



# KI

## Praxisworkshop, Vortrag, Online-Webinar

Fokus: KI-Landkarte, Agenten, Workflows, Entwicklungsumgebungen für KI-“Coder”



**Autor:** Ralf Kluth

**Web:** <https://from-scratch.ai> | X: @fromscratchai | LinkedIn: <https://www.linkedin.com/in/rakidakixyz/>



# Agenda

- 01 KI-Landkarte und News**
- 02 Überblick Agenten, Workflows, ...**
- 03 100 Zeilen Python Code – Basis für KI-Agenten**
- 04 Cursor/ Claude Code und 100 Zeilen Python**
- 05 KI-Entwicklung: Manus, Lovable, Dyad, Kimi, CC**
- 06 Claude Code Setup (u.a. BMAD)**

# 1

## KI-News und KI-Landkarte (Produkte, Lösungen, LLMs)

**Neuigkeiten, OpenAI – Anthropic Claude – Google Gemini => Neue Funktionen**

**Landkarte der KI-Lösungen**

**Open-Source Community und Lösungen**

# KI-News

## Rechenzentren, Investitionen, Firmenbewertungen



Bau von neuen gigantischen Rechenzentren

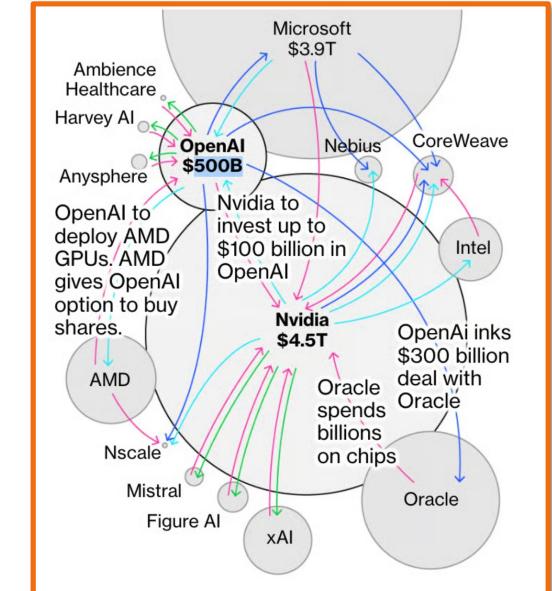
**Anthropic verdreifacht seine Bewertung auf 183 Milliarden Dollar**

Damit wird der OpenAI-Konkurrent zu einem der höchstbewerteten Start-ups der Welt. Der KI-Entwickler will nun international expandieren und vermehrt zu Sicherheit forschen.

**Überbewertung von KI-Firmen?**



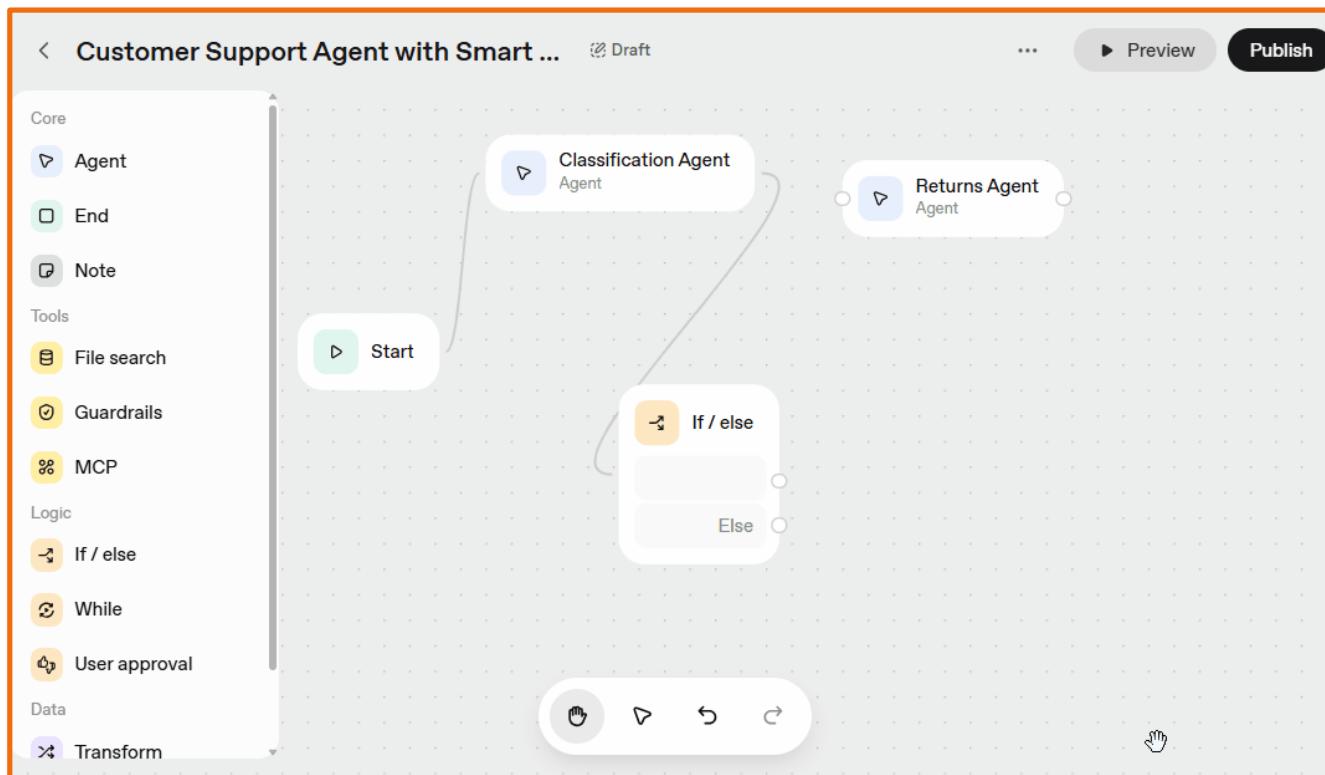
KI verändert die Zukunft  
Ängste vs. Visionen



Geldflüsse in der KI-Bubble

# KI-News

## OpenAI, Anthropic, Google, ...



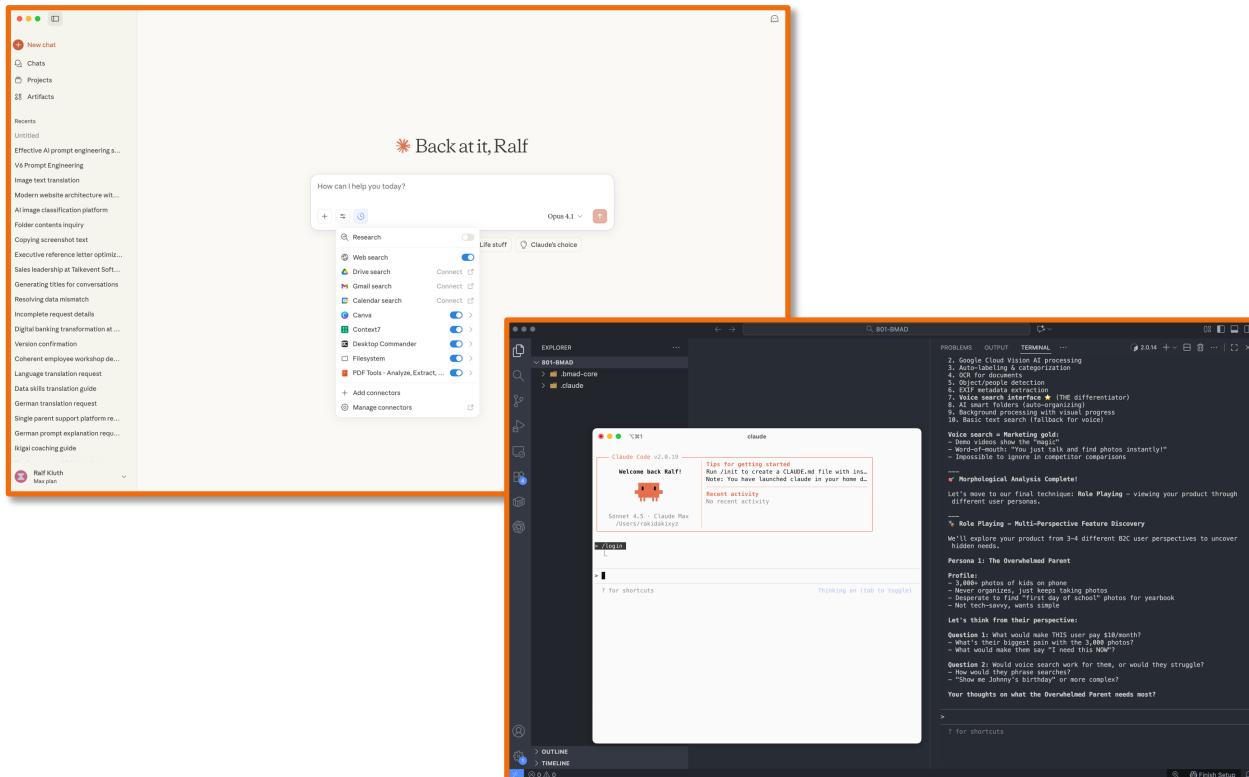
### OpenAI:

- Tools und Konnektoren
- MCP
- Apps und App Store
- Agent Builder
- ChatKit
- Guardrails
- AgentsSDK

OpenAI => komplettes Ökosystem  
(geschlossen/ offen??)  
Analogie zu Apple (iOS, App-Store, ...)

# KI-News

## Anthropic – Neue Modelle, neue Funktionen



### Anthropic – Claude, Claude Code

- Neue Sprachmodell Versionen
- Claude Code – KI-Software Entwickler
- MCP („Erfinder“ ... aber eigentlich sind es „nur“ Schnittstellen – APIs)
- Konnektoren (Gmail, Outlook, Canva, Teams, Sharepoint, ...)
- SDK für die Entwicklung von KI-Anwendungen

# KI-News

## Anthropic Skills (Capabilities)

The screenshot shows the 'Skills' section of the Anthropic platform. On the left, a sidebar menu includes 'Usage', 'Capabilities' (which is selected and highlighted in grey), 'Connectors', and 'Claude Code'. The main area is titled 'Skills' with a 'Preview' button. It contains a list of skills, each with a description, a toggle switch, and three dots for more options. The skills listed are:

- algorithmic-art: Creating algorithmic art using p5.js with seeded randomness and interactive parameter exploration. Use this when users request creating art using code, generative art, algorithmic art, flow fields, or particle systems....
- artifacts-builder: Suite of tools for creating elaborate, multi-component claudia.ai HTML artifacts using modern frontend web technologies (React, Tailwind CSS, shadcn/ui). Use for complex artifacts requiring state management, routing, ...
- brand-guidelines: Applies Anthropic's official brand colors and typography to any sort of artifact that may benefit from having Anthropic's look-and-feel. Use it when brand colors or style guidelines, visual formatting, or company design...
- canvas-design: Create beautiful visual art in .png and .pdf documents using design philosophy. You should use this skill when the user asks to create a poster, piece of art, design, or other static piece. Create original visual designs, never...
- internal-comms: A set of resources to help me write all kinds of internal communications, using the formats that my company likes to use. Claude should use this skill whenever asked to write some sort of internal communications (status...

An 'Upload skill' button is located at the top right of the skills list.

### Anthropic – Skills + Capabilities

- Anweisungen und Wissen, um Fähigkeiten für eine Aufgabe im Chat zu aktivieren, um eine bessere Antwort zu generieren
- Anstelle von Projekt- oder GPT- Instruktionen werden über ZIP-Dateien Anweisungen im Markdown-Format und Wissen in das Anthropic Account hochgeladen und diese neue „Fähigkeit“ ist dann über den Namen im Prompt nutzbar/ aktivierbar.

# KI-News

## Google

The screenshot shows the Google AI Studio interface. On the left, the sidebar includes Home, Chat (selected), Build, Dashboard, and Documentation. The main area has three tabs: Chat prompt, Chat, and Chat with models in the Playground. The Chat tab displays a list of recent chats, including "I Cannot Access Your Photos", "Prompt Engineering Research ...", "Azure Kubernetes Cluster In R...", "Private Deployment Auf Basis ...", and "BAG Bank Website Redesign S...". The Chat with models in the Playground section shows a playground interface with code snippets for generating REST APIs and monitoring usage. The Documentation section provides links to "Get API key", "View status", "Settings", and "rakidaki@from-scrat...". A large orange box highlights the Chat tab and its content.

## Google Ökosystem

- Gemini 2.5
  - TOP Sprachmodell
- Nano Banana
  - TOP KI-Bildgenerator
- Veo 3.1
  - TOP KI-Videogenerator
- Notebook LM
  - TOP Wissensmanagement

# KI-News

## OpenAI, Anthropic, Google, ...

- KI steckt in der „Kinderschuhen“, wird aber schnell „erwachsen“
  - neue Funktionen, Features, ... fast wöchentlich
- Rechenzentren, Chip-Hersteller, Algorithmen, Daten, Talente
  - Kampf um Ressourcen
- Nischenlösungen mit Spezial-KI Lösungen (Texte, Bilder, Videos, Musik, ...)
  - StartUps und etablierte Unternehmen gewinnen Kunden mit KI-Lösungen in den vertikalen und horizontalen Märkten
- Regulatorik (EU-KI-Verordnung, DSGVO, Haftungsrichtlinien, ...)
  - Microsoft Azure AI Plattform Lösungen – Hosting in DE/ EU
  - Mistral Plattform – Hosting in EU
  - Hetzner, OVH, ... diverse Cloud-Anbieter mit Komplett-Lösungen
  - Open-Source Lösungen zum “Selber-Hosten”

# Open Source Community Offline arbeiten mit lokalen LLMs

<https://ollama.com/>

Installation eines **lokalen „ChatGPT“** auf dem eigenen Rechner mit Open-Source Sprachmodellen für die „**offline**“ Nutzung. Kann auch mit OpenAI ChatGPT und anderen „großen“ Sprachmodellen „**online**“ genutzt werden.

<https://lmstudio.ai/>

<https://msty.ai/>

# Open Source Community Agenten Systeme

agent0ai / agent-zero

Code Issues 144 Pull requests 91 Discussions Actions Projects Security Insights

agent-zero Public

main Branches Tags

Add file Code About

Agent Zero AI framework

agent-zero.ai

linux agent ai assistant autonomous zero

Readme View license Activity Custom properties 12k stars 237 watching 2.3k forks Report repository

Releases 36

v0.8.6 - Memory Dashboard Latest 2 weeks ago + 35 releases

Sponsor this project

frdel Jan Tomášek Sponsor Learn more about GitHub Sponsors

<https://github.com/agent0ai/agent-zero>

The-Pocket / PocketFlow

Code Issues 51 Pull requests 4 Actions Projects Security Insights

PocketFlow Public

main Branch 1 Tag

zachary62 Merge pull request #109 from DawiniaJ/fix107 23a36bf · 2 months ago 632 Commits

cursorrules update cursor rules 3 months ago

cookbook FIX: RecursionError when loop flow 2 months ago

docs updates requirements 2 months ago

pocketflow fix: align AsycNode retry mechanism with Node implem... 3 months ago

tests update the pocketflow design 6 months ago

utils update cursor rule files 6 months ago

.cursormules update requirements 2 months ago

.gitignore Release 0.0.2 to 0.0.3 with type hints 3 months ago

LICENSE Create LICENSE 10 months ago

README.md Update README.md 3 months ago

setup.py Release 0.0.2 to 0.0.3 with type hints 3 months ago

README MIT license

About

Pocket Flow: 100-line LLM framework. Let Agents build Agents!

the-pocket.github.io/PocketFlow

workflows artificial-intelligence flow-based-programming agents ai-framework workflow-orchestration ai-frameworks flow-engineering large-language-models large-language-model llm-agent retrieval-augmented-generation llm-framework pocket-flow agent agents agent-framework agentic-workflow agentic-ai pocketflow

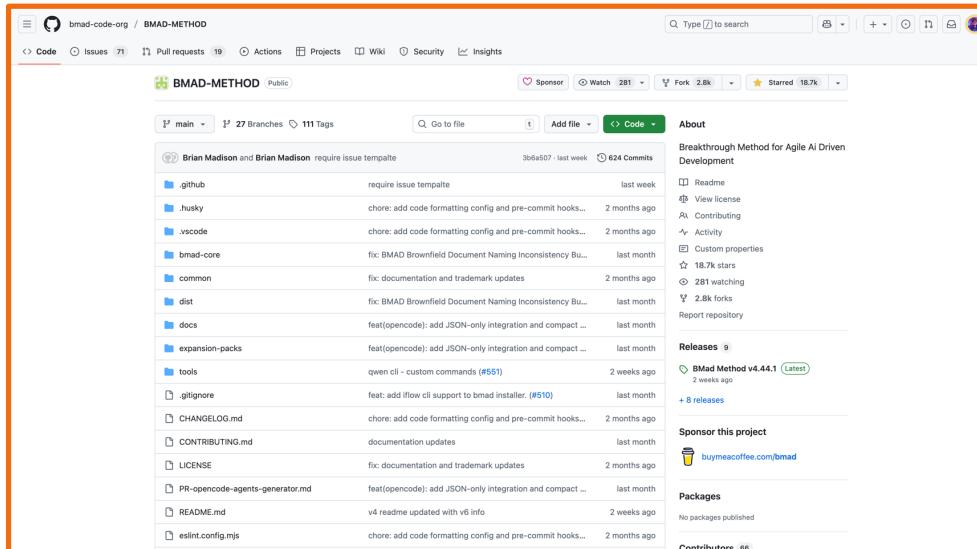
Readme MIT license Activity Custom properties 8.6k stars 94 watching 973 forks Report repository

Releases

<https://github.com/The-Pocket/PocketFlow>

„Leichtgewichtige“ Python (Programmiersprache) Frameworks mit denen man Agenten und KI-Automatisierungen „programmieren“ kann.

# Open Source Community Senior Programmierer als KI-Mitarbeiter



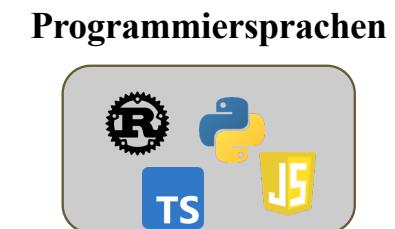
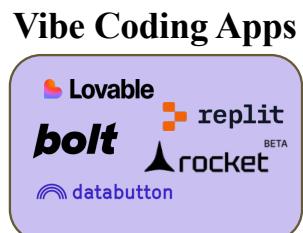
<https://github.com/bmad-code-org/BMAD-METHOD>

Agile Programmierungsmethode in Form von „Code“ als Grundlage für KI-Entwicklungstools bereitgestellt

Also per „Text- oder Spracheingabe“ wie mit einem echten Senior-Entwickler sprechen und dann:

Per Brainstorming Ideen generieren  
Ideen in einen Plan überführen  
Plan in eine IT-Architektur überführen  
Produktanforderungen für die Anwendung erstellen  
Agile Entwicklungsmethode starten  
Scrum-Master definieren  
Sprint-Entwicklung  
Automatisierte Tests der Anwendung  
Automatisierte Dokumentation der Anwendung  
...

# KI-Landkarte Überblick



## Individual KI-Entwicklung

LLM = Large Language Model => Sprachmodelle wie ChatGPT, Wrapper = Webseite

# **Praxisbausteine**

**GPTs, Projekte, MCP, Aktionen, Agent Builder, Claude Skills, N8N, ...**

# GPTs, Projekte, Skills, Agenten, Workflows

## OpenAI, Anthropic, N8N, ...

The screenshot shows the ActionsGPT interface for creating a new GPT. It includes sections for:

- Füge Aktionen hinzu**: Add actions to let your GPT perform tasks outside of ChatGPT.
- Authentifizierung**: Authentication (Keinen).
- Schema**: OpenAPI-Schema input field.
- Datenschutzrichtlinie**: Privacy policy URL (<https://app.example.com/privacy>).
- Vorschau erstellen**: Create preview.
- Modell 5**: Model selection.
- Hilfe von ActionsGPT**: Help link.
- Stelle irgendeine Frage**: Ask any question.

### OpenAI -> GPTs -> Action:

- API/ Schema Abfrage
- Erweiterung von Funktionen von ChatGPT
- Datenabfrage

# GPTs, Projekte, Skills, Agenten, Workflows

## OpenAI, Anthropic, N8N, ...

The screenshot shows two main sections. On the left, under 'Füge Aktionen hinzu', there's a schema editor with JSON code for the 'DataForSEO API'. Below it is a table of available actions:

Name	Methode	Pfad	Test
serp_organic_live_advanced	POST	/v3/serp/{search_engine}/organic/live/advanced.ai	<button>Test</button>
keywords_data_google_ads_search_volume	POST	/v3/keywords_data/google_ads/search_volume/live.ai	<button>Test</button>
keywords_data_dataforseo_trends_explore	POST	/v3/keywords_data/dataforseo_trends/explore/live.ai	<button>Test</button>
on_page_content_parsing	POST	/v3/on_page/content_parsing/live.ai	<button>Test</button>

On the right, under 'Vorschau erstellen' (Model 5), there's an AI-generated response to a question about functions:

Hi Rolf, auf welche funktionen kann ich denn mit Dir zugreifen?

Hi 🤖 — super, dass du fragst! Ich bin Rolf, dein SEO-Experte, und ich kann auf eine ganze Reihe spezialisierter SEO-, Keyword- und Backlink-Funktionen zugreifen, dank meiner Integration mit der DataForSEO API.

Hier ist ein Überblick über alles, was ich für dich tun kann 🤖

**Keyword & Content Recherche**

Ich kann dir helfen, die besten Keywords für deine Website oder Kampagne zu finden:

- 🔍 Keyword Ideas – Finde verwandte Suchbegriffe, Longtails und Nischenkeywords.
- ✖️ Keyword Difficulty – Wie schwer ist es, für ein Keyword zu ranken?
- 💰 CPC & Wettbewerb – Zeigt an, wie teuer und umkämpft ein Begriff ist.
- 📊 Keyword Trends – Zeigt, wie sich das Suchvolumen über Zeit entwickelt.
- 📍 Lokale Keyworddaten – Ich kann Suchvolumen nach Land, Stadt oder Sprache filtern.

