

SMA-O

RAG-System Dokumentation

Sebastian Krüsmann

Co-Founder & Chief Information Officer (CIO) | NeoMINT GmbH



SMAO-Gruppe 1

Muzamil El-Bashir, 3003194

Mohamad Yamen Al Salkini, 2211216

Georg Bitter, 3000723

Ghada Awad, 2211780

Dara Soun, 2123000

Leon Ehrenhard-Dickescheid, 2122451

Inhaltsverzeichnis

1. Überblick.....	3
2. Systemarchitektur (Kurzfassung).....	3
3. Voraussetzung.....	4
4. Datenfluss.....	4
4.1 Ingestion (Dokumentaufnahme)	4
4.2 Query/Chat (Retrieval & Antwort).....	5
5. Datenhaltung.....	5
5.1 PostgreSQL (Metadaten-Speicherung)	5
5.2 Qdrant (Vektordatenbank)	6
6. Zotero (Setup und Netzung)	6
6.1 Setup (Zotero):	6
6.2 Nutzung:	7
6.3 Zotero Datenabruf	7
7. Docling (Chunking).....	8
8. Ollama (Embeddings).....	8
9. n8n Workflow Design	8
10. Verifikation (Gegentest)	9
10.1 System Checks (nach Start)	9
10.2 Pipeline Test (minimal).....	9
10.3 Akzeptanzkriterien	9
11. Häufige Fehler	10

1. Überblick

Dieses Projekt implementiert ein lokal lauffähiges Retrieval-Augmented-Generation (RAG) System.

PDF-Dokumente werden aus einer Zotero-Library ingestiert, in Text-Chunks zerlegt, mit Embeddings versehen und in Qdrant gespeichert. Chunk-Metadaten werden zusätzlich in PostgreSQL persistiert, um Nachvollziehbarkeit und Auswertung zu ermöglichen.

Zielzustand:

- Reproduzierbare, lokale Pipeline von
- Dokument -> Chunk -> Embedding -> Vektor-DB
- Nachvollziehbare Speicherung von Metadaten (Titel, Seitenbereiche, Tagging, Timestamps)
- Grundlage für einen Query-/Chat-Workflow (Retrieval und Antwortgenerierung)

Nicht-Ziel / Abgrenzung:

- Kein vollständiges Frontend-Produkt (nur Workflows/Services)
- Kein vollautomatisches "nur neue Dokumente"-Ingestion

2. Systemarchitektur (Kurzfassung)

Datenquelle

- Zotero: Literaturverwaltung & PDF-Quelle

Orchestrierung

- N8n: Orchestrierung des gesamten Workflows

Dokumentenverarbeitung

- Docling: PDF-Chunking (sync/async)

Embeddings

- Ollama: Embedding-Generierung

Speicher

- PostgreSQL: Persistenz von Chunk-Metadaten
- Qdrant: Vektordatenbank für semantische Suche

Betrieb

- Docker Compose: Lokales Setup & Service-Orchestrierung

3. Voraussetzung

- Docker und Docker Compose (lokal)
- Zugang zu Zotero (Account und Library)
- Zotero API Key
- Zugriff auf n8n (Self-hosted im Compose)
- Ollama (lokal bzw. auf Docker)

4. Datenfluss

4.1 Ingestion (Dokumentaufnahme)

Ablauf

1. Zotero Items listen (Parent Items)
2. Attachements (PDFs) je Parent ermitteln
3. PDF herunterladen
4. Chunking via Docling

5. Chunks formatieren
6. Metadaten in Postegre SQL schreiben
7. Embeddings via Ollama berrechnen (Batching)
8. Embeddings und Payload in Qdrant Speichern

4.2 Query/Chat (Retrieval & Antwort)

Ablauf

1. User-Frage
2. Embedding der Query
3. Qdrant Vector Search
4. Kontextaufbau
5. Antwortgenerierung (LLM) mit klaren Regeln (keine Halluzinationen, nur Kontext)

5. Datenhaltung

5.1 PostgreSQL (Metadaten-Speicherung)

PostgreSQL wird über ein Init-Skript initialisiert:

001_init.sql:

```
CREATE TABLE IF NOT EXISTS public.dokumente_log (
  id BIGSERIAL PRIMARY KEY,
  chunkId TEXT NOT NULL,
  titel TEXT NOT NULL,
  seiteStart INT,
  SeiteEnde INT,
  inhalt TEXT,
  tag TEXT,
  created_at TIMESTAMPTZ DEFAULT now()
);
```

Diese Tabelle speichert alle verarbeiteten Chunks inklusive Metadaten.

5.2 Qdrant (Vektordatenbank)

Die Qdrant Collection wird beim Start automatisch über docker-compose initialisiert:

```
curl -s -X PUT http://qdrant:6333/collections/chunks \
  -H "Content-Type: application/json" \
  --data-raw '{
    "vectors": { "size": 768, "distance": "Cosine" }
  }'
```

- Vektorgroße: 768
- Distanzmetrik: Cosine

6. Zotero (Setup und Netzung)

6.1 Setup (Zotero):

1. Zotero Account anlegen
2. Zotero App herunterladen oder im Browser ausführen
3. Zotero Connector Extension für Firefox installieren
4. Zotero API Key in den Settings erzeugen und sicher speichern

(Optional:) Zotero Gruppe erstellen, damit mehrere auf die gleiche Library zugreifen können.

