— // DOCUMENT INTELLIGENCE · SUSTAIND

Aus *widerspenstigen* Dokumenten wird strukturiertes Wissen.

ARCHITEKTUR & IMPLEMENTIERUNG EINES PRODUKTIONSREIFEN DOCUMENT-INGESTION-WORKERS · PYTHON / CLAUDE / AWS BEDROCK

sustaind, eine Compliance-Plattform — von Nachhaltigkeitsregularien über Produkt-Compliance bis zu Sicherheitsdatenblättern — beantwortet regulatorische Fragen aus den Dokumenten ihrer Kunden. Diese Dokumente sind gescannte PDFs, alte Word-Dateien, Tabellen mit zehntausend Zeilen und Screenshots — Material, das kein Parser von der Stange zuverlässig liest. Ich habe den Worker entworfen und gebaut, der sie alle verarbeitet: eine fehlertolerante Pipeline, die aus beliebigen Uploads semantisch gruppierte Chunks macht — durchsuchbar per Vektorsuche und bis auf den Pixel zur Quelle rückverfolgbar.

01

Ein Worker, jedes Format

PDF · DOC/DOCX · XLSX · PNG/JPEG

02

Drei Extraktions-Engines

Layout, OCR, LLM-Vision — pro Seite vereint

03

Semantisches Chunking

LLM-gruppiert, hierarchisch, quellgenau

04

Sicher im Fehlerfall

Circuit Breaker, Retries, Dead-Letter-Queue

01 KONTEXT

Compliance-Antworten stecken in Kundendokumenten.

sustaind unterstützt Unternehmen bei der regulatorischen Compliance: welche Gesetze gelten, ob Produkte konform sind, wie Sicherheitsdatenblätter zu pflegen sind. Die Belege für jede Antwort liegen in den Uploads der Kunden: Auditberichte, Richtlinien, Zertifikate, Lieferantenfragebögen, Sicherheitsdatenblätter. Bevor die Retrieval- und LLM-Schichten der Plattform damit arbeiten können, muss etwas diese Dateien in saubere, strukturierte, durchsuchbare Daten verwandeln.

Mein Auftrag war Entwurf und Implementierung genau dieser Komponente: **Ingestor**, der Python-Service zwischen Datei-Upload und dem Vektor-Store der Plattform. Er verantwortet alles dazwischen — Formaterkennung, Extraktion, OCR, visuelles Verstehen, semantisches Chunking und Embedding-Generierung.

---

02 DAS PROBLEM

Dokumente aus der Praxis wehren sich.

Der naive Ansatz — Text extrahieren, in Stücke schneiden — scheitert an fast allem, was Unternehmen tatsächlich hochladen:

/01 **Scans ohne Textebene**
Zertifikate und unterschriebene Auditberichte kommen als fotografierte oder gescannte Seiten. Es gibt keinen Text zu extrahieren — nur Pixel.

/02 **Bedeutung, die in Grafiken steckt**
Diagramme, Stempel, Organigramme und eingebettete Abbildungen enthalten Informationen, die reine Textextraktion stillschweigend verwirft.

/03 **Tabellen mit Struktur, die nur Menschen sehen**
Ein Sicherheitsfragebogen kann fünf logische Abschnitte in einer 70-zeiligen Tabelle enthalten — getrennt durch nichts als eine fett gesetzte Zeile.

/04 **Legacy-Formate**
Binäre .doc-Dateien von 2009 tauchen in Compliance-Workflows noch immer auf — und müssen korrekt gelesen werden.

/05 **Kontext, den naives Chunking zerstört**
Splitting nach fester Länge schneidet Tabellen mittendurch und trennt Antworten von ihren Fragen — die Retrieval-Qualität bricht ein.

/06 **Instabile, ratenlimitierte Abhängigkeiten**
Die Pipeline hängt von S3 und LLM-APIs ab. Drosselung und Ausfälle dürfen niemals das Dokument eines Kunden verlieren.