**Domain & Wettbewerbsanalyse**

Ich kann dir helfen zu verstehen, wie deine website im Vergleich zu Mitbewerbern abschneidet:

+ Stelle irgendeine Frage

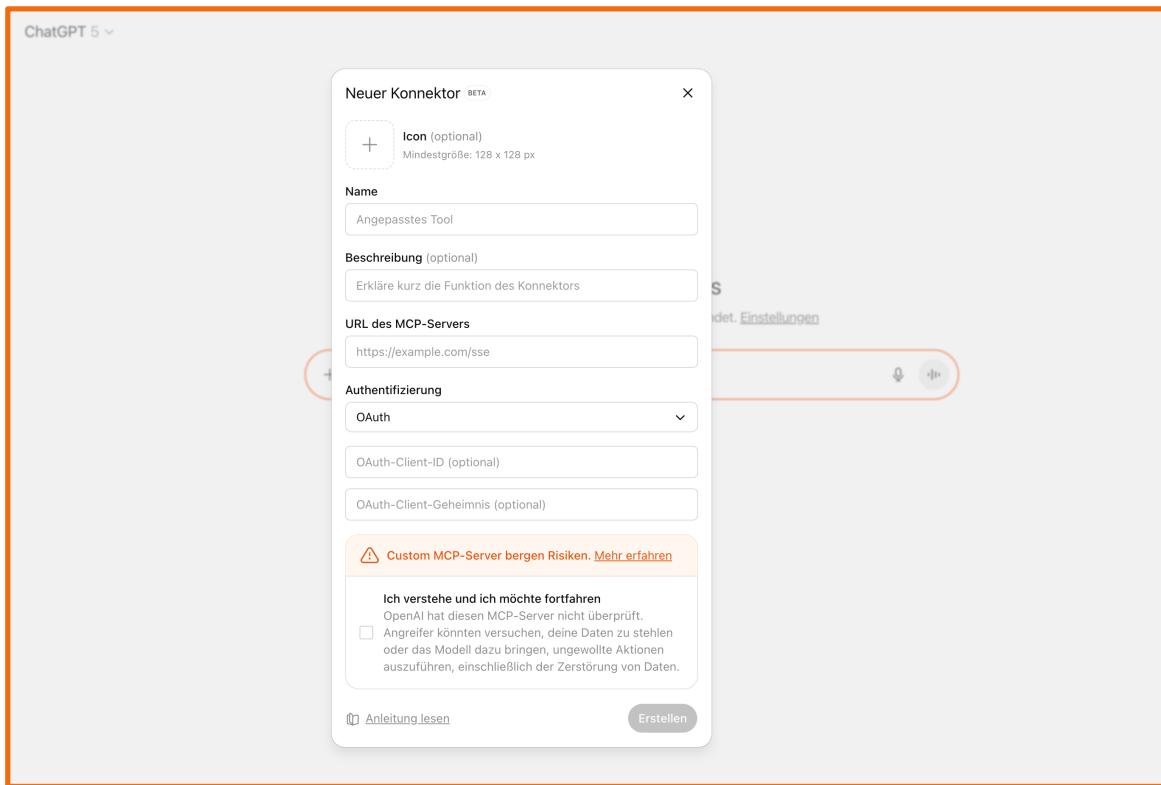
### OpenAI -> GPTs -> Action:

- API/ Schema Abfrage
- Erweiterung von Funktionen von ChatGPT
- Datenabfrage

### Beispiel – Action -> DataSEO

- SEO-Funktionen von DataSEO über Actions (API) bereitgestellt

# GPTs, Projekte, Skills, Agenten, Workflows OpenAI, Anthropic, N8N, ...



## OpenAI -> Konnektoren -> MCP

- Verbindung zu APIs, Tools, ...
- Hier ein Beispiel, um über Klavis (MCP-Marktplatz) mit OpenRouter zu arbeiten (um weitere LLMs einzubinden)

# GPTs, Projekte, Skills, Agenten, Workflows

## OpenAI, Anthropic, N8N, ...

The screenshot shows the Klavis interface. On the left is a sidebar with navigation links: Allgemein, Benachrichtigungen, Personalisierung, Apps und Konnektoren (highlighted), Zeitpläne, Datenkontrollen, Sicherheit, Kindersicherung, and Konto. The main area displays a connector named "Klavis->OpenRouter->LLMs MCP" (DEV). It includes a "Trennen" button and an ellipsis "...". Below this is an "Info" section showing "Verbunden am 11. Nov. 2025" and a URL "https://openrouter-mcp-server.klavis.ai/mcp/?instance\_i". Under "Aktionen", there are four items: "openrouter\_analyze\_model\_performance" (Analyze the performance of a single model across multiple test prompts), "openrouter\_compare\_models" (Compare multiple models by running the same prompt through each), "openrouter\_create\_chat\_completion" (Create a chat completion using OpenRouter, with a "SCHREIBEN" button), and "openrouter\_create\_chat\_completion\_stream" (Create a streaming chat completion using OpenRouter, with a "SCHREIBEN" button).

### OpenAI -> Konnektoren -> MCP

- Verbindung zu APIs, Tools, ...

#### Beispiel OpenRouter

- MCP Server Openrouter ermöglicht die Nutzung von z.B. Anthropic Claude aus ChatGPT heraus

# GPTs, Projekte, Skills, Agenten, Workflows

## OpenAI, Anthropic, N8N, ...

ChatGPT 5 ✓

Welche Sprachmodelle kann ich mit Openrouter nutzen

Gespräch mit connector beendet ✓

( ⓘ) Tool aufgerufen ✓

Kurzfassung: Mit OpenRouter kannst du eine riesige Auswahl an Modellen verschiedenster Anbieter nutzen – von „Frontier“-LLMs über günstige Schnellläufer bis hin zu Coding-, Vision- und Research-Spezialisten. Hier ein übersichtlicher Überblick (Auszug der wichtigsten Familien):

**Top-Anbieter & Modelfamilien**

- **OpenAI**
  - GPT-5 (Pro, Mini, Image), GPT-4.1 (Chat/Mini/Nano), GPT-4o & 4o-mini, o-Serie (o3, o4-mini, o1), GPT-4 Turbo, GPT-3.5
- **Anthropic**
  - Claude 4 / 4.1 Opus, Claude 4 Sonnet, Claude 4.5 Sonnet, Claude 3.7 Sonnet, Claude 3.5 Haiku
- **Google**
  - Gemini 2.5 Pro / Flash / Flash-Lite, Gemini 2.0 Flash/Flash-Lite, Gemma 3 (4B/12B/27B), Gemma 2 (9B/27B)
- **Meta**
  - Llama 4 Maverick, Llama 4 Scout, Llama 3.1 (8B/70B/405B), Llama 3.2 (Vision 11B/90B, 1B/3B)
- **Mistral**
  - Mistral Large 2, Mistral Medium/Small, Devstral (Code), Codestral, Mixtral 8x22B, Mistral/Mistral (Edge)
- **DeepSeek**
  - R1 (Reasoning), R1-0528, V3.1 (Terminus), R1 Distill (Qwen/Llama)
- **Qwen (Alibaba)**
  - Qwen3 (Max/Plus/Turbo, 8B/14B/30B/32.35B), Qwen3-VL (Vision 8B/30B/235B), Qwen3-Coder / Coder-Plus / Flash

ENTWICKLERMODUS

| Stelle irgendeine Frage

+

K Klavis->OpenRouter->...

OpenAI -> Konnektoren -> MCP

- Klavis MCP Marktplatz -> OpenRouter

**Mehrwert:**

  - Zugriff auf andere Sprachmodelle aus OpenAI ChatGPT heraus

The screenshot shows a tool card from the ChatGPT interface. The title is '(5) ChatGPT möchte mit call\_tool sprechen'. The card contains the following information:

- Klavis->OpenRouter->LLMs MCP**
- Allow creation of 5 customer complaint reply emails?**
- Description: Allow creation of 5 professional, empathetic, solution-focused German reply emails addressing customer complaints like late delivery, defective goods, unfriendly support, wrong or incomplete items. [Details](#)
- Action buttons: **Create** (black button), **Verweigern** (white button)

At the bottom, there is a footer note: Die Verwendung von Tools ist mit Risiken verbunden. Mehr erfahren.

# GPTs, Projekte, Skills, Agenten, Workflows

## OpenAI, Anthropic, N8N, ...

The screenshot shows the Claude AI interface with a sidebar on the left containing navigation links: Allgemein, Konto, Datenschutz, Abrechnung, Nutzung, **Fähigkeiten** (which is highlighted with a blue border), Konnektoren, and Claude Code. The main area is titled 'Skills' with a 'Vorschau' button and a 'Fähigkeit hochladen' button. It lists several skills:

- bwa-voba-rk**: Automatisierte Betriebswirtschaftliche Auswertung (BWA) für Geschäftsberichte der VR-Bank. Verarbeitet PDF-Bilanzen und Jahresabschlüsse, extrahiert Kennzahlen, führt Branchenvergleiche durch und erstellt...  
Von dir hinzugefügt - vor 4 Tagen. Status: Off
- audio-meeting-analyzer**: Transcribe audio recordings and generate comprehensive meeting analysis including protocols, summaries, action items, and meeting quality evaluation. Use when user uploads audio files (MP3, WAV, M4A), mentions...  
Von dir hinzugefügt - vor 20 Tagen. Status: Off
- algorithmic-art**: Creating algorithmic art using p5.js with seeded randomness and interactive parameter exploration. Use this when users request creating art using code, generative art, algorithmic art, flow fields, or particle systems....  
Anthropic. Status: On
- artifacts-builder**: Suite of tools for creating elaborate, multi-component claudia.ai HTML artifacts using modern frontend web technologies (React, Tailwind CSS, shadcn/ui). Use for complex artifacts requiring state management, routing, ...  
Anthropic. Status: On
- brand-guidelines**: Applies Anthropic's official brand colors and typography to any sort of artifact that may benefit from having Anthropic's look-and-feel. Use it when brand colors or style guidelines, visual formatting, or company design...  
Anthropic. Status: On
- canvas-design**: Create beautiful visual art in .png and .pdf documents using design philosophy. You should use this skill when the user asks to create a poster, piece of art, design, or other static piece. Create original visual designs, never...  
Anthropic. Status: On

### Anthropic Claude -> Skills:

- Erweiterungen des Funktionsumfangs eines Sprachmodells
- Zusätzliche „Fähigkeiten“ hinzufügen
- Skills können über natürliche Sprache „konfiguriert“ und erstellt werden

# GPTs, Projekte, Skills, Agenten, Workflows

## OpenAI, Anthropic, N8N, ...

Einstellungen

Allgemein  
Konto  
Datenschutz  
Abrechnung  
Nutzung  
Fähigkeiten  
**Konnektoren**  
Claude Code

**Konnektoren**  
Erlauben Sie Claude, andere Apps und Dienste für mehr Kontext zu referenzieren.

Google Drive	<button>Verbunden</button> <button>...</button>
GitHub	<button>Verbunden</button> <button>...</button>
Canva	<button>Konfigurieren</button> <button>...</button>
Gmail Getrennt	<button>Verbinden</button> <button>...</button>
Google Calendar Getrennt	<button>Verbinden</button> <button>...</button>
Infranodus <small>BENUTZERDEFINIERT</small> https://server.smithery.ai/@infranodus/mcp-serv...	<button>Konfigurieren</button> <button>...</button>

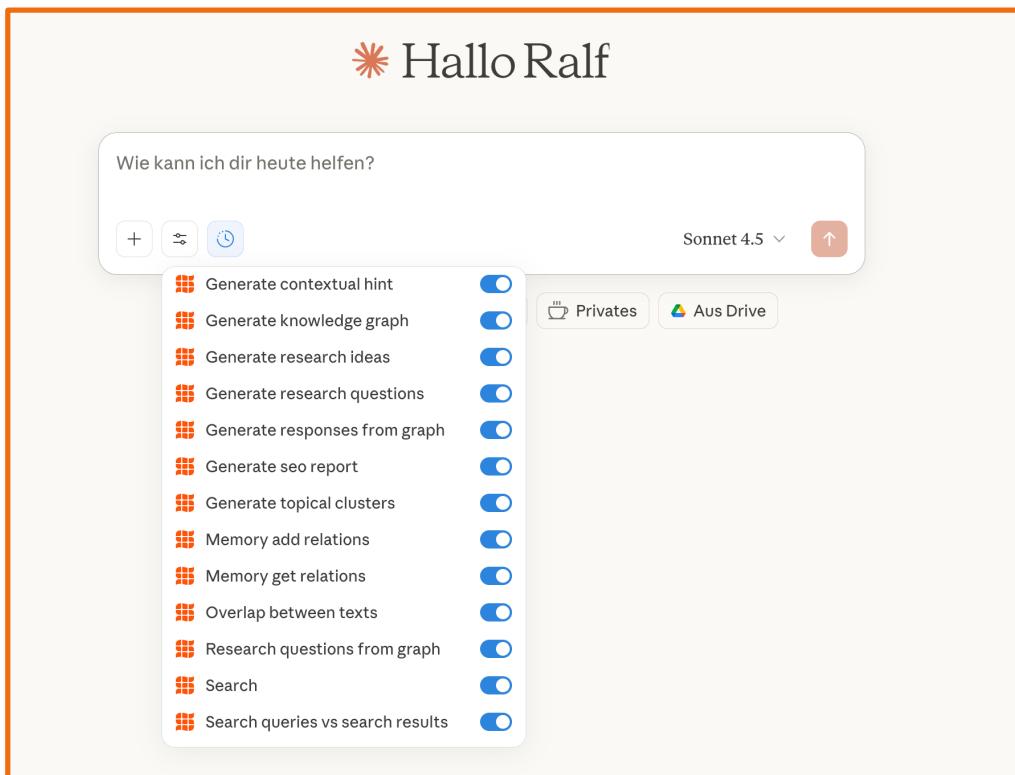
[Connectors durchsuchen](#) [Benutzerdefinierten Connector hinzufügen](#)

### Anthropic Claude -> Konnektoren (MCPs):

- Verbindung (APIs) zu anderen Services u.a. Tools, Daten, ...
- Hier das Beispiel Infranodus (Graph-Wissensmanagement-Technologien)

# GPTs, Projekte, Skills, Agenten, Workflows

## OpenAI, Anthropic, N8N, ...

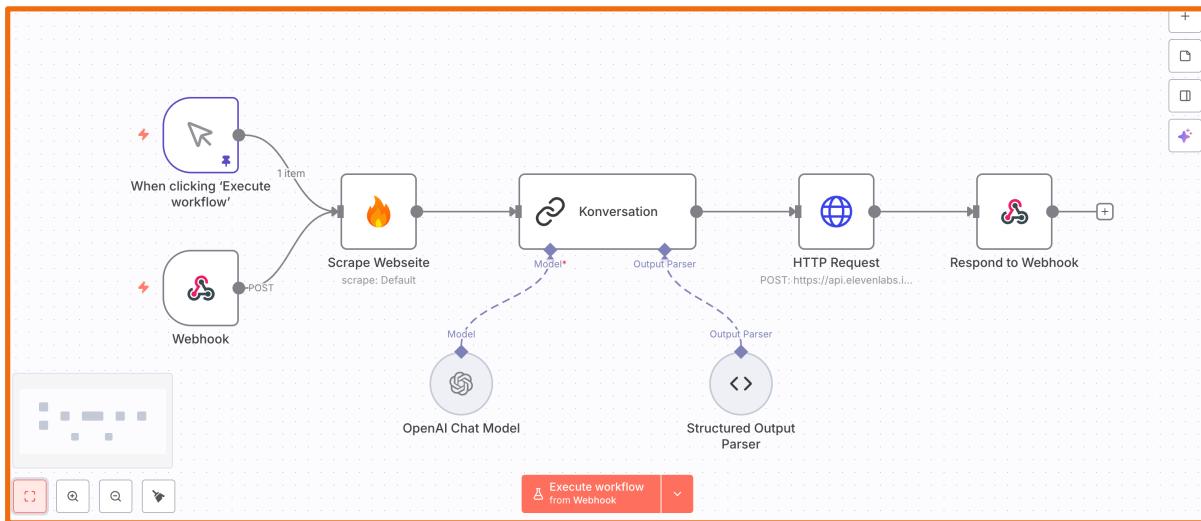


### Claude -> MCP Infranodus -> Neue „Werkzeuge“

- MCP-Konnektoren ermöglichen den Zugriff über API Keys auf Daten und Werkzeuge
- Mit den zuvor genannten Skills kann man damit „komplexe“ Agenten erstellen.

# GPTs, Projekte, Skills, Agenten, Workflows

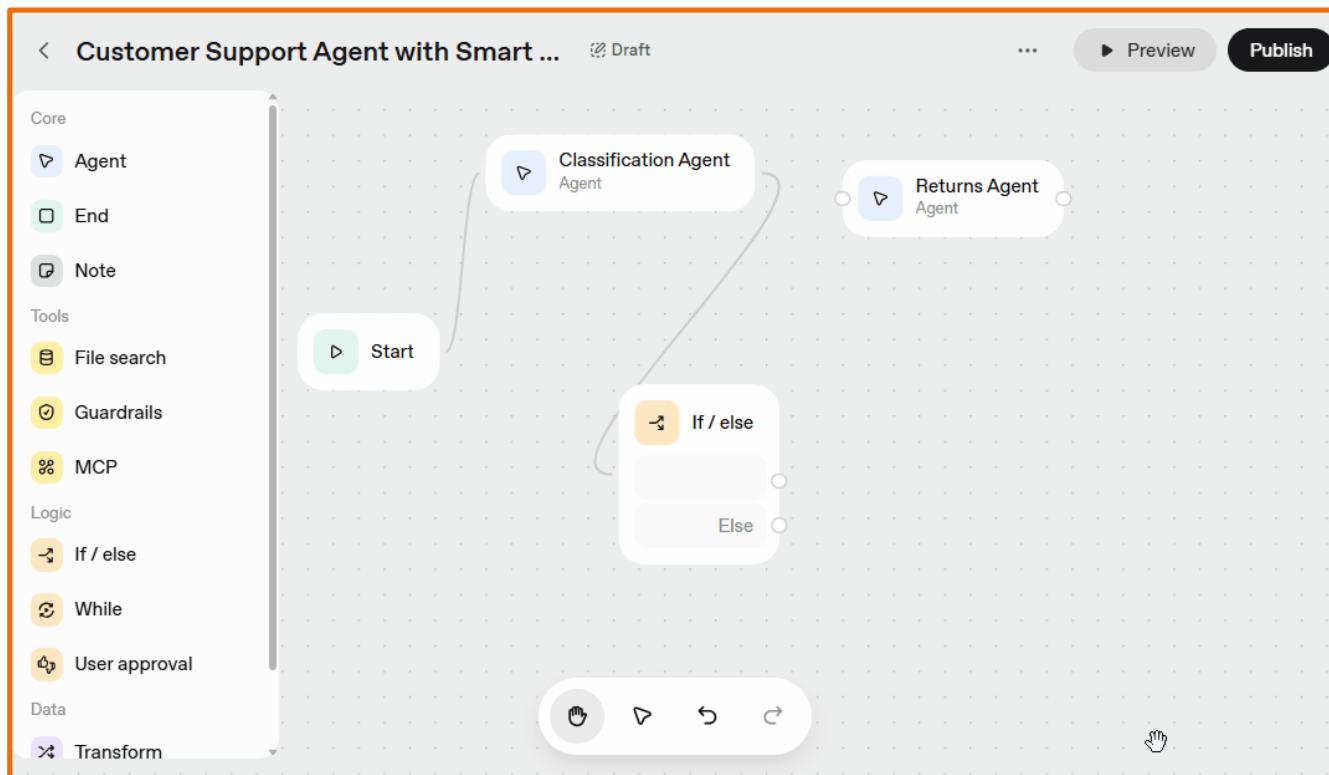
## OpenAI, Anthropic, N8N, ...



### N8N – Automatisierungsplattform:

- Klassische Automatisierungsplattform
- Integration von KI-Nodes
- Integration von APIs von Drittanbietern (Elevenlabs, Firecrawl, ...)

# GPTs, Projekte, Skills, Agenten, Workflows OpenAI, Anthropic, N8N, ...



## OpenAI -> Agent Builder & ChatKit & ...

- Agent Builder
- ChatKit
- Guardrails
- AgentsSDK

OpenAI => komplettes Ökosystem  
(geschlossen/ offen??)  
Analogie zu Apple (iOS, App-Store, ...)

# **Agenten (Node, Graph, Flow), ...**

**Ein paar Grundlagen**

# LLM-Framework mit 100 Zeilen Python Code

## Ein paar Begriffe am Anfang klären

- **LLM** steht für **Large Language Model**, also ein großes Sprachmodell wie ChatGPT
- Neben **ChatGPT** von **OpenAI** gibt es zahlreiche andere **Sprachmodelle**: Claude, Gemini, Grok, DeepSeek, Kimi, ... (>1 Mio. Sprachmodelle siehe u.a. HuggingFace Plattform)
- Für die **Automatisierung** von digitalen Prozessen gibt es Frameworks wie **N8N**, **Zapier**, **Make** und viele mehr, die auch Sprachmodelle integrieren – über **Nodes**
- Ein **Node** (Knoten) in der Automatisierung ist ein Element, dass eine bestimmte Aufgabe erfüllt und könnte z.B. ein Sprachmodell sein
- Ein **Graph** ist ein Prozessablauf, wie z.B. der Prozess für einen Kreditantrag, der in einer Prozessbeschreibung, Verfahrensanweisung, ... festgelegt ist
- Ein **Flow** ist der konkrete Durchlauf eines Prozesses für den jeweiligen Input (Kunde, Trigger, ...)

# LLM-Framework mit 100 Zeilen Python Code

## 🤔 Node: ein einzelner Bearbeitungsschritt

- Ein **Node** ist wie ein Schreibtisch/Abteilung in deiner Volksbank, wo EINE bestimmte Aufgabe erledigt wird:

- 👤 Node 1: "Kundenberatung"  
→ Aufgabe: Kundenwunsch erfassen (Kredithöhe, Zweck)
- 📊 Node 2: "SCHUFA-Prüfung"  
→ Aufgabe: Bonität checken, SCHUFA-Score abrufen
- 💰 Node 3: "Einkommensprüfung"  
→ Aufgabe: Gehaltsnachweise prüfen, Haushaltsrechnung
- ✓ Node 4: "Kreditentscheidung"  
→ Aufgabe: Bewilligen oder Ablehnen basierend auf Prüfungen
- ↳ Node 5: "Vertragsabteilung"  
→ Aufgabe: Kreditvertrag erstellen und versenden

- **Wichtig:** Jeder **Node** ist wie ein **Spezialist** - macht **NUR** seinen Job perfekt!

# LLM-Framework mit 100 Zeilen Python Code

## 💡 Graph: der komplette Kreditprozess

- Ein **Graph** ist der Prozessablauf, wie er in eurer Verfahrensanweisung steht:

[Kundenberatung] → [SCHUFA-Check] → [Einkommensprüfung]



[Kreditentscheidung]



[Genehmigt]



[Abgelehnt]



[Vertrag]



[Alternativvorschlag]

- Der Graph ist wie ein **internes Prozesshandbuch** - zeigt alle Stationen und Entscheidungswege.

# LLM-Framework mit 100 Zeilen Python Code

## 💡 Flow: ein konkreter “Kunde“ durchläuft den Prozess

- Der **Flow** ist, wenn ein Kunde „Herr Müller“ WIRKLICH für seinen Autokredit durch alle Abteilungen läuft:

```
START: Herr Müller möchte 25.000€ für einen VW Golf
↓
1. Beratungsgespräch: 14:00 Uhr, Büro 3 (Node 1 wird ausgeführt)
↓
2. SCHUFA-Abfrage: Score 95, sehr gut! (Node 2 wird ausgeführt)
↓
3. Gehaltsprüfung: 3.500€ netto, passt! (Node 3 wird ausgeführt)
↓
4. Entscheidung: GENEHMIGT (Node 4 wird ausgeführt)
↓
5. Vertrag wird erstellt (Node 5 wird ausgeführt)
↓
ENDE: Herr Müller unterschreibt, Auto kann gekauft werden! 🚗
```

# LLM-Framework mit 100 Zeilen Python Code

🤔 Wie sieht das als Code jetzt aus?

