

# KI-Austausch Tool Calls & MCP in KI-Chatbots

**Uwe Dierolf**

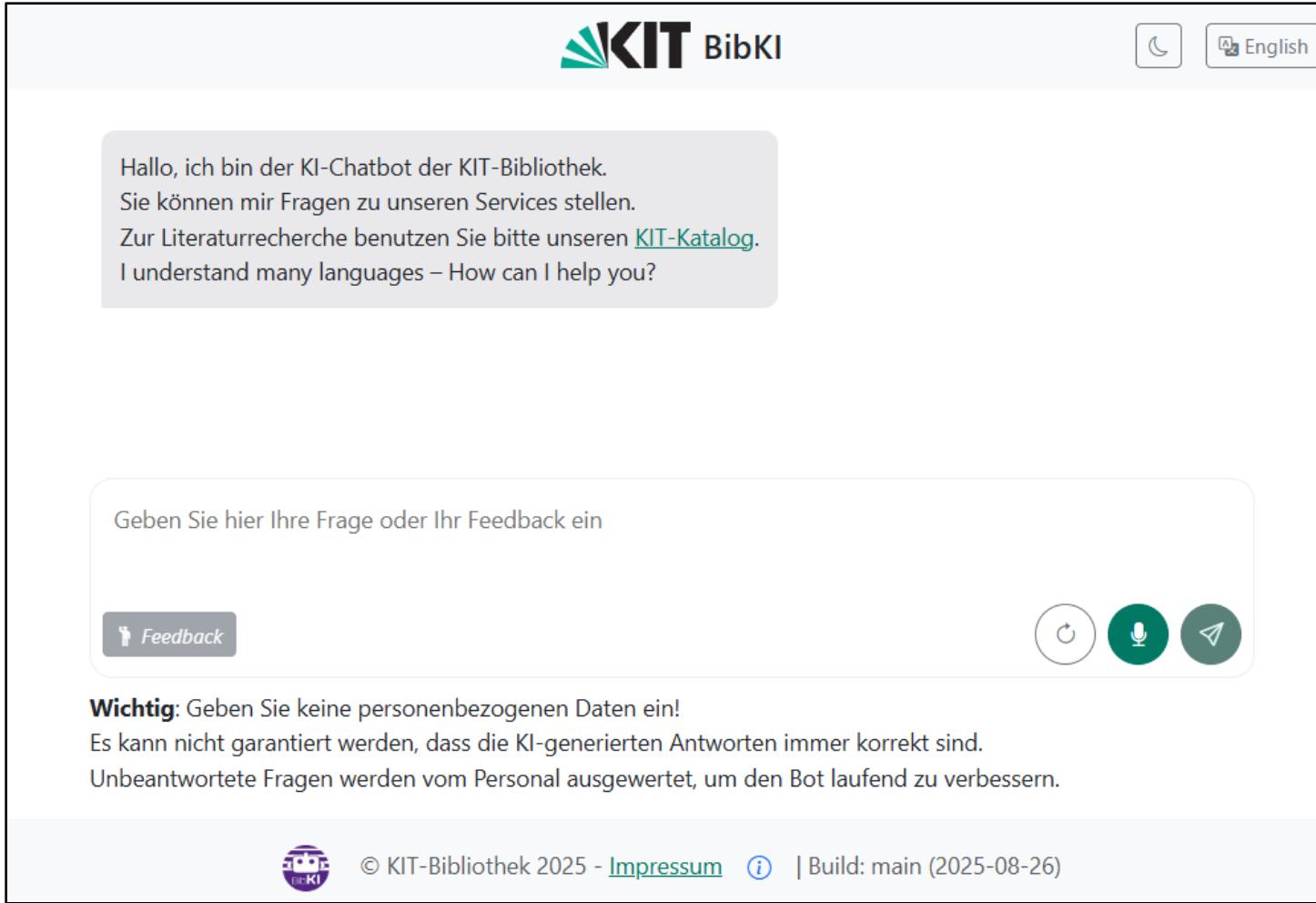


# Übersicht

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

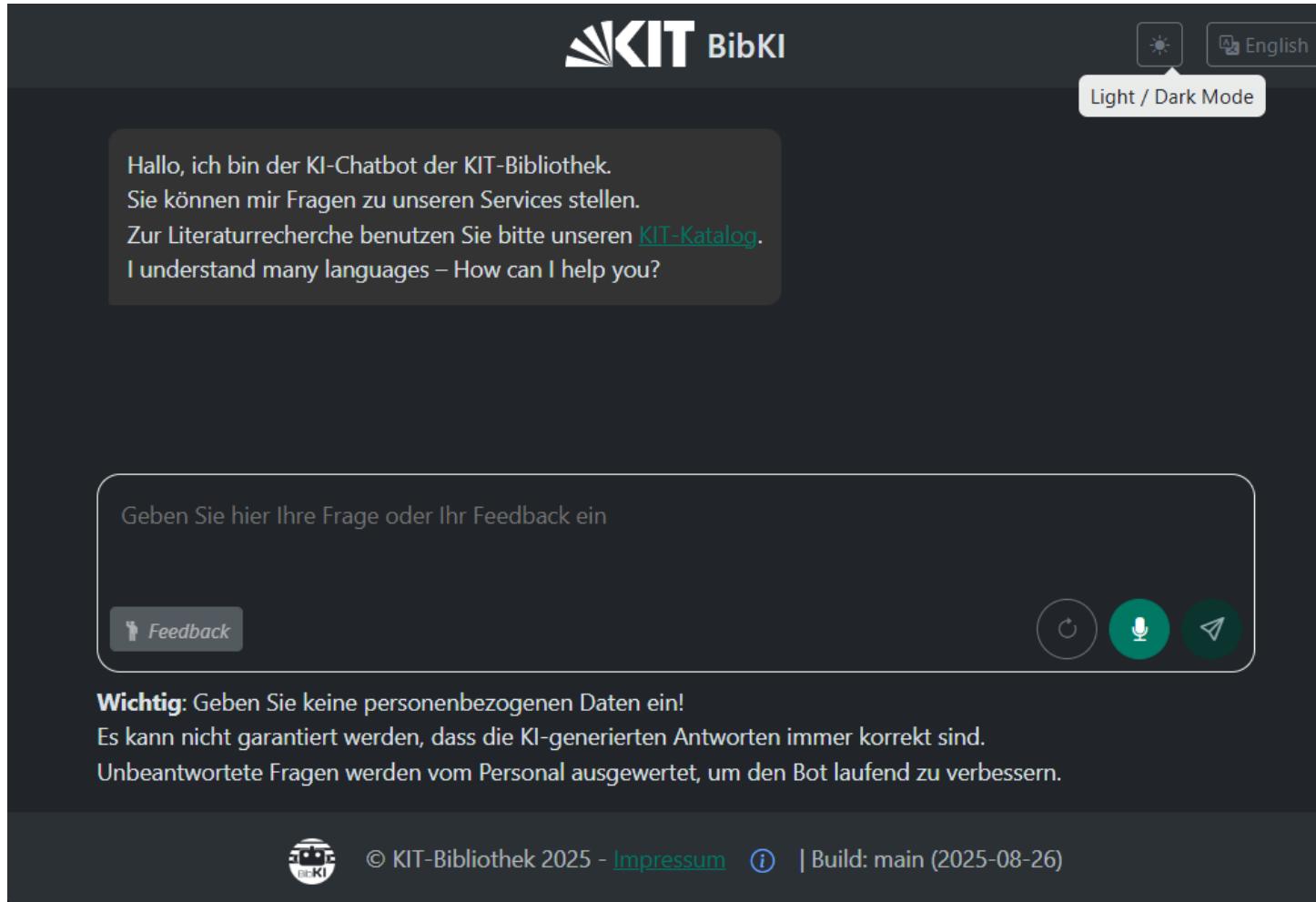


# BibKI – der KI-Service-Chatbot – Light Mode



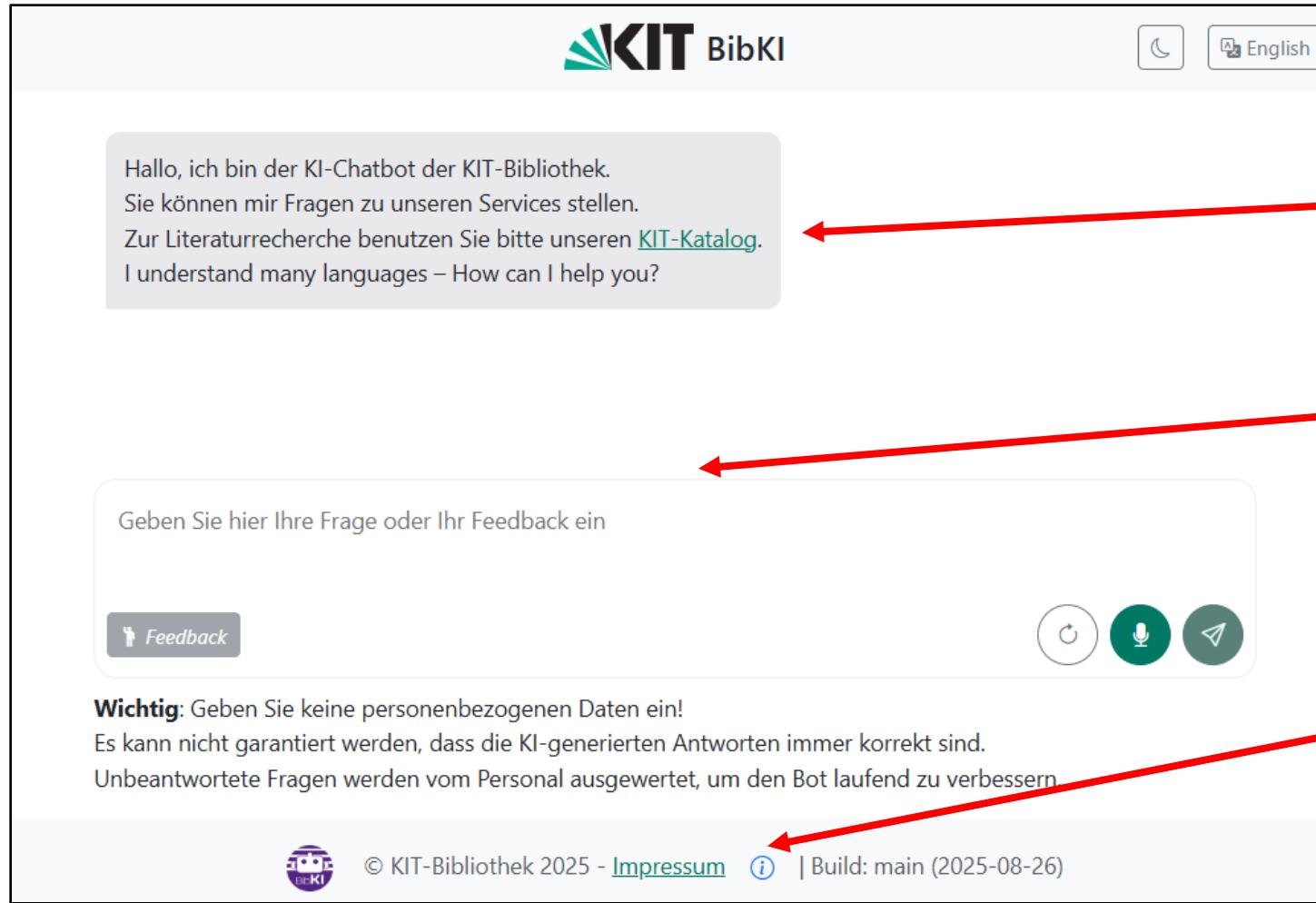
The screenshot shows the BibKI Light Mode chat interface. At the top, the KIT BibKI logo is displayed, along with a language selection button for English. A message from the chatbot is shown: "Hallo, ich bin der KI-Chatbot der KIT-Bibliothek. Sie können mir Fragen zu unseren Services stellen. Zur Literaturrecherche benutzen Sie bitte unseren [KIT-Katalog](#). I understand many languages – How can I help you?" Below this, there is a text input field with placeholder text "Geben Sie hier Ihre Frage oder Ihr Feedback ein". Underneath the input field are three buttons: "Feedback" (grey), a circular refresh icon, a microphone icon, and a circular arrow icon. A note below the input field reads: "Wichtig: Geben Sie keine personenbezogenen Daten ein! Es kann nicht garantiert werden, dass die KI-generierten Antworten immer korrekt sind. Unbeantwortete Fragen werden vom Personal ausgewertet, um den Bot laufend zu verbessern." At the bottom, there is a footer with the KIT logo, copyright information ("© KIT-Bibliothek 2025 - [Impressum](#)"), a build status indicator ("Build: main (2025-08-26)"), and a small blue circular icon.