6.2 Nutzung:

- PDFs werden über den Zotero Connector in die Library importiert
 - Um PDFs in Zotero hinzuzufügen, braucht man die App-Version (Achtung: Die App und Zotero Account synchronisieren)
 - Einsicht in die Library benötigt man nur den Browser Interface
- In n8n:
 - Group Library -> Nutzung der Group ID
 - Persönliche Library -> Nutzung der User ID

6.3 Zotero Datenabruf

Unterscheidung der Zotero-Items:

- Parent Item:
 - Enthält Metadaten (Titel, Autoren, Jahr, etc.)
- Attachment
 - Enthält die zugehörige PDF-Datei

API Requests

- Parent Items:
 - <https://api.zotero.org/groups/<groupid>/items/top>
- Attachment
 - <https://api.zotero.org/groups/<groupid>/items/children>
- PDF-Download:
 - <https://api.zotero.org/groups/<groupid>/items/<key>/file>

Jede Request benötigt:

- Zotero API Key
- Zotero API Version

7. Docling (Chunking)

Verarbeitungslogik nach Dateigröße:

- Kleine PDFs (< 2MB) Synchron
 - <http://docling:5001/v1/chunk/hybrid/file>
- Große PDFs (\geq 2 MB) Asynchron
 - <http://docling:5001/v1/chunk/hybrid/file/async>

8. Ollama (Embeddings)

Base URL:

<http://ollama:11434>

- Batch- Verarbeitung: standardmäßig 3 Chunks pro Batch
- Bei Fehlern pro Batch: Retry
- Bei Dauerhaften Fehler: Chunk als "failed" markieren, statt Workflow komplett zu stoppen

Voraussetzung:

- Embedding-Modell vorhanden (typisch: nomic-embed-text) (siehe häufige Probleme)
- Chat/LLM-Modell vorhanden

9. n8n Workflow Design

Die Ingestion ist in zwei Workflows aufgeteilt, um pro PDF einen sauberen, isolierten Verarbeitungslauf zu gewährleisten und die Pipeline übersichtlicher zu halten:

- Main-Workflow: Holt PDFs aus Zotero, führt das Chunking aus (Docling) und übergibt pro PDF ein Chunk-Array an den Sub-Workflow.
- Sub-Workflow: Verarbeitet das Chunk-Array: speichert den Chunk-Metadaten in PostgreSQL, erzeugt Embeddings (Batching) über Ollama und speichert Vektoren und Payload in Qdrant.

Nach jeden Sub-Workflow-Lauf wird der Ausführungskontext verworfen. Jeder neue Aufruf startet unabhängig.

- Im Sub-Workflow wird ein globaler Counter inkrementiert und für eindeutige Chunk-IDs genutzt.

10. Verifikation (Gegentest)

10.1 System Checks (nach Start)

1. Qdrant Colletion existiert
2. Postgres Tabelle existiert
3. N8n Workflow startet ohne Credential-Fehler

10.2 Pipeline Test (minimal)

1. PDF in Zotero importieren (klein, < 2 MB)
2. Ingestion Workflow starten
3. Erwartung :
Postgres: neue Rows in *dokumente_log*
Qdrant: neue Points in Collection *chunks*
4. Query: Frage stellen, die im PDF beantwortbar ist

10.3 Akzeptanzkriterien

- Ein Dokument wird vollständig verarbeitet, ohne dass der Workflow unkontrolliert abbricht.
- Fehler werden nachvollziehbar geloggt (Dokument erkennbar)
- Qdrant enthält Embeddings und Payload vollständig
- Postgres enthält pro Chunk Metadaten

11. Häufige Fehler

Postgres

- Chunks into Postgres "not any credentials set"

Lösung:

- Daten von .env übernehmen und in "Credential to connect with" einfügen.

Ollama

"Problem in node 'Embeddings1'"

The ressource you are requesting could not be found

Das Embedding-Modell ist in Ollama noch nicht installiert.

Lösung:

Modell pullen (runterladen) in den Ollama-Container

- Terminal "Docker/" Ordner:
docker compose exec ollama ollama pull nomic-embed-text
- prüfen, ob es da ist:
docker compose exec ollama ollama list

"nomic-embed-text" sollte in der Liste sein

Embeddings Langsam/Fehler:

Lösung:

Dokumentation-RAG

- Batch-Größe reduzieren
- Retries aktivieren

RAG-Chat / Web UI

Langsam / verzögert (abhängig von Internet/Netzwerk):

- Der RAG-Chat ist in der Nutzung netzwerkabhängig. Bei schlechter Verbindung kann das UI spürbar langsamer reagieren (z.B. beim Laden/Übertragen von Requests).

Lösung:

- stabile Internet-/ Netzwerkverbindung nutzen oder Test im lokalen Netzwerk durchführen

RAG-Chat funktioniert nicht mit aktiviertem Adblocker:

Ein aktiver Adblocker kann Requests blockieren, wodurch dann nur diese Antwort kommt:

- Es ist ein Fehler aufgetreten. Bitte versuchen Sie es erneut

Lösung:

- Adblocker für die verwendete WEB-UI deaktivieren oder eine Ausnahme (Whitelist) hinzufügen.

PDF-Download leer/fehlt

Lösung:

1. Attachment-Key falsch
2. Item hat kein PDF-Attachment