- Der Kreditantragsprozess als KI-System:

```
# Die NODES (Bearbeitungsstationen)
class KundenberatungNode(Node):
    def exec(self, kundenantrag):
        # Wie dein Beratungstool
        return {
            "kunde": "Herr Müller",
            "betrag": 25000,
            "zweck": "Autokredit"
        }

class SchufaPruefungNode(Node):
    def exec(self, kundendaten):
        # Wie eure SCHUFA-Schnittstelle
        schufa_score = schufa_api.check(kundendaten["kunde"])
        return {"score": 95, "bewertung": "sehr gut"}

class EinkommenspruefungNode(Node):
    def exec(self, antrag_mit_schufa):
        # Wie euer Haushaltsrechner
        return {
            "netto": 3500,
            "verfuegbar": 800,
            "rate_moeglich": 450
        }
```

```
# Der GRAPH (euer Kreditprozess)
kredit_prozess = Flow(
    nodes=[
        KundenberatungNode(),      # Station 1
        SchufaPruefungNode(),      # Station 2
        EinkommenspruefungNode(),   # Station 3
        KreditentscheidungNode(),   # Station 4
        VertragsNode()             # Station 5
    ]
)

# Der FLOW (Herr Müller's Antrag läuft durch)
ergebnis = kredit_prozess.run({"kunde": "Herr Müller", "wunsch": "25000€"})
# Ausgabe: "Kreditvertrag erstellt - GENEHMIGT"
```

.... weitere Nodes

# LLM-Framework mit 100 Zeilen Python Code

🤔 Zusammengefasst – Kreditantrag

Begriff	Volksbank-Beispiel	KI-System Entsprechung
<b>Node</b> 📁	Ein Bearbeitungsschritt (z.B. SCHUFA-Prüfung)	Ein Code-Baustein für genau diese Prüfung
<b>Graph</b> 🌐	Eure Kreditrichtlinie/Prozesshandbuch	Alle Prüfschritte und deren Reihenfolge
<b>Flow</b> 🚶	Ein konkreter Kunde durchläuft den Prozess	Computer arbeitet einen echten Antrag ab

# LLM-Framework mit 100 Zeilen Python Code

🤔 Warum ist das so spannend?

- Compliance konforme Lösung

```
# Jeder Node protokolliert automatisch für die BaFin
class SchufaNode(Node):
    def post(self, shared, prep_res, exec_res):
        # Automatische Dokumentation für Prüfungspfad
        shared["pruefprotokoll"].append({
            "zeitstempel": datetime.now(),
            "pruefung": "SCHUFA",
            "ergebnis": exec_res
        })
```

# LLM-Framework mit 100 Zeilen Python Code

🤔 Warum ist das so spannend?

- Verschiedene Kreditarten – gleiche Nodes

```
# Baufinanzierung? Nutze die gleichen Bausteine!
baufi_prozess = Flow([
    KundenberatungNode(),
    SchufaPruefungNode(),
    EinkommenspruefungNode(),
    GrundbuchNode(),          # NEU: Nur bei Baufi
    KreditentscheidungNode(),
    VertragsNode()
])

# Dispokredit? Vereinfachter Prozess!
dispo_prozess = Flow([
    SchufaPruefungNode(),    # Weniger Nodes!
    KreditentscheidungNode()
])
```

# LLM-Framework mit 100 Zeilen Python Code

🤔 Warum ist das so spannend?

- Risikomanagement eingebaut

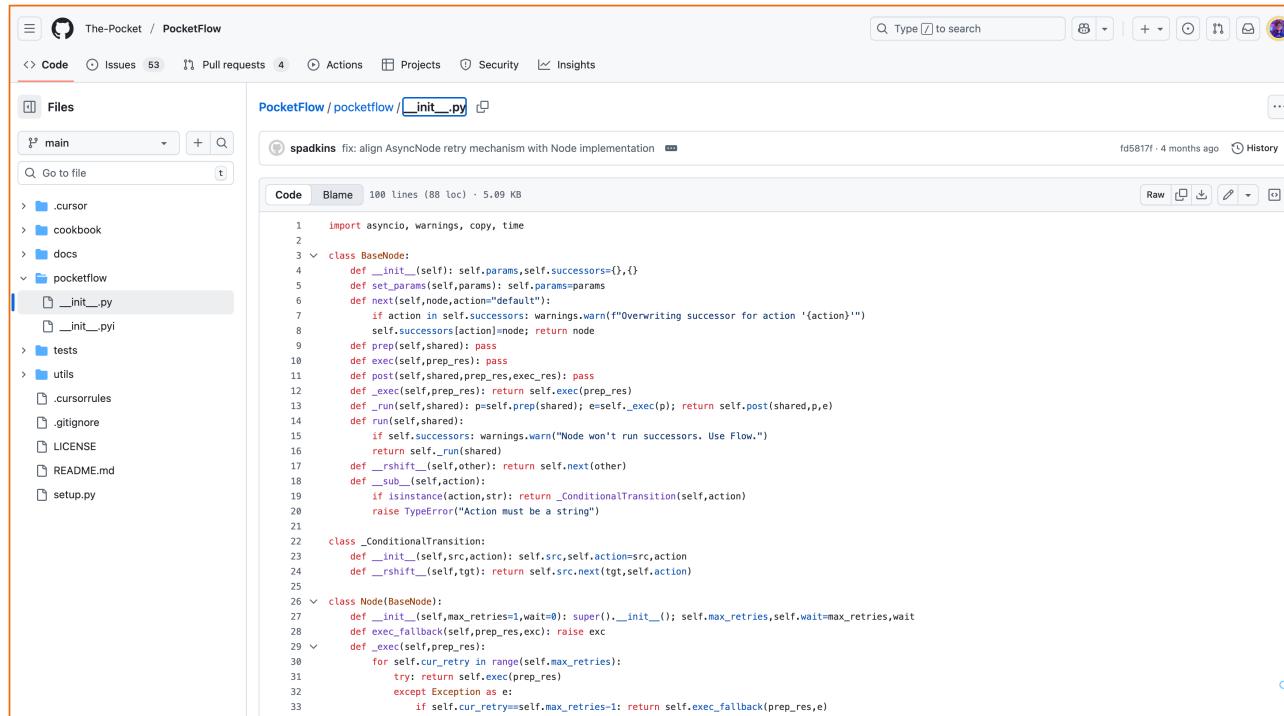
```
class RisikoNode(Node):  
    def post(self, shared, prep_res, exec_res):  
        if exec_res["score"] < 50:  
            return "zur_filialleitung" # → Automatische Eskalation  
        else:  
            return "standard_prozess" # → Normal weiter
```

# Programmierung mit KI

**100 Zeilen in Python – kurzer Blick auf die Elemente und den Code**

# LLM-Framework mit 100 Zeilen Python Code

## Nodes



The screenshot shows a GitHub repository page for 'The-Pocket / PocketFlow'. The main file, `init.py`, is displayed. The code is a compact LLM framework consisting of approximately 100 lines of Python. It includes classes for nodes like `BaseNode` and `Node`, and methods for operations like `exec` and `post`. A commit by spadkins is visible, fixing an issue related to AsyncNode retry mechanism.

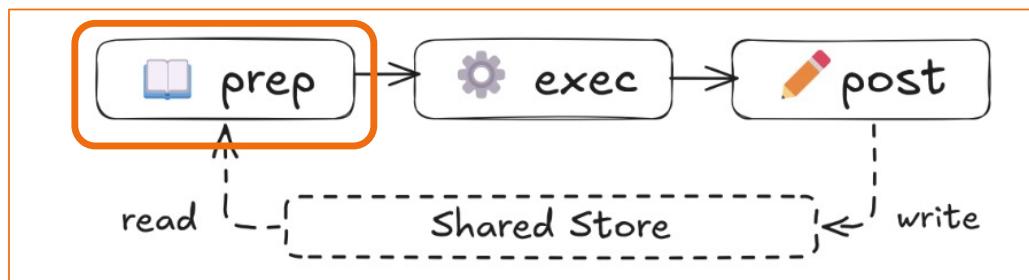
```
import asyncio, warnings, copy, time
...
class BaseNode:
    def __init__(self):
        self.params, self.successors = {}, {}
    def set_params(self, params):
        self.params = params
    def next(self, node, action="default"):
        if action in self.successors:
            warnings.warn(f"Overwriting successor for action '{action}'")
        self.successors[action] = node
    def prep(self, shared):
        pass
    def exec(self, shared):
        pass
    def post(self, shared, prep_res):
        pass
    def _exec(self, prep_res):
        return self.exec(prep_res)
    def _run(self, shared):
        p = self.prep(shared)
        e = self._exec(p)
        self.post(shared, e)
    def run(self, shared):
        if self.successors:
            warnings.warn("Node won't run successors. Use Flow.")
        return self._run(shared)
    def __rshift__(self, other):
        return self.next(other)
    def __sub__(self, action):
        if isinstance(action, str):
            return _ConditionalTransition(self, action)
        raise TypeError("Action must be a string")
    class _ConditionalTransition:
        def __init__(self, src, action):
            self.src, self.action = src, action
        def __rshift__(self, tgt):
            return self.src.next(tgt, self.action)
    class Node(BaseNode):
        def __init__(self, max_retries=3, wait=0):
            super().__init__()
            self.max_retries, self.wait = max_retries, wait
        def exec_fallback(self, prep_res, exc):
            raise exc
        def _exec(self, prep_res):
            for self.cur_retry in range(self.max_retries):
                try:
                    return self.exec(prep_res)
                except Exception as e:
                    if self.cur_retry == self.max_retries - 1:
                        return self.exec_fallback(prep_res, e)
                    else:
                        self.wait += 1000 * (14 ** self.cur_retry)
```

<https://github.com/The-Pocket/PocketFlow/tree/main>

# LLM-Framework mit 100 Zeilen Python Code

## 💡 Nodes

Ein **Node** ist der kleinste Baustein. Jeder Node hat 3 Schritte **prep->exec->post**:



### 1. prep(shared)

Lese und bereite Daten vor aus dem gemeinsamen Speicher.

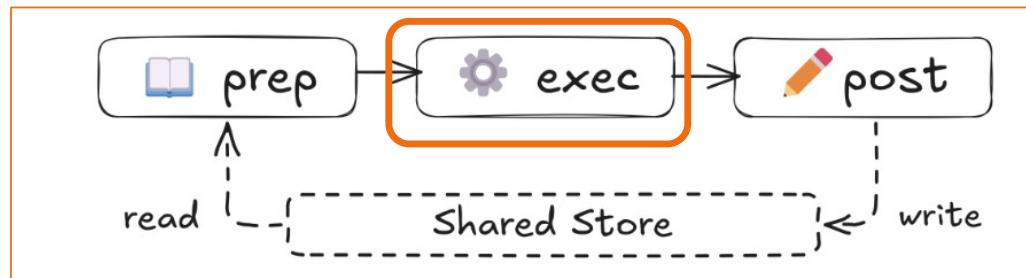
Beispiele: Datenbank abfragen, Dateien lesen oder Daten in einen String serialisieren.

Gibt `prep_res` zurück, welches von `exec()` und `post()` verwendet wird.

# LLM-Framework mit 100 Zeilen Python Code

## 💡 Nodes

Ein Node ist der kleinste Baustein. Jeder Node hat 3 Schritte prep->exec->post:



### 2. exec(prep\_res)

Führe Berechnungslogik aus, mit optionalen Wiederholungen und Fehlerbehandlung.

Beispiele: (hauptsächlich) LLM-Aufrufe, externe APIs, Tool-Nutzung (MCP, ...)

⚠ Dies soll NUR für Berechnungen sein und greift NICHT auf shared Store zu.

⚠ Falls Wiederholungen aktiviert sind, stelle idempotente Implementierung sicher.

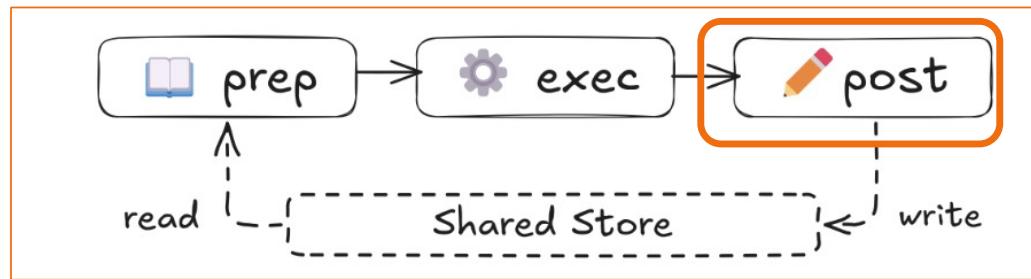
⚠ Überlasse Fehlerbehandlung dem eingebauten Wiederholungsmechanismus des Nodes.

Gibt exec\_res zurück, welches an post() weitergegeben wird.

# LLM-Framework mit 100 Zeilen Python Code

## 🧠 Nodes

Ein Node ist der kleinste Baustein. Jeder Node hat 3 Schritte prep->exec->post:



### 3. post(shared, prep\_res, exec\_res)

Nachbearbeitung und schreibe Daten zurück zu shared.

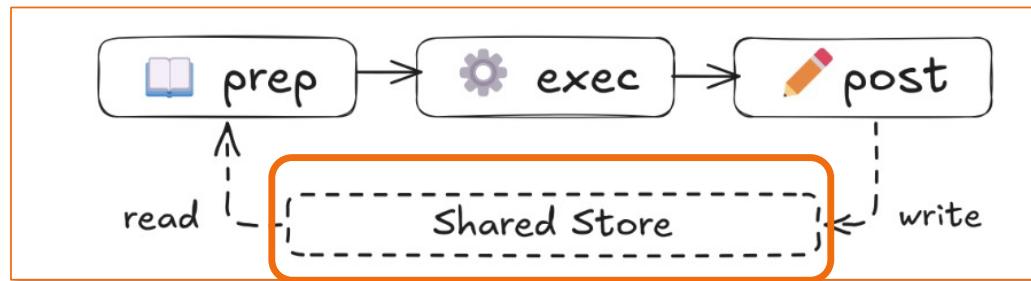
Beispiele: Datenbank aktualisieren, Status ändern, Ergebnisse protokollieren.

Entscheide die nächste Aktion durch Rückgabe eines string (action = "default" falls None).

# LLM-Framework mit 100 Zeilen Python Code

## 💡 Nodes

Ein Node ist der kleinste Baustein. Jeder Node hat 3 Schritte prep->exec->post:



### Shared Store (Gemeinsamer Speicher)

Die gestrichelte Box in der Mitte des Diagramms

**prep** liest aus dem gemeinsamen Speicher

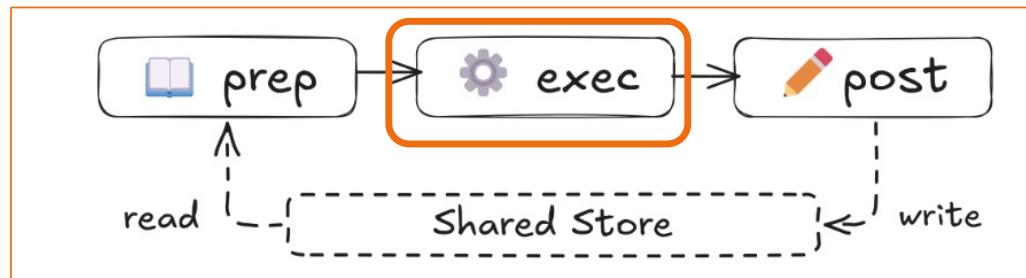
**post** schreibt in den gemeinsamen Speicher

**exec** hat **KEINEN** direkten Zugriff

# LLM-Framework mit 100 Zeilen Python Code

## 💡 Nodes

Ein Node ist der kleinste Baustein. Jeder Node hat 3 Schritte prep->exec->post:



Du kannst exec() wiederholen lassen, wenn es eine Ausnahme (Exception) auslöst, über zwei Parameter beim Definieren des Nodes:

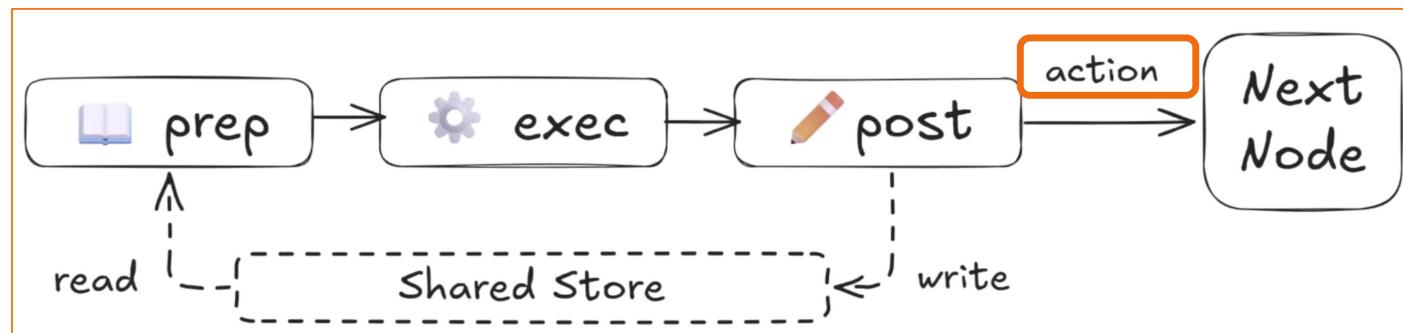
**max\_retries** (Ganzzahl): Maximale Anzahl, wie oft exec() ausgeführt wird. Der Standard ist 1 (keine Wiederholung).

**wait** (Ganzzahl): Die Wartezeit (in Sekunden) vor dem nächsten Versuch. Standardmäßig ist wait=0 (keine Wartezeit). wait ist hilfreich, wenn du auf Ratenlimits oder Kontingentfehler deines LLM-Anbieters stößt und eine Pause einlegen musst.

# LLM-Framework mit 100 Zeilen Python Code

🤔 Flows – Node A >> Node B >> Node C >>...

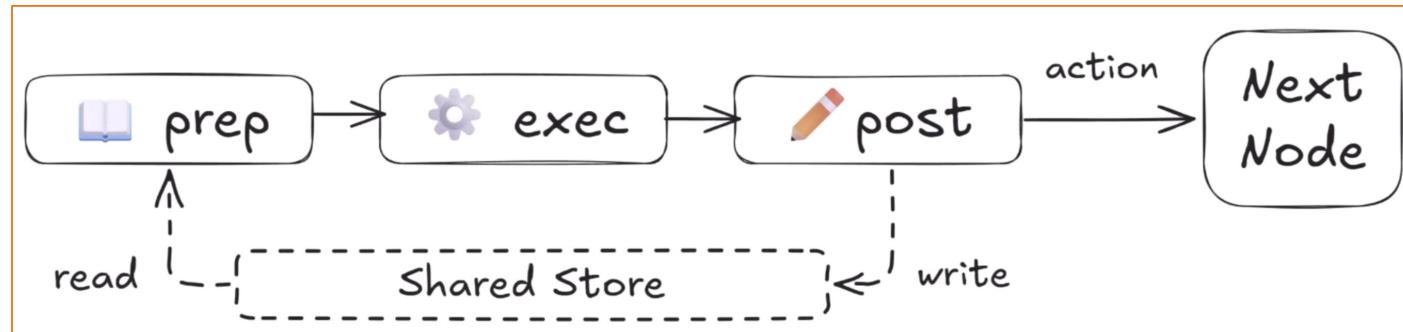
Wie funktioniert die Überleitung zum nächsten Node: per action im post Ergebnis des vorhergehenden Nodes:



# LLM-Framework mit 100 Zeilen Python Code

💡 Flows: Node A >> Node B >> ...

Ein Flow orchestriert einen Graphen von Nodes. Du kannst Nodes in einer Sequenz verketten oder Verzweigungen erstellen, abhängig von den Actions, die von jedem Node's post() zurückgegeben werden.



## 1. Action-basierte Übergänge

Jeder Node's post() gibt einen Action-String zurück. Standardmäßig, wenn post() nichts zurückgibt, behandeln wir das als "default".

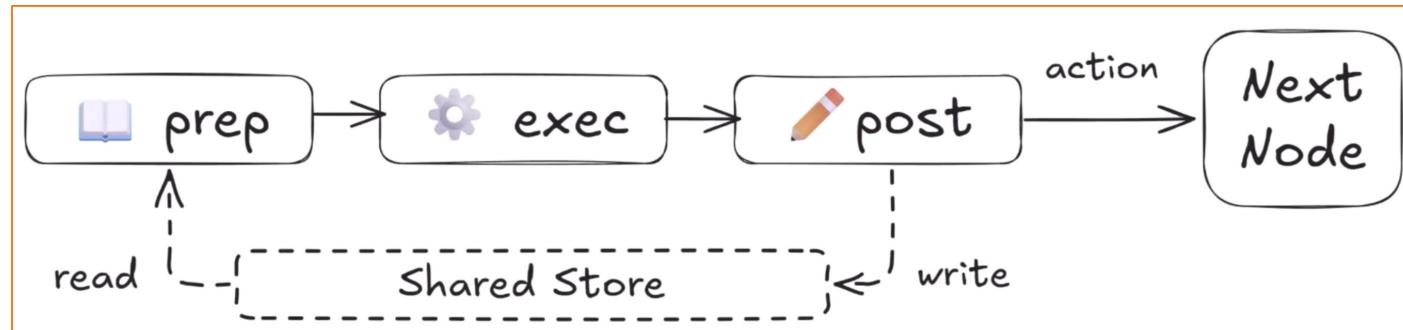
Du definierst Übergänge mit folgender Syntax:

**1. Einfacher Standard-Übergang:** node\_a >> node\_b Das bedeutet: Wenn node\_a.post() "default" zurückgibt, gehe zu node\_b. (Äquivalent zu node\_a - "default" >> node\_b)

# LLM-Framework mit 100 Zeilen Python Code

💡 Flows: Node A >> Node B >> ...

Ein Flow orchestriert einen Graphen von Nodes. Du kannst Nodes in einer Sequenz verketten oder Verzweigungen erstellen, abhängig von den Actions, die von jedem Node's post() zurückgegeben werden.

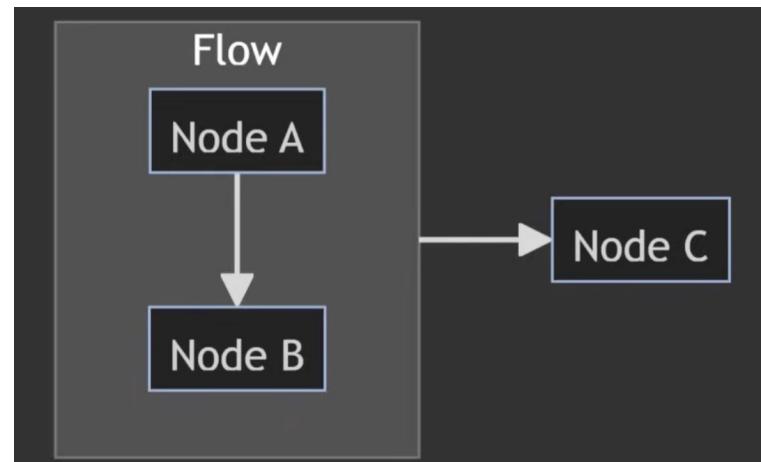


**2. Benannter Action-Übergang:** node\_a - "action\_name" >> node\_b Das bedeutet: Wenn node\_a.post() "action\_name" zurückgibt, gehe zu node\_b.  
Es ist möglich, Schleifen, Verzweigungen oder mehrstufige Flows zu erstellen.