# BibKI – der KI-Service-Chatbot – Dark Mode



The screenshot shows the BibKI AI Service Chatbot interface in Dark Mode. At the top, the KIT BibKI logo is displayed, along with language selection buttons for English and a "Light / Dark Mode" switch. A message from the bot reads: "Hallo, ich bin der KI-Chatbot der KIT-Bibliothek. Sie können mir Fragen zu unseren Services stellen. Zur Literaturrecherche benutzen Sie bitte unseren [KIT-Katalog](#). I understand many languages – How can I help you?" Below this, there is a large input field with placeholder text "Geben Sie hier Ihre Frage oder Ihr Feedback ein". Underneath the input field are three buttons: "Feedback" (with a person icon), a refresh/circular arrow icon, and a microphone icon. A note below the input field states: "Wichtig: Geben Sie keine personenbezogenen Daten ein! Es kann nicht garantiert werden, dass die KI-generierten Antworten immer korrekt sind. Unbeantwortete Fragen werden vom Personal ausgewertet, um den Bot laufend zu verbessern." At the bottom, there is a footer bar with the KIT logo, copyright information ("© KIT-Bibliothek 2025 - [Impressum](#)"), a help icon, and build information ("Build: main (2025-08-26)").

# BibKI – der KI-Service-Chatbot – Light Mode



The screenshot shows the BibKI Light Mode chatbot interface. At the top left is the KIT BibKI logo. At the top right are language selection buttons for German (selected) and English. The main content area contains a welcome message in German, followed by a question input field and a feedback button. Below this is a section with important information and a footer with copyright and build details.

Hallo, ich bin der KI-Chatbot der KIT-Bibliothek.  
Sie können mir Fragen zu unseren Services stellen.  
Zur Literaturrecherche benutzen Sie bitte unseren [KIT-Katalog](#).  
I understand many languages – How can I help you?

Geben Sie hier Ihre Frage oder Ihr Feedback ein

**Wichtig:** Geben Sie keine personenbezogenen Daten ein!  
Es kann nicht garantiert werden, dass die KI-generierten Antworten immer korrekt sind.  
Unbeantwortete Fragen werden vom Personal ausgewertet, um den Bot laufend zu verbessern

© KIT-Bibliothek 2025 - [Impressum](#) | Build: main (2025-08-26)

