# Plannung/Idee Projekt Arbeit Public

- Selfhosted AI-Gateway – Technische Dokumentation Teil 1
- Synthetische testdaten

# Selfhosted AI-Gateway – Technische Dokumentation Teil 1

**Projekt:** Hybrides KI-Gateway mit lokalem Inferenz-Server und Cloud-Fallback **Status:** Prototyp / Proof of Concept **Stand:** 04.09.2026 **Version:** 0.2 **Autor:** *(dein Name)*

---

# 1. Zielsetzung

Aufbau eines zentralen Gateways, das KI-Anfragen von Mitarbeitenden entgegennimmt und je nach Komplexität, Datenschutzanforderung und Kostenlage an unterschiedliche Backends weiterleitet.

## 1.1 Ausgangsproblem

KI-Dienste werden üblicherweise über Pro-Kopf-Abonnements externer Anbieter bezogen. Daraus ergeben sich drei Nachteile:

- **Kosten skalieren linear** mit der Mitarbeiterzahl, unabhängig von der tatsächlichen Nutzung
- **Datenabfluss** an Dritte bei jeder Anfrage, auch bei trivialen Aufgaben
- **Keine Transparenz** über Nutzungsverhalten und Kostentreiber

## 1.2 Lösungsansatz

Ein hybrides Modell: Einfache Anfragen werden lokal auf eigener Hardware verarbeitet, nur rechenintensive oder qualitativ anspruchsvolle Anfragen gehen an Cloud-Anbieter – dort über API-Abrechnung nach Verbrauch statt Pro-Kopf-Abo.

## 1.3 Ziele

| Ziel | Messgröße |
|---|---|
| Kostenreduktion gegenüber Pro-Kopf-Abos | € / Monat, hochgerechnet |
| Datensparsamkeit | Anteil lokal verarbeiteter Anfragen |
| Nachvollziehbarkeit | Vollständiges Logging aller Anfragen |
| Akzeptable Antwortzeiten | Latenz p50 / p95 |

## 1.4 Nicht-Ziele des Prototyps

- Produktivbetrieb mit Echtdaten
- Hochverfügbarkeit / Ausfallsicherheit
- Feinjustierung der Modellqualität
- Integration in bestehende Unternehmenssysteme

---

# 2. Abgrenzung

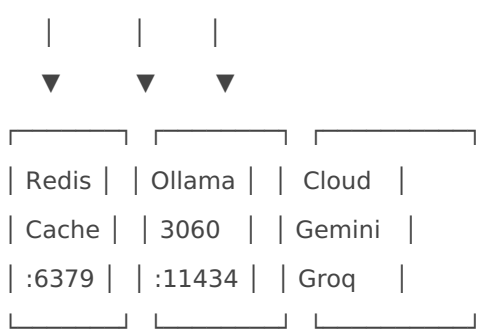
Der Prototyp entsteht **vollständig in privater Umgebung** auf eigener Hardware. Es werden keine Unternehmensdaten, keine internen Systeminformationen und keine realen Nutzerdaten verarbeitet. Sämtliche Testinhalte sind synthetisch.

---

# 3. Architektur

## 3.1 Übersicht






# 3.2 Verarbeitungskette

1. Nutzer sendet Anfrage über Open WebUI
2. Open WebUI leitet an LiteLLM weiter (ein gemeinsamer API-Key)
3. LiteLLM prüft Cache (Redis)
4. Bei Cache-Treffer: sofortige Rückgabe
5. Bei Cache-Miss: Routing-Entscheidung nach gewähltem Modell
6. Antwort wird geloggt, gecacht und zurückgegeben

# 3.3 Zentrale Entwurfsentscheidung

**Open WebUI spricht Ollama nicht direkt an.** Sämtlicher Verkehr läuft über LiteLLM. Andernfalls würde das Gateway umgangen – mit Verlust von Logging, Caching und Kostenerfassung.

Die native Ollama-Anbindung in Open WebUI bleibt deaktiviert.