# LLM-Framework mit 100 Zeilen Python Code

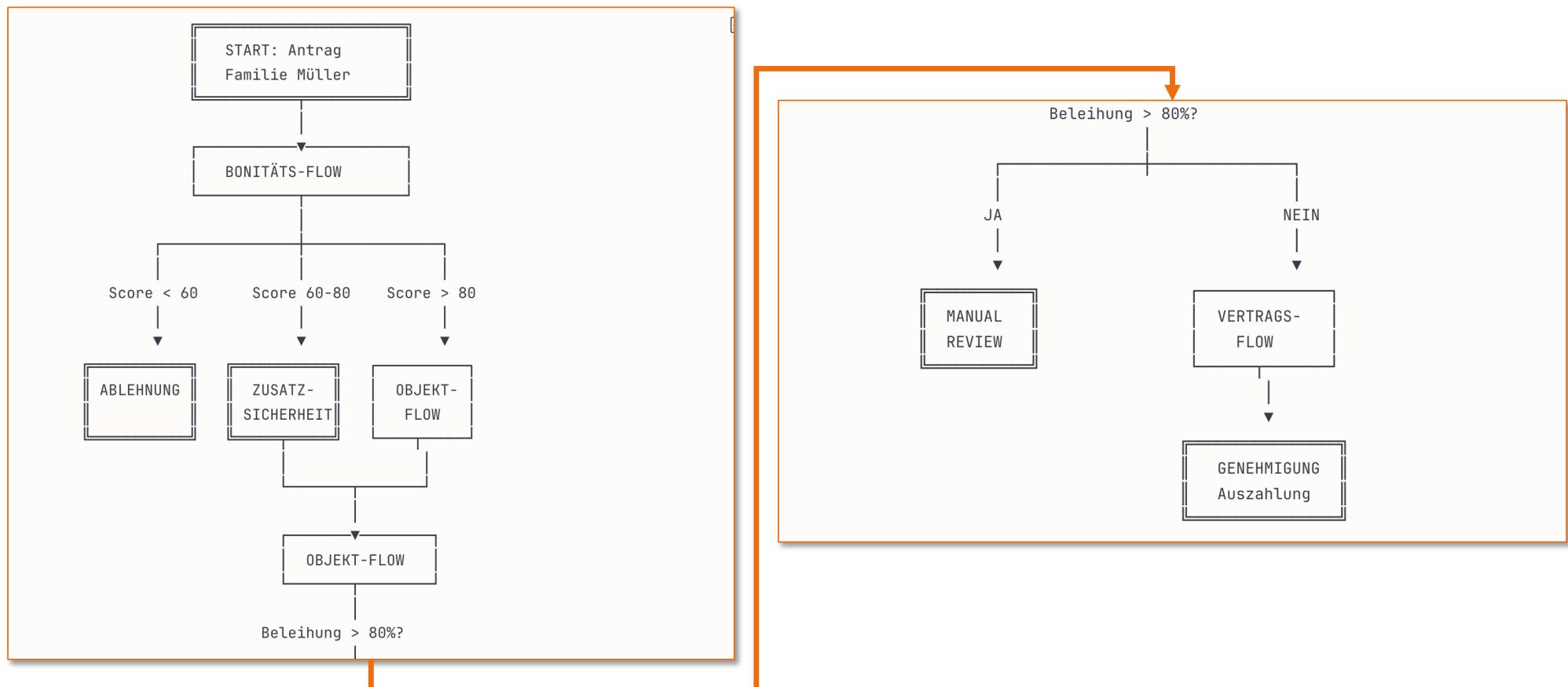
🤔 Flows = Nodes?

Ein Flow wie hier z.B. Node A >> Node B kann wiederum zu einem Node definiert werden, so dass komplexe Workflows/ Prozesse/ Vorgänge umgesetzt werden können



# LLM-Framework mit 100 Zeilen Python Code

🤔 Beispiel für einen komplexen Flow



# LLM-Framework mit 100 Zeilen Python Code

 Beispiel für einen komplexen Flow

Flow	Node	Aufgabe	Output
 <b>Bonitäts-Flow</b>	SCHUFA-Check	Bonitätsprüfung	Score 0-100
	Einkommens-Prüfung	Haushaltsrechnung	Verfügbares Einkommen
	Eigenkapital-Check	Vermögensprüfung	EK-Quote
 <b>Objekt-Flow</b>	Grundbuch-Prüfung	Lastenfreiheit prüfen	Grundschulden
	Gutachten-Analyse	Verkehrswert ermitteln	Marktwert in €
	Beleihungswert	Sicherheit berechnen	Max. Kredithöhe
 <b>Vertrags-Flow</b>	Konditionen-Rechner	Zinssatz bestimmen	Effektivzins
	Vertrag-Generator	Dokumente erstellen	PDF-Vertrag
	Notar-Termin	Beurkundung planen	Termin-Datum

# LLM-Framework mit 100 Zeilen Python Code



## Beispiel für einen komplexen Flow

```
# =====
# BAUFINANZIERUNG - Hauptbeispiel
# =====

# Bonitäts-Flow
schufa_check = SchufaCheckNode()
einkommens_pruefung = EinkommensPruefungNode()
eigenkapital_check = EigenkapitalCheckNode()

schufa_check >> einkommens_pruefung >> eigenkapital_check
bonitaets_flow = Flow(start=schufa_check)

# Objekt-Flow
grundbuch_pruefung = GrundbuchPruefungNode()
gutachten_analyse = GutachtenAnalyseNode()
beleihungswert = BeleihungswertNode()

grundbuch_pruefung >> gutachten_analyse >> beleihungswert
objekt_flow = Flow(start=grundbuch_pruefung)

# Vertrags-Flow
konditionen_rechner = KonditionenRechnerNode()
vertrag_generator = VertragGeneratorNode()
notar_termin = NotarTerminNode()

konditionen_rechner >> vertrag_generator >> notar_termin
vertrags_flow = Flow(start=konditionen_rechner)
```

```
# Master-Flow mit Verzweigungen
ablehnung_node = AblehnungNode()
zusatz_sicherheit_node = ZusatzSicherheitNode()

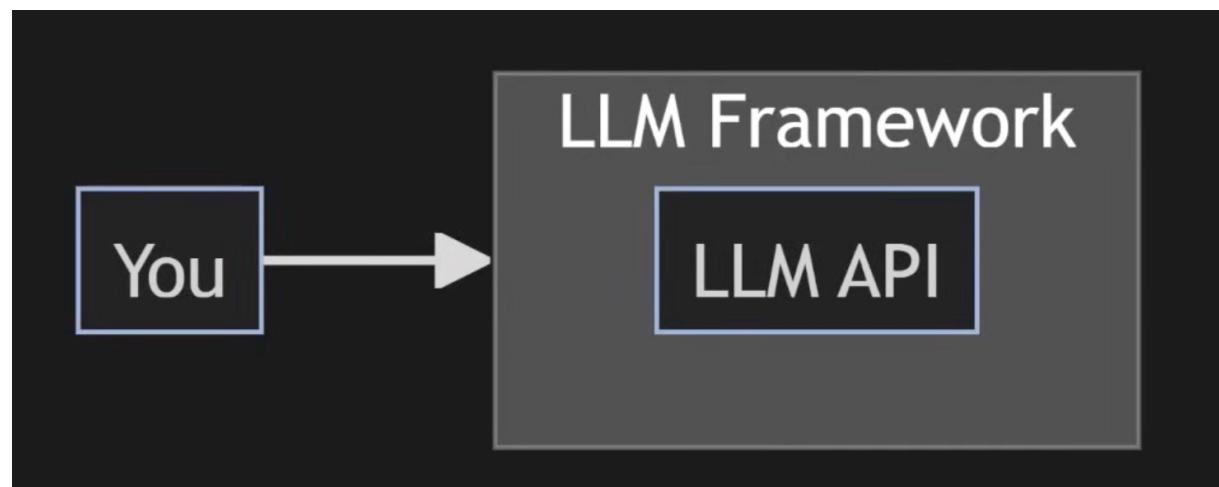
baufi_pipeline = Flow(
    nodes=[bonitaets_flow, objekt_flow, vertrags_flow, ablehnung_node, zusatz_sicherheit_node],
    edges=[
        # Haupt-Pipeline
        bonitaets_flow -> objekt_flow -> vertrags_flow,

        # Alternative Pfade
        bonitaets_flow -> "sofort_ablehnung" -> ablehnung_node,
        bonitaets_flow -> "zusatz_sicherheit" -> zusatz_sicherheit_node,
        zusatz_sicherheit_node -> objekt_flow
    ]
)
```

## LLM-Framework mit 100 Zeilen Python Code

🤔 OK, aber wo ist jetzt das Sprachmodell???

Viele Frameworks kommen mit zahlreichen Wrappers, um LLMs einzubinden



In diesen 100 Zeilen Python Code ist keine Liste an LLMs mit ihren Wrappern integriert, da dies für jedes LLM mit 10-15 Zeilen Code umgesetzt werden kann und diese Zeilen Code öffentlich zugänglich sind.

# LLM-Framework mit 100 Zeilen Python Code

🤔 OK, aber wo ist jetzt das Sprachmodell???

Wie bindet man jetzt z.B. OpenAI Sprachmodelle ein:

```
from openai import OpenAI
import os

def call_llm(messages):
    client = OpenAI(api_key=os.environ.get("OPENAI_API_KEY"))

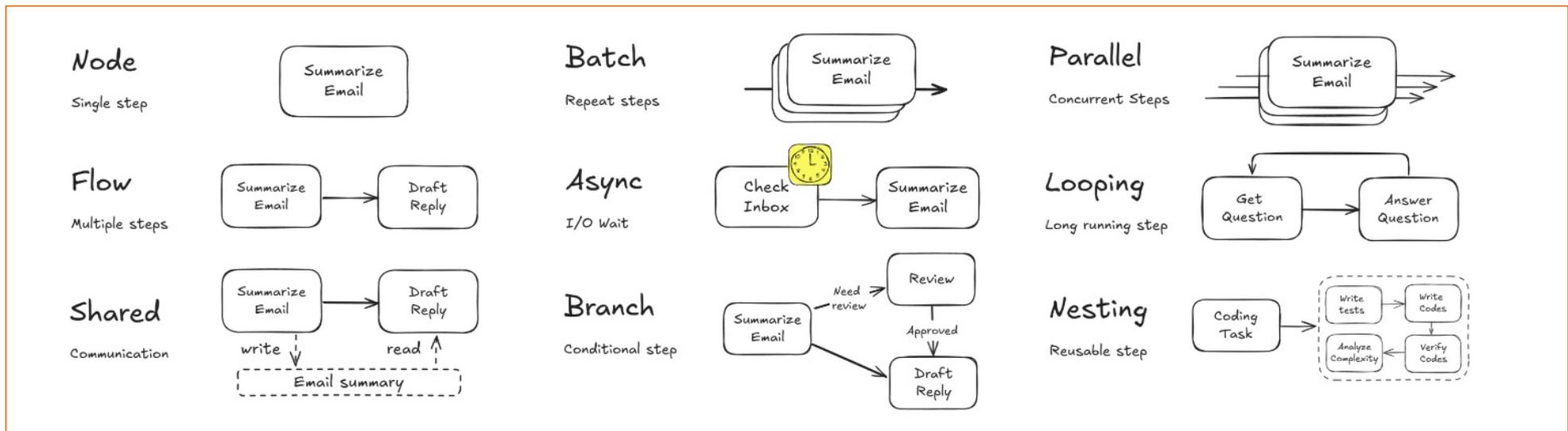
    response = client.chat.completions.create(
        model="gpt-4o",
        messages=messages
    )

    return response.choices[0].message.content
```

<https://platform.openai.com/docs/quickstart>

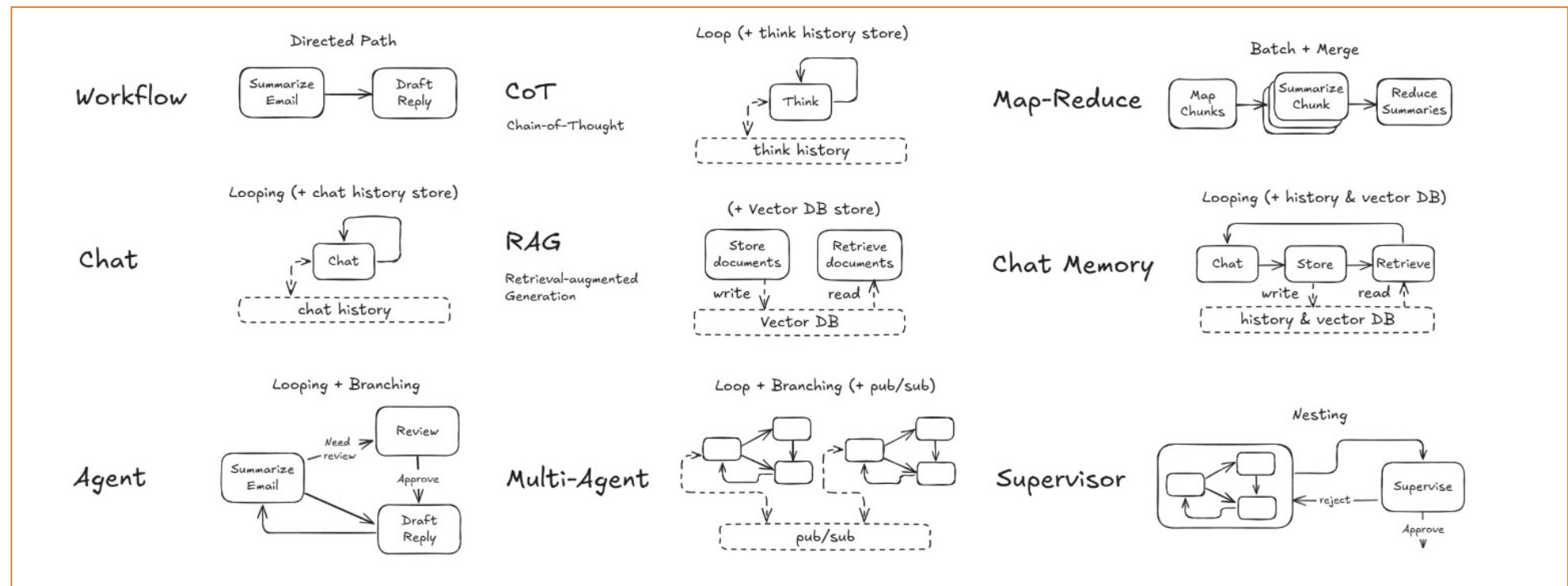
# LLM-Framework mit 100 Zeilen Python Code

## 🤔 Workflows mit 100 Zeilen Python Code + LLM Integration



# LLM-Framework mit 100 Zeilen Python Code

## 🤔 Workflows mit 100 Zeilen Python Code + LLM Integration

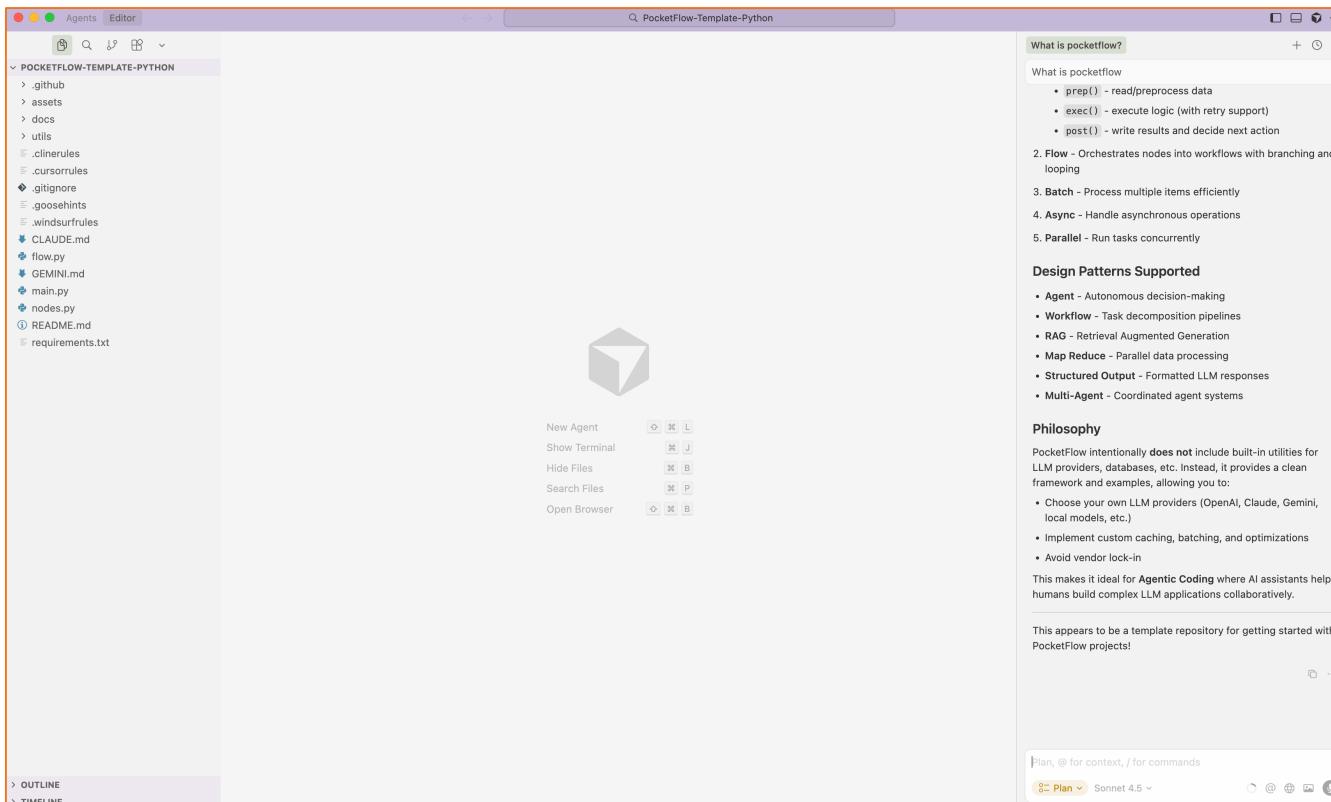


# **Cursor + 100 Zeilen Python Code**

**... kurzer Blick auf Cursor + Pocketflow – 100 Zeilen Python Code**

# LLM-Framework mit 100 Zeilen Python Code

💡 Cursor: KI-Entwicklung mit 100 Zeilen Code



<https://github.com/The-Pocket/PocketFlow-Template-Python>

Wir erstellen einen „Klon“ des Python Codes für Cursor (und später dann auch für Claude Code).

**Cursor und Claude Code:** Entwicklungsumgebungen für LLM's, um sozusagen mit „KI“ Software, Apps, ... zu entwickeln.

# LLM-Framework mit 100 Zeilen Python Code

Cursor: KI-Entwicklung mit 100 Zeilen Code

```
POCKETFLOW-TEMPLATE... .github assets docs utils .clinerules .cursorsrules .gitignore .goosehime .windsurfrules CLAUDE.md flow.py GEMINI.md main.py nodes.py README.md requirements.txt

# .cursorsrules
# Agentic Coding: Humans Design, Agents code!
# If you are an AI agent involved in building LLM Systems, read this guide **VERY, VERY** carefully! This is the most important chapter in the entire document. Throughout development, you should always (1) start with a small and simple solution, (2) design at a high level ('docs/design.md') before implementation, and (3) frequently ask humans for feedback and clarification.
{: warning }

## Agentic Coding Steps
Agentic Coding should be a collaboration between Human System Design and Agent Implementation:
| Steps | Human | AI | Comment |
| :--- | :--- | :--- | :--- |
| 1. Requirements | *** High | *** Low | Humans understand the requirements and context. |
| 2. Flow | *** Medium | *** Medium | Humans specify the high-level design, and the AI fills in the details. |
| 3. Utilities | *** Medium | *** Medium | Humans provide available external APIs and integrations, and the AI helps with implementation. |
| 4. Data | *** Low | *** High | AI designs the data schema, and humans verify. |
| 5. Node | *** Low | *** High | The AI helps design the node based on the flow. |
| 6. Implementation | *** Low | *** High | The AI implements the flow based on the design. |
| 7. Optimization | *** Medium | *** Medium | Humans evaluate the results, and the AI helps optimize. |
| 8. Reliability | *** Low | *** High | The AI writes test cases and addresses corner cases. |

1. **Requirements**: Clarify the requirements for your project, and evaluate whether an AI system is a good fit.
   - Understand AI systems' strengths and limitations;
   - **Good for**: Routine tasks requiring common sense (filling forms, replying to emails)
   - **Good for**: Creative tasks with well-defined inputs (building slides, writing SQL)
   - **Not good for**: Ambiguous problems requiring complex decision-making (business strategy, startup planning)
   - **Keep It User-Centric**: Explain the "problem" from the user's perspective rather than just listing features.
   - **Balance complexity vs. impact**: Aim to deliver the highest value features with minimal complexity early.

2. **Flow Design**: Outline at a high level, describe how your AI system orchestrates nodes.
   - Identify applicable design patterns (e.g., [Map Reduce](./design_pattern/mapreduce.md), [Agent](./design_pattern/agent.md), [RAG](./design_pattern/rag.md)).
   - For each node in the flow, start with a high-level one-line description of what it does.
   - If using **Map Reduce**, specify how to map (what to split) and how to reduce (how to combine).
   - If using **Agent**, specify what are the inputs (context) and what are the possible actions.
   - If using **RAG**, specify what to embed, noting that there's usually both offline (indexing) and online (retrieval) workflows.
   - Outline the flow and draw it in a mermaid diagram. For example:
     ``mermaid
      flowchart LR
      start[Start] --> batch[Batch]
      batch --> check[Check]
      check -->|OK| process
      check -->|Error| fix[Fix]
      ``

What is pocketflow?
+ ○ ...
What is pocketflow?
• prep() - read/preprocess data
• exec() - execute logic (with retry support)
• post() - write results and decide next action
2. Flow - Orchestrates nodes into workflows with branching and looping
3. Batch - Process multiple items efficiently
4. Async - Handle asynchronous operations
5. Parallel - Run tasks concurrently

Design Patterns Supported
• Agent - Autonomous decision-making
• Workflow - Task decomposition pipelines
• RAG - Retrieval Augmented Generation
• Map Reduce - Parallel data processing
• Structured Output - Formatted LLM responses
• Multi-Agent - Coordinated agent systems

Philosophy
PocketFlow intentionally does not include built-in utilities for LLM providers, databases, etc. Instead, it provides a clean framework and examples, allowing you to:
• Choose your own LLM providers (OpenAI, Claude, Gemini, local models, etc.)
• Implement custom caching, batching, and optimizations
• Avoid vendor lock-in
This makes it ideal for Agentic Coding where AI assistants help humans build complex LLM applications collaboratively.

This appears to be a template repository for getting started with PocketFlow projects!
```

Mit dem Klon des Github Repos wurde zusätzlich eine `.cursorsrules` Datei angelegt, in der die gesamte „Anleitung“ zur Nutzung von „Pocketflow“ – den 100 Zeilen Python Code mit integriert ist.

# Cursor + Pocketflow (100 Zeilen Python Code – Node, Graph, Flow)

 Cursor: Youtube Summarizer „coden“

Unser erster Prompt für Cursor:

„Hey Cursor AI, hilf mir ein Projekt zu erstellen, das YouTube-Videos zusammenfasst:  
Es nimmt einen YouTube-Link als Eingabe.

