HTBL Krems

Höhere Lehranstalt für Informationstechnologie
Ausbildungsschwerpunkt: Systemtechnik

DIPLOMARBEIT

Anwendung für eine Firma

Ausgeführt im Schuljahr 2025/26 von:

Jacob Toifl	5AHIT-01
Michael Schaider	5AHIT-02

Betreuer:

Winkler Norbert, MSc
Winkler Norbert, MSc

Krems, am 01.04.2026

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Krems, (Datum)

Verfasser/Verfasserinnen:

Jacob Toifl

Michael Schaider

DIPLOMARBEIT

Bestätigung der Abgabe

Abgabebestätigung

Datum	Name	Unterschrift

Genehmigung der Diplomarbeit

Approbation

Datum	Prüfer*in	Abteilungsleiter*in Direktor*in

DIPLOMARBEIT

Dokumentation

Verfasser*innen
Jacob Toifl, 5AHIT
Michael Schaider, 5AHIT
Abteilung
Informationstechnologie
Ausbildungsschwerpunkt: Systemtechnik
Schuljahr
2025/2026
Thema der Diplomarbeit
Anwendung für eine Firma
Kooperationspartner
MBIT Solutions GmbH

Aufgabenstellung

Realisierung

Ergebnisse

DIPLOMA THESIS

Documentation

Authors

Jacob Toifl, 5AHIT

Michael Schaider, 5AHIT

Department

Information Technology

Specialization: Systems Engineering

Academic year

2025/2026

Thesis Topic

Application for a Company

Cooperation Partner

MBIT Solutions GmbH

Task Description

Implementation

Results

Inhaltsverzeichnis

1. Präambel 8
1.1. Kurzfassung . . . 8
1.2. Abstract . . . 8
1.3. Team . . . 8
1.4. Danksagung . . . 8
1.5. Gendererklärung . . . 9

2. Einleitung 10
2.1. Ausgangslage . . . 10
2.1.1. Systemarchitekturen und Berechtigungssysteme . . . 10
2.1.2. KI-gestützte Klassifizierung und Suchlogiken . . . 10
2.2. Forschungsfrage . . . 10
2.2.1. Dynamische Zugriffskontrolle und Skalierbarkeit von Dokumentensystemen . . . 10
2.2.2. Automatisierte Dokumentenanalyse und Suchoptimierung . . . 10

3. Theoretische Grundlagen 11
3.1. Künstliche Intelligenz zur Dokumentenverarbeitung . . . 11
3.1.1. KI-gestützte Dokumentenklassifikation . . . 11
3.1.2. Architekturen zur Integration externer KI-Dienstleister . . . 13
3.1.3. Merkmalsextraktion und Embedding . . . 13
3.1.4. Modellfamilien zur Dokumenttypenerkennung . . . 17
3.1.5. Datenaufbereitung und Preprocessing für Dokumente . . . 17
3.2. Architekturen für Dokumentensysteme . . . 17
3.2.1. Suche und Filter-Architekturen . . . 17
3.2.2. Rollen- und Berechtigungssysteme . . . 17
3.2.3. Skalierbare plattformunabhängige Systemarchitekturen . . . 18
3.2.4. Sharepoint . . . 20
3.2.5. Microsoft Entra ID . . . 20

4. Dokumentation der Implementierung 21
4.1. Dokumentation - Grundlegend . . . 21
4.1.1. Test Umgebung . . . 21
4.1.2. Technologien . . . 21
4.2. Dokumentation - Funktionen . . . 21
4.2.1. Dokumenten-Upload . . . 21
4.2.2. Dokumenten-Klassifikation . . . 21
4.2.3. Dokumenten-Suche . . . 21
4.2.4. Benutzer- und Rollenverwaltung . . . 21
4.2.5. System-Logging und Monitoring . . . 21
4.2.6. API-Endpunkte . . . 21
4.2.7. Fehlerbehandlung und Ausnahmen . . . 21
4.2.8. Sicherheitsfunktionen . . . 21

5. Beurteilung 22
5.1. Bewertung der Implementierung . . . 22
5.2. Erfüllung der Forschungsfragen . . . 22
5.3. Kritische Reflexion und Grenzen . . . 22

