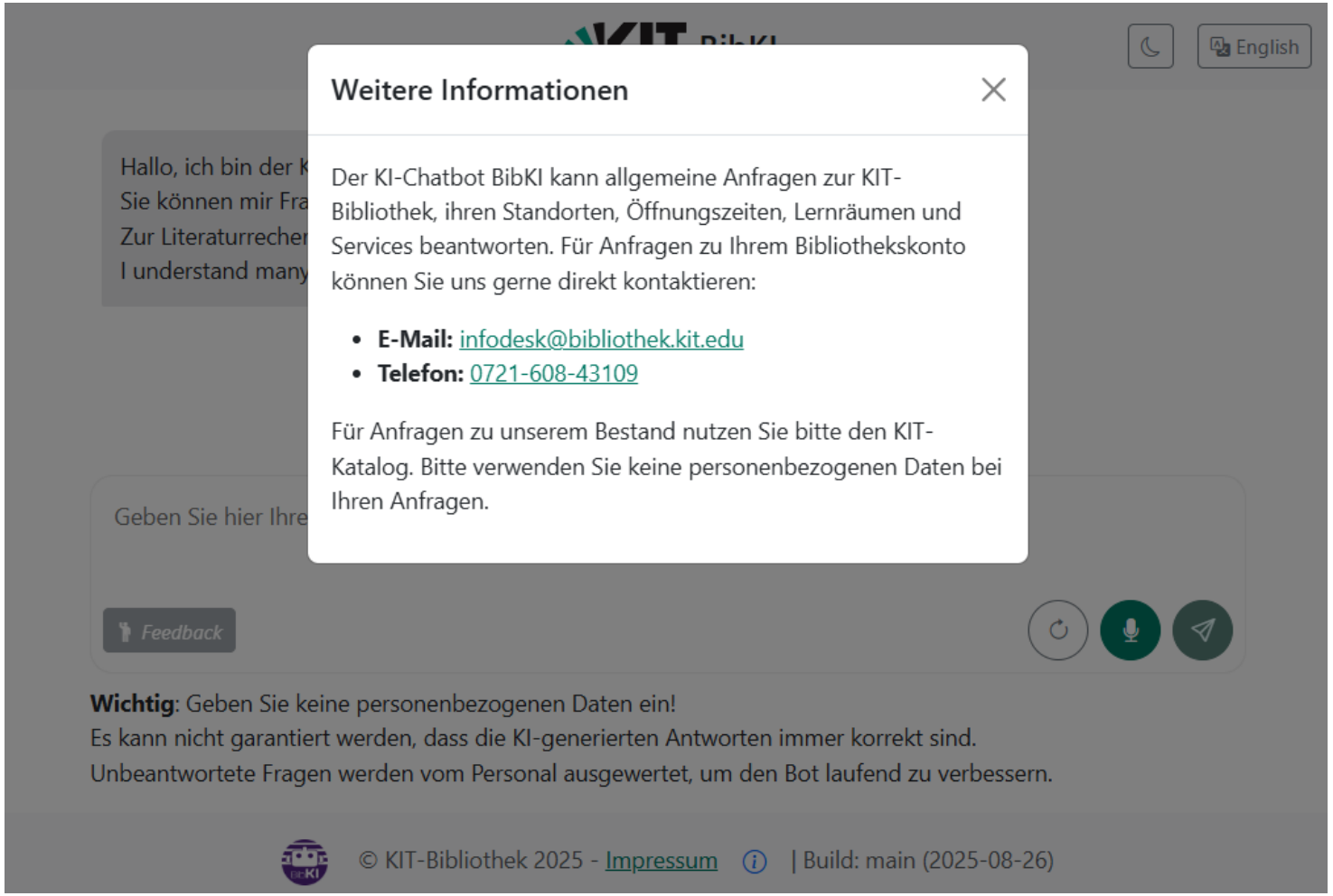
Welcome Message

Frage

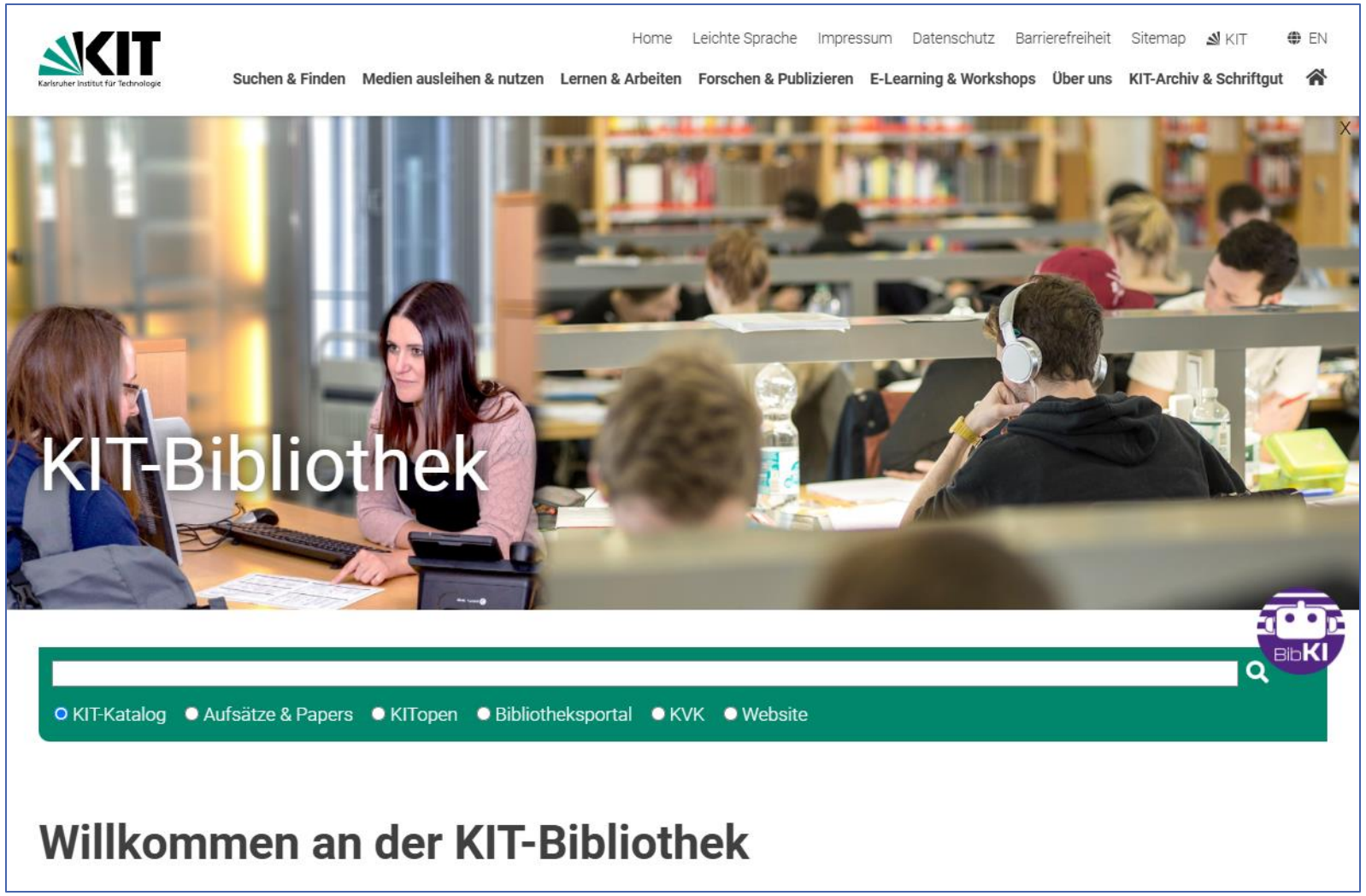
Infos

Alle Texte sind frei konfigurierbar

BibKI – der KI-Service-Chatbot – Light Mode



BibKI – Integration auf einer Webseite



KI-Service-Chatbot

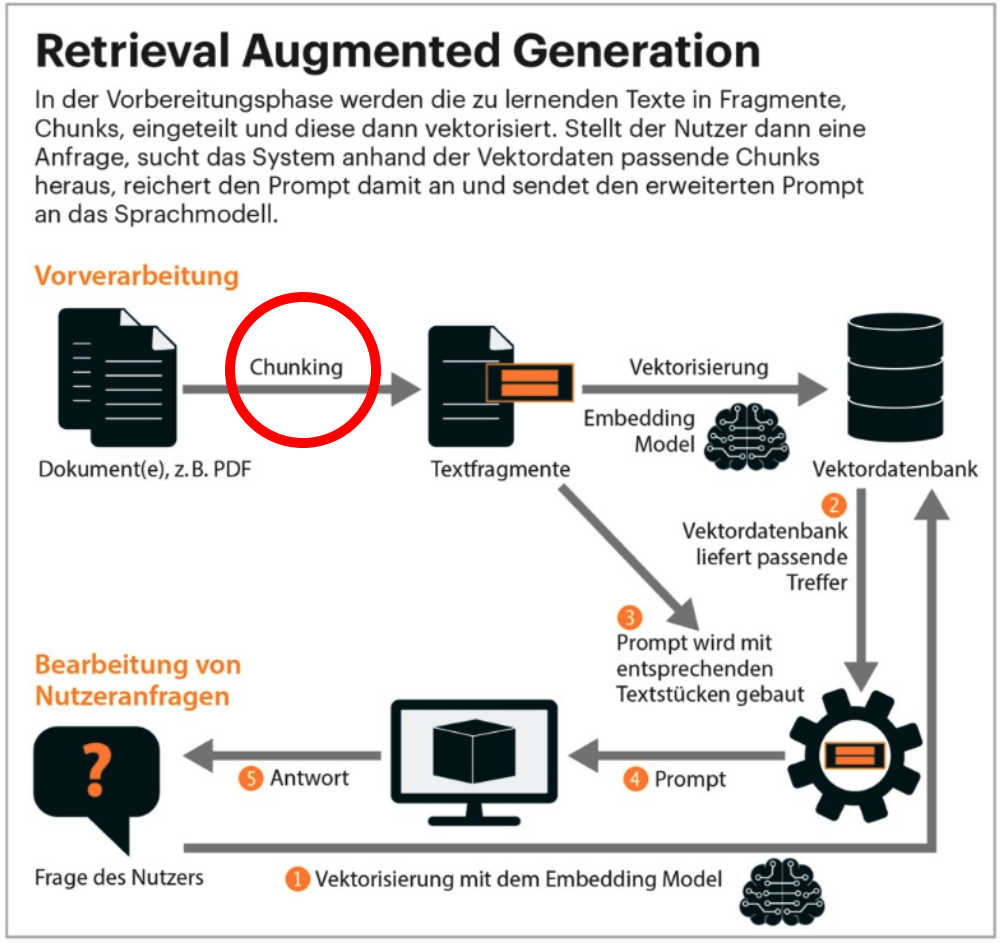
- Man möchte Fragen auf Basis einer eigenen Wissensbasis beantworten können und nicht das Wissen eines großen Sprachmodells nutzen
  - Wissensbasis = meine Daten (Dokumente etc.)
- Dazu benutzen wir RAG = Retrieval Augmented Generation
  - LLM = Large Language Model (großes Sprachmodell)
  - Embeddings
  - Vector Store
  - Context Window

Was macht RAG?

- Mittels RAG wird nach Dokumenten gesucht, die am besten zur Beantwortung einer Frage geeignet sind

- Hierzu wird eine „semantische Suche“ durchgeführt
  - Das ist eine **Ähnlichkeitssuche**
  - Kataloge führen eine Stichwortsuche durch
  - Eine Semantische Suche ist unschärfer.
    Dafür findet man aber auch dann etwas, wenn die Suchbegriffe mit den Begriffen in der Datenbank nicht übereinstimmen
    - Bsp: „Wie kann ich meine Bücher zurückgeben?“
      Findet auch Dokumente zur Rückgabestation.

