Retrieval-Augmented Generation für den Mittelstand
KI-Innovationswettbewerb – Generative KI für den Mittelstand

Projekt ID: 01MK250104 **Projektstart:** 01.02.2025 **Laufzeit:** 36 Monate

Ergebnis 2.1: Einbindung von Schnittstellen für die Basisversion

Publikationslevel	Öffentlich
Zieldatum	M9, 31.10.2025
Abschlussdatum	M9, 31.10.2025
Arbeitspaket	AP2 – Schnittstellen
Ergebnis	E2.1
Typ	Report
Status	Final
Version	1.0

Kurzzusammenfassung: Dieses Deliverable beschreibt die Entwicklung der Schnittstellen für die Basisimplementierung.

Gefördert durch:

Bundesministerium für Wirtschaft und Klimaschutz

aufgrund eines Beschlusses des Deutschen Bundestages

Dieses Projekt wurde finanziert vom Bundesministerium für Wirtschaft und Klimaschutz (01MK250104).

History

Version	Datum	Änderung	Author
0.1	14.07.2025	Erster Entwurf	USU
0.2	17.09.2025	Ergänzungen und Implementierungsdetails	IFDT
0.3	23.09.2025	Ergänzungen	USU
0.4	15.10.2025	Überarbeitung nach Review	IFDT
1.0	03.11.2025	Finale Version abgeschlossen und veröffentlicht	IFDT

Zusammenfassung

Dieses Deliverable beschreibt die Entwicklung der Schnittstellen für die Basisimplementierung. Dazu wurden Anforderungen von relevanten Use Cases an diese Schnittstellen analysiert. Es wurde eine Recherche zu industrie-relevanten Schnittstellen und eine generelle Betrachtung von Schnittstellen durchgeführt. Außerdem wurden Festlegungen für die Entwicklung getroffen. Die Architektur der Basisimplementierung wird vorgestellt und es wird aufgezeigt, wie die Umsetzung der Schnittstellen erfolgt ist.

List of Abbreviations

CSV	Comma-separated values
HTML	Hypertext Markup Language
IFDT	Institut für Digitale Technologien
JSON	JavaScript Object Notation
PDF	Portable Document Format
RAG	Retrieval-Augmented Generation
RE	Requirements Engineering
SST	Schnittstellen
SQL	Structured Query Language
UPB	University of Paderborn
URL	Uniform Resource Locator

Table of Contents

1. Einleitung

Um die wissenschaftlichen Ziele des Gesamtvorhabens praktisch anwendbar zu machen, ist die Implementierung von Schnittstellen (SST) zu relevanten Unternehmensdatenquellen entscheidend. Das Teilarbeitspaket AP2 konzentrierte sich daher auf die Schaffung dieser notwendigen technischen Voraussetzungen.

Das Hauptziel von AP2 ist die Implementierung von Schnittstellen zur Indexierung von Textinhalten aus den in deutschen Unternehmen am häufigsten genutzten Datenquellen. Dieser Prozess wurde durch eine anfängliche Erhebung der wichtigsten Datenquellen in deutschen Unternehmen eingeleitet, gefolgt von der tatsächlichen Implementierung der Indexierungsschnittstellen. Dabei hat sich gezeigt, dass das Schnittstellenverständnis für den Projektkontext für eine sinnhafte gemeinsame Arbeit zu schärfen ist. Die Überlegungen dazu sind nachfolgend dokumentiert. Bei der Erschließung der eigentlichen Datenquellen wurde sich in einem ersten Schritt (für die Basisversion der Implementierung) auf die Partner im Konsortium konzentriert.

Mit dem erfolgreichen Abschluss dieses Arbeitspakets existiert die Grundlage für die effiziente Integration unserer Software-Lösung in die Datenlandschaften der Endnutzer in einer ersten Basisversion. Die entwickelten Schnittstellen bzw. Datenanbindungen ermöglichen eine rapide Integration von Retrieval-Augmented Generation (RAG) in Unternehmen und tragen somit direkt zur Erreichung des übergeordneten Projektziels bei, KMUs den Zugang zu korrekten, aktuellen und erklärbaren Antworten aus großen Sprachmodellen zu ermöglichen. Die entwickelten Integrationsansätze sind dabei so ausgelegt, dass sie bei der Weiterentwicklung der Gesamtlösung kontinuierlich um weitere Datenquellen erweitert werden können.