6. Zusammenfassung und Ausblick 23
6.1. Zusammenfassung . . . 23
6.2. Ausblick . . . 23

I. Literaturverzeichnis 24

II. Abbildungsverzeichnis 25

III. Tabellenverzeichnis 26

IV. Quellcodeverzeichnis 27

A. Anhang 28
A.1. Arbeitsteilung . . . 28
A.2. Kapitelverzeichnis . . . 28
A.3. Projekttagebücher . . . 28
A.3.1. Projekttagebuch Max Mustermann . . . 28
A.3.2. Projekttagebuch Mex Musterjuan . . . 28
A.4. Besprechungsprotokolle . . . 29
A.5. Datenträgerbeschreibung . . . 31

1. Präambel

1.1. Kurzfassung

Die vorliegende Diplomarbeit beschäftigt sich mit der Entwicklung und Implementierung des Systems DropIT, einer modernen Lösung zur strukturierten, sicheren und effizienten Verwaltung von Dokumenten innerhalb einer Organisation. Ziel des Projekts ist es, eine nutzerfreundliche Anwendung zu schaffen, welche den Upload, die Klassifizierung, die Suche sowie die Organisation von Dokumenten zentralisiert und vereinfacht. DropIT integriert sich nahtlos in bestehende Microsoft-Dienste wie SharePoint und Entra ID, wodurch sowohl private als auch unternehmensinterne Anwender von einer verbesserten Übersichtlichkeit, Automatisierung und Datensicherheit profitieren.

1.2. Abstract

DropIT is a user-friendly document management system developed to make it easier to store, organize, and find digital files. The system works together with Microsoft services such as SharePoint and Entra ID, allowing secure login and central storage of documents. With AI-supported classification and metadata extraction, DropIT can automatically recognize document types and important information like contract periods or expiration dates. Features such as full-text search, filters, and a built-in reminder system help users manage documents faster and more efficiently. This thesis describes the idea, design, development, and evaluation of the system. The goal of DropIT is to offer a clear, reliable, and scalable solution that improves document handling for both private users and organizations.

1.3. Team

Das Projektteam besteht aus:

- **Jacob Toifl** – Projektleiter
- **Michael Schaider** – Projektmitglied

1.4. Danksagung

Wir möchten uns an dieser Stelle herzlich bei allen Personen bedanken, die uns während der Erstellung dieser Diplomarbeit unterstützt haben. Besonderer Dank gilt unserem Betreuer **Winkler Norbert, MSc**, für seine fachliche Beratung, seine Unterstützung im Entwicklungsprozess und seine wertvollen Rückmeldungen. Ebenso bedanken wir uns bei der Firma **MBIT Solutions GmbH** für die tolle Zusammenarbeit und die Bereitstellung der notwendigen Ressourcen und Infrastruktur, die maßgeblich zum Erfolg dieses Projekts beigetragen haben.

1.5. Gendererklärung

Zur besseren Lesbarkeit der Diplomarbeit wurde ausschließlich die männliche Form verwendet. Da Begriffe wie „Benutzerinnen und Benutzer“ den Text unleserlich machen, wurde es schlicht auf „Benutzer“ gekürzt, dies soll jedoch keine Geschlechterdiskriminierung zum Ausdruck bringen.

2. Einleitung

2.1. Ausgangslage

2.1.1. Systemarchitekturen und Berechtigungssysteme

Das ist die Ausgangslage von Jacob Toifl.

2.1.2. KI-gestützte Klassifizierung und Suchlogiken

Das ist die Ausgangslage von Michael Schaider.

2.2. Forschungsfrage

2.2.1. Dynamische Zugriffskontrolle und Skalierbarkeit von Dokumentensystemen

Das ist die Forschungsfrage von Jacob Toifl.

2.2.2. Automatisierte Dokumentenanalyse und Suchoptimierung

Das ist die Forschungsfrage von Michael Schaider.

3. Theoretische Grundlagen

3.1. Künstliche Intelligenz zur Dokumentenverarbeitung

3.1.1. KI-gestützte Dokumentenklassifikation

3.1.1.1. Definition von Dokumentenklassifikation

Bei der Dokumentenklassifizierung werden Dokumente bestimmten, zuvor definierten Klassen zugeordnet. Das Dokument wird zunächst erfasst, anschließend werden die enthaltenen Informationen ausgelesen und ausgewertet. So lässt sich erkennen, um welche Art von Dokument es sich handelt, wo es abgelegt werden soll, welche Daten daraus übernommen werden müssen und in welchen Workflow es anschließend einfließen kann. [1]

Zum Einsatz kommen dabei unter anderem OCR und KI, die selbst sehr feine Unterschiede zwischen verschiedenen Dokumentarten identifizieren können. Mithilfe von OCR werden Textinhalte aus Bilddateien ausgelesen, automatisch kategorisiert und in eine strukturierte Form gebracht. Dadurch können Dokumente und ihre Inhalte effizient gespeichert, verwaltet, durchsucht und ausgewertet werden. [1]

Die Begriffe Dokumentklassifizierung und Textklassifizierung werden häufig synonym verwendet, weisen jedoch einige Unterschiede auf, wie in Tabelle 3.1 ersichtlich.

