

Studienarbeit zum Thema KI gestützte Archivierung indexierter PDFs

Studienarbeit

im Studiengang

Softwaretechnik und Medieninformatik

vorgelegt von

Baran Ibrahim Cal und Nicolai Glock

am 24.06.2025

an der Hochschule Esslingen

Erstprüfer/in: Prof. Dr. Dirk Hesse

Zweitprüfer/in: Prof. Dr. Kai Warendorf

Kurzfassung

Gegenstand der hier vorgestellten Arbeit ist die Entwicklung eines Systems zur KI-gestützten Archivierung von indexierten PDF-Dokumenten. Ziel des Projekts ist es, durch den Einsatz von Künstlicher Intelligenz eine weitgehend automatisierte Analyse, Klassifikation und Verschlagwortung von PDF-Dateien zu ermöglichen. Das System soll in der Lage sein, relevante Informationen aus den Dokumenten zu extrahieren und diese übersichtlich darzustellen und für eine Suchfunktion bereitzustellen.

Das Projekt wird im Rahmen des Studiums an der Hochschule Esslingen von den Studierenden Nicolai Glock und Baran Ibrahim Cal durchgeführt. Durch die Umsetzung werden die Kenntnisse in verschiedenen Technologien vertieft, insbesondere in der praktischen Anwendung von KI-Modellen zur Analyse von PDF-Dokumenten. Das entwickelte System reduziert den manuellen Aufwand bei der Archivierung und verbessert die Suche von Inhalten.

Inhaltsverzeichnis

| | |
|--|-----------|
| Kurzfassung..... | 2 |
| Inhaltsverzeichnis | 3 |
| Abbildungsverzeichnis..... | 4 |
| 1 Überblick | 5 |
| 2 Stand der Technik | 6 |
| 3 Installation..... | 8 |
| 4 Architektur..... | 9 |
| 4.1 Design Pattern | 9 |
| 4.2 Datenmodell | 10 |
| 4.2.1 Logisches Datenmodell | 10 |
| 4.2.2 Physisches Datenmodell | 11 |
| 5 Umsetzung | 12 |
| 5.1 Technologien und verwendete Packages | 12 |
| 5.2 Entwicklungsumgebung | 13 |
| 5.3 Umsetzung verschiedener Varianten | 14 |
| 5.3.1 Klassische Pipeline mit Tokenizer | 14 |
| 5.3.2 Einsatz von Gemini | 14 |
| 5.3.3 Begründung der finalen Auswahl | 14 |
| 5.4 Backend | 16 |
| 5.4.1 API..... | 16 |
| 5.4.2 Logik..... | 21 |
| 5.5 Frontend..... | 21 |
| 5.5.1 Allgemein | 21 |
| 5.5.2 Aufbau | 21 |
| 5.5.3 User Interface | 22 |
| 6 Herausforderungen | 28 |
| 7 Zusammenfassung und Ausblick | 29 |
| Literaturverzeichnis | 30 |
| Ehrenwörtliche Erklärung..... | 32 |

Abbildungsverzeichnis

| | |
|--------------------|----|
| Abbildung 1 | 9 |
| Abbildung 2 | 10 |
| Abbildung 3 | 11 |
| Abbildung 4 | 16 |
| Abbildung 5 | 16 |
| Abbildung 6 | 17 |
| Abbildung 7 | 17 |
| Abbildung 8 | 18 |
| Abbildung 9 | 18 |
| Abbildung 10 | 18 |
| Abbildung 11 | 19 |
| Abbildung 12 | 20 |
| Abbildung 13 | 20 |
| Abbildung 14 | 22 |
| Abbildung 15 | 22 |
| Abbildung 16 | 23 |
| Abbildung 17 | 23 |
| Abbildung 18 | 24 |
| Abbildung 19 | 24 |
| Abbildung 20 | 25 |
| Abbildung 21 | 25 |
| Abbildung 22 | 26 |
| Abbildung 23 | 26 |
| Abbildung 24 | 27 |

1 Überblick

Die fortschreitende Digitalisierung führt in vielen Bereichen zu einem massiven Anstieg von Daten, wobei Dokumente im PDF-Format einen wesentlichen Bestandteil darstellen. Die manuelle Archivierung und Verwaltung dieser digitalen Inhalte stellen oft eine erhebliche Herausforderung dar. Dieser Prozess ist nicht nur zeit- und ressourcenintensiv, sondern kann auch ein hohes Fehlerpotenzial bürden, was die effiziente Auffindbarkeit relevanter Informationen sehr erschwert. Besonders bei großen und stetig wachsenden Daten stoßen herkömmliche Methoden schnell an ihre Grenzen.

Deshalb befasst sich das vorliegende Studienprojekt mit der Entwicklung eines Systems, das auf Künstlicher Intelligenz (KI) basiert. Das primäre Ziel ist es, den Aufwand bei der Archivierung und insbesondere bei der Suche nach nützlichen Informationen zu reduzieren. Hierbei liegt der Fokus auf der automatisierten Analyse, Klassifikation und Verschlagwortung von PDF-Dokumenten, um relevante Inhalte schnell und genau zu extrahieren. Dazu wird dieser Inhalt aufbereitet und für eine Suchfunktion bereitgestellt. Dadurch wird die Effizienz bei der Dokumentenverwaltung gesteigert und die Informationsverarbeitung langfristig zu verbessern.

2 Stand der Technik

Die fortschreitende Digitalisierung führt in vielen Organisationen zu einer zunehmenden Datenflut, insbesondere in Form von Dokumenten wie PDF-Dateien (Datagrid, 2025; Alkhamis, 2023). Um diese großen Mengen effizient zu verwalten und zu archivieren, hat der Einsatz von Künstlicher Intelligenz (KI) in den letzten Jahren erheblich an Bedeutung gewonnen (Wolfenstein, 2025; Schneider, 2025; Datagrid, 2025). KI ermöglicht es, die digitale Archivierung von Dokumenten zu beschleunigen und zu standardisieren (Schneider, 2025).

Ein grundlegender Bestandteil im Umgang mit digitalen Dokumenten ist die Optische Zeichenerkennung (OCR). Diese Technologie ist unerlässlich, um gescannte Seiten in bearbeitbaren und durchsuchbaren Text umzuwandeln (Datagrid, 2025). Die anschließende Konvertierung in archivierungsfähige Formate wie PDF/A trägt zur Langzeitlesbarkeit der Informationen bei (Schneider, 2025).

Der aktuelle Stand der Technik in der Dokumentenverarbeitung geht über die reine Texterkennung hinaus und endet im Konzept des Intelligent Document Processing (IDP). IDP kombiniert OCR mit Maschinellern Lernen (ML), um den Kontext von Dokumenten zu erfassen, Fehler zu reduzieren und verschiedene Dokumentenformate zu verarbeiten (Datagrid, 2025). Die Rolle der KI im Dokumentenmanagement umfasst dabei mehrere Schlüsselbereiche:

- **Informationsextraktion (IE):** Die Informationsextraktion ist der Prozess, in der relevante strukturierte Daten aus nicht sehr strukturierten oder unstrukturierten Textquellen, einschließlich PDF-Dokumenten, gewonnen werden (Alkhamis, 2023; Haider, 2025). Hierbei werden Technologien der Natürlichen Sprachverarbeitung (NLP) eingesetzt, die es Maschinen ermöglichen, menschliche Sprache zu analysieren und zu verstehen (Haider, 2025). Standard-IE-Methoden umfassen unter anderem die Identifikation benannter Entitäten (Named Entity Recognition – NER), um spezifische Informationen wie Kontaktdaten schnell und automatisch zu finden (Alkhamis, 2023). Moderne, KI-gestützte Extraktionstools, wie ClickUp lesen, interpretieren und organisieren Inhalte auf eine Weise, die sie direkt nutzbar macht, (Behal, 2025).
- **Dokumentenklassifikation und -kategorisierung:** KI-Modelle ermöglichen die automatisierte Kategorisierung von PDF-Dokumenten in vordefinierte und selbst trainierte Kategorien (PLANET AI GmbH, 2024). Diese Technologie nutzt KI und ML, um Inhalte und Layout der PDF zu analysieren und zu verstehen, wofür es in dem Dokument geht, (PLANET AI GmbH, 2024). Die KI-basierte Klassifikation beruht auf maschinellern Lernen, das Muster in großen Datensätzen

zen erkennt und seine Genauigkeit durch kontinuierliches Lernen stetig verbessert (PLANET AI GmbH, 2024). Dies bietet erhebliche Vorteile, darunter Zeitersparnis durch die Reduzierung manueller Klassifikationsprozesse, erhöhte Konsistenz durch die Vermeidung menschlicher Fehler und eine verbesserte Zugänglichkeit durch eine gute Organisation von Dokumenten (PLANET AI GmbH, 2024).

- **Automatische Verschlagwortung und Indexierung:** Basieren auf der automatischen Erkennung von Angaben können Dokumente automatisch benannt und zugeordnet werden, wobei wichtige Inhalte durch Verschlagwortung gesichert werden, (Schneider, 2025; PLANET AI GmbH, 2024). Die Automatisierung der PDF-Indexierung reduziert repetitive Aufgaben und ermöglicht es sich auf andere Tätigkeiten zu konzentrieren (Datagrid, 2025). Eine standardisierte Kategorisierung reduziert zudem das Risiko, dass Daten falsch abgelegt oder übersehen werden (Datagrid, 2025).

Am Markt sind zahlreiche KI-gestützte PDF-Reader, Analysetools und auch Dokumentenmanagementsysteme vorhanden, die diese und ähnliche Funktionen haben. Beispielsweise sind das 3 solcher Tools: Adobe Acrobat KI-Assistent, UPDF, ChatPDF (Wolfenstein, 2025). Manche solcher Tools bieten das Zusammenfassen, Übersetzen bis hin zur Datenextraktion an.

3 Installation

Über das GitHub-Repository <https://github.com/nicolai99/KIdoc.git> können die Quelldatenbezogen werden.

Für eine lokale Umgebung wird lediglich Docker benötigt. Ist das gegeben, kann der README gefolgt werden. Nach der Initialisierung stehen alle notwendigen Ressourcen bereit und sind bereits mit Standarddaten ausgestattet.

4 Architektur

4.1 Design Pattern

Abbildung 1 zeigt das Design Pattern, welches aus den nachfolgenden Komponenten besteht:

- Client
 - Der Client stellt die Anfragen an den Controller. Er visualisiert die Antworten des Controllers
- Controller:
 - Der Controller verwaltet die Webanfragen und gibt sie an den Service weiter. Er stellt dafür Routen bereit die je nach Operation lesen oder schreiben. Außerdem werden entsprechend der verwendeten Services HTTP Status Codes an den Client zurückgegeben.
- Service
 - Der Service steht zwischen Controller und der Django Query. Er enthält die Logik und sorgt für den notwendigen Programmfluss. Bei Fehlern wirft dieser, entsprechende Ausnahmen, welche dann über einen Globalen Handler vom Controller aufgefangen und an den Client zurückgegeben werden.
- Django Query
 - Das Django Query ist eine Art Abstrahierung von SQL-Abfragen und Teil des ORMs (Object Relational Mapping) von Django. Dadurch lassen sich komplexe Abfragen in teilweise vorgefertigten Methoden sicher durchführen.
- Datenbank
 - Die Datenbank wird als Persistenz zum Verwalten der Daten verwendet.

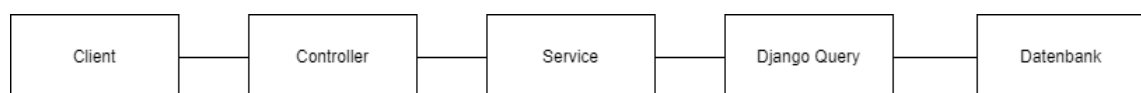


Abbildung 1

4.2 Datenmodell

4.2.1 Logisches Datenmodell

Abbildung 2 zeigt das logische Datenmodell. Das Archiv vereint mit den Attributen gemeinsame Dokumententypen. Jedes Attribut steht für einen wichtigen Parameter, der beispielsweise im Nachgang für die Suche nach dem Dokument von Bedeutung ist. Jedem Attribut wird ein Typ zugewiesen, der einem Datentyp gleicht (date, text, number). Ein Dokument wird für die Archivierung entsprechend den zugehörigen Attributen des Archivs mit den Werten versehen.

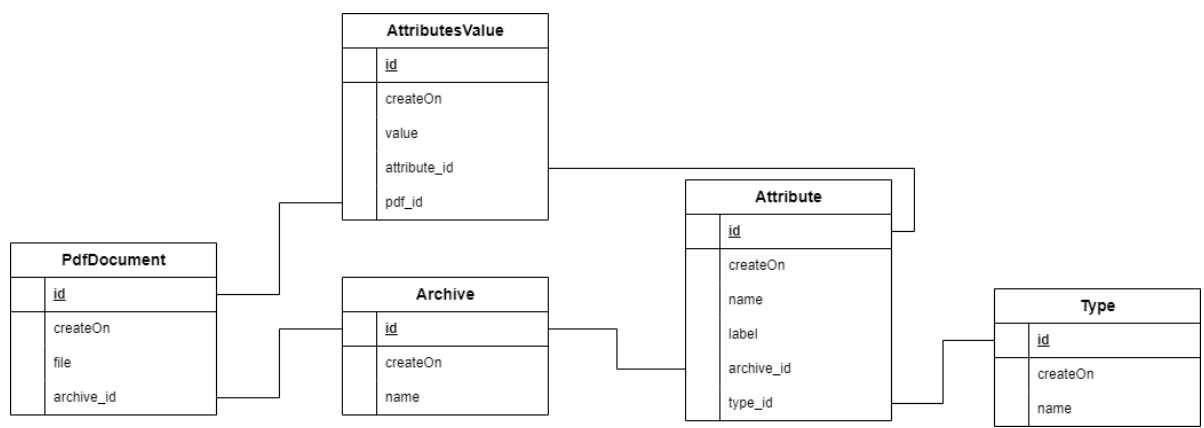


Abbildung 2

4.2.2 Physisches Datenmodell

Abbildung 3 zeigt das physische Datenmodell. Dort liegen die festgelegten Datentypen und die verwendeten Schlüssel. Die Multiplizitäten zeigen die Beziehungsverhältnisse auf. Unter PdfDocument gibt es ein Feld „file“, dieses ist mit Absicht kein Binary Feld, da Binary Felder, nur verwendet werden sollten, wenn diese auch tatsächlich verarbeitet werden müssen, (Medium Harshith Gowda, 2025). In diesem Fall wird lediglich der Pfad zum Dateiverzeichnis gespeichert. Die Datei selbst wird dann vom Betriebssystem verwaltet. Das Feld text in AttributesValue ist deshalb als Textfeld gewählt worden, da es ohne Feldbegrenzung auskommt.