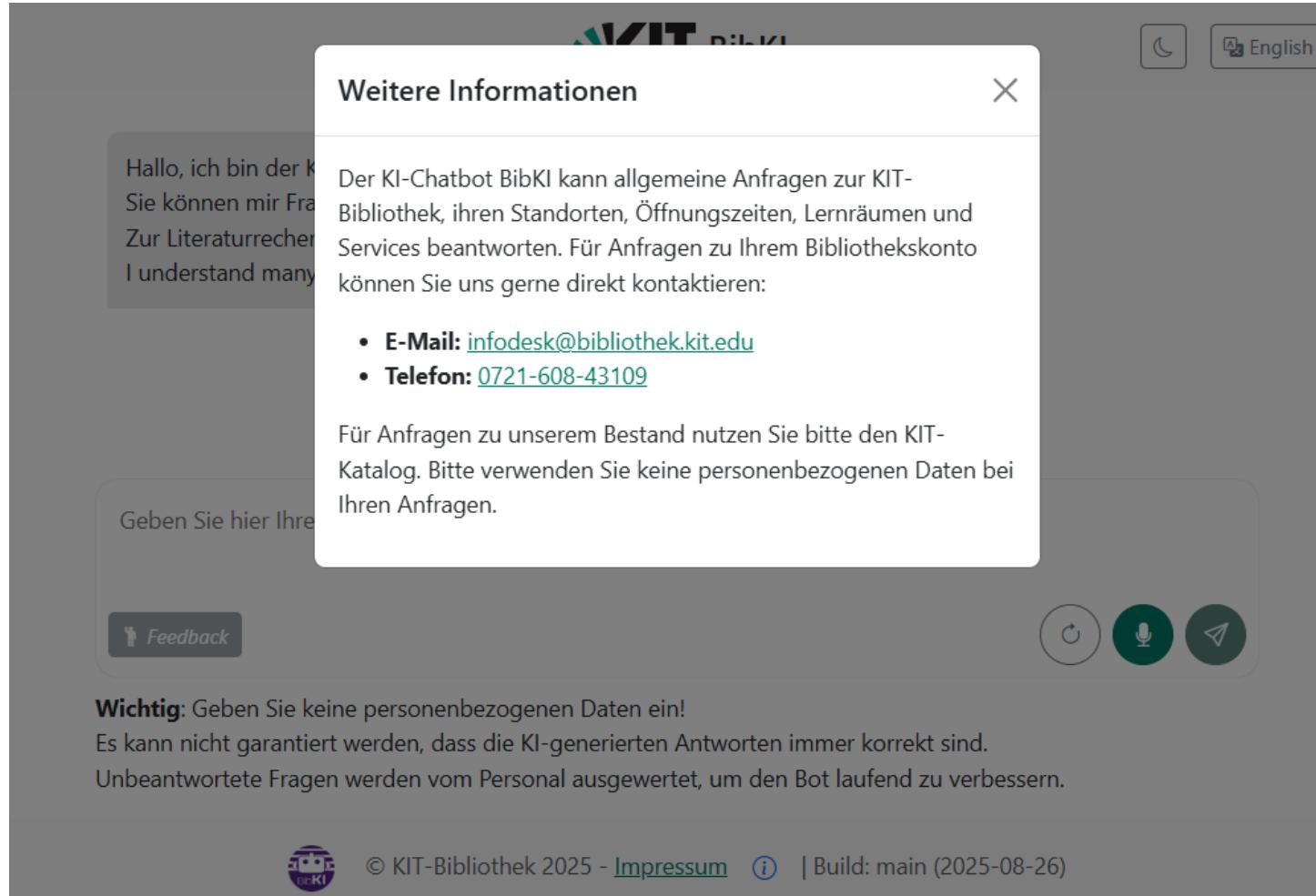
Welcome Message

Frage

Infos

Alle Texte  
sind frei  
konfigurierbar

# BibKI – der KI-Service-Chatbot – Light Mode



Hallo, ich bin der KI-Service-Chatbot BibKI. Sie können mir Fragen zu KIT-Bibliothek und Lernräumen stellen. Zur Literaturrecherche und anderen Services kann ich Ihnen helfen. I understand many languages.

Geben Sie hier Ihre Anfrage ein:

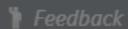
**Weitere Informationen**

Der KI-Chatbot BibKI kann allgemeine Anfragen zur KIT-Bibliothek, ihren Standorten, Öffnungszeiten, Lernräumen und Services beantworten. Für Anfragen zu Ihrem Bibliothekskonto können Sie uns gerne direkt kontaktieren:

- **E-Mail:** [infodesk@bibliothek.kit.edu](mailto:infodesk@bibliothek.kit.edu)
- **Telefon:** [0721-608-43109](tel:0721-608-43109)

Für Anfragen zu unserem Bestand nutzen Sie bitte den KIT-Katalog. Bitte verwenden Sie keine personenbezogenen Daten bei Ihren Anfragen.