Aspekt	**Textklassifizierung**	**Dokumentenklassifizierung**
Geltungsbereich	Analysiert nur Textinhalt.	Analysiert Text sowie Layout- und Bildelemente.
Data Input	Rein textliche Daten (Sätze, Absätze).	Gesamtes Dokument inkl. Bilder und Tabellen.
Anwendungsfälle	Sentiment, Themenzuordnung, Spam-Erkennung.	Rechnungen, Verträge, Formulare.
Techniken	NLP-Methoden.	Kombination aus NLP, Computer Vision und OCR.

Tabelle 3.1.: Textklassifizierung vs. Dokumentenklassifizierung [2]

Im Allgemeinen lässt sich sagen, dass Textklassifizierung eine Teilmenge der Dokumentenklassifizierung ist, die sich ausschließlich auf den Textinhalt konzentriert, während die Dokumentenklassifizierung einen umfassenderen Ansatz verfolgt. [2]

3.1.1.2. Funktion der Dokumentenklassifizierung

Die Dokumentenklassifizierung kann grundsätzlich auf zwei Wegen erfolgen: manuell oder automatisiert. Bei der manuellen Klassifizierung prüft eine Person die Dokumente, identifiziert inhaltliche Zusammenhänge und ordnet sie anschließend den entsprechenden Kategorien zu. Bei der automatischen Dokumentenklassifizierung kommen hingegen Verfahren des maschinellen Lernens bzw. Deep Learnings zum Einsatz. Ziel ist es, Dokumente ohne menschliches Eingreifen systematisch zuzuordnen. Für betriebswirtschaftliche Anwendungen ist es daher wichtig, die unterschiedlichen Dokumentarten sowie die damit verbundenen Geschäftsprozesse zu verstehen. [2]

Strukturierte Dokumente weisen klar definierte, einheitlich formatierte Daten auf (z. B. konsistente Nummerierung, Schriftarten und Layouts). Aufgrund dieser hohen Standardisierung lassen sich Klassifizierungsmodelle für solche Unterlagen vergleichsweise einfach entwickeln und die Ergebnisse sind gut prognostizierbar. [2]

Unstrukturierte Dokumente liegen in einem freien, wenig standardisierten Format vor. Beispiele sind Schreiben, Verträge oder Bestellungen mit variierendem Aufbau und sprachlicher Gestaltung. Durch diese Heterogenität ist die automatisierte Identifikation relevanter Informationen deutlich komplexer, was den Einsatz leistungsfähiger Klassifikationsverfahren erforderlich macht. [2]

3.1.1.3. Funktion der KI-basierten Dokumentenklassifizierung

Die automatisierte Klassifizierung von Dokumenten mit KI erfolgt typischerweise in mehreren aufeinanderfolgenden Schritten:

1. **Datensammlung und Beschriftung**
   Die Basis sind hochwertige, breit gefächerte Datenbestände. Dazu werden Dokumente aus unterschiedlichen Kategorien gesammelt und sauber mit passenden Labels versehen, damit Machine-Learning-Modelle sinnvoll trainiert werden können. [2]
2. **Vorverarbeitung und Feature-Erzeugung**
   Liegt ein Dokument als Scan oder Bild vor, wird der enthaltene Text zunächst per OCR (optische Zeichenerkennung) ausgelesen. Anschließend bereinigen NLP-Verfahren den Text, zerlegen ihn in Tokens und überführen ihn in aussagekräftige Merkmalsrepräsentationen. Parallel dazu wertet Computer Vision das Seitenlayout und visuelle Strukturen aus. [2]
3. **Training des Klassifikationsmodells**
   Überwachte Lernverfahren (etwa Transformer-Modelle oder CNNs) werden mit den gelabelten Beispielen trainiert, um wiederkehrende Muster zu entdecken. Das Modell lernt dabei, die gewonnenen Merkmale den jeweiligen Dokumentkategorien zuzuordnen. [2]