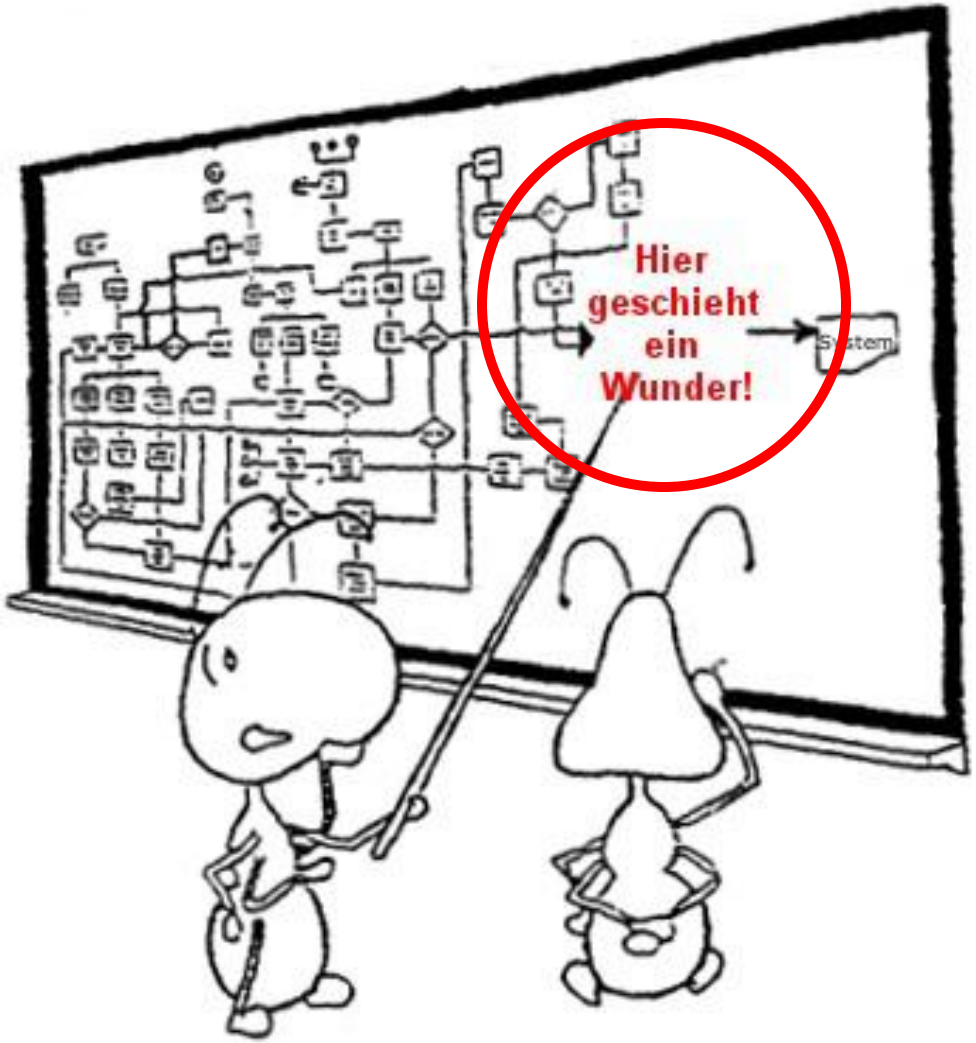
RAG-Workflow von c‘t



Quelle: c‘t Heft 7/2025, S. 145

RAG-Kurzfassung ohne „Mathe“

- Finde Dokumente, die zur Beantwortung einer Frage passen
- Führe dazu eine Ähnlichkeitssuche durch (RAG bzw. file_search)
- Ein LLM erzeugt dann eine Antwort



Sehr gute Arbeit!
Aber sollten wir hier nicht vielleicht etwas detaillierter werden...?