# 3.4 Hardware

| System | Rolle | Ausstattung |
|---|---|---|
| Proxmox-Host | Virtualisierung | 16 Kerne, 16 GB RAM |
| AI-Server | Lokale Inferenz | NVIDIA RTX 3060 (12 GB VRAM) |

# 3.5 Container-Layout

| Container | Zweck | Ressourcen |
|---|---|---|
| `ct-ai-gateway` | Docker-Host: Open WebUI, LiteLLM, Redis | 4 Kerne, 6 GB RAM, Debian 12 |

**Hinweis zur Dimensionierung:** Open WebUI benötigt real 1–2 GB RAM. Zusammen mit LiteLLM und Redis wären 4 GB knapp bemessen; daher 6 GB.

# 4. Softwarekomponenten

| Komponente | Version | Funktion |
|---|---|---|
| Proxmox VE | *(einzutragen)* | Virtualisierung |
| Debian | 12 | Container-Betriebssystem |
| Docker / Compose | *(einzutragen)* | Container-Laufzeit |
| Open WebUI | *(einzutragen)* | Weboberfläche, Benutzerverwaltung |
| LiteLLM Proxy | *(einzutragen)* | Gateway, Routing, Logging |
| Redis | 7 | Antwort-Cache |
| Ollama | *(einzutragen)* | Lokale Inferenz |

**Cloud-Anbieter:**

| Anbieter | Zugang | Einsatz |
|---|---|---|
| Google AI Studio (Gemini) | Free Tier | Komplexe Anfragen |
| Groq | Free Tier | Latenzkritische Anfragen |

⚠ **Begriffsabgrenzung:** *Groq* bezeichnet den Hardware- und Inferenzanbieter (LPU-Architektur), nicht das Sprachmodell *Grok* von xAI. Die Bezeichnungen sind konsequent zu trennen.

---

# 5. Routing-Konzept

## 5.1 Eskalationsstufen

| Stufe | Ziel | Kriterium | Kosten |
|---|---|---|---|
| 0 | Redis-Cache | Identische Anfrage bereits beantwortet | 0 € |
| 1 | Ollama lokal | Standardaufgaben, Datenschutzbedarf | 0 € |
| 2 | Groq | Latenzkritisch | verbrauchsabhängig |
| 3 | Gemini | Hohe Komplexität, lange Kontexte | verbrauchsabhängig |

## 5.2 Modellpräsentation im Frontend

Nutzer sehen keine technischen Modellnamen, sondern aufgabenbezogene Bezeichnungen:

| Anzeigename | Backend | Einsatz |
|---|---|---|
| Standard | Ollama (lokal) | Alltagsanfragen |
| Schnell | Groq | Latenzkritisch |
| Komplex | Gemini | Anspruchsvolle Aufgaben |

Dies reduziert Fehlbedienung und lenkt die Nutzung implizit auf die kostengünstigste Stufe.

---

# 6. Datenschutz und Sicherheit

## 6.1 Grundsatz

Anfragen werden bevorzugt lokal verarbeitet. Eine Weiterleitung an Cloud-Anbieter erfolgt nur bei explizit gewählter Eskalationsstufe.

## 6.2 Kostenfreie Kontingente

Bei den kostenfreien Tarifen der Cloud-Anbieter werden Eingaben typischerweise zur Modellverbesserung verwendet. Für den Prototyp mit synthetischen Testdaten ist dies unkritisch.

**Für einen Produktivbetrieb wäre dies nicht zulässig.** Dort müssten kostenpflichtige Tarife mit vertraglich zugesicherter Nicht-Verwertung der Daten genutzt werden. Dieser Umstand ist ein zentrales Argument für den hybriden Ansatz und in der Wirtschaftlichkeitsbetrachtung zu berücksichtigen.

## 6.3 Netzwerk

Sämtliche Komponenten sind ausschließlich im lokalen Netz erreichbar. Kein Port-Forwarding, keine öffentliche Exposition.

## 6.4 Zugriffskontrolle

Open WebUI erfordert Registrierung; die erste angelegte Person erhält Administratorrechte. Selbstregistrierung wird deaktiviert, Konten werden administrativ angelegt. Damit ist jede Anfrage einem Konto zuordenbar.