> **Die zentrale Anforderung: Nachvollziehbarkeit.** Eine Compliance-Antwort, die niemand prüfen kann, ist wertlos. Jeder extrahierte Chunk muss sich bis zur exakten Stelle im Originaldokument zurückverfolgen lassen — pixelgenau.

„Bevor eine Plattform *denken* kann,
muss etwas das Dokument *lesen*."

03 ARCHITEKTUR

Ein queue-getriebener Worker mit modularen Pipelines.

Der Worker konsumiert Ingestion-Jobs aus einer Redis-Queue, erkennt und validiert den tatsächlichen MIME-Typ jeder Datei und übergibt sie an die passende formatspezifische Pipeline aus einer Registry. Vier Produktions-Pipelines — PDF, Word, Excel, Bilder — teilen sich ein Ausführungsmodell: extrahieren, verstehen, chunken, einbetten, persistieren. Jedes Zwischenergebnis landet unter einem deterministischen Pfad pro Dokument in S3 — jeder Lauf lässt sich nachträglich prüfen, wiederholen und neu verarbeiten.

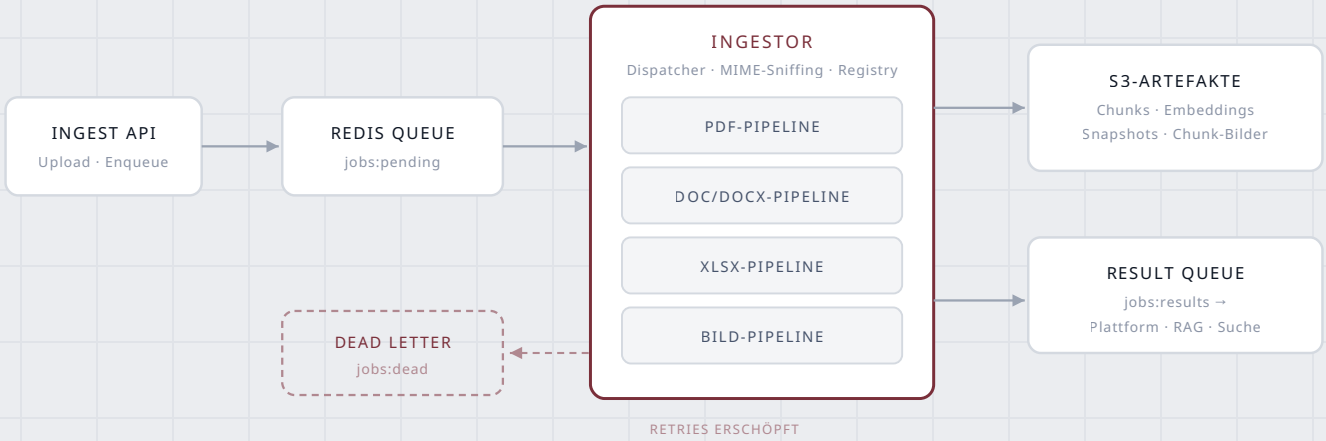

ABB. 01 — INGESTION-TOPOLOGIE. PIPELINES REGISTRIEREN SICH AUF MIME-TYPEN; EIN NEUES FORMAT BERÜHRT KEINEN BESTEHENDEN CODE.

04 EXTRAKTION

Keine Engine liest alles — deshalb arbeiten drei zusammen.

Jede PDF-Seite durchläuft drei komplementäre Extraktoren. **PyMuPDF** liefert native Textblöcke mit exakten Bounding-Boxes und rendert einen Snapshot der Seite. **Tesseract OCR** übernimmt bildlastige oder gescannte Seiten ohne Textebene. **Claude Vision** (über AWS Bedrock) liest die gerenderte Seite wie ein Mensch — erkennt Tabellen, Diagramme, Logos, Stempel und Abbildungen und beschreibt, was sie bedeuten.