RAG alleine genügt heutzutage nicht mehr

- Nachteil an RAG
  - Dokumente sind statisch
  - Ein LLM kennt normalerweise nicht mal die Uhrzeit

- Man möchte Funktionen nutzen können, die Echtzeitinformationen ermitteln. Das LLM soll
  - diese Funktionen auswählen,
  - die Parameter dazu aus der Anfrage ermitteln,
  - die (meistens maschinenlesbare) Antwort im Klartext für Endnutzer ausgeben

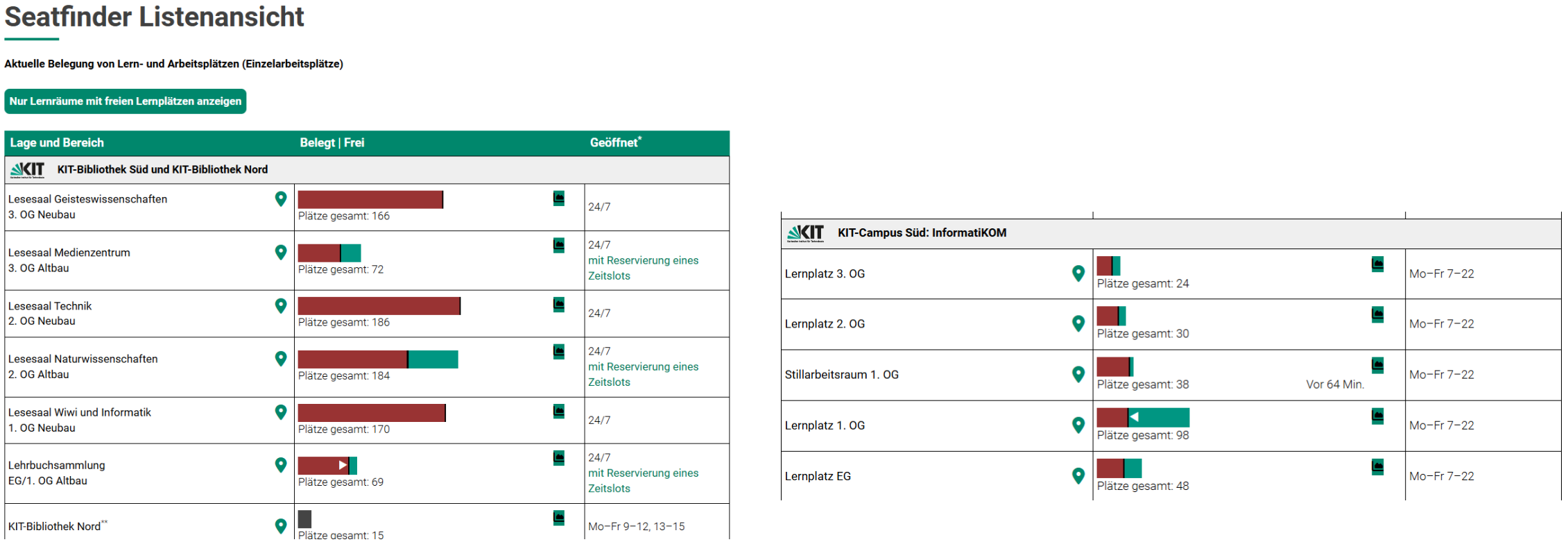
Erweiterung von Chatbots um „Tool Calls“

- Integration von Online-Services mit Echtzeit-Informationen
  - Suche in natürlicher Sprache zu aktuellen Informationen
- Beschreibung eines Tools in JSON genügt
  - LLM entscheidet, ob File Search (RAG) oder Tool Call durchgeführt wird

- Beispiele in Karlsruhe
  - Seatfinder
  - Mittagessen
  - Gebäude- und Raum-/Hörsaalsuche
  - Simple Form der Literaturrecherche
  - Uhrzeit, Wetter als einfache Tool Calls

Seatfinder

- https://www.bibliothek.kit.edu/freie-lernplaetze.php



Seatfinder Listenansicht

Aktuelle Belegung von Lern- und Arbeitsplätzen (Einzelarbeitsplätze)