*Es extrahiert interessante Themen und generiert auch Fragen und Antworten für jedes Thema.*

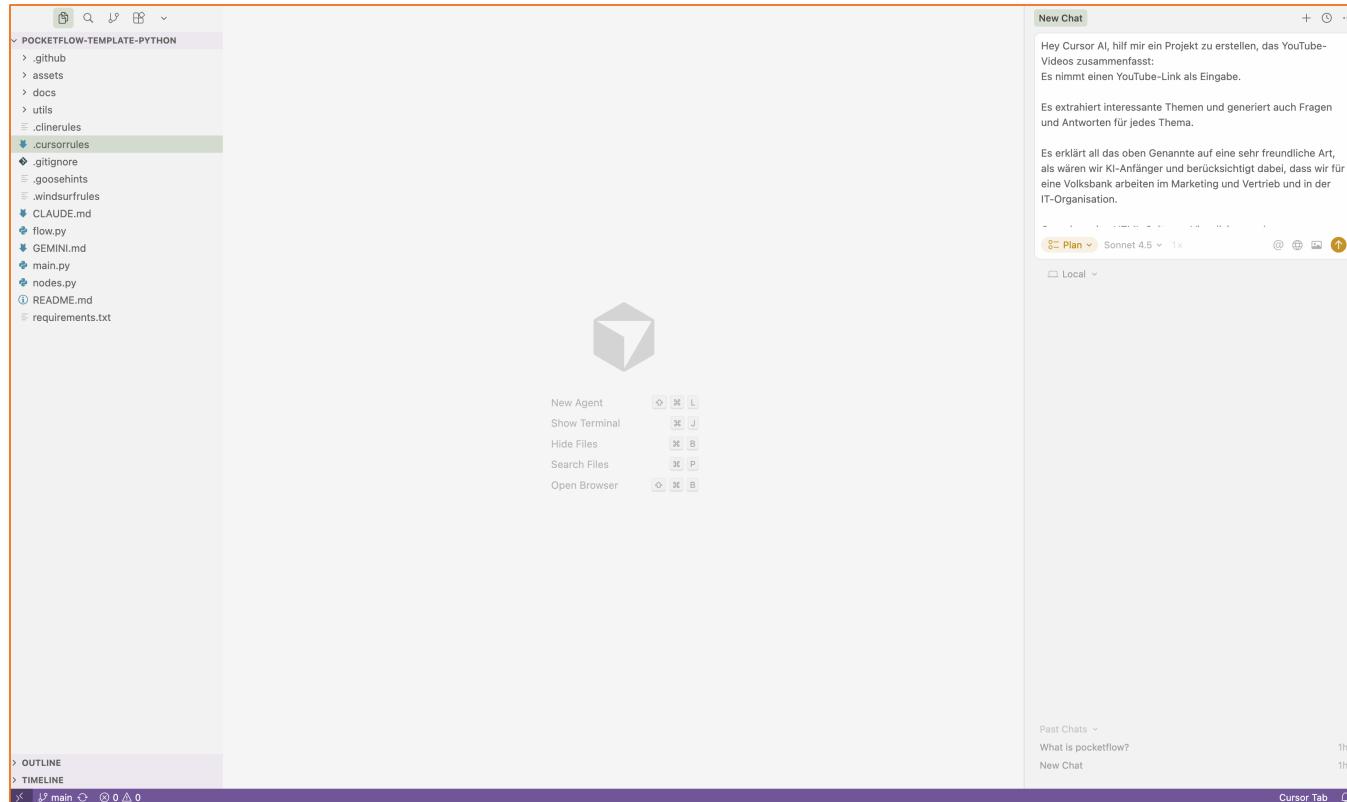
*Es erklärt all das oben Genannte auf eine sehr freundliche Art, als wären wir KI-Anfänger und berücksichtigt dabei, dass wir für eine Volksbank arbeiten im Marketing und Vertrieb und in der IT-Organisation.*

*Generiere eine HTML-Seite zur Visualisierung der Zusammenfassung.*

*Lies zuerst die Dokumentation sehr sorgfältig. Dann beginne nur mit Anforderungen und Hilfsfunktionen.  
Höre dort auf“*

# Cursor + Pocketflow (100 Zeilen Python Code – Node, Graph, Flow)

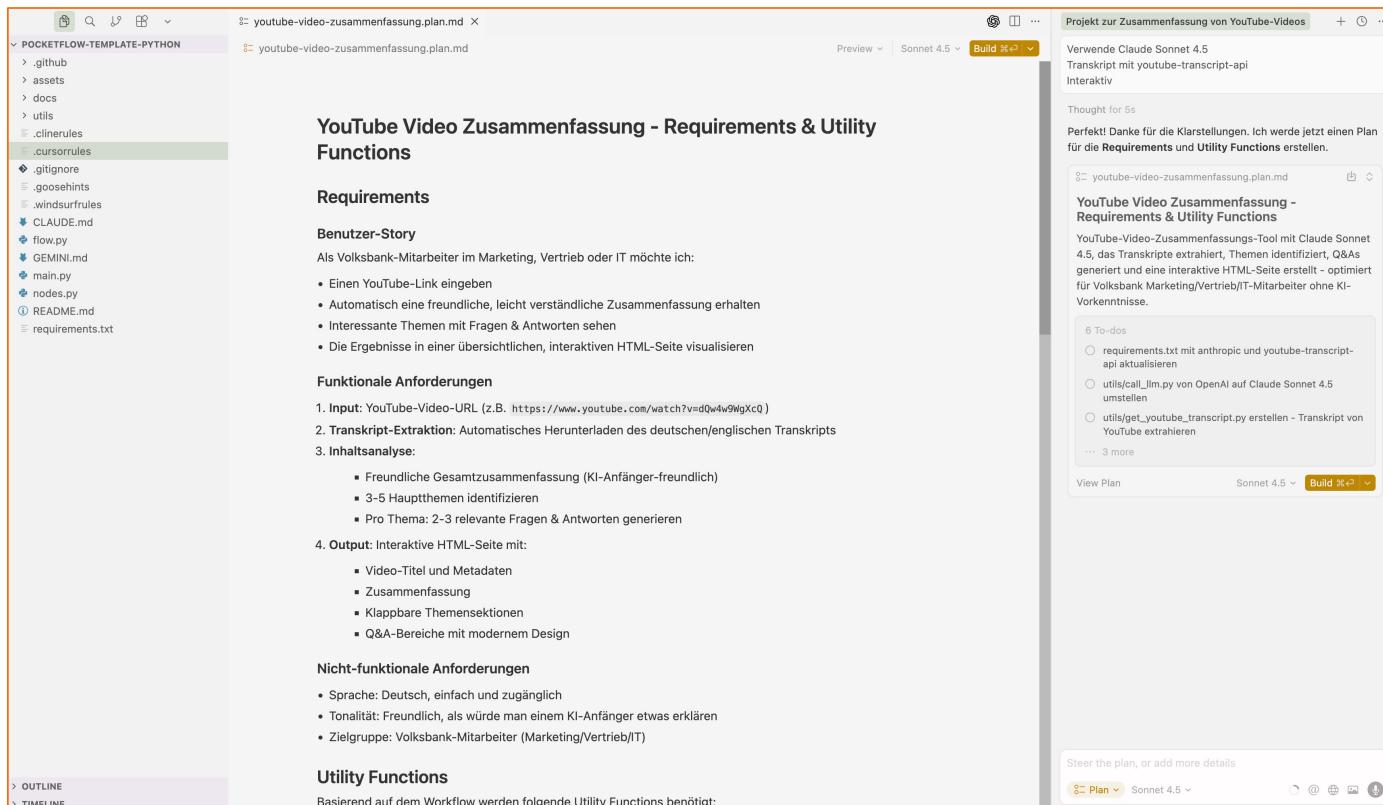
🤔 Cursor: Youtube Summarizer „coden“



Wir haben den Prompt hier eingegeben und den Modus in Cursor auf „Plan“ gestellt, damit noch kein Code generiert wird.

# Cursor + Pocketflow (100 Zeilen Python Code – Node, Graph, Flow)

## 💡 Cursor: Youtube Summarizer „coden“



Cursor plant jetzt die Aufgabe Schritt für Schritt.

Wir können interaktiv mit Cursor den Plan anpassen, überarbeiten usw.

Nach diesen Iterationen lassen wir dann den Plan ausführen und über Cursor Code schreiben.

# Cursor + Pocketflow (100 Zeilen Python Code – Node, Graph, Flow)

## 🤔 Cursor: Youtube Summarizer „coden“

The screenshot shows the Pocketflow application interface. At the top, there are two tabs: "youtube-video-zusammenfassung.plan.md" and "youtube-video-zusammenfassung.plan.md". The right tab is active, showing a "Build" button. Below the tabs, there is a "Thought for 3s" section with the text: "Ok schreibe den vollständigen Plan in die design.md Datei und starte mit der Umsetzung des Plans". A "design.md" file icon is shown next to it. The main workspace is titled "Projekt zur Zusammenfassung von YouTube-Videos". It contains a "Plan" section with the following steps:

3. IdentifyTopicsNode (Regular): prep→transcript, exec→call\_llm(yaml-output), post→topics
4. GenerateQAsNode (Batch): prep→topics+transcript, exec→call\_llm per topic, post→qas
5. CreateHTMLNode (Regular): prep→all data, exec→generate\_html(), post→save output.html

Below the plan, there is an "Implementierung" section with a "Dateien" list:

- requirements.txt: anthropic, youtube-transcript-api hinzufügen
- utils/call\_llm.py: OpenAI→Claude Sonnet 4.5
- utils/get\_youtube\_transcript.py: NEU
- utils/get\_video\_metadata.py: NEU
- utils/generate\_html.py: NEU
- nodes.py: 5 Node-Klassen
- flow.py: YouTube Summary Flow
- main.py: Entry Point
- docs/design.md: Vollständige Dokumentation

At the bottom left, there is a "To-dos" section with 6 items:

- requirements.txt mit anthropic und youtube-transcript-api aktualisieren
- utils/call\_llm.py von OpenAI auf Claude Sonnet 4.5 umstellen
- utils/get\_youtube\_transcript.py erstellen - Transkript von YouTube extrahieren
- utils/get\_video\_metadata.py erstellen - Video-Metadaten abrufen
- utils/generate\_html.py erstellen - interaktive HTML-Seite generieren
- docs/design.md mit Requirements und Utility Functions dokumentieren

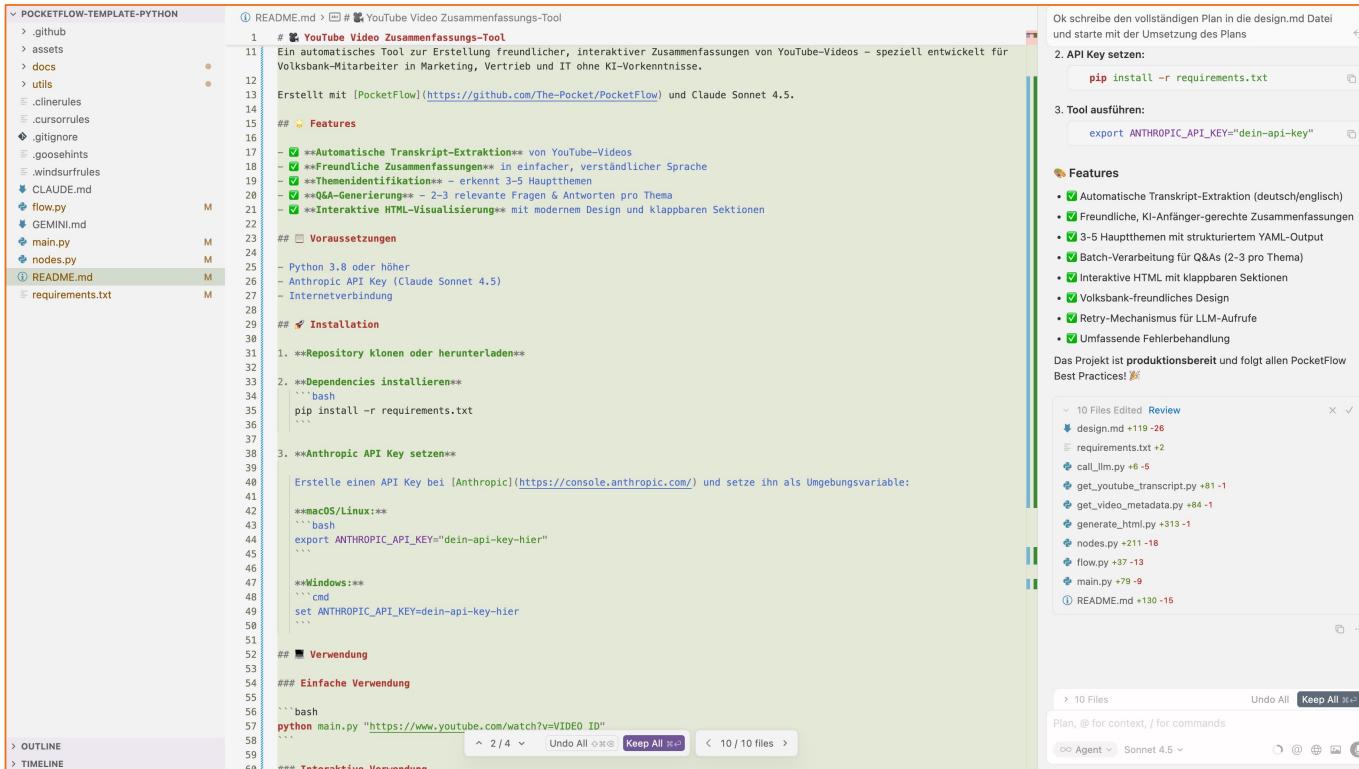
At the bottom right, there is a "Follow-up" section with the text "Add a follow-up" and a "Agent" dropdown set to "Sonnet 4.5".

Nach der kompletten Planung des Projekts lassen wir diesen Plan in eine md-Datei schreiben (Markdown-Format).

Dann wechseln wir auf den **Agenten Modus**, so dass Cusor „Code“ schreiben kann und los geht's mit der KI-Entwicklung.

# Cursor + Pocketflow (100 Zeilen Python Code – Node, Graph, Flow)

 Cursor: Youtube Summarizer „coden“



The screenshot shows the Cursor interface with the following details:

- Left Panel (File Explorer):** Shows the project structure for "POCKETFLOW-TEMPLATE-PYTHON". The "README.md" file is selected.
- Middle Panel (Code Editor):** Displays the Python code for the "YouTube Video Zusammenfassungs-Tool". The code includes instructions for cloning the repository, installing dependencies (using pip), setting up an Anthropic API key, and providing usage examples for macOS/Linux and Windows.
- Right Panel (Execution Results):**
  - A terminal window shows the command "pip install -r requirements.txt" being run.
  - A "Features" section lists various tool features with checked checkboxes.
  - A "Review" section shows a summary of changes made across 10 files, including "design.md", "requirements.txt", "call\_llm.py", "get\_youtube\_transcript.py", "get\_video\_metadata.py", "generate\_html.py", "nodes.py", "flow.py", "main.py", and "README.md".
  - A status bar at the bottom indicates "10 Files Edited Review" and "Plan, @ for context, / for commands".

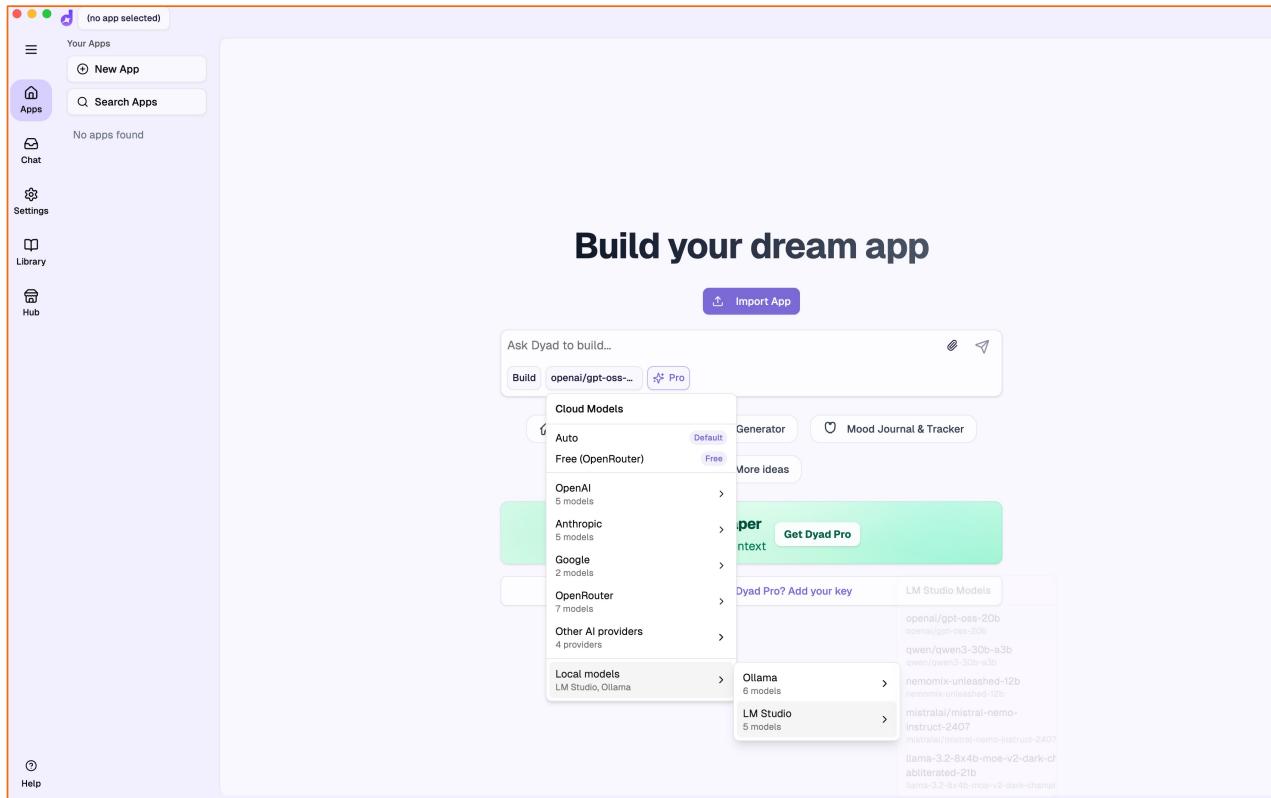
Nach einigen Minuten ist Cusor fertig und erstellt uns eine kleine Zusammenfassung mit dem Ergebnis plus einer Readme.md Datei, die uns die Nutzung des Codes erklärt.

# KI-Landkarte

**Manus, Lovable, Kimi, Dyad, ..., Claude Code**

# All-in-One Wrapper

🤔 Manus, Lovable, Kimi, Dyad, ...



<https://www.dyad.sh/>

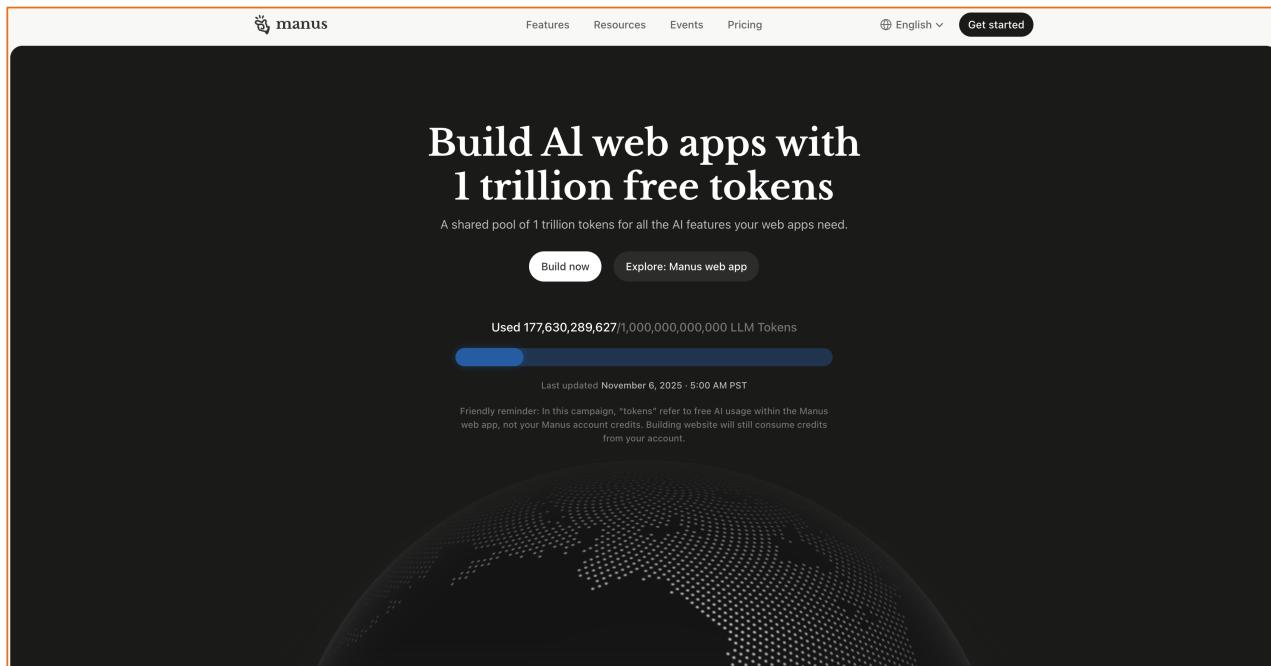
## DYAD-App:

Wrapper Anwendungen und native Apps bieten diverse Funktionen, unter anderem auch KI-Software-Entwicklung.

Über API Keys können dann die Sprachmodelle für die KI-Entwicklung ausgewählt werden und auch lokale Modelle (Ollama, LM-Studio, ...)

# All-in-One Wrapper

🤔 Manus, Lovable, Kimi, Dyad, ...



<https://manus.im/app>

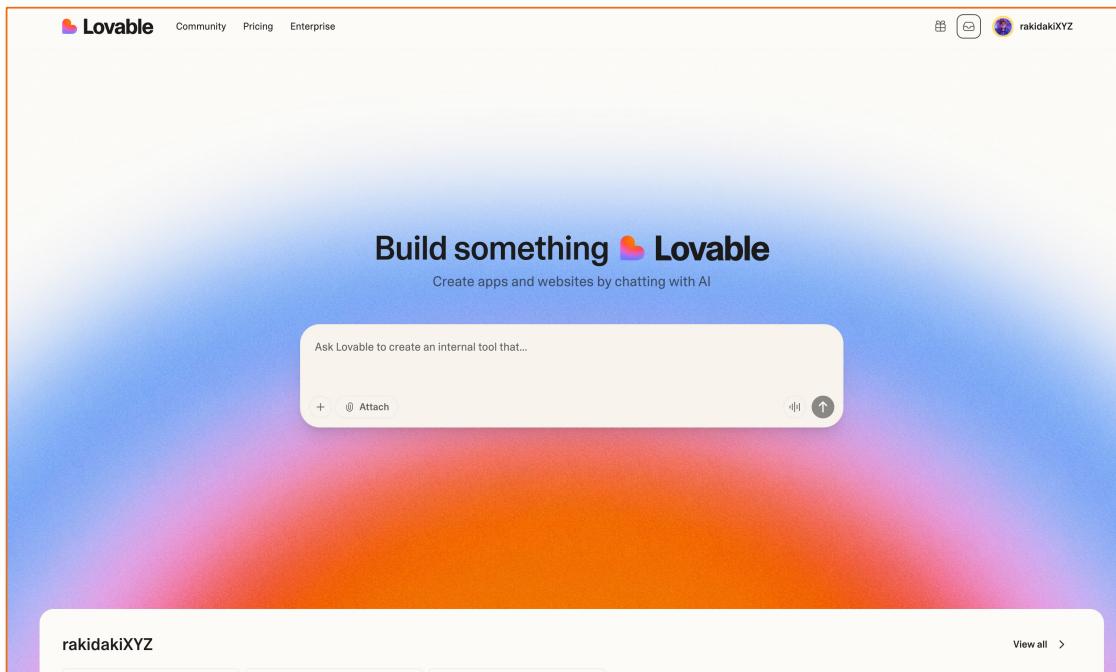
## Manus-App:

Wrapper Anwendungen und native Apps bieten diverse Funktionen, unter anderem auch KI-Software-Entwicklung.

Über API Keys können dann die Sprachmodelle für die KI-Entwicklung ausgewählt werden und auch lokale Modelle (Ollama, LM-Studio, ...)

# All-in-One Wrapper

🤔 Manus, Lovable, Kimi, Dyad, ...