4. **Evaluation und Feintuning**
   Im Anschluss wird das Modell mit bislang unbekannten Testdaten geprüft, um Kennzahlen wie Genauigkeit, Präzision und Recall zu bestimmen. Durch Anpassung von Hyperparametern und ggf. Modellvarianten wird die Performance weiter verbessert. [2]
5. **Produktivbetrieb und laufende Anpassung**
   Nach der Implementierung ordnet das Modell neue Dokumente automatisch in Echtzeit den passenden Klassen zu. Über Nutzerfeedback und zusätzliche Trainingsdaten wird es regelmäßig nachtrainiert und kann seine Treffgenauigkeit im Zeitverlauf kontinuierlich steigern. [2]

3.1.2. Architekturen zur Integration externer KI-Dienstleister

3.1.3. Merkmalsextraktion und Embedding

3.1.3.1. Definition von Merkmalsextraktion

Die Merkmalsextraktion ist ein wichtiger Schritt in der Datenanalyse und im maschinellen Lernen. Dabei werden aus Rohdaten gezielt die Informationen herausgefiltert, die für eine spätere Auswertung oder ein Modell relevant sind. Ziel ist es, aussagekräftige Merkmale hervorzuheben und unwichtige oder störende Anteile zu reduzieren, sodass die Daten kompakter und besser nutzbar werden. [3]

Je nach Datentyp kommen dafür unterschiedliche Methoden zum Einsatz – von einfachen statistischen Verfahren bis hin zu maschinellen Lernverfahren, die Muster automatisch erkennen. Die gewonnenen Merkmale werden häufig in einer strukturierten Form zusammengefasst (z. B. als Merkmalsvektor) und bilden dann die Grundlage für weitere Schritte wie Klassifikation oder Vorhersagen. [3]

Wie in Tabelle 3.2 ersichtlich, werden im Folgenden wichtige Begriffe der Merkmalsextraktion mit kurzer Beschreibung dargestellt.

Begriff	**Beschreibung**
Merkmal	Eine quantitativ oder qualitativ erfassbare Eigenschaft, die ein Datenobjekt beschreibt.
Feature Extraction	Verfahren, bei dem aus Rohdaten gezielt die aussagekräftigen Eigenschaften herausgelöst bzw. abgeleitet werden.
Vektor	Geordnete Sammlung von Merkmalwerten, die ein Objekt in strukturierter Form repräsentiert.
Dimensionalität	Anzahl der enthaltenen Merkmale bzw. Einträge eines Vektors.

Tabelle 3.2.: Wichtige Begriffe der Merkmalsextraktion [3]

3.1.3.2. Funktion der KI-gestützten Merkmalsextraktion

KI-gestützte Merkmalsextraktion ist ein zentraler Bestandteil moderner Datenanalyse. Dabei werden automatisierte Verfahren eingesetzt, um aus großen Datensätzen gezielt relevante Informationen herauszuarbeiten. Durch den Einsatz unterschiedlicher KI-Algorithmen können Muster und Zusammenhänge präzise erkannt und die Daten effizient für weitere Analysen oder Modelle aufbereitet werden. [3]

Die in Tabelle 3.3 aufgeführten KI-Algorithmen sind zentrale Methoden für die Merkmalsextraktion.

Algorithmus	**Beschreibung**
Support Vector Machines (SVM)	Geeignet für Klassifikation und Regression; bestimmt eine optimale Trennlinie bzw. Entscheidungsgrenze zwischen Datenpunkten.
Decision Trees	Erstellen Entscheidungsbäume anhand von Merkmalen und Regeln, um Vorhersagen oder Klassen zu bestimmen.
Random Forest	Kombiniert viele Entscheidungsbäume und erhöht dadurch meist Genauigkeit und Stabilität der Ergebnisse.

Tabelle 3.3.: Relevante Algorithmen zur Merkmalsextraktion [3]

Maschinelles Lernen unterstützt die Merkmalsextraktion, indem Modelle aus Trainingsdaten lernen, welche Merkmale für eine Aufgabe wichtig sind, und weniger relevante Informationen ausblenden. Für Textdaten ist besonders Natural Language Processing (NLP) relevant, da damit Bedeutungen und Zusammenhänge in Sprache erkannt werden können. Wichtige Methoden sind Tokenisierung, Lemmatisierung und Named Entity Recognition (NER). Moderne Modelle wie BERT und GPT-3 nutzen große Datenmengen, um Muster im Kontext zu erfassen und Texte präziser zu analysieren. [3]