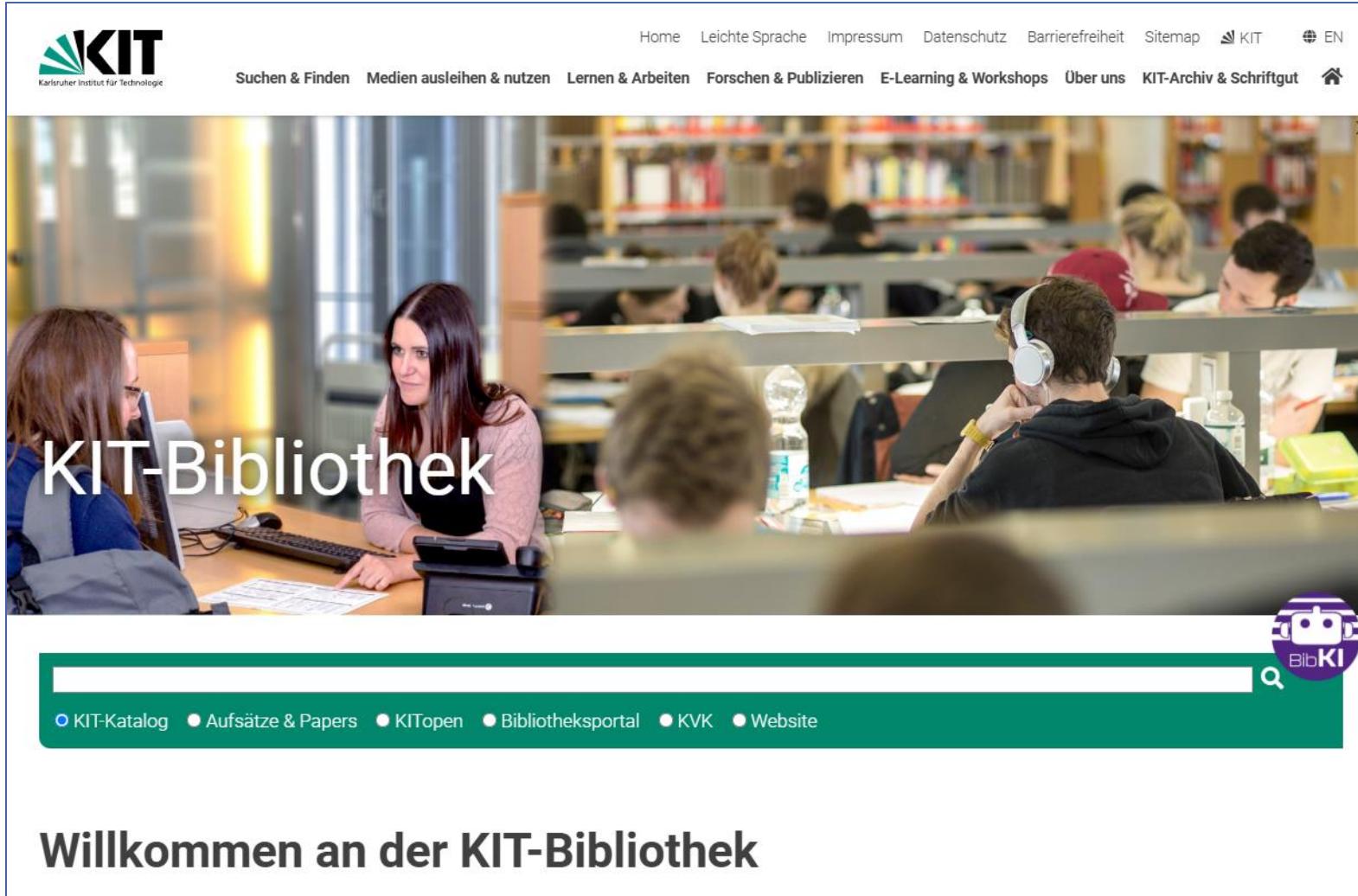
**Wichtig:** Geben Sie keine personenbezogenen Daten ein!  
Es kann nicht garantiert werden, dass die KI-generierten Antworten immer korrekt sind.  
Unbeantwortete Fragen werden vom Personal ausgewertet, um den Bot laufend zu verbessern.

 Feedback

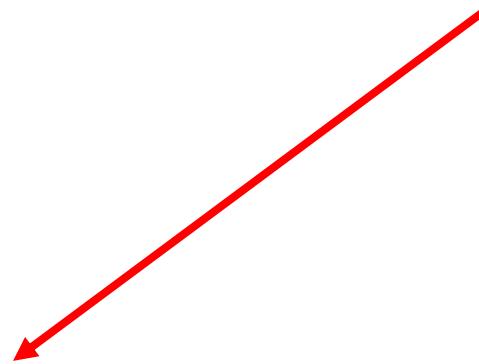
 © KIT-Bibliothek 2025 - [Impressum](#) | Build: main (2025-08-26)

# BibKI – Integration auf einer Webseite



The screenshot shows the homepage of the KIT-Bibliothek website. At the top, there is a navigation bar with links to Home, Leichte Sprache, Impressum, Datenschutz, Barrierefreiheit, Sitemap, a KIT logo, and EN. Below the navigation bar is a banner image of students studying in a library. Overlaid on the banner is the text "KIT-Bibliothek". At the bottom of the page is a green footer bar containing a search bar with a placeholder "Suchen", a purple BibKI logo with a camera icon, and links to various services: KIT-Katalog, Aufsätze & Papers, KITopen, Bibliotheksportal, KVK, and Website.