2. Analyse der Anforderungen

In Deliverable E1.1 wurden die Anforderungen an die Basisversion erhoben. Hier werden diese im Hinblick auf Arbeitspaket 2 analysiert und Implikationen abgeleitet.

USABILITY
GUI für das Setup eines Administrators (MUST):
Es wird eine graphische Benutzeroberfläche zur Konfiguration der Learn2RAG Basisversion geschaffen. Die vom Administrator darin vorgenommene Konfiguration wird in einem JSON gespeichert (user_config.json). Die Werte hieraus werden im Backend verwendet. In der Basisversion handelt es sich dabei um einem vom Nutzer gewählten Namen für den Use Case, den Pfaden zu den Systemen, die angebunden werden sollen und der Auswahl des Large Language Modells.

BETRIEB UND IT-SERVICE-MANAGEMENT
Self-Service Konnektor Verwaltung (COULD):
Der Administrator kann bereits in der Basisversion zwischen verschiedenen Konnektoren für unterschiedliche Datenquellen wählen. Diese sind in der Basisversion ein File-System Adapter und ein Webcrawler.

FUNKTIONALE ANFORDERUNGEN
Kontrolle über Quellen (MUST):
In der Basisversion kann der Administrator weitere Quellen zu einer schon existierenden Datenbasis hinzufügen, indem er denselben Namen für den Use Case wählt. In der

Basisversion werden alle einmal hinzugefügte Datenquellen verwendet. Ein Ausschluss von Datenquellen für einzelne Nutzer oder das Entfernen von Daten wird erst zu einem späteren Zeitpunkt ermöglicht.

QUALITÄT, WARTBARKEIT UND ERWEITERBARKEIT

Modularer Aufbau (COULD):
Das Backend der Basisversion ist bereits modular aufgebaut. Als API-Schnittstellen stehen bereits drei Endpunkte für die Suche und die Antwortgenerierung zur Verfügung.

Konfiguration-als-Code (COULD):
Die von der Pipeline optimierten Werte für die Parameter des RAG-Systems werden in einem JSON (opt_config.json) gespeichert.

Um weitere spezifische Anforderung an die Schnittstellen aufzunehmen wurde ein Fragebogen entwickelt. Der Fragbogen ist diesem Deliverable angehängt und wird bei den Workshops genutzt, um weitere Anforderungen der Anwendungspartner und der KMUs aufzunehmen. Dazu erfolgt eine Integration des aus technischer Anforderungssicht entwickelten Fragebogens in speziell E1.2. Der Fragebogen wurde bereits von den projektinternen Use Case Partnern DRK und USU ausgefüllt. Daraus ergeben sich folgende Anforderungen für eine erste Plattformversion:

	DRK Katastrophenfall	DRK Regelbetrieb	USU Release Notes	USU Requirements Engineering
Daten	Öffentlich + intern	Öffentlich	Intern	Intern
Nutzer	Öffentlich + intern	Öffentlich	Intern	Intern
Datenquellen	Web, spezifische Adressen und unspezifische wie Pressemitteilungen	Webseiten von DRK Verbänden, Webdrive, Dateisysteme	Jira + Doku + alte Release Notes	Jira + Doku
Datenformat	Text, PDF, CSV, HTML	Text, PDF, HTML	PDF, CSV	PDF, CSV
Aktualisierungs-intervall	Mehrmals am Tag	Unbekannt	< Täglich	< Täglich
Besonderheiten	Evtl. ortsbezogene Daten	Ortsbezogene Daten		

Implikationen:

- Zugriff auf spezifische URLs (Webcrawler), PDF reader, HTML reader, CSV reader (Interpretation!)
- Berechtigungssystem: autorisierte Nutzer können mehr Datenquellen sehen als die Öffentlichkeit
- Zugriff aufs Web mit unspezifische Webadressen bei Pressemitteilungen (Websearch Tool, RAG und Websuche parallel)
- Ortsbezug in Metadaten speichern oder Chunks mit Ort augmentieren
- Trigger, um neueste Infos aus Datenquellen zu importieren und damit dann zu arbeiten
- Es müssen unstrukturierte und semi-strukturierte Daten verarbeitet werden

Mit Hilfe des Fragebogens (s. Anhang) und im Rahmen der World Cafés im Rahmen der

Workshops soll Rücksprache und Abstimmung mit KMUs und Industriepartnern zu offenen Anforderungen gehalten werden.

