DocGenerator

Sichere Suche & Berichtsentwürfe aus technischen Reports (Rapid Prototype)

Ausgangslage

Problem

Heute

- Wissen steckt in PDFs und Ordnerstrukturen
- Suche kostet Zeit (Dateinamen, Stichworte, Lesen)
- Doppelarbeit bei ähnlichen Berichten
- Skalierung: mehr Reports ⇒ linear mehr Aufwand

Pain

Zeit, Qualität, Wiederverwendung

Zielbild

In Minuten statt Stunden

Was wird besser?

- Semantisch finden: auch ohne exakte Schlagworte
- Entwürfe aus Quellen: kurz, strukturiert, nachvollziehbar
- Immer mit Quellen/Seiten/Nachweis
- Angebotsprozess: Textbausteine/Checklisten (keine Kalkulation aus Reports)

Leitsatz: KI liefert Vorschläge – Entscheidungen & Freigaben bleiben beim Menschen.

Was ist DocGenerator?

3 Kernfunktionen

1) Ingest

PDFs einlesen & strukturieren

Abschnitte + Metadaten

2) Search

Semantische Suche

Treffer + Quellen

3) Generate

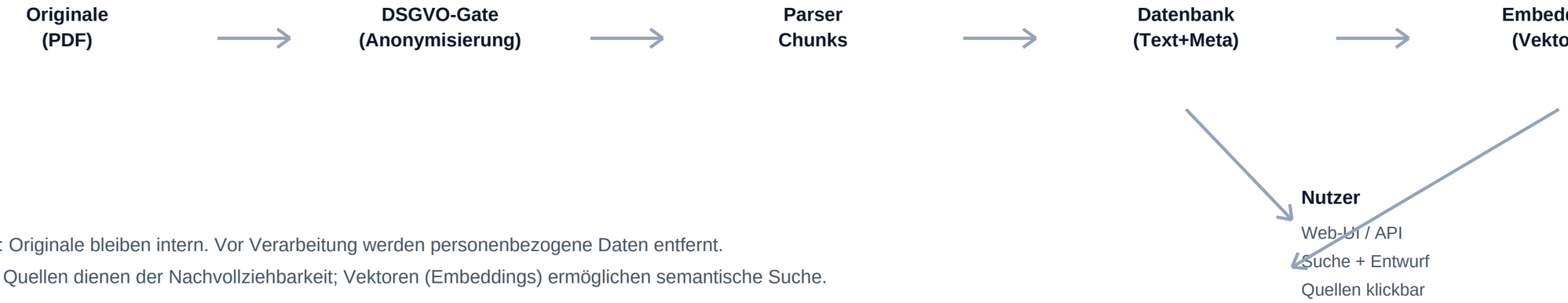
Berichtsentwurf

aus Top-Fundstellen

Architektur (management-tauglich)

lokal betreibbar • skalierbar

Datenfluss (vereinfacht)



Erklärung: Originale bleiben intern. Vor Verarbeitung werden personenbezogene Daten entfernt.
Chunks + Quellen dienen der Nachvollziehbarkeit; Vektoren (Embeddings) ermöglichen semantische Suche.

Bedenken: Halluzinationen

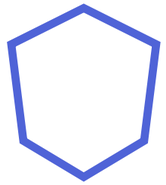
Wie wir Vertrauen schaffen

Sorge

- KI erfindet plausible Aussagen
- Unklare Nachvollziehbarkeit

Design-Gegenmaßnahmen

- Suche liefert Originaltext-Ausschnitte
- Generierung basiert auf Top-Fundstellen
- Quellenpflicht (Dokument/Seite)
- Output als Entwurf (Human-in-the-loop)



Qualität & Nachvollziehbarkeit

messbar • prüfbar

Qualitätssicherung

- Nutzer prüft Treffer + Quellen vor Nutzung
- Entwurf ≠ Freigabe (Fachprüfung bleibt Pflicht)
- Erfolgsmessung im Feldversuch: Zeit, Trefferqualität, Fehlerklassen

Betrieb & Datenschutz

lokal / im eigenen Rechenzentrum

Schutz vor Verlust & Fremdzugriff

- Dokumente verbleiben intern (keine externe Plattform notwendig)
- Nachvollziehbarkeit: Chunks speichern Quelle (Dokument/Seite) und werden archiviert
- Schutz: Datenbank + Embeddings regelmäßig sichern (Backup/Archiv)
- Schutz vor Fremdzugriff: Zugriffskontrollen; optional Verschlüsselung im Ruhezustand

Rapid Prototype – aber produktionsnah

professioneller Unterbau

Rapid Prototype (heute)

- kleiner Scope
- schnelle Iteration
- messbarer Nutzen

Produktionsnaher Unterbau

- klare Module
- Docker-fähig
- erweiterbar (Rechte/Logging)

Zeitersparnis

Business Case (messbar)

Nutzen

- Schneller finden: Sekunden statt Minuten
- Schneller starten: Entwurf aus Quellen statt Copy/Paste
- Messplan: Vorher/Nachher Zeit + Nutzerfeedback

Roadmap

Migration zuerst • dann Test + Feldversuch

Phasen + Freigabe-Punkte

Schritt 1

Portierung auf
DB Hardware
(im eigenen RZ)

Gate: IT/Security

Schritt 2

Test mit Original-
Dokumenten
1 Woche

Gate: Datenschutz

Schritt 3

Feldversuch
10 Nutzer
4 Wochen

Gate: Review

Schritt 4

Rollout
Nachbarabteilungen

Gate 1: Entscheidung

Freigabe Phase 1

Freigabe (risikoarm, messbar)

- Umfang: ~2 Berichtsjahre (~400 Dateien)
- Ziel: Zeitersparnis + Qualitätskennzahlen + Fehlerklassen
- Ergebnis: Messbericht (Zeit/Qualität) + Empfehlung für Migration & Testgruppe

Portierung: Hostinger → DB Hardware

Einschätzung

Durchführbarkeit: hoch

- Container-/Service-Design ist für Betrieb im eigenen Rechenzentrum geeignet
- Zeitaufwändig typischerweise: Security/Compliance, Netzwerk/TLS, Berechtigungen, Modellbetrieb
- Empfehlung: IT/Security früh einbinden, klare Betriebsprozesse (Backup/Restore, Updates)

Feldversuch

kontrolliert • 10 Nutzer

Pilot-Setup

- 10 Nutzer, definierte Use-Cases (Suche, Entwurf, Quellenprüfung)
- Feedback-Schleife + KPI-Review
- Ergebnis: Entscheidung für Rollout & Prioritäten

Funktionsausbau

nach Pilot priorisiert

Beispiele

- Konservativer Modus: nur zitieren + zusammenfassen
- Quellen klickbar, Export (PDF/Word)
- Rollen/Rechte: Suche vs. Generierung
- Feedbackbutton: hilfreich/falsch/Quelle fehlt

Software & Lizenzen

Transparenz für den Einsatz im Unternehmen

Stack (Rapid Prototype)

- FastAPI/Uvicorn (API) – Open Source
- SQLite (DB) – Open Source / Public Domain
- Ollama (Model-Server) – Open Source (Betrieb intern)
- Modelle (Embeddings/LLM) – Lizenz pro Modell prüfen
- PDF-Tools (pdfplumber/pypdf/reportlab) – Open Source

Hinweis: Vor Rollout werden Modell-Lizenzen und interne Standards formal geprüft.