Nur Lernräume mit freien Lernplätzen anzeigen

Lage und Bereich	Belegt | Frei	Geöffnet*
KIT-Bibliothek Süd und KIT-Bibliothek Nord		
Lesesaal Geisteswissenschaften 3. OG Neubau	Plätze gesamt: 166	24/7
Lesesaal Medienzentrum 3. OG Altbau	Plätze gesamt: 72	24/7 mit Reservierung eines Zeitslots
Lesesaal Technik 2. OG Neubau	Plätze gesamt: 186	24/7
Lesesaal Naturwissenschaften 2. OG Altbau	Plätze gesamt: 184	24/7 mit Reservierung eines Zeitslots
Lesesaal Wiwi und Informatik 1. OG Neubau	Plätze gesamt: 170	24/7
Lehrbuchsammlung EG/1. OG Altbau	Plätze gesamt: 69	24/7 mit Reservierung eines Zeitslots
KIT-Bibliothek Nord**	Plätze gesamt: 15	Mo–Fr 9–12, 13–15
KIT-Campus Süd: InformatiKOM		
Lernplatz 3. OG	Plätze gesamt: 24	Mo–Fr 7–22
Lernplatz 2. OG	Plätze gesamt: 30	Mo–Fr 7–22
Stillarbeitsraum 1. OG	Plätze gesamt: 38 Vor 64 Min.	Mo–Fr 7–22
Lernplatz 1. OG	Plätze gesamt: 98	Mo–Fr 7–22
Lernplatz EG	Plätze gesamt: 48	Mo–Fr 7–22

Tool Call Beispiele für Seatfinder & more

- Thema Lernraum
  - Gibt es in der Bib noch freie Plätze zum Lernen?
  - Sind im LSG noch Plätze zum Lernen frei?

- Essenspläne - Mensa & more
  - Ich bin Veganer, finde ich heute in der Mensa was für mich zum Essen?
  - Welche Salate gibt es diese Woche in der Mensa in der Moltkestraße?
  - Welche Pasta-Gerichte gibt es diese Woche?

- Wo ist der „Eiermann“
  - Hörsaal-Spitzname
  - Ausgabe von Campus-Plan- und Google-Maps-Link

MCP – Model Context Protocol

- Damit können Sie Ihre Tool Calls auch anderen zur Verfügung stellen

- Von Anthropic, den Machern von Claude
- https://modelcontextprotocol.io/introduction
  - Standardisierte Schnittstelle, über die Sprachmodelle externe Tools und APIs verwenden können, ohne dass diese APIs speziell auf das Modell angepasst werden müssen
- Liste aller Server
  - https://github.com/modelcontextprotocol/servers
  - https://mcpmarket.com/

MCP – Beispiele Airbnb & Blender

- Suche in natürlicher Sprache
  - Bsp. Airbnb:
    - ich suche ein Zimmer für 2 Personen in Paris am 30.10. für unter 120 €
  - LLM extrahiert die Parameter

```
{
  `adults`      : 2,
  `location`    : `Paris, France`,
  `checkin`     : `2025-10-30`,
  `checkout`    : `2025-10-31`,
  `maxPrice`    : 120
}
```

- YouTube: „MCP will change the world” von c’t 3003
  - https://www.youtube.com/watch?v=S_4VUJ-x8hE
  - Hier wird das 3D-Programm “Blender” per Sprache von einem absoluten Blender-Laien bedient

MCP-Ablauf

- Client fragt mit „list tools“ beim Service-Anbieter an
  - „Sag mir, was Du kannst“ bzw.
  - „Welche API endpoints bietest Du an?“ bzw.
  - „Wie sieht Deine Schnittstelle aus?“
- Service-Anbieter (MCP-Server) liefert in JSON alle „Tools“ mit „description“ und Parametern
- Sollte eine „description“ vom LLM als passend zur Beantwortung einer Frage im Chat erkannt werden, versucht das LLM dieses „Tool“ aufzurufen
  - Dazu werden alle Parameter (sofern möglich) aus der Anfrage extrahiert und an die function übergeben
- Das Ergebnis wird ausgewertet
  - das klappt, da das LLM die Antwort (JSON, XML etc.) gut „verstehen“ kann