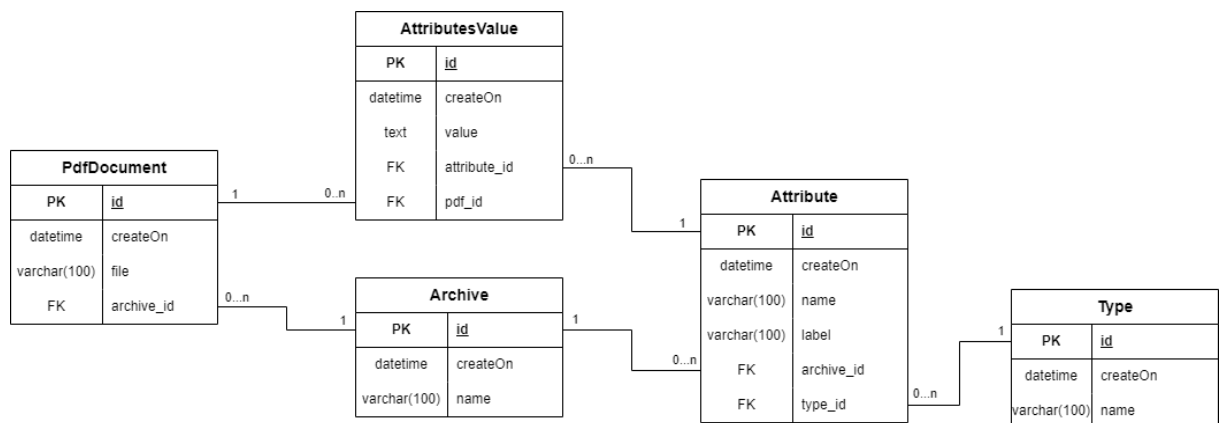


Abbildung 3

5 Umsetzung

5.1 Technologien und verwendete Packages

Für die Entwicklung des Systems zu KI-gestützten Archivierung indexierter PDFs wurden verschiedenste Technologien verwendet, die jeweils ihre Aufgaben innerhalb des Projekts erfüllen.

- **Django:** Django ist ein kostenloses und quelloffenes Web-Framework, das die Entwicklung von Webanwendungen mit Python beschleunigt. Es unterstützt eine schnelle Entwicklung und ein klares, pragmatisches Design. Django hilft schnell funktionsreiche, sichere und skalierbare Webanwendungen zu erstellen und bietet eine hohe Effizienz an (IBM).
- **Django Ninja:** Django Ninja ist ein Webframework zum Erstellen von APIs mit und für Django. Es ist einfach zu bedienen und gestaltet sich intuitiv. Es bietet eine schnelle Ausführung und die Verwendung von Typ-Hinweisen und automatischer Dokumentation ermöglicht es sich auf die Entwicklung der Anwendung zu konzentrieren. Das Framework basiert auf Open API und JSON-Schema und hat eine gute Integration mit Django core als auch ORM (Django Ninja).
- **Vue:** Vue oder auch VueJs ist ein JavaScript Framework zum Erstellen von Benutzeroberflächen. Es baut auf HTML, CSS und JavaScript auf aber bietet ein komponentenbasiertes Programmiermodell an, dass die Entwicklung effizienter gestaltet. Dabei ist Vue flexibel einsetzbar und kann für verschiedene Anwendungsfälle wie von der Erstellung und Verbesserung statischer Seiten bis hin zu komplexen Single-Page Applikationen eingesetzt werden (Vue.js).
- **TypeScript:** TypeScript ist ein Superset (eine erweiterte Version) von JavaScript. Sie fügt statische Typisierung und weitere Funktionen hinzu, die die Entwicklung von Anwendungen vereinfachen und vor allem die Fehleranfälligkeit reduziert. Damit ist es robuster, besser lesbar und leichter zu warten (Kinsta Inc., 2023).
- **PrimeVue:** PrimeVue ist eine UI-Komponentenbibliothek für VueJs. Sie bietet eine breite Palette von vorgefertigten UI-Komponenten, die die Entwicklung um einiges vereinfachen, da vieles anpassbar und wiederverwendbar ist. Es unterstützt verschiedene Theming-Optionen um auch mit CSS-Frameworks wie Tailwind zu funktionieren (PRIMEVUE).
- **Pinia Storemanagement:** Pinia ist die offizielle State-Management-Bibliothek für VueJs, die es ermöglicht Daten und Logik über verschiedene Komponenten und Seiten hinweg, zu teilen (Pinia).

- **PostgreSQL:** PostgreSQL ist ein open-source und objektrelationales Datenbanksystem, das auf der SQL-Sprache basiert und für seine Zuverlässigkeit, Datenintegrität und hohen Funktionsumfang bekannt ist. PostgreSQL kann relationale als auch nicht-relationale Daten verarbeiten und für viele Anwendungen benutzt werden. Von simplen und kleinen Webanwendungen bis hin zu großen und komplexen Unternehmensdatenbanken (POSTGRESQL).
- **Tailwind CSS:** Tailwind ist ein Utility-First-CSS-Framework, das eine alternative Herangehensweise für das Styling von Webanwendungen bietet. Es liefert eine Vielzahl von CSS-Klassen, mit denen man sehr schnell und einfach Benutzeroberflächen gestalten kann. Diese Klassen werden direkt mit HTML kombiniert (HubSpot Inc., 2022).
- **Gemini API:** Die Gemini API bietet Entwicklern Zugang zu Googles KI-Modellen darunter Gemini 2.5 Flash. Diese Modelle können verschiedene Eingabeformate verarbeiten wie Text, Bilder, Dokumente oder auch Audio und daraus Zusammenfassungen, Code, Beschreibungen oder strukturierte Daten generieren (Seng, 2025).

5.2 Entwicklungsumgebung

Eine Effiziente Entwicklungsumgebung ist entscheidend für die Produktivität als auch die Zusammenarbeit im Projekt. Sie gewährleistet, dass alle mit denselben Konfigurationen arbeiten und die Entwicklung ohne Probleme verläuft.

- **Docker:** Docker ist eine open-source Plattform, die die Entwicklung, Bereitstellung und Verwaltung von Software mithilfe von Containern vereinfacht. Container sind standardisierte, ausführbare Einheiten, die eine Anwendung und alle notwendigen Komponenten, wie OS-Bibliotheken und Abhängigkeiten bündeln, sodass sie in jeder Umgebung konsistent ausgeführt werden können (Susnjara, et al., 2024).
- **PyCharm:** PyCharm ist eine weit verbreitete Integrierte Entwicklungsumgebung (IDE) von JetBrains, die auf die Python-Programmierung zugeschnitten ist. Sie ist kompatibel mit allen gängigen Betriebssystemen und bietet umfangreiche Tools für eine effiziente Entwicklung. Zudem unterstützt PyCharm Webtechnologien wie HTML, CSS und JavaScript und auch Frameworks wie Django. Es wird auch für Data Science und auch Machine Learning Projekte verwendet (DataScientest, 2023).

5.3 Umsetzung verschiedener Varianten

Im Rahmen der Systementwicklung wurden zwei verschiedene Umsetzungsansätze zur automatisierten Analyse und Abfrage von PDF-Dokumenten konzipiert und getestet. Ziel war es, ein Verfahren zu finden, das sowohl in Bezug auf Geschwindigkeit als auch auf Präzision eine effektive Nutzung im späteren Betrieb ermöglicht. Die beiden Varianten unterschieden sich grundlegend in ihrer technischen Umsetzung: eine klassische Pipeline mit Tokenizer und Question-Answering-Modul, sowie ein moderner Ansatz unter Verwendung des Large Language Models *Gemini* von Google.

