

# 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.	Spezifische Ausgangslage - Jacob Toifl . . . . .	10
2.1.2.	Spezifische Ausgangslage - Michael Schaider . . . . .	10
2.2.	Forschungsfrage . . . . .	10
2.2.1.	Spezifische Forschungsfrage - Jacob Toifl . . . . .	10
2.2.2.	Spezifische Forschungsfrage - Michael Schaider . . . . .	10
3.	Theoretische Grundlagen	11
3.1.	Künstliche Intelligenz zur Dokumentenverarbeitung . . . . .	11
3.1.1.	KI-gestützte Dokumentenklassifikation . . . . .	11
3.1.2.	Feature-Extraktion und Embeddings . . . . .	13
3.1.3.	Modellfamilien zur Dokumenttypenerkennung . . . . .	13
3.1.4.	Ableitung zur Forschungsfrage von Michael Schaider . . . . .	13
3.2.	Architekturen für Dokumentensysteme . . . . .	13
3.2.1.	Suche und Filter-Architekturen . . . . .	13
3.2.2.	Rollen- und Berechtigungssysteme . . . . .	13
3.2.3.	Skalierbare plattformunabhängige Systemarchitekturen . . . . .	14
3.2.4.	Ableitung zur Forschungsfrage von Jacob Toifl . . . . .	16
4.	Dokumentation der Implementierung	17
4.1.	Dokumentation - Grundlegend . . . . .	17
4.1.1.	Test Umgebung . . . . .	17
4.1.2.	Technologien . . . . .	17
4.2.	Dokumentation - Funktionen . . . . .	17
4.2.1.	Dokumenten-Upload . . . . .	17
4.2.2.	Dokumenten-Klassifikation . . . . .	17
4.2.3.	Dokumenten-Suche . . . . .	17
4.2.4.	Benutzer- und Rollenverwaltung . . . . .	17
4.2.5.	System-Logging und Monitoring . . . . .	17
4.2.6.	API-Endpunkte . . . . .	17
4.2.7.	Fehlerbehandlung und Ausnahmen . . . . .	17
4.2.8.	Sicherheitsfunktionen . . . . .	17
5.	Zusammenfassung und Ausblick	18
5.1.	Zusammenfassung . . . . .	18
5.2.	Ausblick . . . . .	18

---

I.	Literaturverzeichnis	19
II.	Abbildungsverzeichnis	20
III.	Tabellenverzeichnis	21
IV.	Quellcodeverzeichnis	22
A.	Anhang	23
A.1.	Arbeitsteilung . . . . .	23
A.2.	Kapitelverzeichnis . . . . .	23
A.3.	Projekttagebücher . . . . .	23
A.3.1.	Projekttagebuch Max Mustermann . . . . .	23
A.3.2.	Projekttagebuch Mex Musterjuan . . . . .	23
A.4.	Besprechungsprotokolle . . . . .	24
A.5.	Datenträgerbeschreibung . . . . .	26

# 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. Spezifische Ausgangslage - Jacob Toifl

Das ist die Ausgangslage von Jacob Toifl.

#### 2.1.2. Spezifische Ausgangslage - Michael Schaider

Das ist die Ausgangslage von Michael Schaider.

### 2.2. Forschungsfrage

#### 2.2.1. Spezifische Forschungsfrage - Jacob Toifl

Das ist die Forschungsfrage von Jacob Toifl.

#### 2.2.2. Spezifische Forschungsfrage - Michael Schaider

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.

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.

Die Begriffe Dokumentklassifizierung und Textklassifizierung werden häufig synonym verwendet, weisen jedoch einige Unterschiede auf:

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

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.

### 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.

#### **Strukturierte Dokumente**

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.

#### **Unstrukturierte Dokumente**

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.

### 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. 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.

#### **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.

**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.

**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.

**3.1.2. Feature-Extraktion und Embeddings****3.1.3. Modelfamilien zur Dokumenttypenerkennung****3.1.4. Ableitung zur Forschungsfrage von Michael Schaider****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 3.2.: 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 3.3.: 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.

### 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.

### 3.2.4. Ableitung zur Forschungsfrage von Jacob Toifl

## 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. Zusammenfassung und Ausblick

### 5.1. Zusammenfassung

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

### 5.2. Ausblick

## I. Literaturverzeichnis

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

## II. Abbildungsverzeichnis

### III. Tabellenverzeichnis

3.1. Textklassifizierung vs. Dokumentenklassifizierung . . . . .	11
3.2. Merkmale von Softwaresystemen . . . . .	14
3.3. Techniken zur Leistungsoptimierung . . . . .	14
A.1. Kapitelverzeichnis . . . . .	23
A.2. Arbeitstagebuch Toifl . . . . .	23
A.3. Arbeitstagebuch Schaider . . . . .	23

## 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	Besprechung der Programmanforderungen
Di 29.11.16	3h	5h	Datenbankmodell erstellt
Mi 30.11.16	1h	6h	Datenbankmodell überarbeitet
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	Besprechung der Programmanforderungen

Tabelle A.3.: Arbeitstagebuch Schaider

## A.4. Besprechungsprotokolle

... Hier können auch pdf Dateien eingebunden werden!

**Betreuungsprotokoll zur Diplomarbeit****Ifd. Nr.:**

Themenstellung:

Kandidaten/Kandidatinnen:

Jahrgang:

Betreuer/in:

Ort:

Datum:

Zeit:

Besprechungsinhalt:

Name	Notiz

Aufgaben:

Name	Notiz	zu erledigen bis

## A.5. Datenträgerbeschreibung