3. Recherche zu industrie-relevanten Datenquellen

Im Arbeitspaket wurde eine Recherche zu industrierelevanten Schnittstellen durchgeführt, um die Integration von unternehmensspezifischen RAG-Pipelines zu erleichtern. Da der direkte Zugang zu Unternehmen neben den im Konsortium vertretenen Unternehmen und deren spezifischen Daten erst im weiteren Projektverlauf über die Workshops erfolgen wird, wurde der Ansatz gewählt, zu prüfen, welche Datenimport und -verarbeitungsmöglichkeiten in verschiedenen technischen Lösungen existieren, die bei der Unterstützung und Umsetzung von RAG-Pipelines unterstützen könnten. Der grundlegende Gedanke dabei ist, dass üblicherweise benötigte Schnittstellen sich auch den verschiedenen existierenden Tools wiederfinden werden.

Dazu wurde eine Analyse relevanter, communitybasierter Tools wie LangChain, SpringAI, Apache Spark und Deeplearning4J durchgeführt. Der Fokus lag dabei auf den integrierten **Document Readers** und **Document Transformers**, die für die Indexierung von Textinhalten essenziell sind.

Die Untersuchung hat gezeigt, dass die Verwendung von Open-Source-Frameworks wie LangChain und SpringAI erhebliche Vorteile bietet. Diese Toolsets werden von einer aktiven Entwicklergemeinschaft kontinuierlich weiterentwickelt und bieten eine breite Palette an vorgefertigten Schnittstellen (Document Readers) für diverse Datenformate und -quellen (z.B. PDFs, Word-Dokumente, Webseiten, Datenbanken). Insbesondere sind auch die in der Erhebung bei den beteiligten Partnern benötigten Schnittstellen für den Datenimport vorhanden. Weiterhin werden robuste Document Transformers bereitgestellt, die eine effiziente Vorverarbeitung der extrahierten Daten für die weitere Verarbeitung in einer RAG-Pipeline ermöglichen. Es zeigt sich, dass in diesem Sinne auch eine Transformation von Daten und eine erweiterte Auszeichnung (bspw. mit Metadaten) notwendig ist.

Die Entscheidung, diese communitybasierten Tools und ihre Schnittstellen zu verwenden, ist strategisch sinnvoll. Sie ermöglichen nicht nur eine schnelle und flexible Anbindung an die Datenlandschaft der Unternehmen, sondern stellen auch sicher, dass die technologische Basis des Projekts stets aktuell bleibt und von den neuesten Entwicklungen im Bereich der Künstlichen Intelligenz profitiert. Diese Vorgehensweise minimiert den Entwicklungsaufwand für kundenspezifische Schnittstellen und maximiert die Kompatibilität mit einer Vielzahl von Industriestandards.

Durch die Nutzung der in LangChain, SpringAI und ähnlichen Tools vorhandenen Document Readers und Transformers können wir die Implementierung von Schnittstellen und die Nutzung bzw. Transformation von Daten beschleunigen und gleichzeitig eine hohe Anpassungsfähigkeit an die vielfältigen Datenquellen der Unternehmen gewährleisten. Dies schafft eine solide Grundlage für die effiziente Integration der RAG-Pipelines und trägt maßgeblich zur Erreichung der Projektziele bei.

4. Übersicht zu den SST

Als Ergebnis wurden im Arbeitspaket Schnittstellen im Allgemeinen beschrieben und auf die Passfähigkeit im Projekt untersucht. Im Ergebnis liegt eine Tabelle mit den unterschiedlichen Schnittstellen vor, auf deren Grundlage die Auswahl für die Basisversion getroffen wurde.

4.1 Einordnung

Die Anzahl unterschiedlicher Schnittstellen hängt von der Branche, dem Digitalisierungs- und Vernetzungsgrad ab. Zwar kann davon ausgegangen werden, dass Unternehmen jeder Branche digitale Daten erzeugen, empfangen, verarbeiten und versenden. Für den Datenaustausch zwischen unterschiedlichen Datenverarbeitungssystemen (Software) gibt aber es kein einheitliches Austauschformat. Dabei werden die folgenden beiden Typen unterschieden:

1. Typ A: Gerade in der Industrie haben sich dadurch standardisierte Datenaustauschformate und -protokolle etabliert, die jedoch lediglich auf dedizierte Software-Systeme anwendbar sind. Zumeist sind diese SST lizenzrechtlich geschützt und kostenpflichtig.