Willkommen an der KIT-Bibliothek



# 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

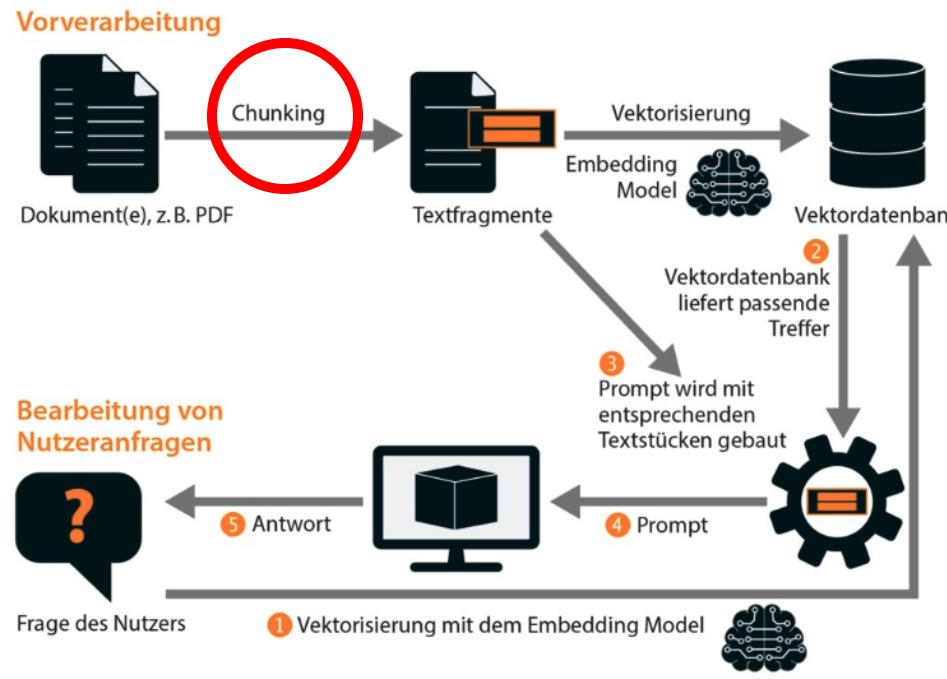
# Was macht RAG?

- 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

## Retrieval Augmented Generation

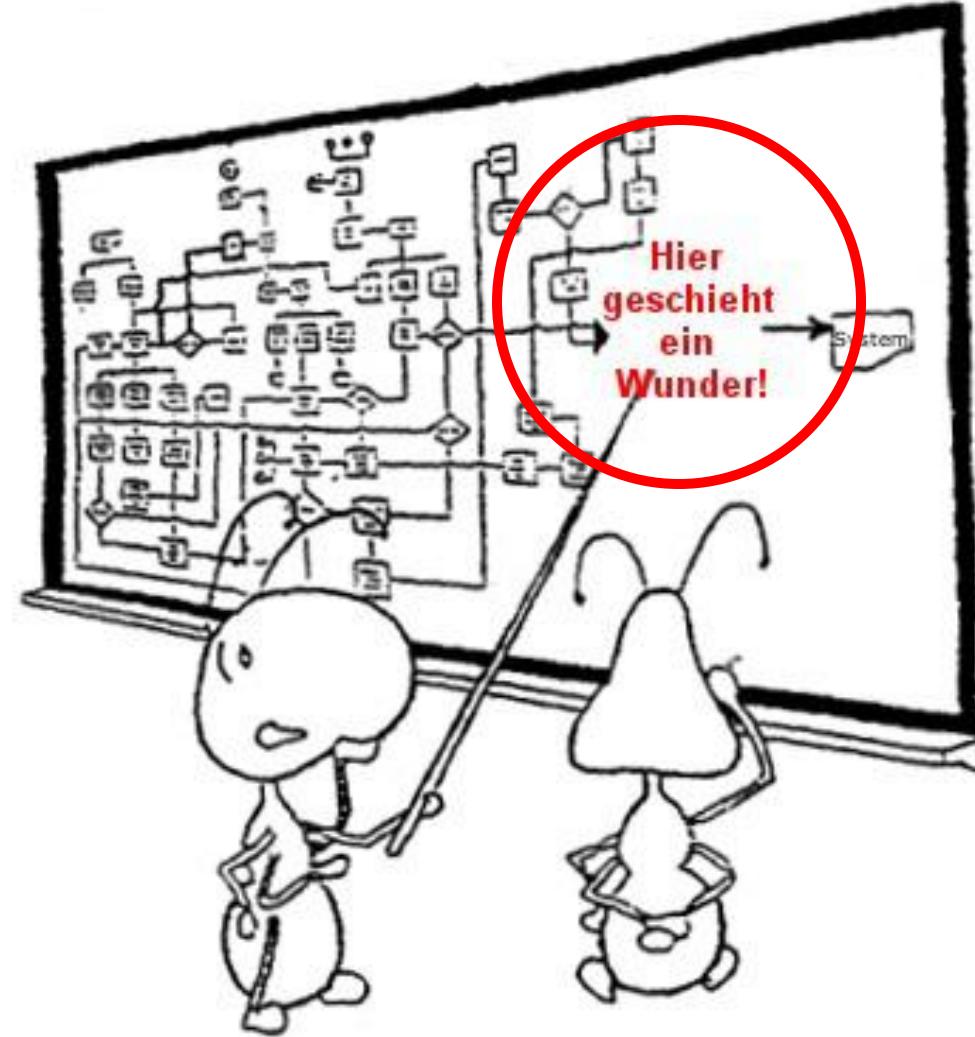
In der Vorbereitungsphase werden die zu lernenden Texte in Fragmente, Chunks, eingeteilt und diese dann vektorisiert. Stellt der Nutzer dann eine Anfrage, sucht das System anhand der Vektordaten passende Chunks heraus, reichert den Prompt damit an und sendet den erweiterten Prompt an das Sprachmodell.