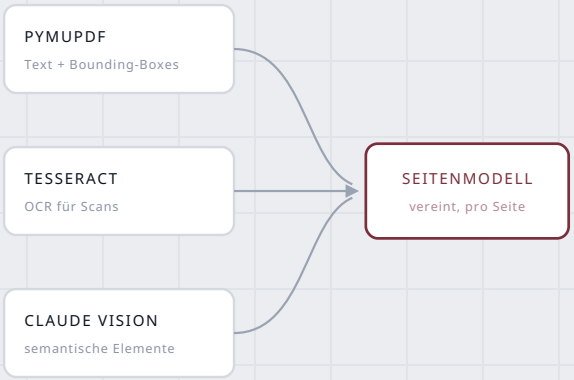

ABB. 02 — DREI ENGINES, EIN VEREINTES SEITENMODELL.

Die drei Ergebnisse werden zu einem gemeinsamen Seitenmodell zusammengeführt: präzise Geometrie vom Parser, aus Pixeln zurückgewonnener Text, semantisches Verständnis vom LLM. Dieselben drei Engines tragen die Bild-Pipeline; Word-Dateien erhalten für alte .doc-Uploads einen Konvertierungsschritt über LibreOffice headless.

Chunks, denen ein Retrieval-System vertraut — und die ein Mensch prüfen kann.

Statt Text an willkürlichen Zeichengrenzen zu zerschneiden, liest ein **LLM-gesteuerter Chunker** die vereinten Seitenmodelle über das gesamte Dokument und gruppiert zusammengehörige Elemente zu semantischen Chunks — flach oder hierarchisch, mit zusammenfassenden Eltern über den Detail-Kindern. Eine Tabelle bleibt eine Tabelle; eine Frage bleibt bei ihrer Antwort.

Entscheidend: Die Bounding-Box jedes Elements bleibt seinem Chunk zugeordnet. Nach dem Chunking führt der Worker die Boxen pro Seite zusammen, schneidet die Region aus dem Seiten-Snapshot aus und legt sie in S3 ab.

ABB. 03 — HIERARCHISCHE CHUNKS; JEDER TRÄGT SEINE QUELLKOORDINATEN.

Jede Antwort der Plattform kann den exakten Ausschnitt des Originaldokuments zeigen, aus dem sie stammt.

Der Chunk-Text wird anschließend mit Amazon Titan Text Embeddings V2 eingebettet — 1024-dimensionale, normalisierte Vektoren, erzeugt in parallelen Batches — und zusammen mit den Chunks an den Vektor-Store der Plattform geliefert.

Tabellen: die Struktur finden, die nur Menschen sehen.

Tabellenkalkulationen brauchten einen eigenen Ansatz. Claude analysiert Sheets in **gleitenden Fenstern von 250 Zeilen mit 50 Zeilen Überlappung**, sodass Tabellen über Fenstergrenzen hinweg vollständig erkannt werden. Das Modell arbeitet multimodal — rohe Zelldaten plus eingebettete Bilder — und liefert typisierte Regionen: Tabellen, Textblöcke, Überschriften, Listen, Illustrationen.

Innerhalb von Tabellen identifiziert es **semantische Gruppen** — die logischen Abschnitte, die ein Mensch an einer fett gesetzten Trennzeile erkennt. Aus einem 70-zeiligen Sicherheitsfragebogen werden Eltern-Chunks wie „Identity & Access Management“, mit jeder Anforderungszeile als abrufbarem Kind-Chunk. Überlappende Erkennungen benachbarter Fenster werden deterministisch zusammengeführt, bevor gechunkt wird.