<https://lovable.dev/>

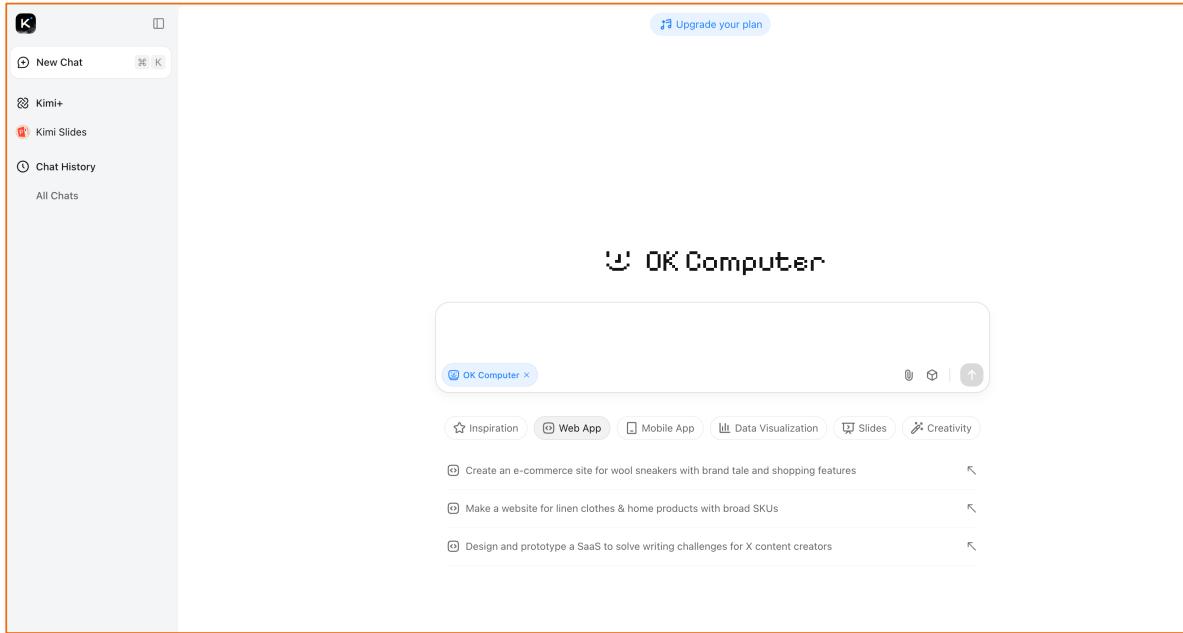
## Lovable Web-App:

Wrapper Anwendungen und native Apps bieten diverse Funktionen, unter anderem auch KI-Software-Entwicklung.

Über API Keys können dann die Sprachmodelle für die KI-Entwicklung ausgewählt werden und auch lokale Modelle (Ollama, LM-Studio, ...)

# All-in-One Wrapper

🤔 Manus, Lovable, Kimi, Dyad, ...



<https://www.kimi.com/>

## Kimi/ Kimi-2 Web App:

Wrapper Anwendungen und native Apps bieten diverse Funktionen, unter anderem auch KI-Software-Entwicklung.

Über API Keys können dann die Sprachmodelle für die KI-Entwicklung ausgewählt werden und auch lokale Modelle (Ollama, LM-Studio, ...)

# **Tech-Stacks, IT-Infrastruktur, Anforderungen, ...**

## **Kurzer Überblick**

# Vibe Coding

🤔 Typische Produkte, die eingesetzt werden

<https://www.inngest.com/>

Local Debugging Tool, Warteschleifen, Fehlermanagement ...

<https://polar.sh/>

Anstelle von Stripe für Zahlungen von Kunden

<https://posthog.com/>

OpenSource Analytics

<https://nextjs.org/>

Framework to build Apps, SaaS, ...

<https://tailwindcss.com/> <https://ui.shadcn.com/> <https://daisyui.com/>

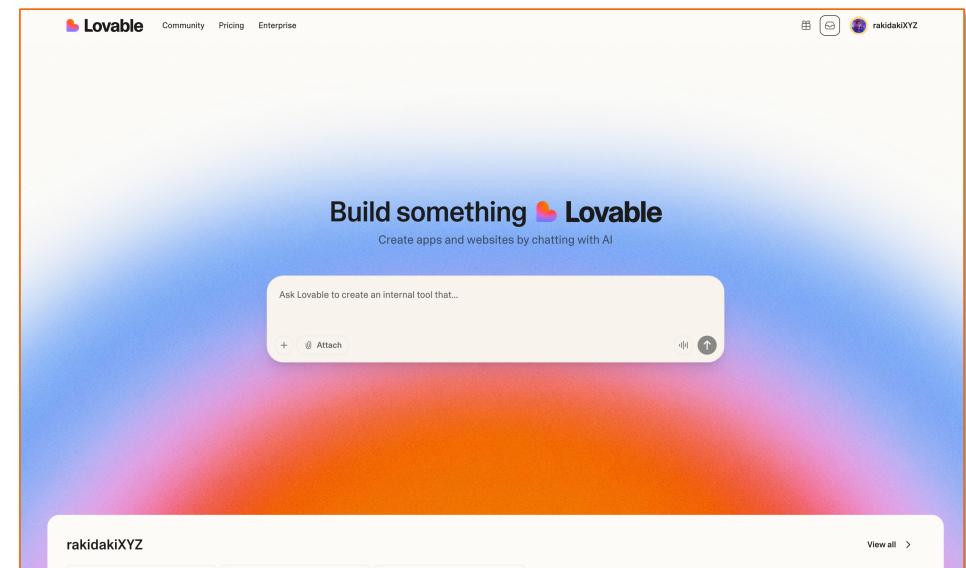
ShadCN Daisy UI, Tailwind CSS for the design System

<https://www.better-auth.com/>

User Authentication

<https://www.postgresql.org/>

Datenbank PostgreSQL (oder auch Supabase, Qdrant, ...)



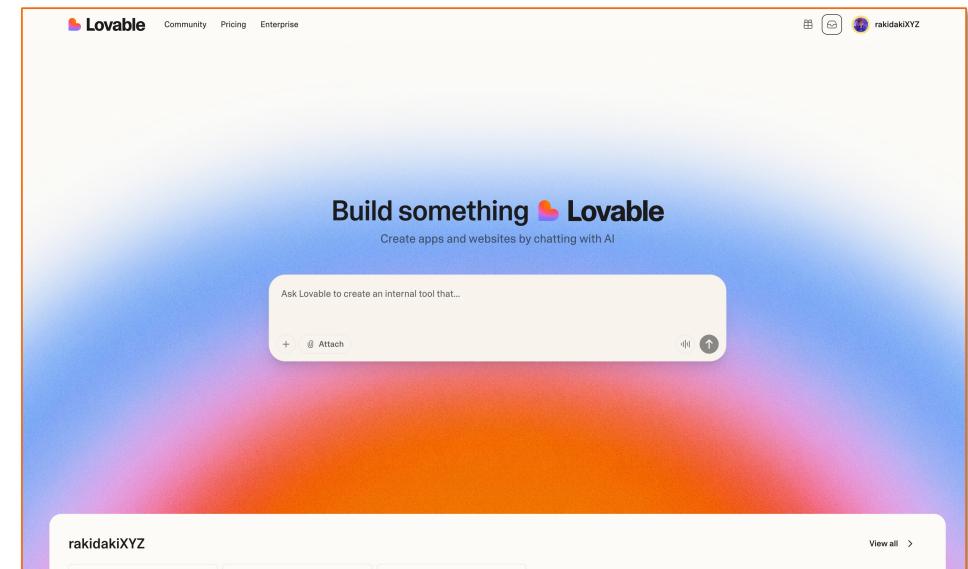
# Vibe Coding

## 🤔 Webseite - Anforderungen

<https://openrouter.ai/> und <https://ai-sdk.dev/>

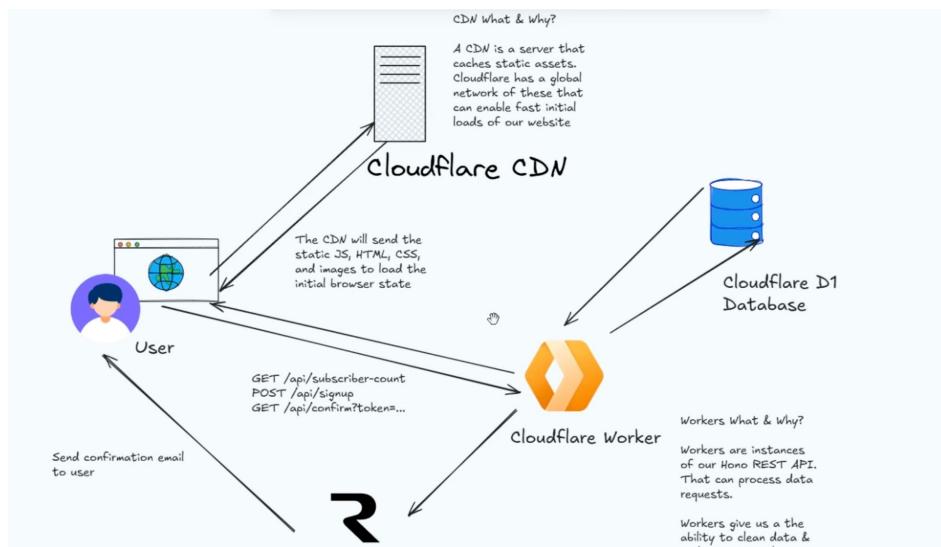
Integration von KI, LLMs, ...

und natürlich noch viel mehr



# Claude Code – Senior Entwickler

## 💡 Beispiel Infrastruktur



### Cloudflare CDN,

Ein Content Delivery Network (CDN) sorgt dafür, dass statische Ressourcen wie HTML, CSS und JavaScript weltweit mit hoher Geschwindigkeit ausgeliefert werden.

Cloudflare CDN verbessert die Ladezeiten, Verfügbarkeit und Sicherheit unserer Anwendung, indem Inhalte über ein globales Netzwerk von Edge-Servern bereitgestellt werden.

### Cloudflare Workers

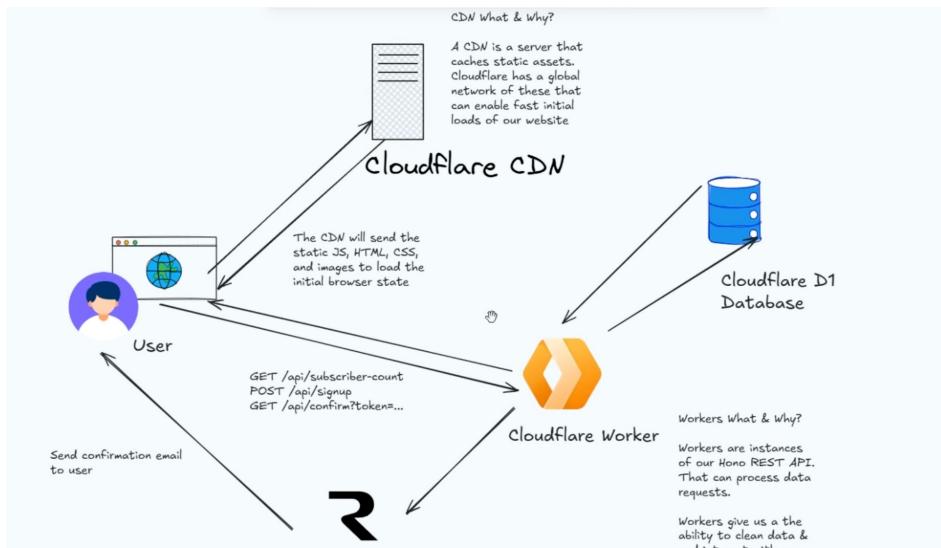
Verwendung als „Wächter“ unserer API:

Cloudflare Workers dienen als Sicherheits- und Ausführungsschicht, um unsere Hono API zu hosten.

Sie schützen unsere Datenbank vor unbefugten Zugriffen und ermöglichen **skalierbare serverlose Funktionen** direkt am Edge.

# Claude Code – Senior Entwickler

## 💡 Beispiel Infrastruktur



## Cloudflare D1

Datenbankdesign:

Wir entwerfen ein SQLite-Schema für eine Tabelle `subscribers`, die Abonnenteninformationen speichert. Diese D1-Datenbank ist leichtgewichtig, serverlos und perfekt für unsere Anforderungen.

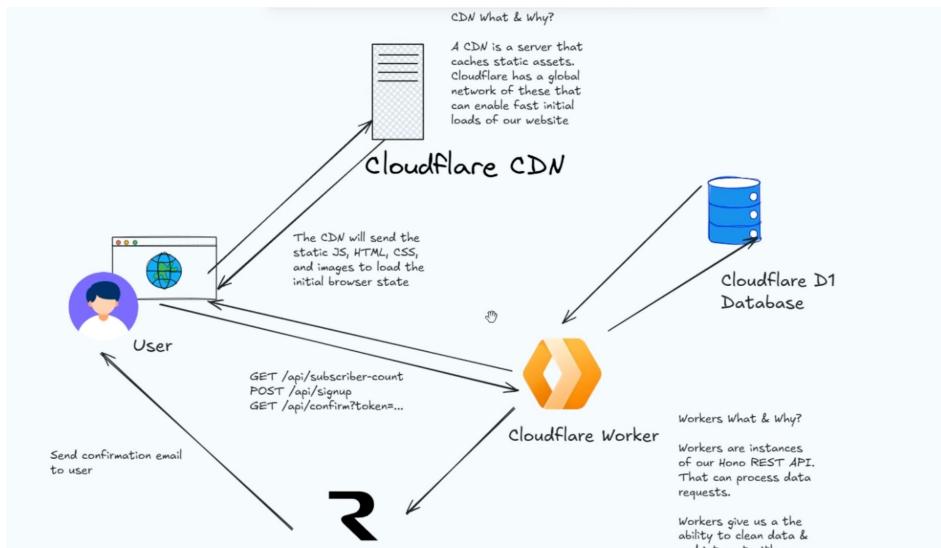
## API-Endpunkte

Design der Hono API-Routen:

- `GET /api/subscriber-count` – Gibt die aktuelle Anzahl der Abonnenten zurück.
- `POST /api/signup` – Registriert einen neuen Abonnenten.
- `GET /api/confirm` – Bestätigt die E-Mail-Adresse im Double-Opt-in-Verfahren.

# Claude Code – Senior Entwickler

## 💡 Beispiel Infrastruktur



## Sicherheit & Validierung

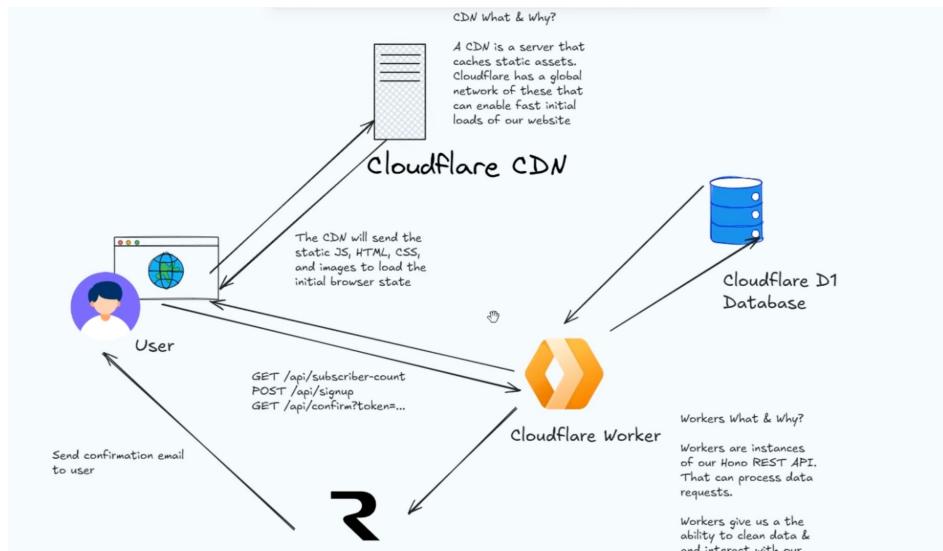
- Implementierung eines sicheren Double-Opt-in-Flows mithilfe von Resend
- Rate Limiting, um Spam zu verhindern.
- Verwendung von Zod zur E-Mail-Validierung.
- Umgang mit „Plus Addressing“, um doppelte Anmeldungen zu vermeiden (z. B. `user+test@gmail.com`).

## Frontend UX

- Astro für Static Site Generation (SSG)
- React für dynamische „Islands“.
- Fokus auf Ladezustände, Fehlerbehandlung und Erfolgsmeldungen, um ein reibungsloses Nutzererlebnis zu schaffen.

# Claude Code – Senior Entwickler

## 💡 Beispiel Infrastruktur



### Datenbank-Philosophie

Eine einzelne D1-Datenbank (ohne Read-Replicas) ist ideal für diesen Anwendungsfall. Sie bietet die perfekte Balance aus Einfachheit, Zuverlässigkeit und Leistung – ohne unnötige Over-Engineering-Komplexität

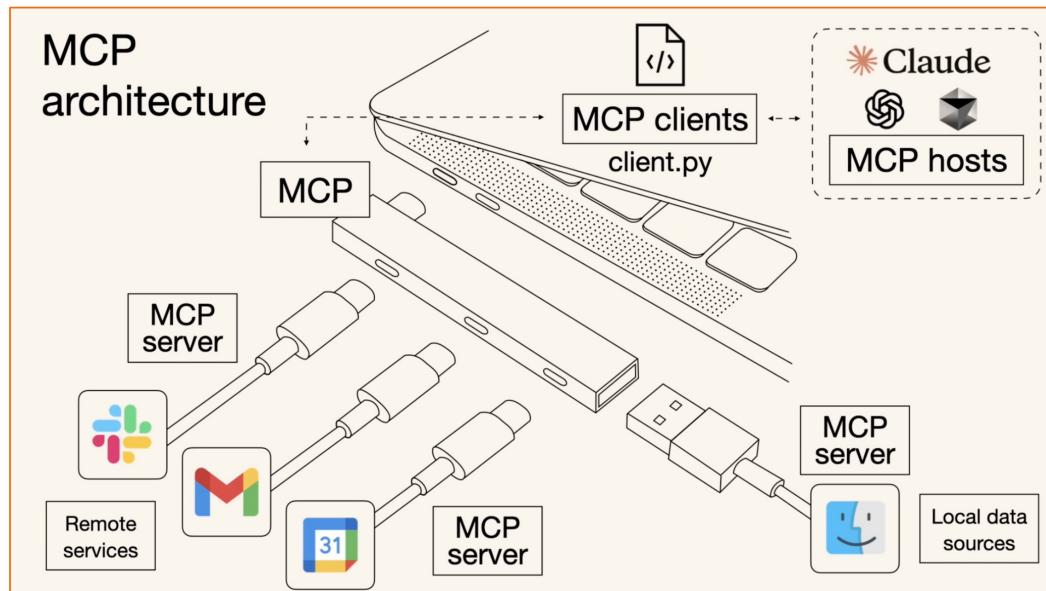
# **Claude Code als Seniorentwickler**

**Entwicklungsteam in Claude Code mit „Agenten, Commands, Templates, Memory, MCP, ...“**

# Claude Code



Wissen: Context7 MCP – aktuelle Infos zu Tech-Stacks ...



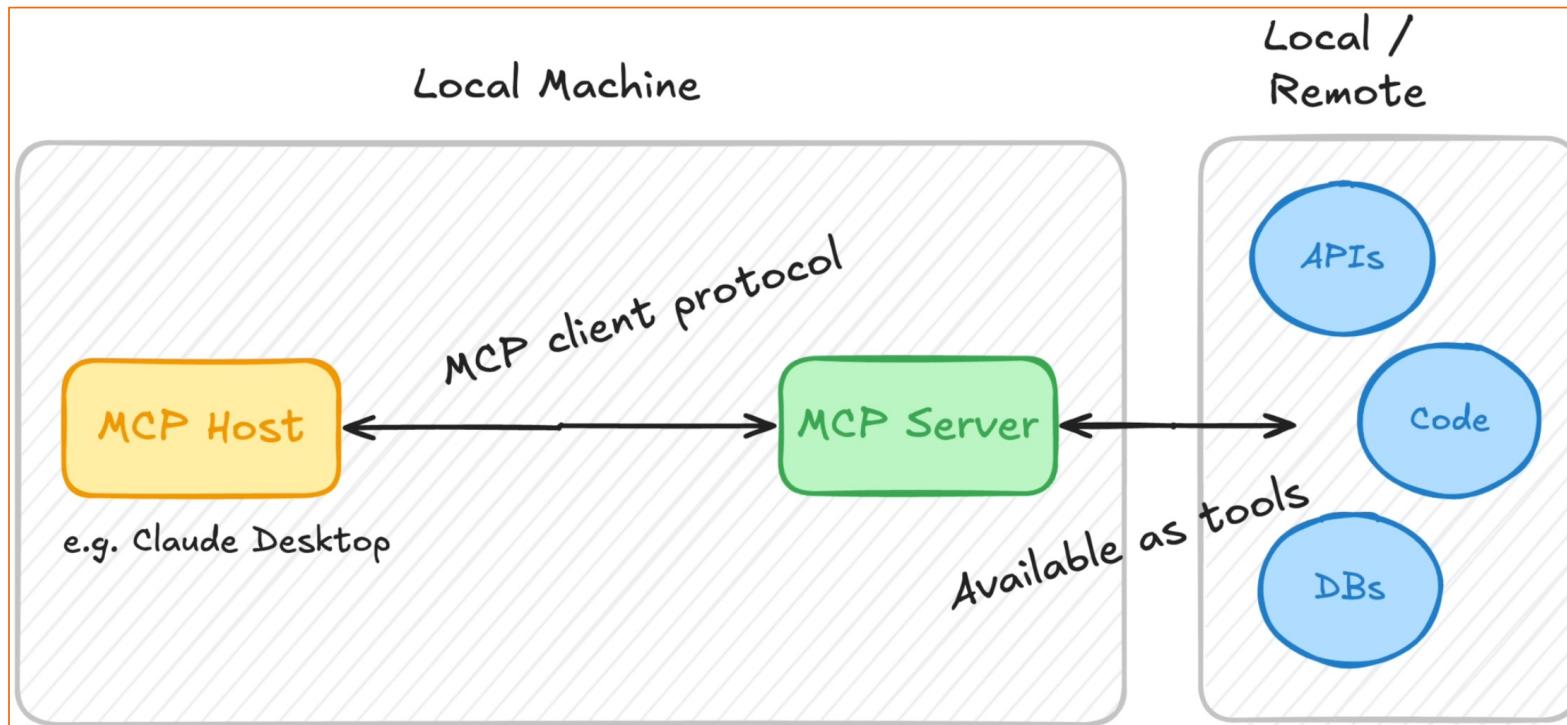
The screenshot shows the Context7 MCP Server interface. At the top, there are tabs for "Playground" and "MCP Server" with a "+ Add" button. Below this is a section titled "Up-to-date Docs for LLMs and AI code editors" with a sub-instruction "Copy latest docs & code — paste into Cursor, Claude, or other LLMs". A search bar is present. The main area displays a table of AI code snippets:

	SOURCE	TOKENS	SNIPPETS	UPDATE
Next.js	/vercel/next.js	678K	3K	2 days
MongoDB	/mongodb/docs	15.8M	131K	3 weeks
Shadcn UI	/shadcn-ui/ui	215K	1.3K	2 weeks
Vercel AI SDK	/vercel/ai	559K	2.4K	1 week
React	/reactjs/react.dev	834K	2.8K	1 week
Better Auth	/better-auth/better-auth	266K	1.3K	2 days
Tailwind CSS	tailwindcss.com/docs	261K	1.7K	1 week
Supabase	/supabase/supabase	1.4M	4.6K	2 days