5.3.1 Klassische Pipeline mit Tokenizer

Bei der ersten Variante wurde zunächst das eingereichte Dokument durch einen sogenannten One-Shot-Classifer geleitet, der eine Grobklassifikation vornahm. Anschließend erfolgte die Zerlegung des Textinhalts mittels eines Tokenizers, um die Informationen in verarbeitbare Einheiten zu überführen. Auf dieser Grundlage wurde ein Question-Answering-Modul eingesetzt, das gezielte Anfragen an das Dokument stellen konnte. Dieses Vorgehen ermöglichte prinzipiell eine flexible Verarbeitung und war vollständig lokal realisierbar. Allerdings zeigte sich im Testbetrieb ein erheblicher Nachteil: Die Verarbeitung eines einzelnen Dokuments dauerte in der Regel zwischen 40 und 50 Sekunden, was insbesondere bei einer größeren Anzahl von Dateien zu einer deutlichen Verzögerung führte. Die Ursache hierfür lag unter anderem im Fehlen einer dedizierten GPU, wodurch die Ausführung ressourcenintensiver Modelle erheblich verlangsamt wurde.

5.3.2 Einsatz von Gemini

Die zweite getestete Variante basierte auf dem KI-Modell Gemini, einem multimodalen Large Language Model, das über eine API angesprochen wurde. Der Prozess gestaltete sich hier deutlich einfacher: Nach dem Hochladen eines Dokuments konnte direkt über gezielte Prompts mit dem Modell interagiert werden, um relevante Inhalte zu extrahieren oder Zusammenfassungen und Klassifikationen zu erhalten. Diese Lösung überzeugte insbesondere durch ihre hohe Geschwindigkeit – eine typische Anfrage wurde in lediglich 4 bis 5 Sekunden verarbeitet. Darüber hinaus lieferte Gemini bei der Textsuche und semantischen Analyse deutlich präzisere und kontextuell stimmigere Ergebnisse als das lokal implementierte System.

5.3.3 Begründung der finalen Auswahl

Aufgrund der deutlich besseren Performance in Bezug auf Antwortzeit und Qualität der Analyse fiel die Entscheidung zugunsten der Variante mit Gemini. Trotz der Abhängigkeit von einem externen Anbieter erwies sich dieser Ansatz als wesentlich effizienter,

benutzerfreundlicher und robuster. Die einfache Integration, die hohe Flexibilität bei der Prompt Gestaltung sowie die überdurchschnittlich gute Texterkennung und -verarbeitung machten Gemini zur bevorzugten Lösung für das weitere Projekt.

5.4 Backend

Das Backend enthält die in der Architektur festgelegten Komponenten. Nachfolgend werden Stück für Stück die Umsetzung der einzelnen Bereiche aufgezeigt.

5.4.1 API

Die API ist mit Django-Ninja umgesetzt worden. Dabei wird mittels Annotation über einer Methode eine neue Route angelegt und entsprechende Werte an die Methode übergeben (Abbildung 4)

```
@archiveRouter.get(path="/", tags=["Archive"], summary="Get all archives", response=List[ArchiveSchema])
def getArchives(request):
    return Archive.objects.all()
```

Abbildung 4

5.4.1.1 Routen

Die Routen sind nach Entitäten aufgeteilt und enthalten GET, POST, PUT und DELETE Operationen. Abbildung 5 zeigt die Routen, die für die Authentifizierung notwendig sind. Zunächst benötigt der Client den CSRF-Token, um Posts machen zu können. Diese und die Login Route sind die einzigen, die sinnvollerweise keine Authentifizierung benötigen. Alle anderen Routen enthalten den impliziten Aufruf zur Authentifizierung.

| Auth | | |
|------|----------------------|----------------|
| GET | /api/auth/csrf-token | Get CSRF token |
| POST | /api/auth/login | Login |
| POST | /api/auth/logout | Logout |
| GET | /api/auth/user | Get User |

Abbildung 5

Abbildung 6 zeigt die Routen der PDFs. Dort können einzelne PDFs hochgeladen werden. Neben der exakten Suche mit der Id sind auch komplexere Suchen über ein Array im Body über alle PDFs eines Archivs möglich

Pdf

| | | |
|--------|---------------------------------|--------------------------------|
| POST | /api/pdfs/upload | Upload a PDF file |
| GET | /api/pdfs/{pdf_id} | Get PDF content |
| DELETE | /api/pdfs/delete/{pdf_id} | Delete PDF |
| POST | /api/pdfs/pdfSearch/{archiveId} | List all Pdf by Seachparameter |

Abbildung 6

Abbildung 7 zeigt die Routen der Archive. Neben der Auflistung aller Archive, können Archive mit der Id zurückgegeben werden. Außerdem ist eine Umbenennung eines Archivs möglich.

| Archive | | |
|---------|----------------------------|----------------------|
| GET | /api/archives/ | Get all archives |
| POST | /api/archives/ | Create a new archive |
| PUT | /api/archives/edit/{id} | Edit archive name |
| GET | /api/archives/{archive_id} | Get archive by ID |

Abbildung 7

Abbildung 8 zeigt die Attribute, die zu einem Archiv gehören. Hier können zu einem bestehenden Archiv Attribute hinzugefügt werden oder gelöscht werden.

Attribute

| | | |
|--------|----------------------|------------------|
| POST | /api/attributes/ | Add Attribute |
| DELETE | /api/attributes/{id} | Delete Attribute |

Abbildung 8

Abbildung 9 zeigt die Routen für die Types. Diese können gelistet und ergänzt werden. Eine Standardauswahl wird zu Beginn festgelegt bzw. importiert.

Type

| | | |
|------|-------------|-------------------|
| GET | /api/types/ | Get all types |
| POST | /api/types/ | Create a new type |

Abbildung 9

Abbildung 10 zeigt die Routen der AttributeValues. Neben den üblichen GET- und POST-Routen gibt es noch eine Route, die die Informationen von Gemini zurückgibt.

AttributeValues

| | | |
|------|--|---|
| GET | /api/attributeValues/list/{id} | Get Attribute Values By Pdf |
| GET | /api/attributeValues/listValues/{id} | Get Attribute Values By Pdf |
| POST | /api/attributeValues/{id} | Set Attribute Values By Pdf |
| GET | /api/attributeValues/geminiValues/{id} | Get Attribute Values By Pdf from Gemini |

Abbildung 10

5.4.1.2 Schema und Serialisierung

Django-Ninja kann in den Routen auch definierte Antwort und Anfrage Schemata verwenden (Abbildung 11).

```
class ArchiveSchema(ModelSchema):  # Nicolai Glock
    id: Optional[int] = None
    createOn: Optional[datetime] = None
    attributes: Optional[List[AttributesSchema]] = None

class Config:  # Nicolai Glock
    model = Archive
    model_fields = "__all__"
```

Abbildung 11

Diese bieten gleiche mehrere Vorteile:

- Zuordnung von JSON (Client) zur Schema Klasse
- Zuordnung von Django Querys zur Schema Klasse
- Automatische spezifische Fehlermeldung an den Client bei Nichteinhaltung des geforderten Schemas
- Klare Struktur durch Objektorientierung

Abbildung 12 zeigt die verwendeten Schemata.