## 6.5 Bekannte Einschränkung

Open WebUI und LiteLLM verwalten Benutzer bzw. Schlüssel getrennt voneinander. Im Prototyp nutzt Open WebUI **einen** LiteLLM-Key für alle Anfragen.

**Konsequenz:** Im LiteLLM-Log ist erkennbar, dass eine Anfrage aus Open WebUI stammt, nicht jedoch von welchem Nutzer. Eine nutzerbezogene Budgetierung ist damit im Prototyp nicht möglich.

Eine Kopplung beider Systeme ist als Ausblick vorgesehen (siehe Kapitel 9).

---

# 7. Messkonzept

## 7.1 Erhobene Kennzahlen

| Kennzahl | Quelle | Zweck |
|---|---|---|
| Input-/Output-Tokens | LiteLLM-Log | Kostenbasis |
| Latenz p50 / p95 | LiteLLM-Log | Nutzerakzeptanz |
| Cache-Hit-Rate | Redis / Log | Einsparpotenzial |
| Routing-Verteilung | LiteLLM-Log | Lokalanteil |
| Antwortqualität | Manuelle Bewertung | Machbarkeitsnachweis |

## 7.2 Qualitätsbewertung

| Note | Bedeutung |
|---|---|
| 5 | Vollständig verwendbar, keine Nacharbeit |
| 4 | Kleinere Anpassungen nötig |
| 3 | Brauchbar, aber deutliche Nacharbeit |
| 2 | Größtenteils unbrauchbar |
| 1 | Fehlerhaft oder unbrauchbar |

**Kriterium:** Eine Anfrage gilt als „lokal lösbar", wenn das lokale Modell mindestens Note 4 erreicht.

## 7.3 Kostenberechnung

Da kostenfreie Kontingente genutzt werden, entstehen im Prototyp keine realen Kosten. Die Berechnung erfolgt daher rechnerisch:

```
Kosten = (Input-Tokens × Preis_Input) + (Output-Tokens × Preis_Output)
```

Grundlage sind die öffentlichen Listenpreise der jeweiligen Produktivtarife. Preisstände sind mit Abrufdatum zu dokumentieren.

**Formulierungsvorschlag für die Auswertung:**

> Die Validierung erfolgte über die kostenfreien Kontingente der Anbieter. Die Kostenberechnung basiert auf den öffentlichen Listenpreisen der jeweiligen Produktivtarife, hochgerechnet auf den gemessenen Token-Verbrauch.

## 7.4 Technische Randbedingungen

Die kostenfreien Kontingente unterliegen Anfragebegrenzungen pro Minute und pro Tag. Für Testläufe sind daher erforderlich:

- Wiederholungslogik bei HTTP-429-Antworten
- Drosselung im Testskript
- Überwachung der Tageskontingente

---

# 8. Testfallkatalog

| Kategorie | Anzahl | Erwartete Stufe |
|---|---|---|
| Textzusammenfassung (kurz) | 15 | Lokal |
| Übersetzung | 15 | Lokal |
| Formulierungshilfe | 15 | Lokal |
| Code-Erklärung | 15 | Lokal / Groq |
| Faktenfragen | 10 | Cloud |

| Kategorie | Anzahl | Erwartete Stufe |
|---|---|---|
| Komplexe Analyse | 10 | Cloud |
| Wiederholungen (Cache-Test) | 20 | Cache |

**Gesamt:** 100 Testfälle

Alle Prompts sind synthetisch und enthalten keine realen Daten.

---

# 9. Offene Punkte

| # | Punkt | Status |
|---|---|---|
| 1 | IP-Adressierung AI-Server | offen |
| 2 | Docker-Installation auf Proxmox | offen |
| 3 | Vorhandene Ollama-Modelle | offen |
| 4 | Semantisches Caching | geplant |
| 5 | Monitoring (Prometheus / Grafana) | geplant |
| 6 | Preisstände Cloud-Anbieter erheben | offen |
| 7 | Kopplung Open-WebUI-Nutzer ↔ LiteLLM-Keys | Ausblick |

---

# 10. Änderungshistorie

| Datum | Version | Änderung |
|---|---|---|
| 04.09.2026 | 0.1 | Ersterstellung |
| 04.09.2026 | 0.2 | Open WebUI als Client-Ebene ergänzt; Kapitel 2, 5.2, 6.4, 6.5, 7.4 neu |