2. Typ B: Dem gegenüber entwickelten sich weit verbreitete Standards für den Datenaustausch zwischen Software-Systemen, die von einer Vielzahl von Nutzern eingesetzt werden, oft ohne spezielle Kenntnisse darüber, welche Protokolle im Hintergrund ablaufen. Solche SST stehen im Regelfall unter kostenfreien Lizenzbedingungen.

In Bezug auf die Datenbeschaffung zur Kontextualisierung von Suchanfragen sind Schnittstellen beider Typen (A und B) notwendig. Schnittstellen agieren zwischen Software-Systemen, weshalb neben der Betrachtung industrie-relevanter Schnittstellen auch Software-Systeme einbezogen werden, die aus dem Blickwinkel von KMU unterschiedliche Datenquellen darstellen.

4.2 Abgrenzung zu vorhandenen Schnittstellen

Bei einem Austausch von Signalen sind Sender und Empfänger auf Schnittstellen angewiesen. Dabei transportieren Schnittstellen diese Signale oder formatieren die Signale um, bspw. in das Empfängerprotokoll. Eine Interpretation oder weitergehende Verarbeitung der Signale gehört nicht zu den Aufgaben einer Schnittstelle. Je nachdem in welcher Ebene die Kommunikation zwischen Sender und Empfänger und demnach der Transport der Signale abläuft, können Schnittstellen klassifiziert werden.

a) Schnittstellen für analoge Signale auf rein physikalischer Ebene: Natur
b) Hardware-Schnittstellen: Elektrotechnik
c) Software-Schnittstellen: Datenverarbeitung
d) Netzwerk-Schnittstellen: Netzwerkmanagement
e) Benutzerschnittstellen: Mensch-Computer-Interaktion

Im Projekt Learn2RAG werden Software-Schnittstellen (c) zur Integration von Daten in RAG-Pipelines benötigt. Die Schnittstellen a, b, d, e bleiben für den Projektkontext zumindest für die zunächst betrachtete Basisversion unberücksichtigt. Die Gestaltung der Benutzerinteraktion wird unabhängig von der hier erfolgten Schnittstellen-Gestaltung speziell in AP4 betrachtet.

4.3 Abgrenzung zu strukturierte und unstrukturierte Daten

Digitale Daten sind grundsätzlich Signale, die analog über elektronische oder Lichtwellenleiter übertragen und dann in eine digitale Darstellung transformiert werden. Software-Schnittstellen verarbeiten digitale Daten, die strukturiert oder unstrukturiert sein können. Da eine Software-Schnittstelle jedoch die Daten nicht interpretiert, kann die Schnittstelle nicht zwischen strukturierten und unstrukturierten Daten unterscheiden. Eine Ausnahme bilden Schnittstellen, die zusätzlich als Konverter fungieren. Denn die Struktur der Daten bildet ein Modell für die Daten. Bei strukturierten Daten muss daher von einem Modell ausgegangen werden, das aus den Daten selbst, einem Datenformat und einer Interpretationsvorschrift besteht. Somit kann aus einem solchen Modell auch ein Kontext abgeleitet werden, der mit den

Daten die eigentliche Information darstellt. Im Projekt Learn2RAG werden größere zusammenhängende Daten, ob strukturiert oder unstrukturiert, in Chunks aufgebrochen. Dabei bricht grundsätzlich auch das Modell auf, wobei die ursprüngliche Kontextualisierung im Regelfall verloren geht, sofern eine reine Text-Extraktion erfolgt.
Es gibt jedoch vorgelagerte Prozessschritte, die mithilfe von entsprechenden Tools – beispielsweise LangChain – eine Chunkifizierung ermöglichen, bei der die Kontextualisierung berücksichtigt und zum Teil beibehalten wird.
Für eine RAG-Pipeline soll eine Auswahl von Chunks durch imperative Suchverfahren präqualifiziert zur Verfügung stehen, wodurch dennoch ein anzunehmender Kontext zu einem spezifischen KI-Prompt entsteht.
Ob und wie ein KI-Prompt beantwortet wird, ist die Aufgabe des KI-Modells und auch, ob ein kontextualisierter Chunk mit Kontext und Auszeichnungssprache Berücksichtigung findet.
Eine Unterscheidung von Schnittstellen hinsichtlich des Transports von strukturierten oder unstrukturierten Daten erscheint vor diesem Hintergrund als unerheblich und wird deshalb nicht berücksichtigt.