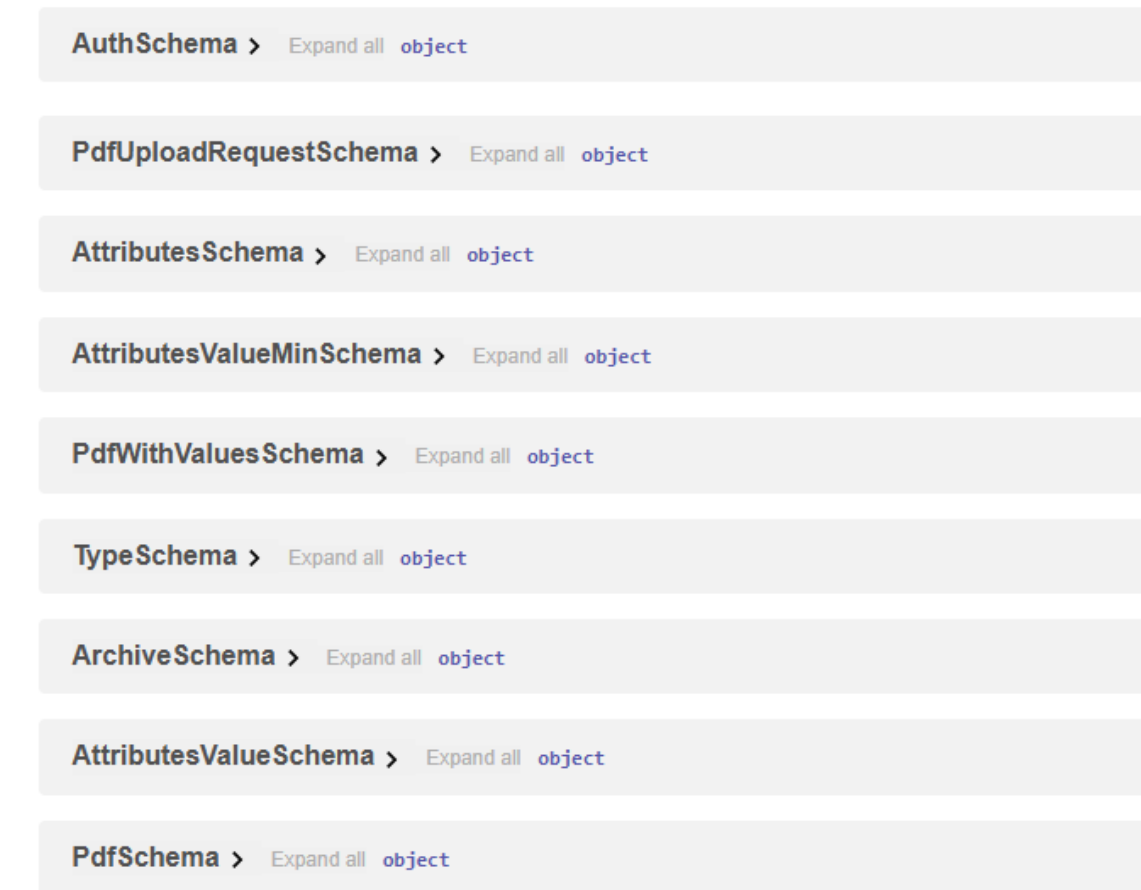


Abbildung 12

Alle Schemata und alle Routen werden durch die Verwendung in den Annotationen automatisch zu einer OpenApi.json konvertiert, diese kann dann als API-Client in der Entwicklung verwendet werden (Abbildung 13).

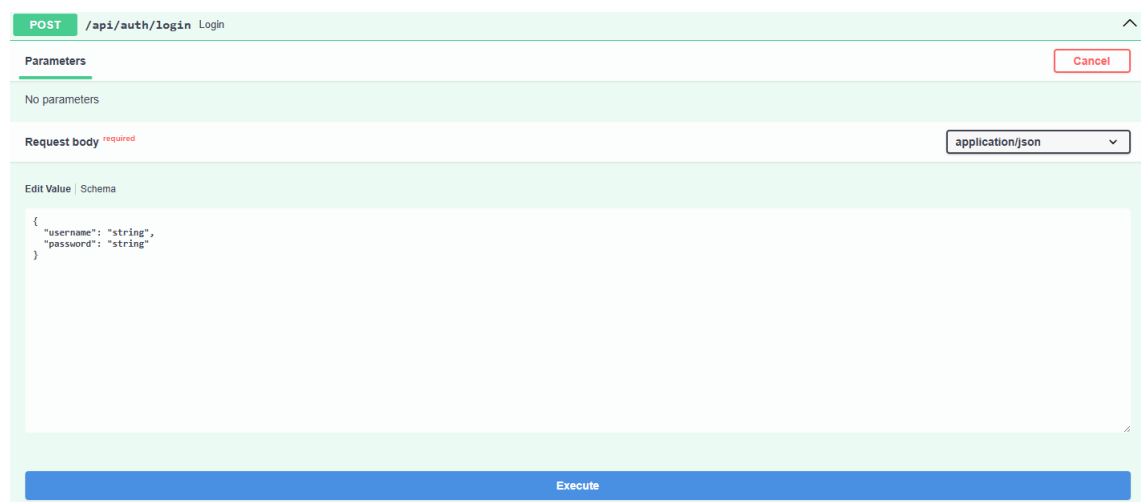


Abbildung 13

5.4.2 Logik

Die Logik der Anwendung wird in den Services der jeweiligen Entitäten definiert. Neben den üblichen CRUD (Create Read Update Delete) Operationen sind auch komplexere Teile notwendig. Beispielsweise für die Kommunikation zwischen dem Service und Gemini.

Zusammengefasst sieht die Logik folgendes vor. Ein Dokument kann zu einem bestimmten Archiv hochgeladen werden. Nach der Verarbeitung intern, wird das Dokument mithilfe der Gemini-API hochgeladen. Im Anschluss wird über ein Prompt Kommando Gemini aufgefordert die entsprechenden Attribute im Dokument zu suchen und als Array zurückzugeben. Dieser Array wird als Vorschlag an den Client gesendet. Der Client kann dann die Daten verifizieren, ggf. anpassen und speichern.

5.5 Frontend

5.5.1 Allgemein

Der Fokus im Frontend liegt auf einfache und schlichte Bedienung. Das User Interface sollte sich an verschiedene Bildschirmgrößen anpassen, ohne dass ein weiteres scrollen oder gar zoomen notwendig ist.

5.5.2 Aufbau

Das Frontend besteht in kleinster Stufe aus einer Komponente, die bestenfalls mehrfach verwendet werden kann. Eine oder mehrere Komponenten bilden eine View. Jede View entspricht genau einer Route. Die Route ist der verfügbare Pfad, der auf direkt über die Adressleiste erreicht werden kann (Abbildung 14).

Neben den Layouts werden für die Verarbeitung der Daten (Anfrage, Antwort) Stores verwendet. Diese halten den Stand global synchron. Dadurch wird vermieden, dass Komponenten bei Änderungen viele andere Komponenten benachrichtigen müssen, das übernimmt dann der Store.

