**KalliGPT Projekthandbuch**

Willkommen im digitalen Projekthandbuch für das KalliGPT-System – hier dokumentieren wir alle zentralen Komponenten, Setups, Workflows und Best Practices rund um die Entwicklung, Pflege und Weiterentwicklung unserer semantischen Wissensplattform.

### Inhaltsverzeichnis (wird nachgeführt)

1. Render Setup Checkliste
2. Supabase Struktur & Zugriff
3. Gradio Frontend-Logik
4. Einbindung von ElevenLabs
5. API Keys & .env Management
6. Erweiterungsplanung (Vektorsuche, Pagination, etc.)

### **Kapitel 1: Render Setup Checkliste**

**Checkliste: Neues Render.com-Projekt mit Auto-Deploy**

✅ Schritt 1: Neues Webservice in Render erstellen

1. Login bei <https://render.com>
2. Klick auf **“New” > “Web Service”**
3. Wähle “Deploy from a Git repository”
4. GitHub-Zugriff erlauben (falls nicht bereits erfolgt)
5. Neues Repo auswählen (z. B. kalligpt)

✅ Schritt 2: Grundeinstellungen setzen

| Einstellung | Wert |
| --- | --- |
| Name | z. B. kalligpt-app |
| Branch | main |
| Build Command | pip install -r requirements.txt |
| Start Command | python kalli\_frontend.py (anpassen!) |
| Runtime | Python 3.11 oder laut runtime.txt |

✅ Schritt 3: Port-Konfiguration im Python-Code

Falls du Gradio nutzt, im Python-Script (z. B. kalli\_frontend.py) sicherstellen:

iface.launch(server\_name="0.0.0.0", server\_port=int(os.environ.get("PORT", 7860)))

✅ Schritt 4: Environment Variables setzen

Im Tab **Environment**:

* SUPABASE\_URL
* SUPABASE\_SERVICE\_ROLE
* OPENAI\_API\_KEY
* ggf. DEBUG\_MODE

✅ Schritt 5: Auto-Deploy aktivieren

1. Reiter **“Settings” > “Deploy”**
2. Haken setzen bei “Auto-Deploy”
3. “Pull Request Previews” ggf. deaktivieren

✅ Schritt 6: (Optional) Custom Domain verknüpfen

* “Custom Domains” > Domain hinzufügen (z. B. kalligpt.de)
* DNS bei IONOS anpassen (A-Record oder CNAME)
* SSL-Zertifikat aktivieren (Let’s Encrypt)

✅ Schritt 7: Test & Finalisierung

* Projekt einmal manuell deployen
* Frontend aufrufen und testen
* Logs im Auge behalten
* Wenn alles passt:
  + ☑ Diagnose-Button entfernen (optional)
  + ☑ Style-Feinschliff (optional)

**Fertig ist dein stabiles Render-Deployment mit GitHub-Auto-Deploy!**

### Kapitel 2: Supabase Struktur & Zugriff

#### Tabellenstruktur

* antraege – enthält Anträge mit Metadaten und Embedding-Vektor
* anfragen\_muendlich – mündliche Anfragen inkl. Datum & Inhalt
* anfragen\_klein – kleine BVV-Anfragen
* anfragen\_gross – große BVV-Anfragen

#### Rollen & Rechte

* anon – Lesezugriff über Frontend aktiviert (RLS aktiviert)
* service\_role – Schreibrechte für Embedding-Script & Adminfunktionen

#### RLS-Regeln (Row-Level-Security)

* Zugriff auf Tabellen nur über definierte Policies
* Policy für anon erlaubt nur select mit Einschränkung (z. B. published = true)
* Admin-Scripte nutzen service\_role, z. B. via Python .env