4.4 Verschlüsselte Datenquellen und verschlüsselte Datenübertragung

Verschlüsselte Datenquellen sind ohne Entschlüsselung weder von Mensch noch von Maschine lesbar. Verschlüsselte Datenquellen werden im Zusammenhang mit Schnittstellen und der Datenübertragung im Projekt Learn2RAG nicht berücksichtigt. Eine verschlüsselte Datenübertragung soll in Learn2RAG-System zurzeit nicht verwendet werden.

Lfd. Nr	Schnittstelle	Einordnung	Relevanz für Learn2RAG
1	SQL-Datenbank (ISO/IEC 9075)	A | c	ja
2	Filesystem (MS Office, LibreOffice, Text, CVS)	A, B | c	ja
3	World Wide Web (HTML)	A, B | c	ja
4	Anwendungsspezifische Schnittstellen (z B. Jira)	A | c	ja
A: industrie-relevant; B: public, social; c: Software-Schnittstelle			

Tabelle 1: Aufstellung der untersuchten Schnittstellen

5. Festlegung der SST für Basisversion

5.1 Entwurf der Software-Architekturen und Programmiersprachen für die SST

Für die Umsetzung der Schnittstellen im Projekt Learn2RAG wurde vom IFDT ein eigenständig nutzbares Softwareartefakt entwickelt und als OpenSource-Lösung auf Github unter https://github.com/Learn2RAG/learn2rag-importer zugänglich gemacht. Die Entwicklung erfolgte mit dem Ziel, die eigenständige Lösung leicht in die weiteren Softwareartefakte des Projektes einzugliedern. Dazu wurde in Verbindung mit den anderen Arbeitspakten des Vorhabens Learn2RAG ein Architekturzielbild entwickelt, welches in Abbildung 1 dargestellt ist.

Wenngleich das Artefakt aus AP2 eigenständig nutzbar ist, ist das Ziel die Integration in das Learn2RAG Toolset bzw. die Pipelinestruktur. Vor diesem Hintergrund wurde auch die Konfiguration und die Nutzung des Outputs über JSON festgelegt.

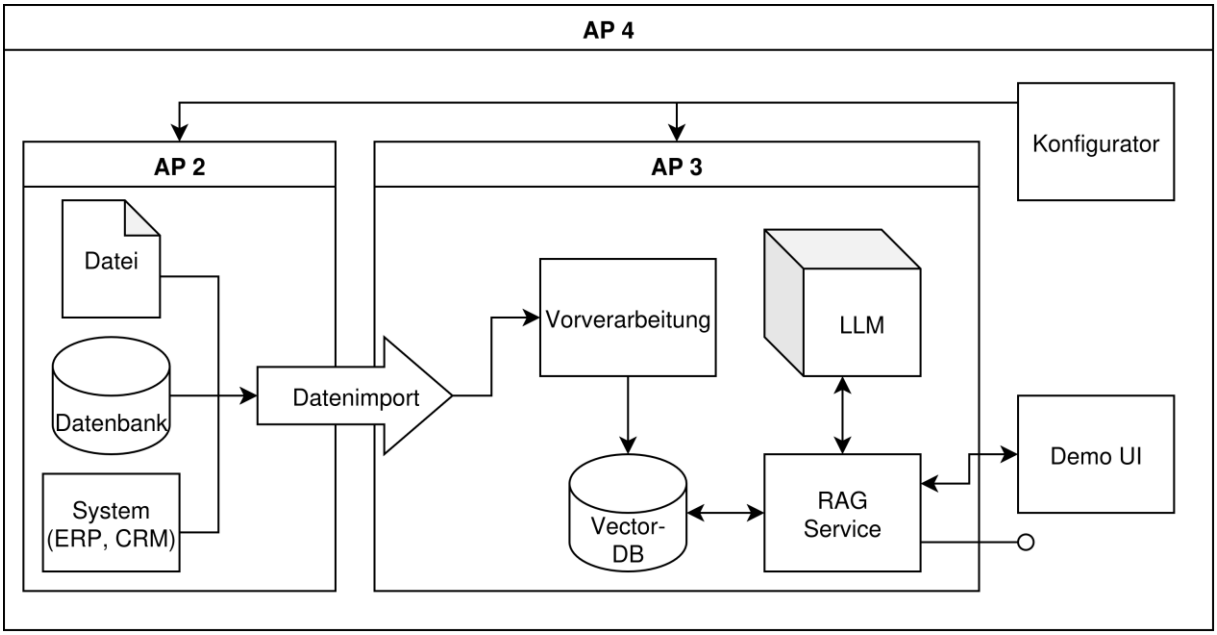

Abbildung 1 - Architekturzielbild und Zuordnung zu den Arbeitspaketen in Learn2RAG

Entwicklungsumgebung

Die Entwicklung erfolgte durch das IFDT mit Python 3.13 in einer modularen Architektur. Als Entwicklungsumgebung wurde Visual Studio Code verwendet, um eine konsistente und effiziente Entwicklung zu gewährleisten.