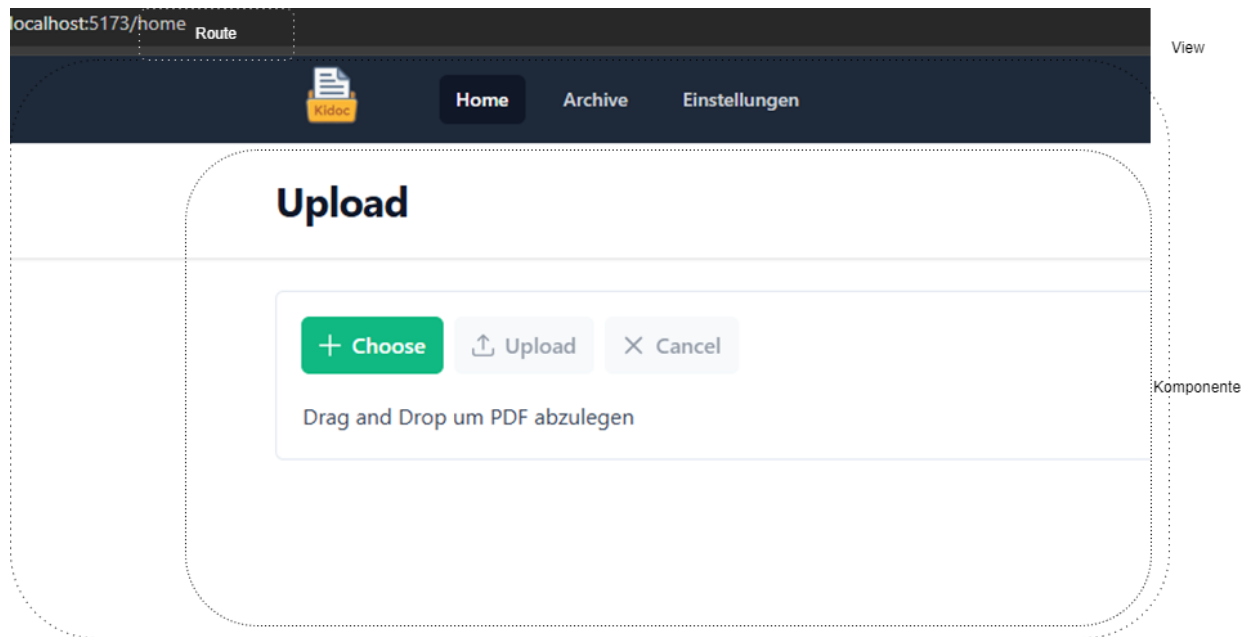


Abbildung 14

5.5.3 User Interface

Django bietet neben der Authentifizierung auch eine Administrationsseite, die für schnelle Änderungen ersatzweise als User Interface dienen kann. Änderungen, die bspw. nur gelegentlich oder von administrativer Hand ausgeführt werden, können darüber abgebildet werden (Abbildung 15).

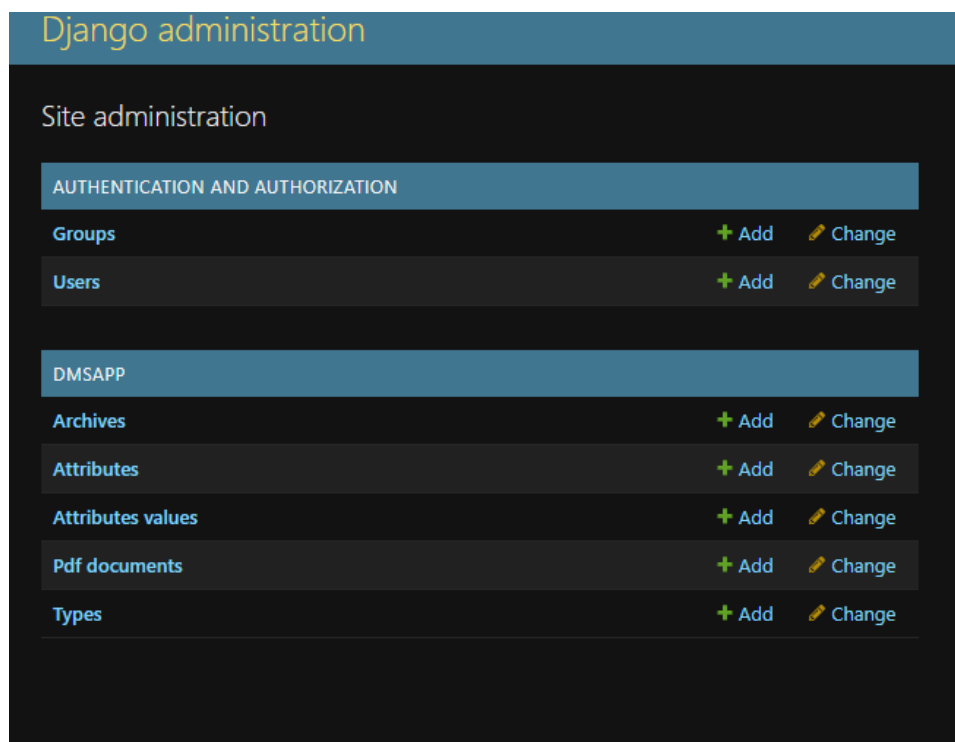


Abbildung 15

Jede einzelne Entität wird automatisch in eine ausfüllbare Form konvertiert (Abbildung 16).

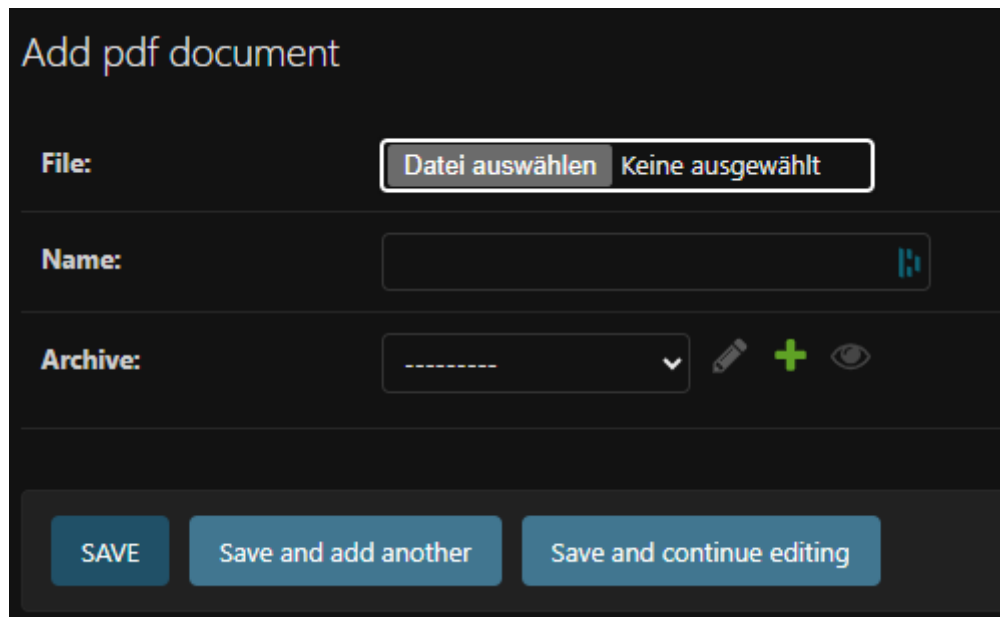
The image shows a web form titled "Add pdf document" on a dark background. It has three main input sections: "File:" with a button "Datei auswählen" and the text "Keine ausgewählt"; "Name:" with a text input field and a blue icon; and "Archive:" with a dropdown menu showing "-----" and icons for edit, add, and view. At the bottom, there are three buttons: "SAVE", "Save and add another", and "Save and continue editing".

Abbildung 16

Die Login Seite ist für die Authentifizierung notwendig (Abbildung 17). Bei Falschein-gabe wird eine Nachricht mit dem Hinweis gegeben, dass eine korrekte Eingabe erforderlich ist.

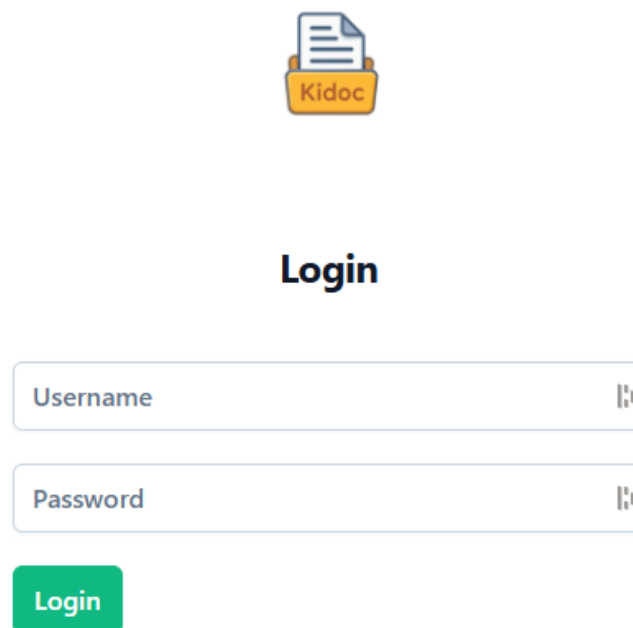
The image shows the login page for "Kidoc". At the top is a logo consisting of a document icon and the word "Kidoc". Below the logo is the heading "Login". There are two input fields: "Username" and "Password", each with a blue icon on the right. Below the password field is a green "Login" button.