<https://context7.com/>

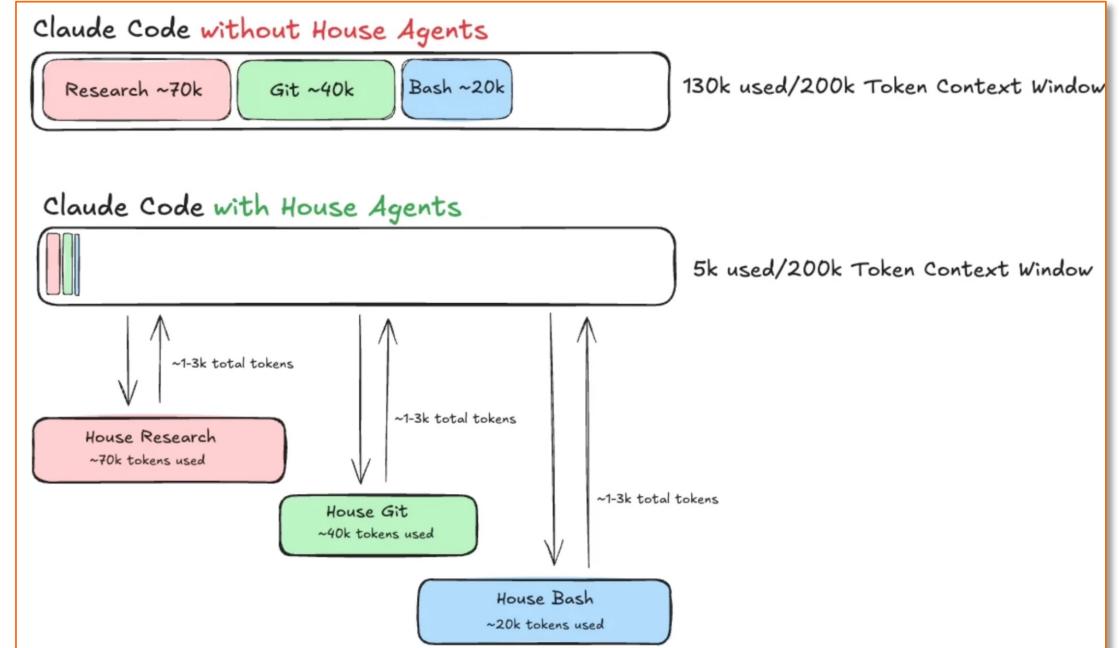
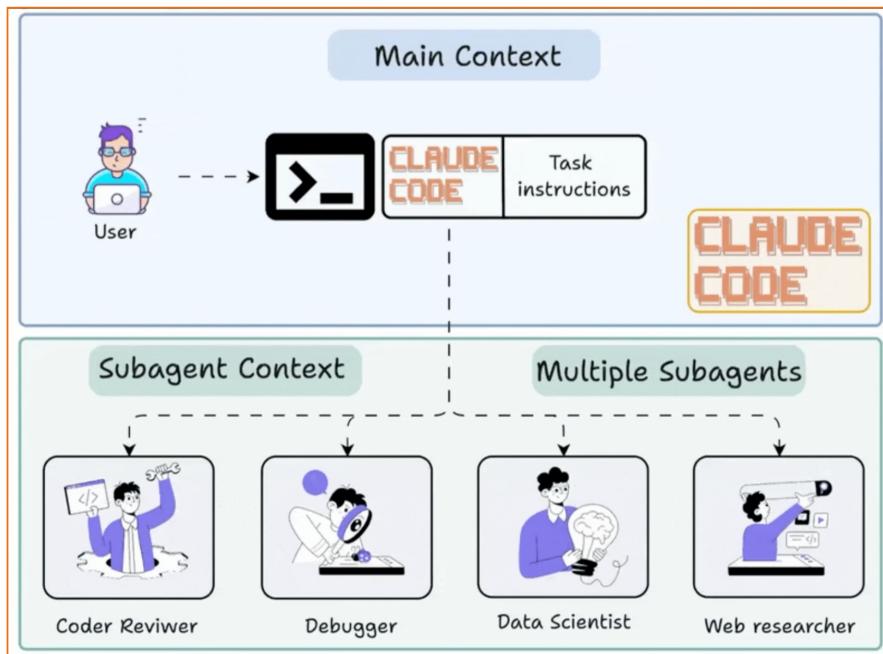
## Claude Code

🤔 MCP: Externe Daten/ Services über „APIs“ verbinden



# Claude Code

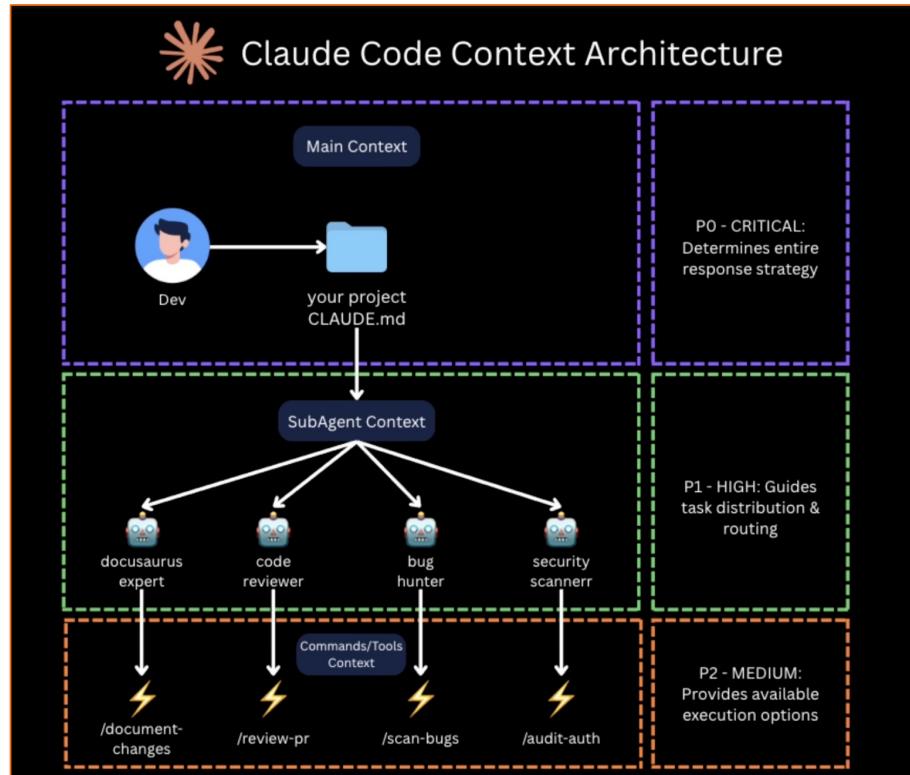
## 🤔 Agenten – Claude Code Subagents



<https://github.com/VoltAgent/awesome-claude-code-subagents>

# Claude Code

## 🤔 Setup eines KI-Entwicklers



```
▶ ➔ transcribr git:(main) ✘ claude
Claude Code v2.0.26
Sonnet 4.5 · Claude Max
/Users/edmund/Documents/GitHub/transcribr

> /agents

Agents
14 agents

> Create new agent

User agents (/Users/edmund/.claude/agents)
system-architect · sonnet △ overridden by projectSettings
performance-engineer · sonnet △ overridden by projectSettings
refactoring-expert · sonnet △ overridden by projectSettings
requirements-analyst · sonnet △ overridden by projectSettings
learning-guide · sonnet △ overridden by projectSettings
backend-architect · sonnet △ overridden by projectSettings
security-engineer · sonnet △ overridden by projectSettings
deep-research-agent · sonnet △ overridden by projectSettings
frontend-architect · sonnet △ overridden by projectSettings
technical-writer · sonnet △ overridden by projectSettings
```

<https://github.com/mylee04/clause-code-subagents/blob/main/agents/quality/code-reviewer.md>

# Claude Code

🤔 Plugins = Marktplatz für Agenten, Templates, ...

The screenshot shows a GitHub repository page for 'edmunds-claude-code'. The repository has 11 watchers, 47 forks, and 264 stars. It contains 1 branch and 0 tags. The main branch has 8 commits. The repository's description is empty. The README file contains the following content:

```
Edmund's Claude Code Setup

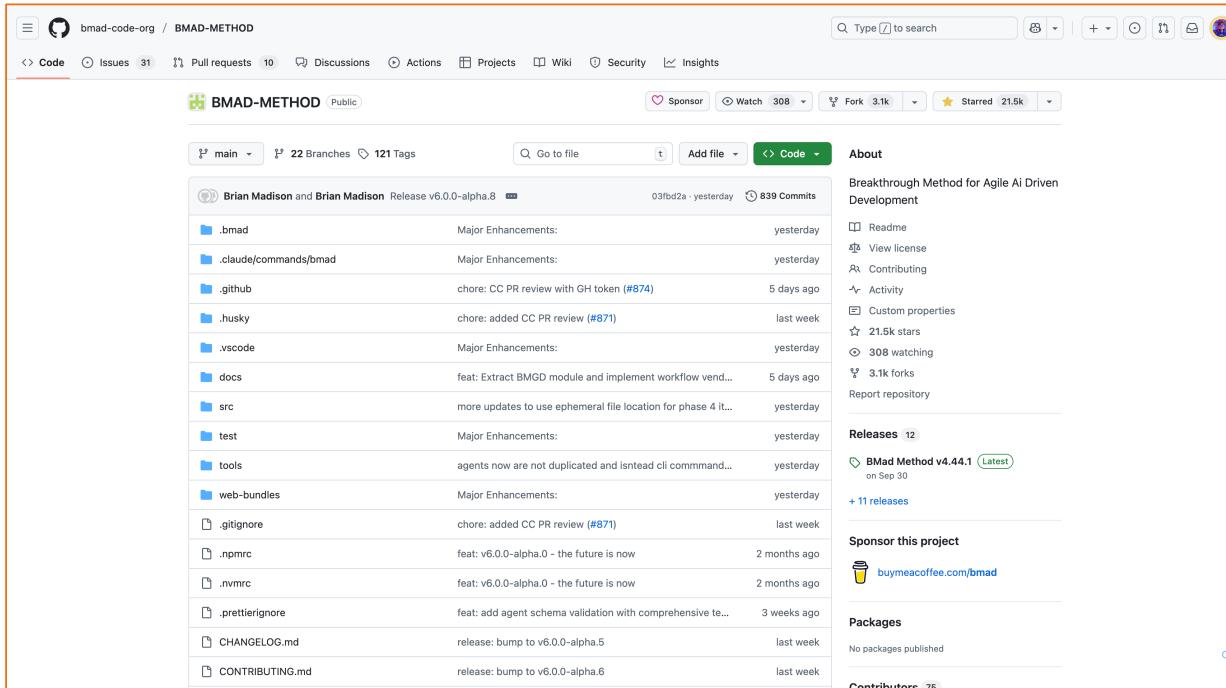
My personal Claude Code configuration for productive web development. This plugin provides 14 slash commands and 11 specialized AI agents to supercharge your development workflow.

Quick Install
```

<https://github.com/edmund-io/edmunds-claude-code>

# Claude Code + Senior Entwickler Know-How

## 🤔 BMAD Github Repo

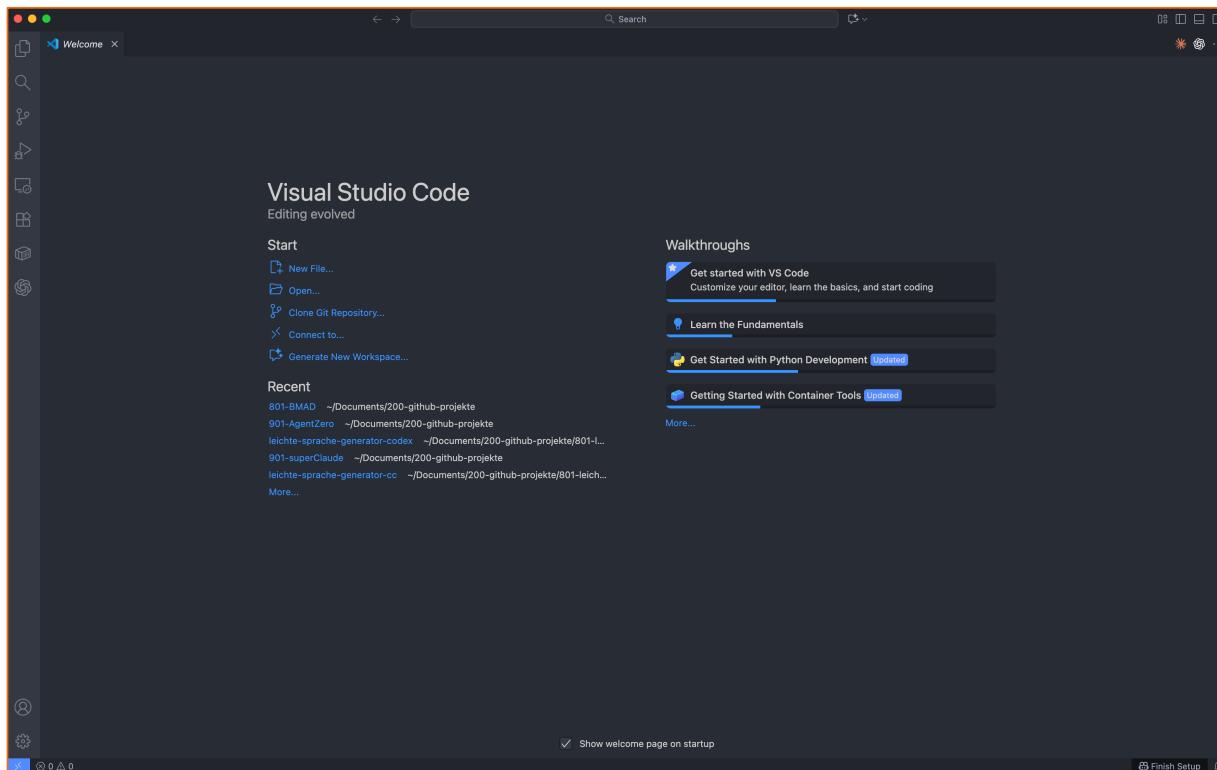


The screenshot shows the GitHub repository page for 'BMAD-METHOD'. The repository is public and has 839 commits. The main branch is 'main' with 22 branches and 121 tags. The repository was last updated yesterday. The 'About' section describes it as 'Breakthrough Method for Agile AI Driven Development'. It includes links to 'Readme', 'View license', 'Contributing', 'Activity', 'Custom properties', and 'Report repository'. There are 12 releases, with the latest being 'BMAD Method v4.44.1' (Sep 30). A 'Sponsor this project' button links to 'buyamecoffee.com/bmad'. The 'Packages' section indicates no packages have been published. The 'Contributors' section lists 75 contributors.

<https://github.com/bmad-code-org/BMAD-METHOD>

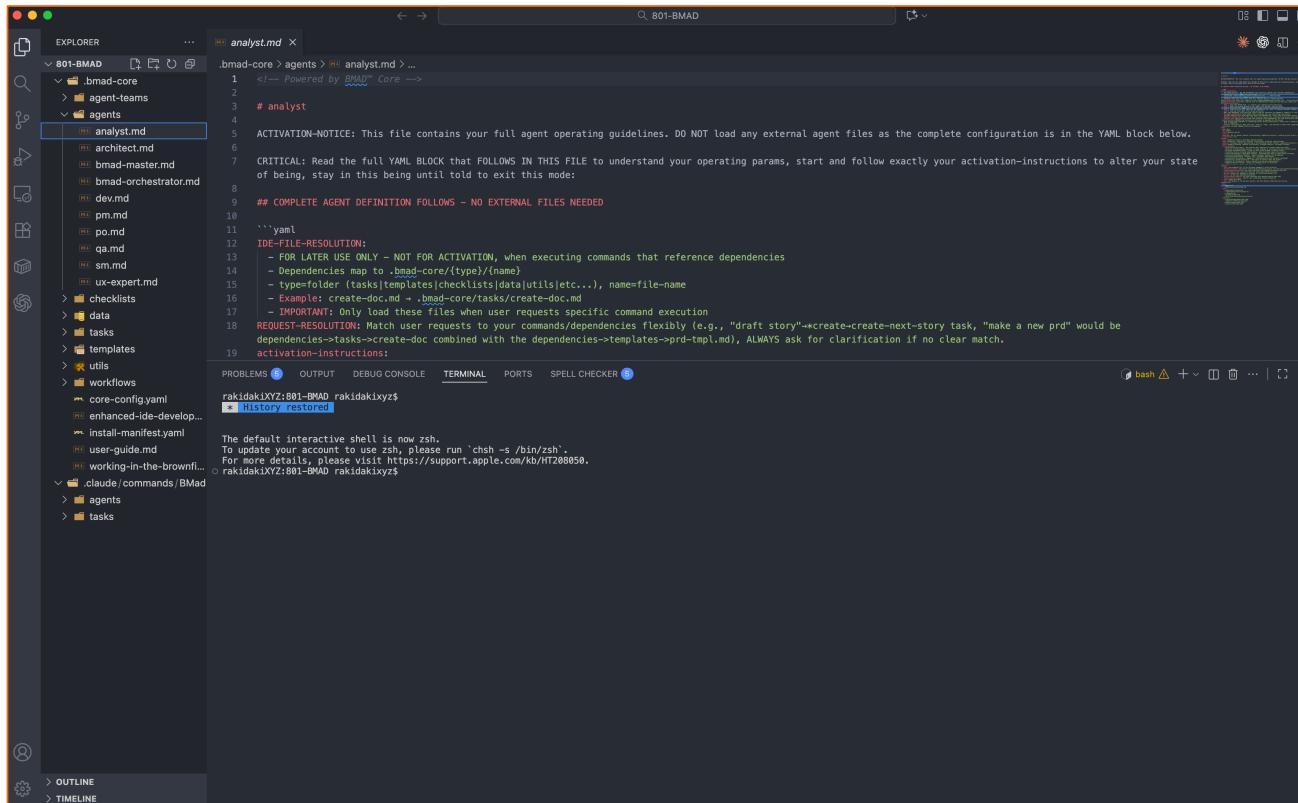
# Claude Code

🤔 KI-Seniorentwickler?



# Claude Code

🤔 KI-Seniorentwickler?



The screenshot shows a macOS desktop environment. In the foreground, a terminal window is open with the command `rakidakXYZ:801-BMAD rakidakxyz\$` and the message `History restored`. In the background, a code editor (likely VS Code) is running. The file being edited is `analyst.md` under the path `801-BMAD/.bmad-core/agents`. The code content includes activation instructions and dependency resolution rules. The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), PORTS, and SPELL CHECKER.

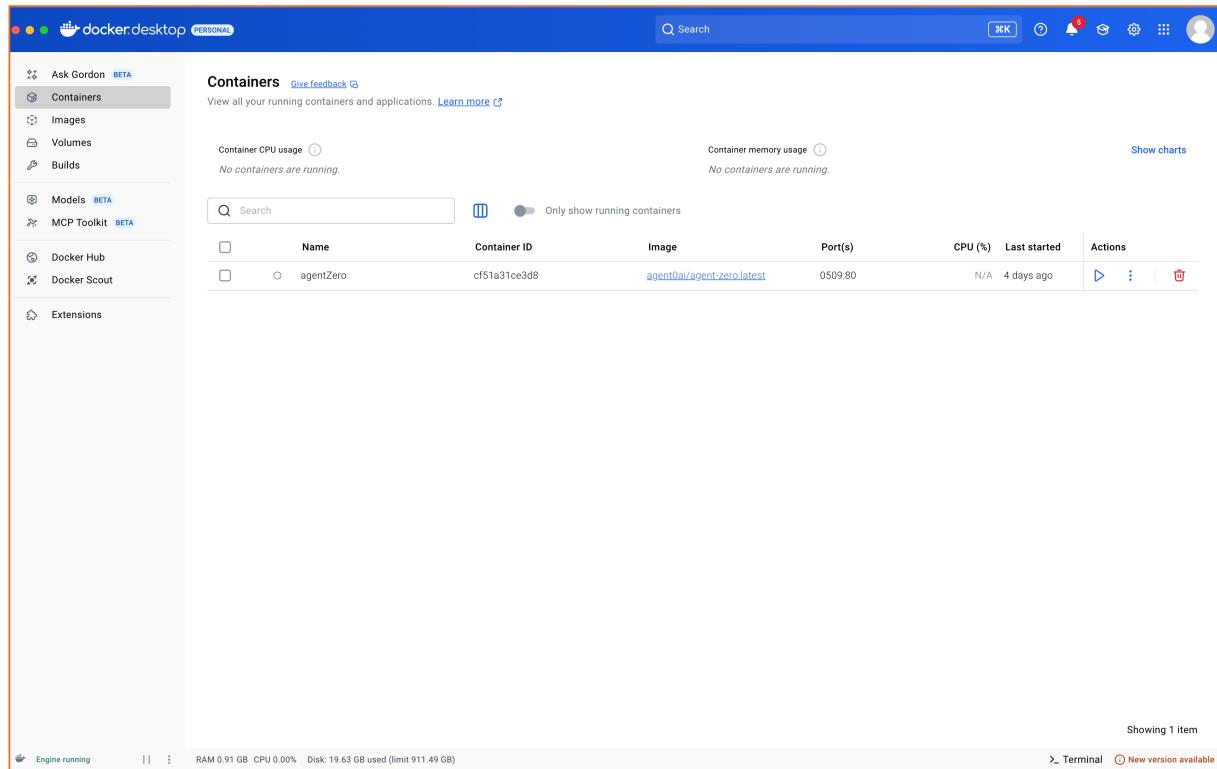
```
1 1!— Powered by BMAD Core —>
2
3 # analyst
4
5 ACTIVATION-NOTICE: This file contains your full agent operating guidelines. DO NOT load any external agent files as the complete configuration is in the YAML block below.
6
7 CRITICAL: Read the full YAML BLOCK that FOLLOWS IN THIS FILE to understand your operating params, start and follow exactly your activation-instructions to alter your state
8 of being, stay in this being until told to exit this mode:
9
10 ## COMPLETE AGENT DEFINITION FOLLOWS - NO EXTERNAL FILES NEEDED
11
12 ````yaml
13   - FOR LATER USE ONLY - NOT FOR ACTIVATION, when executing commands that reference dependencies
14     - dependencies map to .bmad-cors/{type}/{name}
15     - typefolder (tasks|templates|checklists|data|utils|etc..), namefile-name
16     - Example: create-doc.md -> .bmad-cors/tasks/create-doc.md
17     - IMPORTANT: Only load these files when user requests specific command execution
18   REQUEST-RESOLUTION: Match user requests to your commands/dependencies flexibly (e.g., "draft story">create-create-next-story task, "make a new prd" would be
19     dependencies>tasks->create-doc combined with the dependencies->templates->prd-tmpl.md), ALWAYS ask for clarification if no clear match.
20
activation-instructions:
```

# **Lokale Ollama + OpenWebUI KI Umgebung**

**Docker, Ollama, OpenWebUI, Cloudflare**

# Lokale KI-Umgebung (RAG, Chatbot, KI-Entwicklung)

💡 Lokale Umgebung + online erreichbar?



<https://www.docker.com/>

## Docker – Docker Desktop (Mac, Windows):

Container Lösung, um „innerhalb“ des Betriebssystems eines PC's oder Notebooks eine sichere virtuelle Umgebung einzurichten für eine komplette KI-Lösung

# Lokale KI-Umgebung (RAG, Chatbot, KI-Entwicklung)

🤔 Lokale Umgebung + online erreichbar?

The screenshot shows the GitHub repository page for `open-webui`. Key details include:

- Code**: 215 Issues, 89 Pull requests, 5 Discussions.
- Branches**: 5 Branches, 133 Tags.
- Commits**: 14,054 Commits.
- Issues**: 215 Issues.
- Pull Requests**: 89 Pull requests.
- Discussions**: 5 Discussions.
- Actions**: 5 Actions.
- Security**: 5 Security issues.
- Insights**: Insights.