Programmiersprache

Python wurde als Hauptprogrammiersprache gewählt aufgrund seiner:

- umfangreichen Bibliotheken für Dokumentenverarbeitung;
- nativen Unterstützung für JSON- und YAML-Konfigurationen;
- starken Community im Bereich Natural Language Processing;
- einfachen Integration mit Machine Learning Frameworks;
- gewünschten Konsistenz zwischen den Arbeitspaketen im Projekt Learn2RAG, bei welchen sich insgesamt für Python als Programmiersprache entschieden wurde.

Genutzte Libraries
Die Anwendung basiert auf folgenden zentralen Bibliotheken:

Dokumentenverarbeitung:
- langchain-community: Kernframework für Dokumenten-Loader
- unstructured[*]: Erweiterte Unterstützung für strukturierte Dokumente
- beautifulsoup4: HTML-Parsing und Web-Content-Extraktion

Dateityp-Erkennung:
- python-magic-bin (Windows) / python-magic (Linux/macOS): Automatische Dateityp-Erkennung durch libmagic

Konfiguration und Logging:
- PyYAML: YAML-basierte Konfiguration für Logging
- colorlog: Farbige Konsolen-Ausgaben für bessere Entwickler-Experience

Utility-Bibliotheken:
- hashlib: Content-Hash-Generierung für Dokumenten-Integrität
- requests: HTTP-Client für Web-Content-Abruf
- keyboard: Benutzerinteraktion (ESC-Taste für Prozess-Abbruch)

5.2 Übersicht über Schnittstellen

Die folgenden Schnittstellen wurden für die Basisversion entsprechend der erhobenen und beschriebenen Anforderungen entwickelt:

Schnittelle DirectoryLoader

- **Datenquelle:** Dateisysteme, d.h. (auf einer Festplatte) abgelegte Dateien, die dem RAG-System zugänglich gemacht werden sollen
- **Unterstützte Dateiformate:** .csv, .doc, .docx, .eml, .epub, .html, .json, .md, .odt, .pdf, .ppt, .pptx, .rst, .rtf, .txt, .tsv, .cls, .xlsx, .xml
- **Funktionalität:** Rekursive Verzeichnis-Durchsuchung mit konfigurierbarer Tiefe
- **Datenquelle:** Lokale Dateisysteme

Schnittstelle HTMLLoader

- **Datenquelle:** über das Internet mit einer URL abrufbare Inhalte
- **Unterstützte Formate:** Webseiten (HTML)
- **Funktionalität:** Web-Content-Extraktion mit konfigurierbarer Link-Verfolgung
- **Datenquelle:** URLs und Webseiten
- **Besonderheiten:** Meta-Tag-Extraktion, automatische Link-Erkennung

Transformation- bzw. Ausgabe-Schnittstelle

- **Format:** JSON (Standard ist eine loaded_documents.json- Datei)
- **Struktur:** Einheitliche Metadaten als Text + Content-Struktur für alle Loader
- **Metadaten**: Source-Pfad, Content-Hash, Verarbeitungszeit, Loader-Typ, spezifische Eigenschaften

5.3 Entwicklung der ausgewählten Schnittstellen

Zur Umsetzung wurde folgende modulare Struktur erstellt:

```
learn2rag-importer/
├── config/
│   ├── config.json          # Loader-Konfiguration
│   └── logging.yaml         # Logging-Konfiguration
├── loaders/
│   ├── directory_loader.py  # DirectoryLoader-Implementierung
│   ├── html_loader.py       # HTMLLoader-Implementierung
│   ├── csv_loader.py        # CSVLoader-Implementierung
│   └── process_loaders.py   # Loader-Orchestrierung
├── utils/
│   ├── config_loader.py     # Konfigrations-Management
│   └── logging_setup.py     # Logging-Setup
├── globals.py               # Globale Variablen
└── main.py                  # Haupt-Anwendung
```