Text Mining ist ein Teilbereich der Merkmalsextraktion, der sich auf unstrukturierte Textdaten konzentriert. Ziel ist es, Texte computergestützt so auszuwerten, dass relevante Informationen herausgefiltert und unwichtige Inhalte reduziert werden, um daraus verwertbare Erkenntnisse zu gewinnen. Dabei werden insbesondere Verfahren des maschinellen Lernens und der Natural Language Processing (NLP) eingesetzt, um Muster, Bedeutungen und Zusammenhänge in großen Textmengen zu erkennen. Typische Anwendungsfelder sind unter anderem die Analyse von Kundenkommunikation sowie der Einsatz in Forschungskontexten, z. B. zur automatisierten Durchsicht umfangreicher Dokumente. [3]

3.1.3.3. Definition von Embedding

Ein Embedding ist eine Darstellung von Objekten wie Text, Bildern oder Audio als Zahlenvektoren in einem kontinuierlichen Vektorraum. Dabei sind ähnliche Inhalte im Raum näher beieinander angeordnet, sodass Machine-Learning-Modelle Ähnlichkeiten und Zusammenhänge leichter erkennen können. [4]

Embeddings werden in vielen Anwendungen genutzt, z. B. für Suche, Empfehlungssysteme, Chatbots oder Betrugserkennung. Im Unterschied zu handgebauten Merkmalen entstehen sie meist automatisch durch Lernverfahren (z. B. neuronale Netze), die Muster und Beziehungen direkt aus den Daten ableiten. Dadurch können Modelle nicht nur einzelne Wörter oder Elemente isoliert betrachten, sondern auch deren Kontext und Bedeutung besser erfassen. [4]

3.1.3.4. Prinzip und Funktionsweise von Embeddings

Bei der Embedding-Funktionsweise werden Rohdaten zunächst in ein numerisches Format überführt, weil viele ML-Algorithmen nur mit Zahlen arbeiten (z. B. Text als Bag-of-Words, Bilder als Pixelwerte oder Graphdaten als Matrix). Ein Embedding-Modell erzeugt daraus Vektoren – also Zahlenlisten –, die ein Objekt als Punkt in einem hochdimensionalen Raum repräsentieren. Jede Zahl steht für die Position entlang einer Dimension; je nach Aufgabe können es sehr viele Dimensionen sein. Ähnliche Objekte liegen im Vektorraum näher beieinander, und diese Nähe wird mit Ähnlichkeitsmaßen wie Cosinus-Ähnlichkeit oder euklidischer Distanz bewertet. [4]

In Abbildung 3.1 ist dieses Prinzip schematisch dargestellt: semantisch ähnliche Begriffe liegen im Vektorraum näher beieinander (z. B. Cat, Dog, Wolf), während thematisch andere Begriffe weiter entfernt positioniert sind (z. B. Apple, Banana). Die räumliche Distanz dient damit als Maß für Ähnlichkeit.

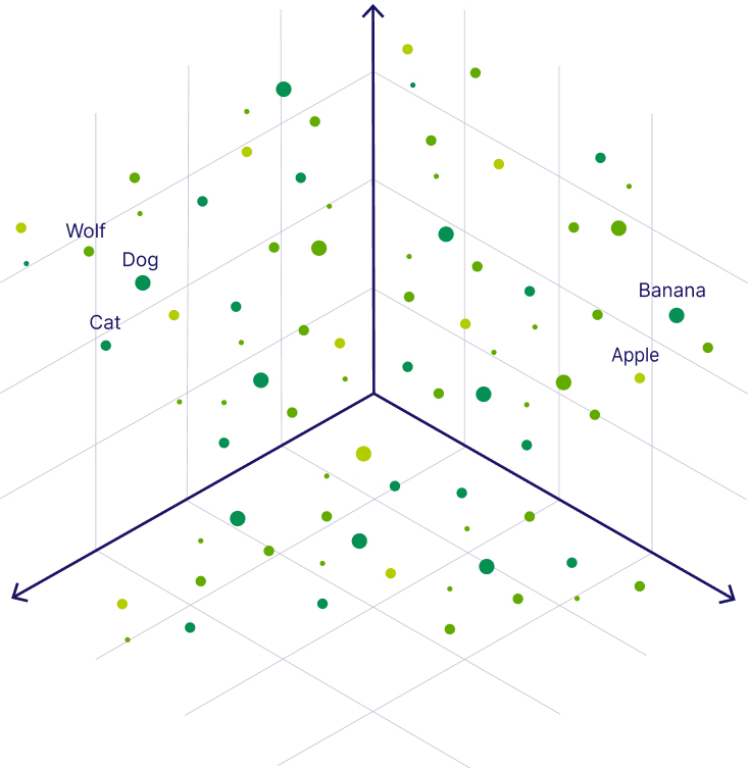

Abbildung 3.1.: Übersicht der Vektordarstellung von Embeddings [5]