Abbildung 17

Im Home-Bereich kann ein Dokument entweder über die Auswahl oder per Drag & Drop hochgeladen werden. Derzeit ist nur das PDF-Format zugelassen

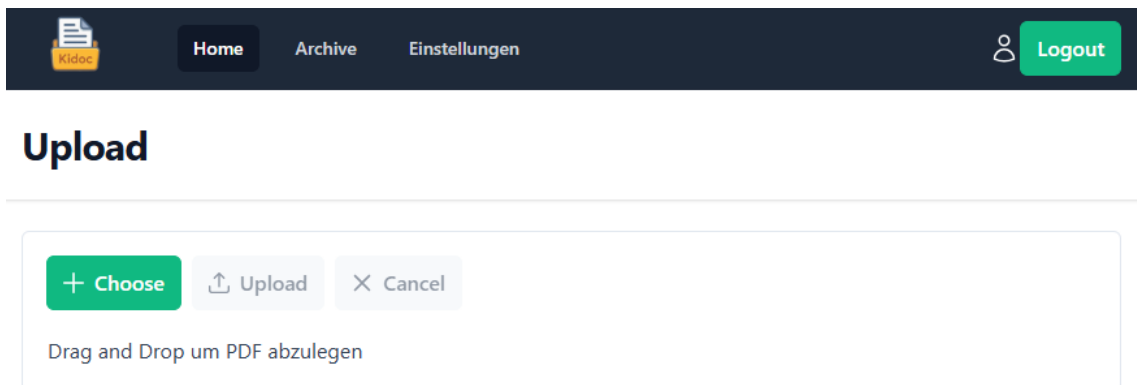


Abbildung 18

Nach dem Upload wird der Nutzer dazu aufgefordert ein Zielarchiv auszuwählen (Abbildung 19).

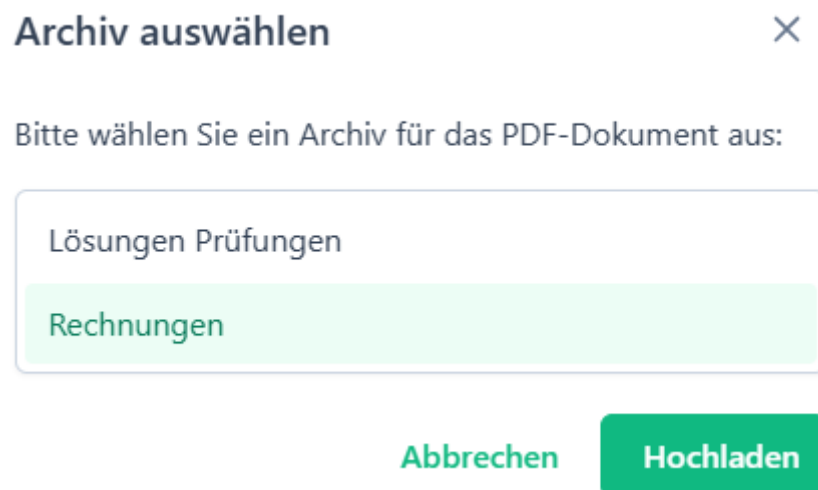


Abbildung 19

Im Anschluss wird das Dokument von Gemini verarbeitet und mit einem Vorschlag an den Nutzer zurückgegeben (Abbildung 20).

Attribute

Rechnungsdatum
24.09.2021

Rechnungstext
Commercial invoice

Rechnungsbetrag
1428,00

Attribute speichern

Commercial invoice

Webware Internet Solutions GmbH • Teichstr. 14-16 • 34130 Kassel

Agoratech
Teichstr. 14-16
34130 Kassel
Germany

Date:
Due Date:
Buyer ref.:
Delivery date:

24.09.2021
24.10.2021
139877
01.11.2021


Invoice Nr.: 2021_10

Commercial invoice

| Pos. | Description | Qty | Price | Total |
|----------------|--------------------|------------|------------|---------------------|
| 10 | Project management | 2,00 Days | 500,00 EUR | 1.000,00 EUR |
| 20 | Consulting | 5,00 Hours | 40,00 EUR | 200,00 EUR |
| Subtotal | | | | 1.200,00 EUR |
| Tax 19,00% | | | | 228,00 EUR |
| Total | | | | 1.428,00 EUR |
| Paid | | | | 0,00 EUR |
| Balance | | | | 1.428,00 EUR |

Abbildung 20

Unter dem Navigationspunkt Archive, werden alle bestehenden Archive gelistet (Abbildung 21). Hier kann neben der Neuerstellung eines Archivs auch eine Suchmaske geöffnet werden.

| | | | |
|---|------|---------|---------------|
|  | Home | Archive | Einstellungen |
|---|------|---------|---------------|

Archive

Neues Archiv

| Name | Erstellt am |
|------------------------------------|--------------------------|
| Lösungen Prüfungen | 2025-06-07T16:44:17.336Z |
| Rechnungen | 2025-06-09T12:41:28.424Z |

Abbildung 21

In der Suche kann ein Dokument über die gespeicherten Attribute gesucht werden (Abbildung 22).

Suche

Rechnungen

Suchen

| Dokumentenname | Rechnungsdatum | Rechnungstext | Rechnungsbetrag |
|-----------------------|----------------|---------------|-----------------|
| keine Suchergebnisse. | | | |

Abbildung 22

Archive können neu erstellt werden. Neben einer Bezeichnung werden Attribute hinzugefügt (Abbildung 23). Diese werden jeweils einem Datentyp zugewiesen (Abbildung 24).

Neues Archiv

Neues Attribut

Attribute

| Bezeichnung | Typ |
|-------------|-----|
|-------------|-----|

Archiv Speichern

Abbildung 23

Neues Attribut

Label

Name

Feldart

text

date

number

Abbildung 24

6 Herausforderungen

Während der Umsetzung des Studienprojekts traten verschiedene Herausforderungen auf. Eine der zentralen Herausforderungen lag in der Suche und Auswahl einer passenden Künstlichen Intelligenz (KI) für die Aufgabe der Analyse, Klassifikation und Verschlagwortung von PDF-Dateien. Es gibt eine Vielzahl von KI-Modellen und APIs, die sich in ihren Fähigkeiten und Komplexitäten erheblich unterscheiden.

- Vielfalt und Komplexität der KI-Modelle: Es gab eine große Auswahl an KI-Modellen, von generischen Sprachmodellen bis hin zu Modellen, die auf bestimmte Dateitypen zugeschnitten sind. Die Entscheidung für das optimale Modell erforderte eine genaue Analyse und vor allem ein Testlauf, mit den verschiedensten Modellen, damit die Anforderungen des Projekts am besten abgedeckt werden.
- Kostenlose Modelle: Ein weiteres Problem war oft die mangelnde Zuverlässigkeit vieler kostenloser KI-Modelle. Diese zeigten sich in den Tests häufig als weniger präzise in ihren Ergebnissen, was dann immer ein Ausschlusskriterium ist. Dies führte immer zu einem zeitlichen Aufwand.