Abbildung 2 - Grundstruktur der umgesetzten Anwendung

Umsetzung der modularen Struktur

1. Konfigurationsschicht (config**)**

- JSON-basierte Loader-Konfiguration mit validierter Struktur
- YAML-basiertes Logging mit UTF-8-Unterstützung für Unicode-Zeichen
- zentrale Konstanten für Pfade und Versioning

Für eine Konfiguration in einem Einsatzsetting ist die /config/config.json enstprechend der jeweiligen Anforderungen anzupassen. Im Learn2RAG Projekt soll diese Konfiguration in Verbindung mit den weiteren Arbeitspaketen über den in AP4 entwickelten Konfigurator erfolgen. Dazu ist für jede Datenquelle ein konfigurierender Loader als Eintrag aufzunehmen. Die Grundstruktur der Datei config.json ist:

```
{
    "loaders": [
        {
            "loader_type": "[TYPE_OF_LOADER]",
            [options_for the loader]
        },
        {
            "loader_type": "[TYPE_OF_LOADER]",
            [options_for the loader]
        },
        {
            ...
        }
    ]
}
```

Beispielkonfiguration Schnittstelle DirectoryLoader

```
{
   "loader_type": "DirectoryLoader",
   "path": "C:\\Users\\foo",
   "recursive": "True"
 },
```

wobei „path“ der gewünschte Einstiegspfad und „recursive“ mit „True“ oder „False“ angibt, ob Unterverzeichnisse berücksichtigt werden sollen.

Beispielkonfiguration Schnittstelle HTMLLoader

```
 {
   "loader_type": "HTMLLoader",
   "url": "https://learn2rag.de",
   "depth": 0
```

```
},
```

wobei „url“ die gewünschte Internetseite und „depth“ angibt, bis zu welcher Verfolgungstiefe Links auf der Webseite ebenfalls ausgelesen werden sollen.

2. Loader-Schicht (loaders**)**

Jeder Loader implementiert eine einheitliche Schnittstelle:

```
def load_from_[type](path/url, **options) -> list[Document]:
    # Gemeinsame Metadaten für alle Loader:
    # - content_hash (SHA256)
    # - process_date/time
    # - loader_type
    # - source information
```

Welche durch Loader-spezifische Metadaten ergänzt werden können.

Beispieloutput Schnittstelle DirectoryLoader:

```
{
    "metadata": {
        "source": "C:\\Users\\foo\\Manuscript.docx",
        "content_hash":
“e18e509d138cf86c22df0b0dfafc5ca5b8f1e266f5e3470de68190f3ebe495b0",
        "source_path": "C:\\Users\\foo",
        "file_extension": "docx",
        "process_date": "2025-07-28",
        "process_time": "14:42:02",
        "loader_type": "DirectoryLoader"
    },
    "content": "A Corpus-based Real-time Text Classification and Tagging Approach
for Social Data..."
},
```

Beispieloutput Schnittstelle HTMLLoader:

```
{
        "metadata": {
            "source": "https://learn2rag.de",
            "content_hash":
"ad31e0478b3390eb4425c5b26d41c8677f79e70b6a9c1021256c04b1db091636",
            "process_date": "2025-07-28",
            "process_time": "14:42:02",
            "loader_type": "HTMLLoader",
```

```
            "meta_properties": {
                "description": "Website",
                "og:type": "website",
                "og:locale": "de_DE",
                "og:site_name": "Learn2RAG",
                "og:title": "Learn2RAG",
                "og:url": "https://learn2rag.de//",
                "og:description": "Website",
                "og:image":
"https://learn2rag.de//assets/images/Learn2RAG_Header.png",
                "viewport": "width=device-width, initial-scale=1.0"
            }
        },
        "content": "\n\nWorkshops 2025\n\nIm September und Oktober 2025
organisieren wir Workshops zum Thema RAG. Mehr dazu hier\n\nIn der heutigen
schnelllebigen Geschäftswelt sind Unternehmen und öffentliche Einrichtungen
gefordert, ihre Daten effizient zu nutzen, um w..."
    }
```


3. Utility-Schicht (utils)

- **Config-Loader:** Validierung und Laden der JSON-Konfiguration
- **Logging-Setup:** UTF-8-kompatibles Logging mit Datei- und Konsolen-Ausgabe

4. Orchestrierung (process_loaders.py)

- Zentrale Verarbeitung aller konfigurierten Loader
- Thread-basierte ESC-Taste-Überwachung für Benutzer-Interaktion
- Einheitliche Fehlerbehandlung und Logging