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“

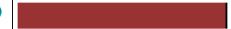
- Integration von Online-Services mit Echtzeit-Informationen
  - Suche in natürlicher Sprache zu aktuellen Informationen
- Beschreibung eines Tools in JSON genügt
  - LLM entscheidet, ob File Search (RAG) oder Tool Call durchgeführt wird
- Beispiele in Karlsruhe
  - Seatfinder
  - Mittagessen
  - Gebäude- und Raum-/Hörsaalsuche
  - Simple Form der Literaturrecherche
  - Uhrzeit, Wetter als einfache Tool Calls

■ <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*
<b>KIT-Bibliothek Süd und KIT-Bibliothek Nord</b>		
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

<b>KIT-Campus Süd: InformatiKOM</b>			
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.
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](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  
    }  
}
```

```
def get_seatfinder_data():  
    url = ("https://seatfinder.bibliothek.kit.edu/karlsruher-institut-fuer-technologie/seatestimate?values=seatestimate,manualcount,location&location=" + ",".join(LOCATIONS) + "&limit=100")  
    ...  
    ...  
    ...
```

Zugehöriger Python Code  
für dieses Tool

## Funktionsbeschreibung in JSON

# 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)
    ...
    
```

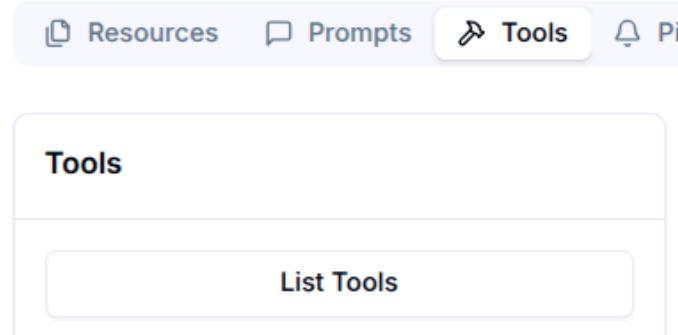
Decorator erzeugt das JSON für die Funktionsbeschreibung aus der Python def

# 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)



Transport Type

SSE

URL

`http://localhost:8000/sse`

[Server Entry](#)  [Servers File](#)

[Authentication](#)

[Configuration](#)

Request Timeout [?](#)

10000

Reset Timeout on Progress [?](#)

True

Maximum Total Timeout [?](#)

60000

Inspector Proxy Address [?](#)

Proxy Session Token [?](#)

f24a9218e5fc7b3646876e8aa191f1

# Diskussion

- Welche Tool Calls finden Sie sinnvoll?



Vielen Dank für Ihre Aufmerksamkeit

Fragen?

[uwe.dierolf@kit.edu](mailto:uwe.dierolf@kit.edu)



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