Ein anschauliches Beispiel sind Wort-Embeddings: Wörter werden als Vektoren dargestellt, etwa „Papa“ und „Mama“. Auch wenn beide inhaltlich verwandt sind, wäre zu erwarten, dass „Vater“ im Vektorraum noch näher an „Papa“ liegt als „Mama“, weil die Bedeutung stärker übereinstimmt. [4]

Beispielhaft als Vektoren:

- „Papa“ = [0, 1548, 0, 4848, . . . , 1, 864]
- „Mama“ = [0, 8785, 0, 8974, . . . , 2, 794]

In Empfehlungssystemen werden sowohl Nutzer als auch Artikel als Embedding-Vektoren gelernt. Die Grundidee: Ein Nutzer und ein Artikel passen umso besser zusammen, je größer das Punktprodukt ihrer Vektoren ist. Der Empfehlungsscore wird dabei typischerweise so berechnet:

$$score = u * i$$

mit

- Empfehlungsscore $score$,
- Embedding des Nutzers u,
- Embedding des Artikels i,

Im Training werden diese Embeddings anhand historischer Interaktionen (z. B. Klicks, Käufe, Bewertungen) angepasst, sodass hohe Scores mit tatsächlichen Präferenzen möglichst gut übereinstimmen. Danach können Artikel mit den höchsten Scores als Top-N-Empfehlungen ausgegeben werden. [4]

3.1.3.5. Häufige Objekte für Embeddings

Embeddings sind flexible Repräsentationen, die sich auf unterschiedliche Datentypen anwenden lassen. Die häufigsten Objekte, die eingebettet werden können, sind in Tabelle 3.4 dargestellt.

Objekt	**Beschreibung**
Wörter	Dichte Vektorrepräsentationen einzelner Wörter, die Bedeutung und Kontextbeziehungen im Sprachkorpus abbilden.
Text	Vektoren für ganze Texte, die den Gesamtinhalt semantisch zusammenfassen und Vergleiche/Klassifikation erleichtern.
Bilder	Embeddings, die visuelle Merkmale und Bildinhalt als Vektor kodieren, z. B. für Ähnlichkeitssuche oder Objekterkennung.
Audio	Vektorrepräsentationen relevanter Klang- und Sprachmerkmale, nutzbar für Spracherkennung, Klassifikation oder Musikanalyse.
Graphen	Embeddings für Knoten oder ganze Netzwerke, die Struktur und Beziehungen erfassen, z. B. für Link Prediction oder Community Detection.

Tabelle 3.4.: Relevante Objekte für Embeddings [4]

3.1.4. Modellfamilien zur Dokumenttypenerkennung

3.1.5. Datenaufbereitung und Preprocessing für Dokumente

3.2. Architekturen für Dokumentensysteme

3.2.1. Suche und Filter-Architekturen

3.2.2. Rollen- und Berechtigungssysteme

3.2.3. Skalierbare plattformunabhängige Systemarchitekturen

3.2.3.1. Skalierbarkeit

Unter einer skalierbaren Systemarchitektur versteht man ein System, das sich problemlos an steigende Anforderungen anpassen lässt. Fehlende Skalierbarkeit führt dazu, dass Systeme bei wachsender Nutzerzahl oder zunehmendem Datenvolumen spürbare Performanceprobleme entwickeln. Während des Entwicklungsprozess sind unterschiedliche Prinzipien zu beachten:

Merkmal	**Beschreibung**
Modularität	Einzelne Komponenten der Software sollen unabhängig sein und sich flexibel aktualisieren oder erweitern lassen.
Flexibilität	Systeme müssen sich dynamisch an verändernde Anforderungen anpassen können.
Fehlertoleranz	Das System soll auf Fehler reagieren können und automatisch geeignete Maßnahmen ergreifen.

Tabelle 3.5.: Merkmale von Softwaresystemen

Um eine skalierbare Architektur zu gewährleisten, sind Optimierungsfunktionen von Bedeutung, um die Effizienz und Leistung zu optimieren. Wichtige Optimierungsstrategien sind unter anderem:

Technik	**Beschreibung**
Load Balancing	Verteilung von Anfragen auf mehrere Server, um Überlastung zu vermeiden.
Caching	Speicherung häufig abgerufener Daten, um Zugriffszeiten zu verkürzen.
Partitioning	Aufteilung von Daten in kleinere Einheiten, die parallel verarbeitet werden können.
Asynchrone Verarbeitung	Ermöglicht es Systemen, Aufgaben im Hintergrund auszuführen, was die Reaktionszeit verbessert.