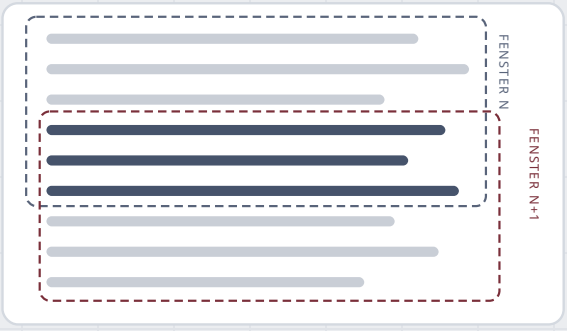

ABB. 04 — GLEITENDE FENSTER ZUR REGIONSERKENNUNG IN GROSSEN SHEETS.

06 PRODUKTIONSHÄRTUNG

Fehler passieren. Dokumente gehen nie verloren.

- **Circuit Breaker** schützen S3 und Bedrock: Fünf Fehlschläge in Folge öffnen den Breaker, Anfragen scheitern sofort; nach einer Abkühlphase (60 s / 120 s) schließen ihn zwei erfolgreiche Probe-Anfragen wieder.
- **Retries mit exponentiellem Backoff und Jitter** — bis zu fünf Versuche pro Job, mit drosselungsbewussten Wartezeiten für LLM-Rate-Limits.
- **Dead-Letter-Queue**: Jobs mit erschöpften Retries landen mit vollem Kontext in `jobs:dead` — niemals stille Fehler.

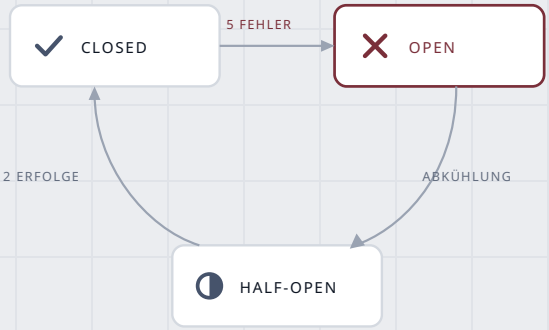

ABB. 05 — CIRCUIT BREAKER PRO ABHÄNGIGKEIT (S3, BEDROCK).

- **Kostendisziplin**: Parallelitäts-Limits auf jedem LLM- und Embedding-Aufruf, lokaler llama.cpp-Provider in der Entwicklung (null Cloud-Kosten), ein Fallback-Modell bei fehlerhafter LLM-Ausgabe, Kosten-Accounting pro Job.
- **Observability**: durchgängig strukturiertes Logging, Sentry-Tracking und -Tracing, Graceful Shutdown für laufende Jobs.

07 ERGEBNIS

Ein Worker, jedes Dokument, jede Antwort überprüfbar.

04	03	0	100%
PRODUKTIONS-PIPELINES — PDF, WORD, EXCEL, BILDER	EXTRAKTIONS-ENGINES PRO SEITE VEREINT	STILLE FEHLER — RETRIES, BREAKER, DEAD-LETTER-QUEUE	DER CHUNKS BIS ZUM QUELLPIXEL NACHVOLLZIEHBAR

Ingestor verarbeitet heute jedes Format, das Kunden hochladen, und liefert der Plattform semantisch gruppierte Chunks samt Embeddings — jeder einzelne bis zur Quelle rückverfolgbar. Neue Formate kommen als eigene Pipeline-Module hinzu, ohne bestehenden Code zu berühren; jeder Schritt bleibt in S3 einsehbar.

STACK — Python 3.11 · Redis · AWS S3 · AWS Bedrock (Claude) · Amazon Titan Embeddings V2 · llama.cpp · PyMuPDF · Tesseract · LibreOffice · python-docx · openpyxl · Pillow · structlog · Sentry · Kubernetes / Tilt

Dokumente, die Ihre Software nicht lesen kann?

Ich konzipiere und baue Document-Intelligence- und LLM-Pipelines — von der Architektur bis zum stabilen Produktionsbetrieb.

PROJEKT STARTEN ↗

hello@bastian-schoentag.de