Tool Call beim Response API

```
FUNCTION_TOOLS = [
    {
        "type": "function",
        "name": "get_weather",
        "description": "Get current weather for a city",
        "parameters": {
            "type": "object",
            "properties": {
                "city": {"type": "string", "description": "City name"}
            },
            "required": ["city"]
        }
    }
]
```

```
def get_weather(city: str):
    debug(f"get_weather called with city: {city}")
    # Demo-Implementierung
    return {
        "city": city,
        "when": datetime.now().isoformat(),
        "forecast": "sunny",
        "temp_c": 24.3
    }
```

Zugehöriger Python Code
function-name = tool-name

Funktionsbeschreibung in JSON

Tool Call beim Response API

```
SEATFINDER_TOOL = {
    "type": "function",
    "name": "get_seatfinder_data",
    "description": (
        "Sucht nach freien Lern- und Arbeitsplätzen. "
        "Gibt dazu die aktuellen Sitzplatzdaten für alle KIT-Lernräume zurück. "
        "Es werden die festen Lernräume LSG, LSM, LST, LSN, LSW, LBS, "
        "BIB-N, FBC, FBP, LAF, FBA, FBI, FBM, FBH, FBD, BLB und WIS abgefragt. "
        "Sowohl seatestimate als auch manualcount werden berücksichtigt."
    ),
    "parameters": {
        "type": "object",
        "properties": {}  # keine Eingabeparameter nötig
    }
}
```

Funktionsbeschreibung in JSON

```
def get_seatfinder_data():
    url = ("https://seatfinder.bibliothek.kit.edu/karlsr
           "?values=seatestimate,manualcount,locatio
           "&location=" + ",".join(LOCATIONS) + "&lim
    …
    …
```

Zugehöriger Python Code für dieses Tool

Tool Call beim Response API

```
FUNCTION_TOOLS.append(SEATFINDER_TOOL)

TOOLS = [
    {"type": "file_search", "vector_store_ids": [VECTOR_STORE_ID] if VECTOR_STORE_ID else []},
    *FUNCTION_TOOLS
]

response = client.responses.create(
    model="gpt-4o-mini",
    input=[sys_msg,
           {"role": "user", "content": [{"type": "input_text", "text": prompt}]}],
    tools=TOOLS,
    max_output_tokens=800,
)
```

Erlaube file_search und tool calls

MCP – Python-Code-Beispiel

```
from fastmcp import FastMCP
…
…
@mcp_server.tool()
async def get_seatfinder_data():
    """Fetch current seat availability data from the KIT Seatfinder API.
       Gib am Ende immer die URL zum Seatfinder
       https://www.bibliothek.kit.edu/freie-lernplaetze.php aus."""
    try:
       data = await get_seatfinder_data_internal()
       return format_seatfinder_result(data)
….
```

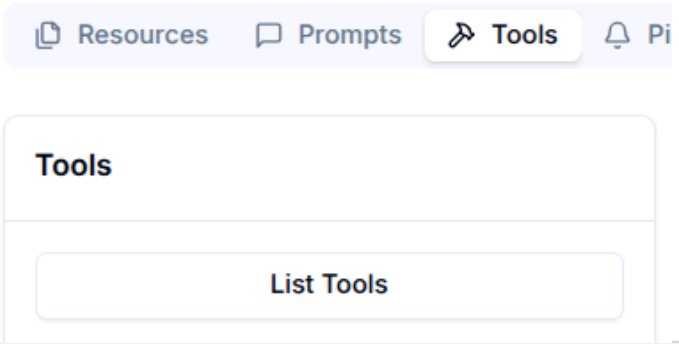
Decorator erzeugt das JSON für die Funktionsbeschreibung aus der Python def