Tabelle 3.6.: Techniken zur Leistungsoptimierung

Das Hauptziel einer skalierbaren Architektur besteht darin, auch bei steigender Nutzerzahl und wachsendem Datenvolumen eine hohe Verfügbarkeit sowie eine konsistent hohe Leistung sicherzustellen. Gleichzeitig muss das System so gestaltet sein, dass es leicht wartbar und problemlos erweiterbar bleibt, um zukünftige Anforderungen effizient integrieren zu können. Insgesamt ist eine durchdacht geplante skalierbare Architektur ein entscheidender Faktor für die langfristige Stabilität, Flexibilität und den Erfolg moderner Softwaresysteme. [6]

3.2.3.2. Plattformunabhängigkeit

Ein Computerprogramm benötigt eine Umgebung, in der es gestartet werden kann und während der gesamten Laufzeit stabil funktioniert. Ein Programm gilt als plattformunabhängig oder plattformübergreifend, wenn es auf verschiedenen Computersystemen ausgeführt werden kann, also auf Geräten mit unterschiedlicher Hardware, verschiedenen Prozessoren oder unterschiedlichen Betriebssystemen. Der Grad dieser Unabhängigkeit wird als Portierbarkeit (oder Portabilität) bezeichnet.

Die Portabilität kann z. B. geschätzt werden über

$$P = 1 - \frac{U + A}{E}$$

mit

- Übertragungsaufwand U (insbesondere Neukompilierung),
- Anpassungsaufwand A (Änderung des Quellcodes, z. B. bei Austausch von Betriebsschnittstellen),
- Entwicklungsaufwand E für Neuentwicklung.

Eine Portabilität von $P = 1$ bedeutet vollständige Kompatibilität; das Programm ist also ohne Änderungen auf dem Zielsystem lauffähig, was genau dann gilt, wenn $U = A = 0$.

Eine Quellcode-Portabilität liegt im Regelfall vor, wenn die Gesamt-Portabilität über 90% liegt. Dies entspricht einem Anpassungsaufwand von $A = 0$ und einem Übertragungsaufwand von $U < 0{,}1E$, da bei

$$P = 1 - \frac{U}{E} > 0{,}9$$

der Wert für U kleiner als ein Zehntel von E sein muss. Eine Portabilität nahe 0 entspricht hingegen einer nahezu vollständigen Neuentwicklung des Programms, wobei in diesem Fall $P \approx 0$ und somit $U + A \approx E$ gilt.

Portabilität ist kein Maß für die Lauffähigkeit eines Programms auf der Zielplattform, d. h. selbst eine Portabilität von 99 % bedeutet nicht unbedingt, dass das Programm nutzbar ist, sondern lediglich, dass eine Portierung im Vergleich zu einer Neuentwicklung deutlich weniger Aufwand erfordert. Damit ist nicht nur gemeint, dass ein Programm auf mehreren Plattformen laufen kann, sondern auch, wie viel Aufwand nötig ist, um es dafür anzupassen. Dieser Vorgang wird Portierung oder Migration genannt. [7]

3.2.4. Sharepoint

3.2.5. Microsoft Entra ID

4. Dokumentation der Implementierung

4.1. Dokumentation - Grundlegend

4.1.1. Test Umgebung

4.1.2. Technologien

4.2. Dokumentation - Funktionen

4.2.1. Dokumenten-Upload

4.2.2. Dokumenten-Klassifikation

4.2.3. Dokumenten-Suche

4.2.4. Benutzer- und Rollenverwaltung

4.2.5. System-Logging und Monitoring

4.2.6. API-Endpunkte

4.2.7. Fehlerbehandlung und Ausnahmen

4.2.8. Sicherheitsfunktionen

5. Beurteilung

5.1. Bewertung der Implementierung

5.2. Erfüllung der Forschungsfragen

5.3. Kritische Reflexion und Grenzen

6. Zusammenfassung und Ausblick

6.1. Zusammenfassung

Zusammenfassend war diese Diplomarbeit ein sehr lehrreiches Projekt, bei dem wir viele neue Erfahrungen gemacht haben. ...

6.2. Ausblick

I. Literaturverzeichnis

[1] SER Group: *Document Classification – mit KI zum optimalen Input Management*, Januar 2024. Online in Internet: URL: `https://www.sergroup.com/de/knowledge-center/blog/document-classification-mit-ki.html`.