# Synthetische testdaten

## Kategorie A: Einfach (`standard-klein` / `standard`)

**1. Rechtschreibkorrektur**

> Korrigiere Rechtschreibung und Grammatik in folgendem Text, ändere sonst nichts:
>
> Sehr geehrte Frau Weber, hiermit bestätige ich ihnen den Erhalt ihrer Anfrage vom 3. Septmber. Wir werden die Unterlagen prüfen und melden uns spätestens bis ende nächster Woche bei ihnen zurück. Bei rückfragen stehe ich ihnen gerne zur verfügung.

**2. Umformulierung / Tonalität**

> Formuliere diese Nachricht höflicher und professioneller:
>
> Ich hab dir letzte Woche schon geschrieben dass die Zugangsdaten fehlen. Kannst du das jetzt endlich mal schicken? Ohne komm ich hier nicht weiter.

**3. Zusammenfassung**

> Fasse folgenden Text in maximal drei Sätzen zusammen:
>
> Im Rahmen der Systemumstellung am kommenden Wochenende wird das ERP-System von Freitag 18:00 Uhr bis Montag 06:00 Uhr nicht erreichbar sein. Betroffen sind alle Module inklusive Lagerverwaltung, Bestellwesen und Rechnungsstellung. Mitarbeitende der Logistik werden gebeten, offene Kommissionierungen bis Freitag 16:00 Uhr abzuschließen. Für dringende Fälle steht eine Notfall-Hotline unter der Durchwahl 4711 zur Verfügung. Das Ticketsystem bleibt durchgehend erreichbar, Anfragen werden jedoch erst ab Montag bearbeitet.

---

# Kategorie B: Mittel (`standard` / `schnell`)

**4. E-Mail aus Stichpunkten**

Schreibe eine E-Mail an einen Kunden aus folgenden Stichpunkten:
- Liefertermin verschiebt sich von 15.09. auf 29.09.
- Grund: Lieferengpass beim Vorlieferanten
- Ersatzweise Teillieferung am 18.09. möglich
- Um Rückmeldung bis Freitag wird gebeten

**5. Übersetzung mit Fachkontext**

Übersetze ins Englische, geschäftlicher Kontext:

Bitte beachten Sie, dass die Gewährleistungsfrist für die gelieferten Komponenten 24 Monate ab Inbetriebnahme beträgt. Verschleißteile sind hiervon ausgenommen. Eine Verlängerung der Frist bedarf der Schriftform.

**6. Strukturierte Extraktion**

Extrahiere aus folgendem Text alle Termine, Verantwortlichen und Aufgaben als Tabelle:

Kurzprotokoll Jour Fixe: Herr Braun übernimmt die Abstimmung mit dem Betriebsrat, Rückmeldung bis 12.09. Frau Kaya bereitet die Präsentation für den Lenkungskreis vor, Termin ist der 19.09. um 14 Uhr. Die Testphase startet nach Freigabe durch die IT, verantwortlich ist Herr Özdemir, Zieltermin Ende September. Nächster Jour Fixe am 26.09.

---

# Kategorie C: Komplex (`komplex`)

**7. Analyse mit Abwägung**

Wir überlegen, unsere Dokumentenablage von einem Netzlaufwerk auf SharePoint umzustellen. Nenne die drei wichtigsten Risiken dieser Migration, jeweils mit einer konkreten Gegenmaßnahme. Berücksichtige, dass rund 40 Mitarbeitende betroffen sind und der Großteil wenig IT-Affinität hat.

**8. Mehrschrittige Logik**

Ein Kunde hat am 02.09. eine Bestellung über 12 Artikel aufgegeben. Am 04.09. wurden 8 Artikel geliefert, am 09.09. weitere 3. Ein Artikel wurde am 05.09. storniert. Der Kunde reklamiert nun, dass die Rechnung über 12 Artikel ausgestellt wurde. Erkläre, welcher Betrag korrekt wäre und welche Schritte zur Klärung nötig sind. Formuliere anschließend eine kurze Antwort an den Kunden.