5. Globales State-Management (globals.py)

- Shared State für Prozess-Kontrolle (stop loading)
- Thread-sichere Kommunikation zwischen Modulen

Als besondere weitere Implementierungsfeatures sind zu nennen:

Content-Integrität:

- SHA256-Hash für jeden Dokumenten-Inhalt
- Änderungserkennung durch Hash-Vergleich

Unicode-Unterstützung:

- UTF-8-Encoding für alle Datei-Operationen
- Windows-kompatible Zeichencodierung im Logging

Erweiterbarkeit:

- Plugin-ähnliche Loader-Architektur
- Einfache Integration neuer Datenquellen durch einheitliche Schnittstelle

Fehlerbehandlung:

- Graceful Handling von Encoding-Fehlern
- Detailliertes Logging mit Datei- und Typ-Informationen
- Robuste Verarbeitung bei fehlenden Dependencies (libmagic)

Diese modulare Struktur ermöglicht eine einfache Erweiterung um zusätzliche Datenquellen und Loader-Typen, während die Konsistenz der Ausgabe-Formate gewährleistet bleibt.

6. Anhang

Fragebogen

1. *Beschreiben Sie kurz Ihren Anwendungsfall allgemein:*

2. *Welche Nutzer(gruppen) würden in dem beschriebenen Anwendungsfall mit dem System Informationen abrufen?*

3. *Für den beschriebenen Anwendungsfall, welche Informationen sollten die Nutzenden vom System erhalten können?*

4. *Welche Fragen sollen die Nutzenden dafür dem System stellen können bzw. welchen Antworten würden Sie sich in Ihrem Anwendungsfall wünschen?*

5. *Welche Daten und Information aus Ihrem Unternehmen benötigt das Learn2RAG System aus Ihrer Sicht, um sinnvolle Antworten bieten zu können?*

6. *Wo liegen diese Daten in Ihrem Unternehmen (Mehrfachnennung möglich)?*
   o Dateisystem
   o Auf dem lokalen System
   o Auf einem entfernten System (bspw. Netzlaufwerk)
   o im Web oder Intranet (erreichbar über eine URL)
   o Confluence
   o MediaWiki
   o anderes System: _____
   o im einem Webdrive:
   o Amazon S3
   o Microsoft Azure / OneDrive / Sharepoint / Office 365
   o Dropbox
   o GoogleCloud
   o anderes: _____
   o Github
   o In einer Datenbank:
   o Microsoft SQL
   o PostgreSQL
   o MariaDB / MySQL
   o andere: _____
   o An anderer Stelle: _____
   o unbekannt

7. *Welche der genannten Datenquellen aus (6.) sind für Ihr Unternehmen am*

wichtigsten?

8. *In welcher Form sind diese Daten in den Systemen gespeichert bzw. zugänglich (Mehrfachnennung möglich)?*
   o Text-Dateien oder Abruf als Text
   o PDF-Dateien
   o CSV-Listen
   o als Webseiten (HTML)
   o JSON-formatiert
   o Markdown
   o über SQL-Queries
   o als Microsoft Office-Dateien
   o andere: _____
   o unbekannt

9. *Wie oft ändern sich diese Daten signifikant?*
   o täglich
   o wöchentlich
   o monatlich
   o anderer Zeitraum: _____
   o niemals
   o unbekannt

10. *Haben alle Mitarbeitenden, die im von Ihnen beschriebenen Anwendungsfall Fragen an das Learn2RAG-System stellen würden in gleicher Weise Zugriff auf die verbundenen Daten und Informationen im Unternehmen?*
    o Ja
    o Nein
    o unbekannt

11. *Wenn sie die letzte Frage (Frage 10) mit "Nein" beantwortet haben, bitte spezifizieren Sie näher, welche Berechtigungsstruktur für den Anwendungsfall relevant ist und wie Nutzergruppen sich gegenüber dem System authentifizieren können sollten:*

12. *Welches Betriebssystem(e) verwenden Sie (Bevorzugt das Betriebssystem, auf dem das Learn2RAG System laufen soll)?*

13. *Werden die Nutzer den gleichen PC benutzen, auf dem das Learn2RAG System läuft oder rufen sie das System über das Netzwerk auf?*

14. *Was ist die beste Hardware, die ihr Unternehmen für ein System wie Learn2RAG zur Verfügung stellen könnte?*

15. *Hat diese Hardware die Möglichkeit ein großes Sprachmodell (LLM) aus dem Internet herunterzuladen?*