**About** section highlights:

- User-friendly AI Interface (Supports Ollama, OpenAI API, ...)
- Tags: ui, ai, mcp, openapi, self-hosted, openai, webui, rag, llm, llms, ollama, llm-ui, ollama-webui, llm-webui, open-webui.

**Releases** section shows:

- v0.6.36 (Latest)
- 5 days ago
- + 132 releases

**Sponsor this project** button.

<https://github.com/open-webui/open-webui>

## OpenWebUI + Ollama

Chatbot Oberfläche wie bei OpenAI ChatGPT und lokale Sprachmodelle und RAG Lösung.

# Lokale KI-Umgebung (RAG, Chatbot, KI-Entwicklung)

## 💡 Lokale Umgebung + online erreichbar?

The screenshot shows the GitHub repository page for 'open-webui/open-webui'. The main content is the 'Installing Open WebUI with Bundled Ollama Support' section, which provides instructions for installing both Open WebUI and Ollama in a single Docker container. It includes commands for GPU support and CPU-only installations. Below this, there's a 'Troubleshooting' section and an 'Example Docker Command' section. To the right, a terminal window is open, showing the output of a 'docker run' command. The logs show the download of the 'ghcr.io/open-webui/open-webui:ollama' image, with many layers being pulled from the Docker registry.

```
docker run -d -p 3000:8080 -v ollama:/root/.ollama -v open-webui:/app/backend/d
Users/rakidakixyz/.shrc:30: command not found: X
docker run -d -p 3000:8080 -v ollama:/root/.ollama -v open-webui:/app/backend/data --name ollama
Pulling from open-webui/open-webui:ollama
Digest: sha256:d6e7e2d0d6800229ddefdcfe55021d157a1689a6ee2ef16c42b083936ea388a
Status: Downloaded newer image for ghcr.io/open-webui/open-webui:ollama
a5b9beb417d885403281a1f867292de92cf5050293a138de720703aca3f61880
```

<https://github.com/open-webui/open-webui>

**Terminal für die Installation des „Docker-Pakets mit Ollama und Open WebUI als Bundle“:**

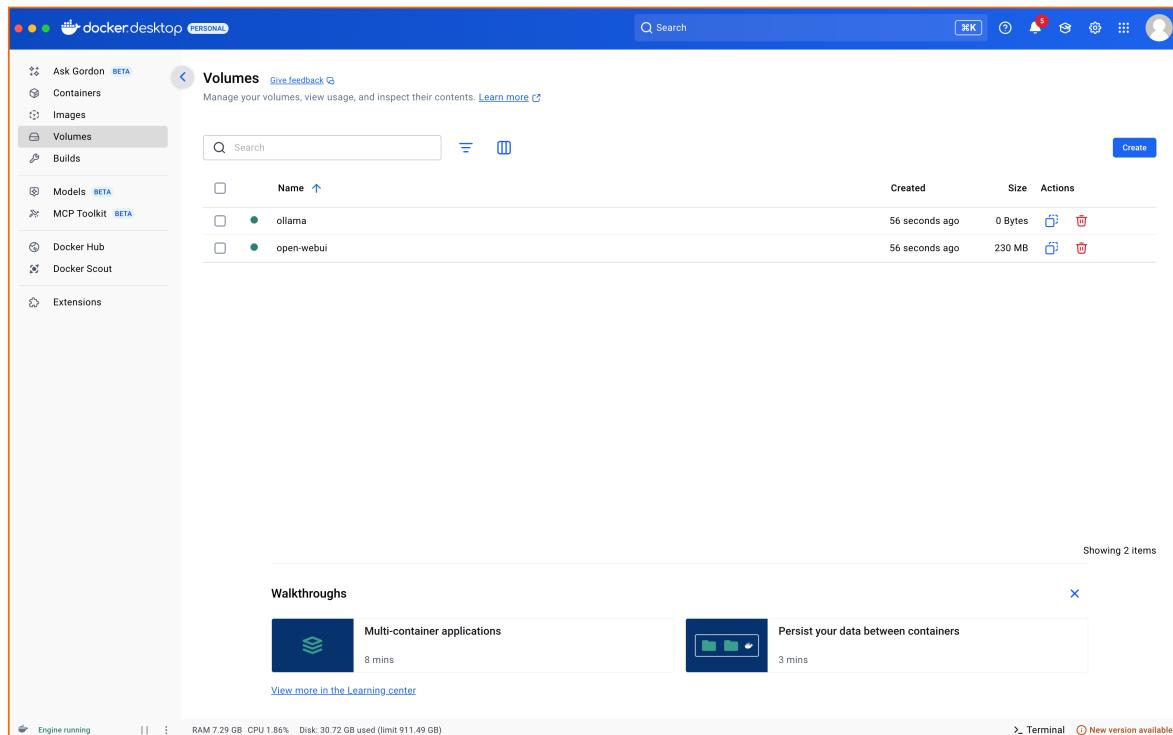
Windows Rechner mit Grafikkarte - also GPU's dann die Zeile für das Terminal mit GPU Support verwenden

Apple Rechner (MacBooks) die Zeile mit CPU Only Support verwenden (klappt trotzdem sehr gut ...

```
s ghcr.io/open-webui/open-webui:ollama
Unable to find image 'ghcr.io/open-webui/open-webui:ollama' locally
ollama: Pulling from open-webui/open-webui
4f4fb700ef54: Pull complete
48ac9344b5f8: Pull complete
162e2af9357: Pull complete
9bba408f3273: Downloading 312.5MB/877.7MB
12a55acfcaee: Pull complete
31fe0fdada64: Pull complete
43c2416e54fe: Pull complete
6047e63ff5f9: Pull complete
9b21e315b4c2: Pull complete
d9693f4bdbf8: Pull complete
691ced9925c1: Downloading 37.75MB/333.9MB
691ced9925c1: Downloading 117.4MB/2.004GB
9bb408f3273: Download complete
974c2c4bbf5a: Download complete
91abefb055a0: Download complete
310547e280b0: Download complete
998b9f57c27: Download complete
```

# Lokale KI-Umgebung (RAG, Chatbot, KI-Entwicklung)

🤔 Lokale Umgebung + online erreichbar?

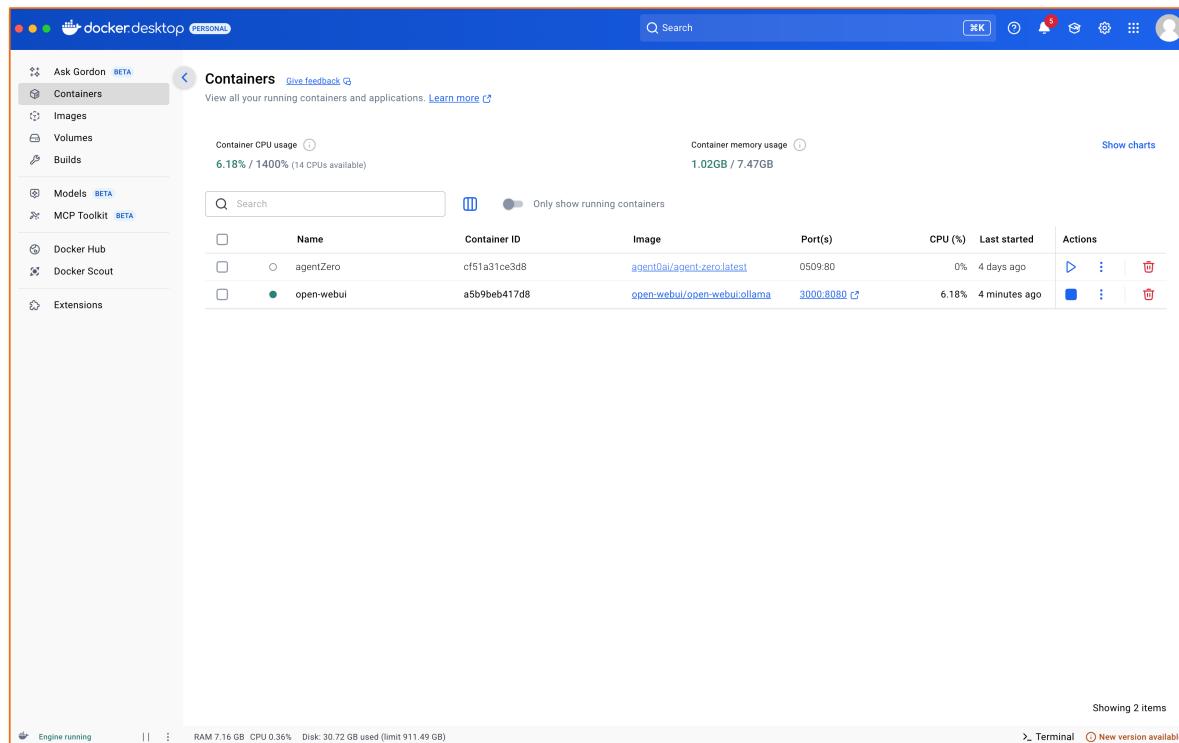


## Installation im Terminal erfolgreich:

Sobald im Terminal ohne Fehlermeldung die Installation erfolgreich abgeschlossen ist (Sie sehen keine Erfolgsmeldung im Terminal nur einen blinkenden Cursor für den nächsten Befehl), können Sie in Docker Desktop unter den Volumes die beiden installierten Pakete ollama und open-webui aufgelistet sehen.

# Lokale KI-Umgebung (RAG, Chatbot, KI-Entwicklung)

💡 Lokale Umgebung + online erreichbar?



## Docker Paket starten:

Wenn man in Docker Desktop dann auf „Containers“ geht, findet man alle virtuellen Container mit den jeweiligen Anwendungen, die Sie installiert haben.

Hier sieht man das Paket open-webui mit dem integrierten Ollama, welches schon gestartet ist und über einen Link in der Spalte Ports aufgerufen werden kann.

# Lokale KI-Umgebung (RAG, Chatbot, KI-Entwicklung)

💡 Lokale Umgebung + online erreichbar?

Dive into knowledge wherever you are

Get started

**Lokale Web-Anwendung Ollama + OpenWebUI**

Sobald man den Link angeklickt hat, startet der lokale Webserver localhost mit der Anwendung.

Man muss jetzt ein Account anlegen

**Get started with Open WebUI**

ⓘ Open WebUI does not make any external connections, and your data stays securely on your locally hosted server.

Name  
rakidakiXYZ

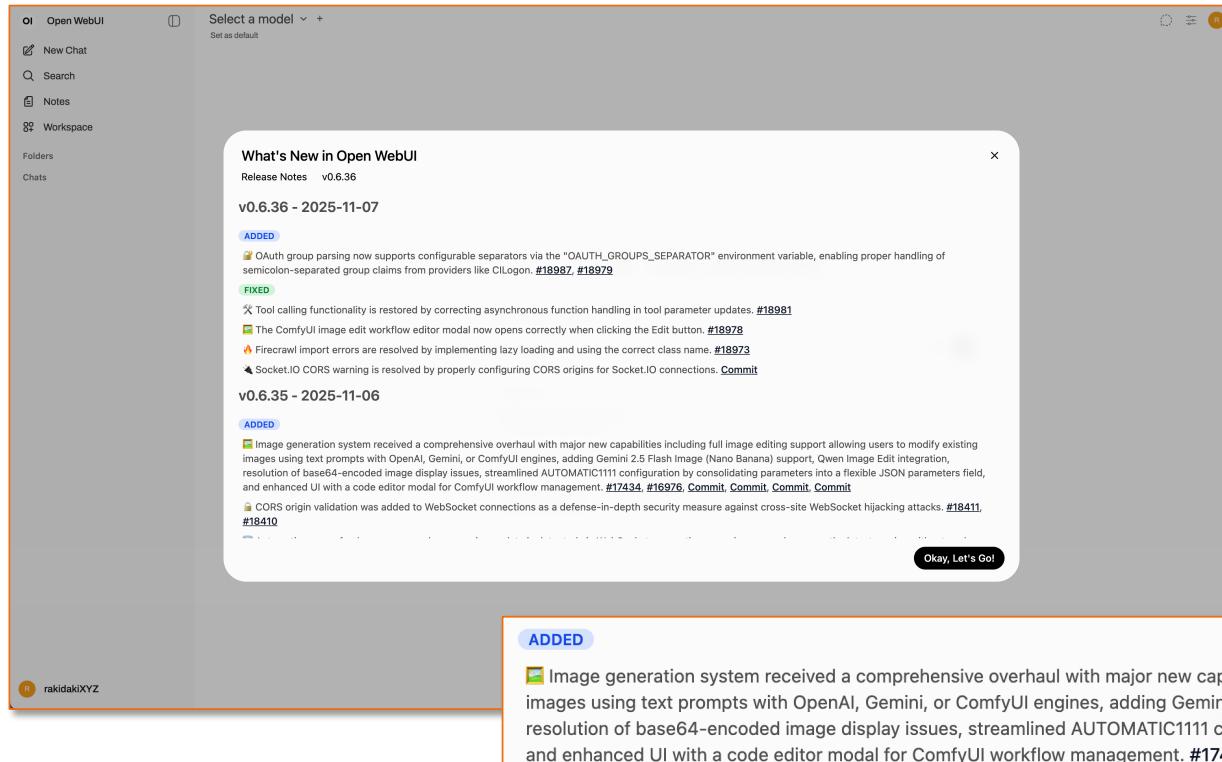
Email  
ralf@from-scratch.ai

Password  
.....

Create Admin Account

# Lokale KI-Umgebung (RAG, Chatbot, KI-Entwicklung)

💡 Lokale Umgebung + online erreichbar?



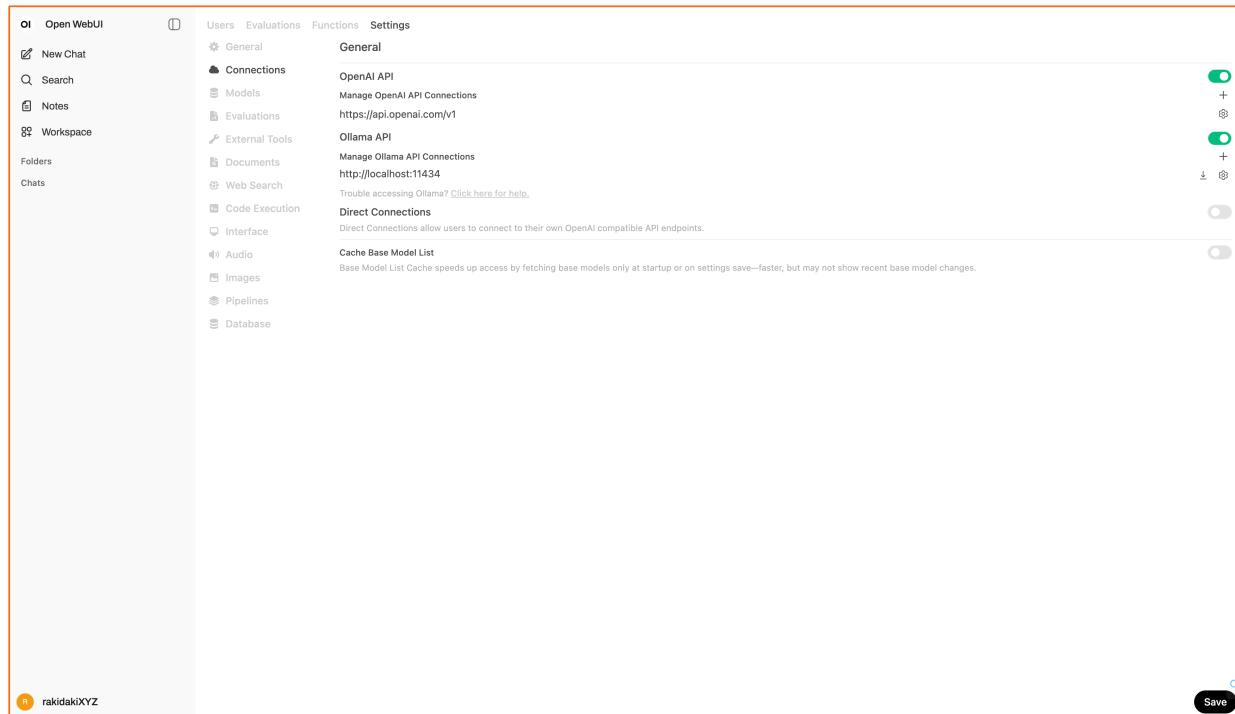
## Erfolgreicher Login:

Nach der ersten Anmeldung wird man dann erfolgreich in der Anwendung begrüßt und die letzten Änderungen werden im Changelog aufgelistet.

Hier sieht man auch, dass es eine aktive Community gibt, die dieses Produkt weiterentwickelt (Menge und Inhalt der Changes).

# Lokale KI-Umgebung (RAG, Chatbot, KI-Entwicklung)

🤔 Lokale Umgebung + online erreichbar?



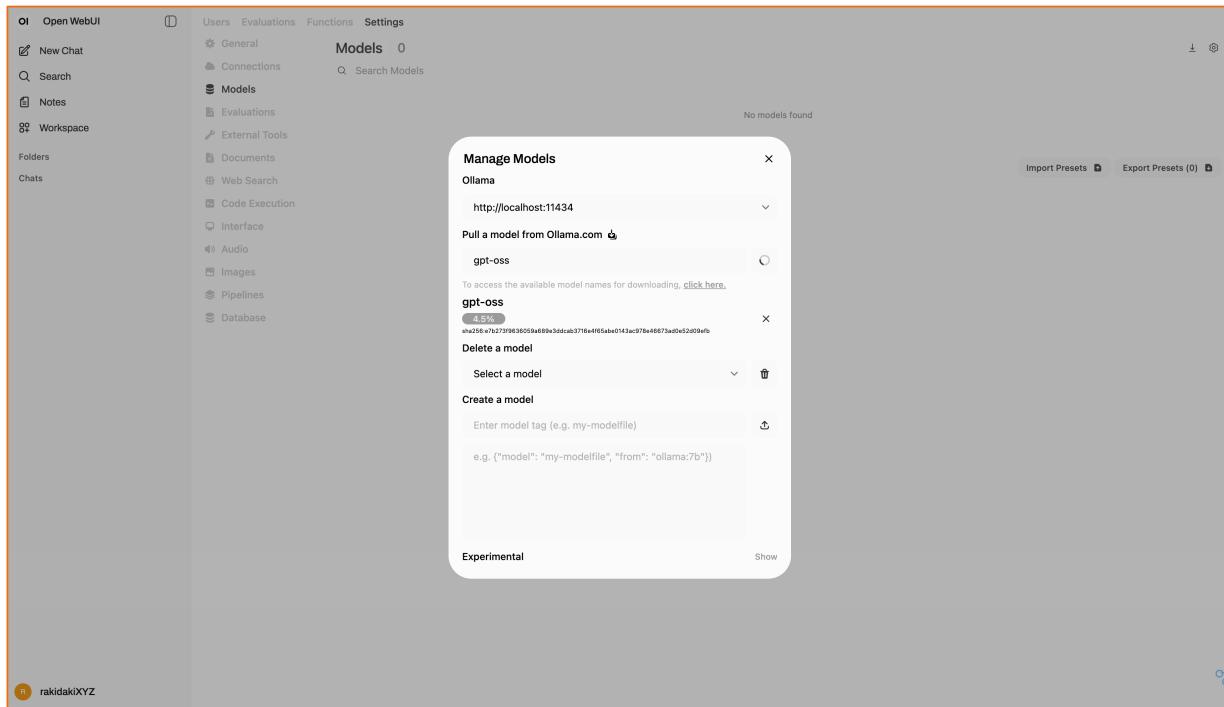
## Einstellungen

Über das Profil-Icon rechts oben kann man auf die Administration und die Einstellungen zugreifen.

Hier sieht man u.a. die Connections – also die Einstellungen zu OpenAI (per Default schon da) und die Verbindung zu Ollama (hier im gleichen Container – mit einer anderen Port Adresse).

# Lokale KI-Umgebung (RAG, Chatbot, KI-Entwicklung)

🤔 Lokale Umgebung + online erreichbar?



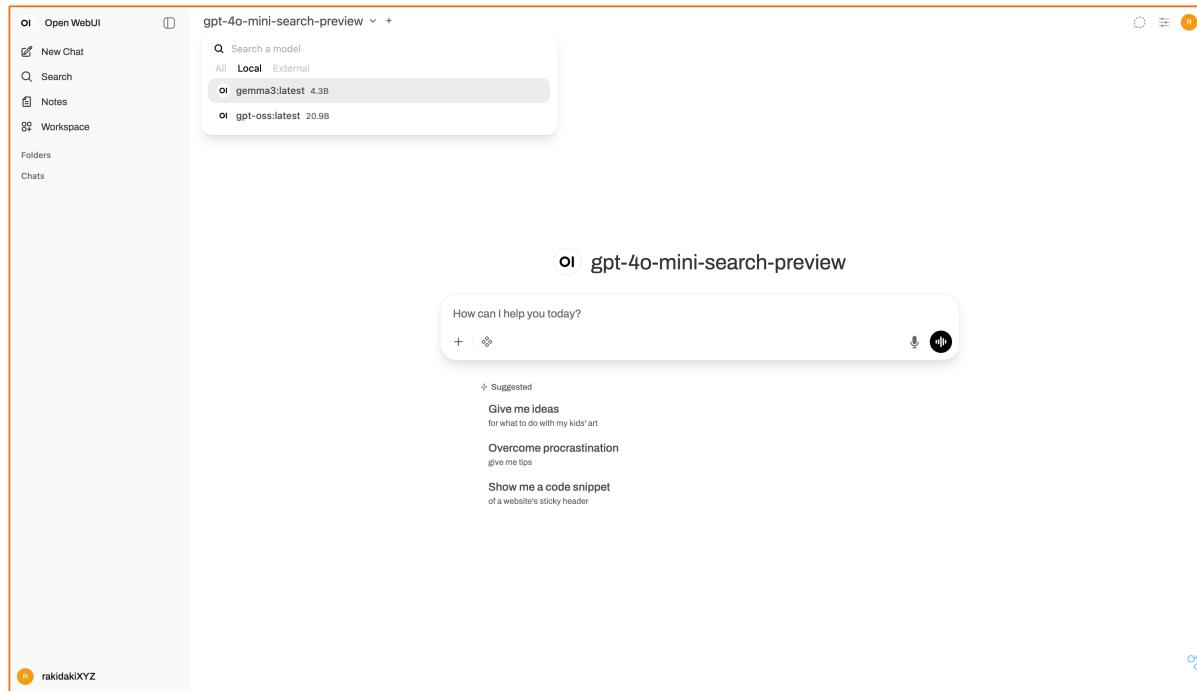
## Modelle (Sprachmodelle auswählen über Ollama)

Über den Eintrag Models kann man jetzt mit Ollama Sprachmodelle für die Nutzung in unserer lokalen Chat-KI Lösung auswählen und herunterladen.

Idealerweise über die Webseite von Ollama (Ollama.com) die Modelle suchen, den Namen kopieren und hier dann eingeben, um das jeweilige Modell herunterzuladen.

# Lokale KI-Umgebung (RAG, Chatbot, KI-Entwicklung)

🤔 Lokale Umgebung + online erreichbar?



## So, los geht's:

Wenn man die Verbindung zu den „großen“ LLMs über deren API Keys hergestellt hat und/ oder mit lokalen Sprachmodellen arbeitsfähig ist über Ollama (Download der Modelle), kann man wie bei OpenAI loslegen

## Chat



66

Ralf Kluth

@ [ralf@from-scratch.ai](mailto:ralf@from-scratch.ai)

X: fromscratchai

LinkedIn: <https://www.linkedin.com/in/rakidakixyz>

99