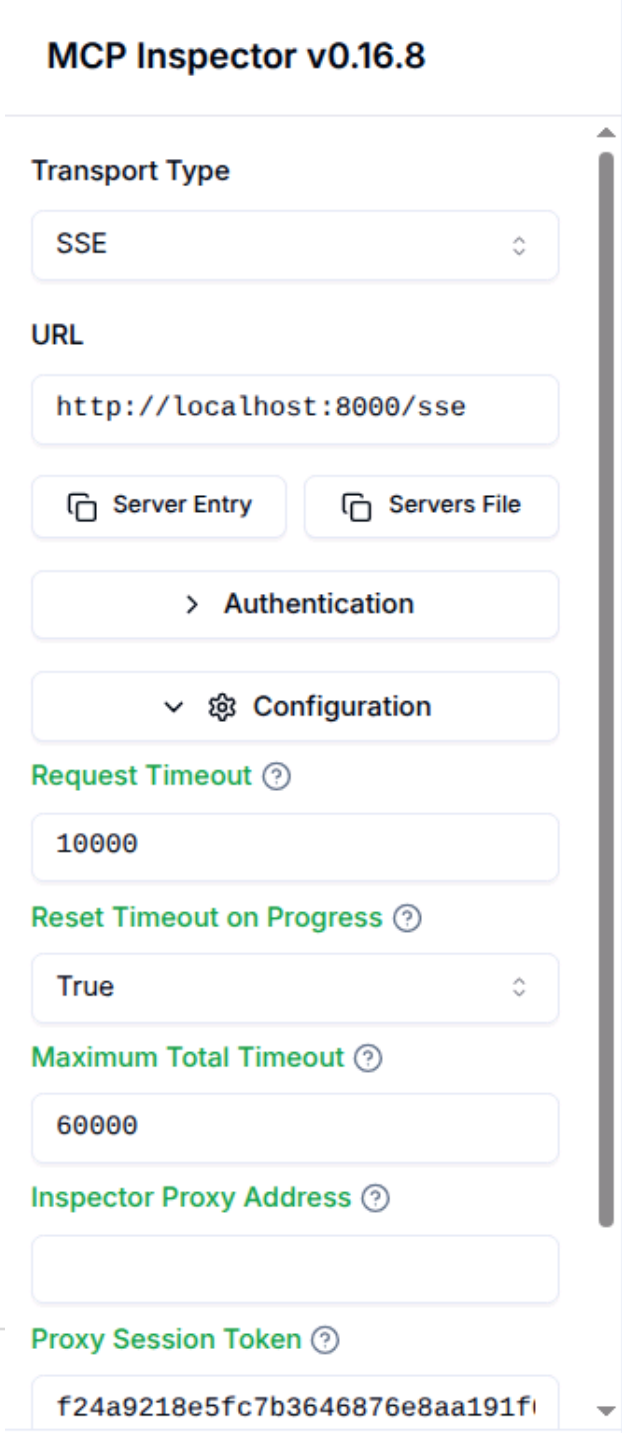
Selber ausprobieren

- Integration von MCP in Claude / ChatGPT
  - MCP-Service läuft lokal und kommuniziert über stdio
  - MCP-Service läuft auf einem Server und wird per streamable http (SSE) benutzt
- Debuggen von MCP
  - MCP-Inspector

Debugging mit MCP inspector

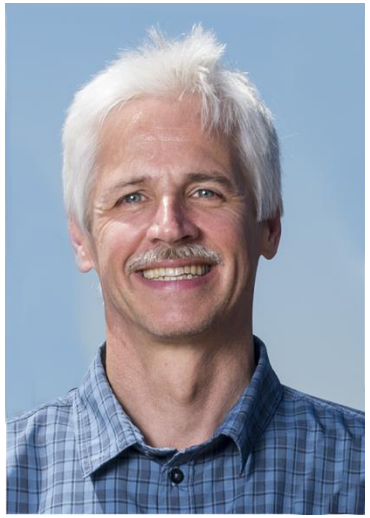
- 1. Konsole: MCP Server starten
- 2. Konsole: MCP Inspector starten
  - npx @modelcontextprotocol/inspector
- Configuration öffnen, falls Connect nicht klappt
- Transport Type einstellen
- URL, falls SSE
- Proxy Session Token eintragen
  - Erscheint beim Starten in der Konsole
- List Tools
- Tools auswählen
- Tool ausführen (Run Tool)





Diskussion

- Welche Tool Calls fänden Sie sinnvoll?



Vielen Dank für Ihre Aufmerksamkeit

Fragen?

uwe.dierolf@kit.edu

https://cloud.bibliothek.kit.edu/index.php/s/WaXPABEWXawsStw