[2] Shaip: *KI-basierte Dokumentenklassifizierung – Vorteile, Prozesse und Anwendungsfälle*, Juli 2025. Online in Internet: URL: `https://de.shaip.com/blog/ai-based-document-classification/`.

[3] Evoluce GmbH: *Merkmalsextraktion mit KI: Relevantes erkennen, Unwichtiges ausblenden.* Online in Internet: URL: `https://evoluce.de/merkmalsextraktion/` (Zugriff: 03.02.2026). Veröffentlichungsdatum auf der Seite nicht angegeben.

[4] Barnard, Joel: *Was ist Embedding?* Online in Internet: URL: `https://www.ibm.com/de-de/think/topics/embedding` (Zugriff: 03.02.2026).

[5] Dascalescu, Dan and Zain Hasan: *Vector embeddings explained*, January 2023. Online in Internet: URL: `https://weaviate.io/blog/vector-embeddings-explained` (Zugriff: 03.02.2026).

[6] StudySmarter: *Skalierbare Architekturen: Definition, Arten & Beispiele*, 2024. Online im Internet: URL: `https://www.studysmarter.de/schule/informatik/technische-informatik/skalierbare-architekturen/`, abgerufen am 27. Januar 2024.

[7] Wikipedia-Autoren: *Plattformunabhängigkeit — Wikipedia, Die freie Enzyklopädie*, Dezember 2024. Online im Internet: URL: `https://de.wikipedia.org/wiki/Plattformunabh%C3%A4ngigkeit`, abgerufen am 27. Januar 2026.

[8] abc: *DB-Engine Ranking*, März 2016. Online in Internet: URL: `http://db-engines.com/de/ranking`.

[9] Griesch, Leon, Leon Görgen und Kurt Sandkuhl: *Ki-als-service: Vergleich von plattformen zur dokumentenklassifikation.* In: *INFORMATIK 2024*, Seiten 1505–1518. Gesellschaft für Informatik eV, 2024.

II. Abbildungsverzeichnis

3.1. Übersicht der Vektordarstellung von Embeddings [5] 15

III. Tabellenverzeichnis

3.1. Textklassifizierung vs. Dokumentenklassifizierung [2] 11
3.2. Wichtige Begriffe der Merkmalsextraktion [3] 13
3.3. Relevante Algorithmen zur Merkmalsextraktion [3] 14
3.4. Relevante Objekte für Embeddings [4] 17
3.5. Merkmale von Softwaresystemen 18
3.6. Techniken zur Leistungsoptimierung 18

A.1. Kapitelverzeichnis 28
A.2. Arbeitstagebuch Toifl 28
A.3. Arbeitstagebuch Schaider 28

IV. Quellcodeverzeichnis

A. Anhang

A.1. Arbeitsteilung

Kurze Beschreibung, wer was gemacht hat (Überblick).

A.2. Kapitelverzeichnis

Kapitel	Editor
?? ??	Max Mustermann
?? ??	Mex Musterjuan

Tabelle A.1.: Kapitelverzeichnis

A.3. Projekttagebücher

A.3.1. Projekttagebuch Max Mustermann

Tag	Zeit	kumulativ	Fortschritt
Mo 28.11.16	2h	2h	Besprechnung der Programmanforderungen
Di 29.11.16	3h	5h	Datenbankmodell erstellt
Mi 30.11.16	1h	6h	Datenbankmodellüberarbeitet
Do 01.12.16	3h	9h	Pflichtenheft erstellt

Tabelle A.2.: Arbeitstagebuch Toifl

A.3.2. Projekttagebuch Mex Musterjuan

Tag	Zeit	kumulativ	Fortschritt
Mo 28.11.16	2h	2h	Besprechnung der Programmanforderungen

Tabelle A.3.: Arbeitstagebuch Schaider

A.4. Besprechungsprotokolle

. . . Hier können auch pdf Dateien eingebunden werden!

htl bildung mit zukunft	**HTL Krems** **Höhere Lehranstalt für Informationstechnologie** **Ausbildungsschwerpunkt**	**Reife- und Diplomprüfung**

Betreuungsprotokoll zur Diplomarbeit lfd. Nr.:

Themenstellung:
Kandidaten/Kandidatinnen:

Jahrgang:
Betreuer/in:
Ort:
Datum:
Zeit:

Besprechungsinhalt:

Name	Notiz

Aufgaben:

Name	Notiz	zu erledigen bis

A.5. Datenträgerbeschreibung