Letzten Endes fiel die Wahl auf die Google Gemini API für die KI-gestützten Funktionen. Diese API und auch der Chatbot Gemini sind für allgemeine Zwecke konzipiert und ermöglichen eine Interaktion über Prompts. Hierbei war es auch herausfordernd, konsistente Ausgaben im gleichen Format bei jeder Antwort der KI sicherzustellen. Das wurde Zeitintensiv durch Trial-and-Error gelöst. Trotzdem erwies sich Gemini hier, den anderen getesteten KI-Modellen, weit überlegen und wurde gewählt, um das Projekt zu realisieren.

7 Zusammenfassung und Ausblick

Die Studienarbeit befasste sich mit der Entwicklung eines KI-gestützten Systems zur Archivierung indexierter PDF-Dokumente. Das zentrale Ziel den manuellen Aufwand zu reduzieren und Informationen Auffindbarer zu machen, dank der KI, konnte erfolgreich umgesetzt werden. Das entwickelte Projekt ermöglicht es relevante Informationen aus den PDFs durch die KI zu erhalten und dazu gibt es auch eine Suchfunktion, um diese Informationen besser zu finden. Durch die Kombination von Django im Backend, Vue im Frontend und der Google Gemini API wurde eine zufriedenstellende Lösung geschaffen. Wichtige Erkenntnisse des Projekts umfassten, zum einen die Herausforderung das richtige KI-Modell zu finden, bei der großen Auswahl an vorhandenen Modellen und auch die Einarbeitung in die verschiedenen Technologien, auch wenn man viele doch schonmal benutzt hat. Natürlich war es auch das erste Mal, das man so richtig mit KI-Modellen in Berührung kam und auch da hat man sich das anders vorgestellt. Zum Schluss erwies sich die Google API als überlegen, auch wenn die Ausgabeformate anfangs nicht sehr konsistent waren.

Für die zukünftige Weiterentwicklung des Systems bieten sich zahlreiche Ansatzpunkte. Eine mögliche Erweiterung wäre das Hinzufügen von weiteren gängigen Dokumentdateien und nicht nur PDF-Dateien. So würde man ein größeres Spektrum abdecken und noch mehr Anwendungsfälle unterstützen. Natürlich wäre ein nächster Schritt aber erstmal die Unterstützung von nicht-indexierten PDFs, um auch bildbasierte oder gescannte PDF-Dokumente verarbeiten zu können. Dies würde die Funktionalität des Systems erheblich erweitern, insbesondere im Hinblick auf die Bereiche in denen häufig mit eingescannten Dokumenten gearbeitet wird. Zusätzlich müsste auch ein Performentest durchgeführt werden, um die Effizienz und Skalierbarkeit des Systems unter realen Bedingungen sicherzustellen und gegebenenfalls Optimierungen vorzunehmen. Es gibt viele weitere Möglichkeiten, das Projekt sinnvoll weiterzuentwickeln, wie dass das Modell durch Nutzerfeedback dazulernt, aber hiermit wäre die Basis für die zukünftige Weiterentwicklung gelegt.

Literaturverzeichnis

Alkhamis, Mahmoud Ahmad. 2023. *Extraktion textueller Informationen aus heterogenen PDF-Dokumenten.* [Online] 15. 03 2023. [Zitat vom: 19. 06 2025.] <https://eprints.dbis.informatik.uni-rostock.de/1096/1/15.03.23.Mahmoud.Khamis.pdf>.

Behal, Garima. 2025. *10 Best AI PDF Data Extractors for Information Mining In 2025.* [Online] 06. 06 2025. [Zitat vom: 19. 06 2025.] <https://clickup.com/blog/pdf-data-extractors/>.

Datagrid. 2025. *Master Automated PDF Indexing Using Ddatagrid's AI Platform.* [Online] 11. 02 2025. [Zitat vom: 19. 06 2025.] <https://www.datagrid.com/blog/automate-pdf-indexing>.

DataScientest. 2023. *PyCharm: Alles über die beliebteste Python-IDE.* [Online] 07. 05 2023. [Zitat vom: 20. 06 2025.] <https://datascientest.com/de/pycharm>.

Django Ninja. [Online] [Zitat vom: 19. 06 2025.] <https://django-ninja.dev/>.

Haider, Khurram. 2025. *Informationsextraktion mittels natürlicher Sprachverarbeitung (NLP).* [Online] 10. 06 2025. [Zitat vom: 19. 06 2025.] <https://www.astera.com/de/type/blog/nlp-information-extraction/>.

HubSpot Inc. 2022. *Tailwind CSS: What It Is, Why Use It & Examples.* [Online] 31. 05 2022. [Zitat vom: 19. 06 2025.] <https://blog.hubspot.com/website/what-is-tailwind-css>.

IBM. *Was ist Django?* [Online] [Zitat vom: 19. 06 2025.] <https://www.ibm.com/de-de/topics/django>.

Kinsta Inc. 2023. *Was ist TypeScript? Ein umfassender Leitfaden.* [Online] 09. 10 2023. [Zitat vom: 19. 06 2025.] <https://kinsta.com/de/wissensdatenbank/was-ist-typescript/>.

Medium Harshith Gowda. 2025. *medium.* [Online] 03. 06 2025. [Zitat vom: 15. 06 2025.] <https://medium.com/%40harshithgowdakt/databases-vs-blob-storage-what-to-use-and-when-d5b1ec0d11cd>.

Pinia. *Introduction.* [Online] [Zitat vom: 19. 06 2025.] <https://pinia.vuejs.org/introduction.html>.

PLANET AI GmbH. 2024. *PDF-Klassifikation: Effizienz durch KI-basierte, intelligente Automatisierung.* [Online] 16. 04 2024. [Zitat vom: 19. 06 2025.] <https://planet-ai.com/de/pdf-klassifikation/>.

POSTGRESQL. *About.* [Online] [Zitat vom: 19. 06 2025.] <https://www.postgresql.org/about/>.

PRIMEVUE. *Introduction.* [Online] [Zitat vom: 19. 06 2025.]

<https://primevue.org/introduction/>.

Schneider, Maximilian. 2025. *Digitale Archivierung durch künstliche Intelligenz.*

[Online] 16. 02 2025. [Zitat vom: 19. 06 2025.] <https://konfuzio.com/de/digitale-archivierung/>.

Seng, Steven Ang Cheong. 2025. *Guide: What is Google Gemini API and How to Use it?*

[Online] 13. 05 2025. [Zitat vom: 19. 06 2025.] <https://apidog.com/blog/google-gemini-api/>.

Susnjara, Stephanie und Smalley, Ian. 2024. *Was ist Docker?* [Online] 06. 06 2024.

[Zitat vom: 19. 06 2025.] <https://www.ibm.com/de-de/think/topics/docker>.

Vue.js. [Online] [Zitat vom: 19. 06 2025.] <https://vuejs.org/guide/introduction>.

Wolfenstein, Konrad. 2025. *KI-gestützte PDF-Reader und Alternativen zu Adobe Acrobat: Ein umfassender Vergleich für Unternehmen, Einzelanwender und Schüler.*

[Online] 20. 02 2025. [Zitat vom: 19. 06 2025.] <https://xpert.digital/ki-gestuetzte-pdf-reader/>.

Ehrenwörtliche Erklärung

| | | | |
|---------------|--------|--------------|------------------------------------|
| Name: | Glock | Vorname: | Nicolai |
| Matrikel-Nr.: | 768801 | Studiengang: | Softwaretechnik / Medieninformatik |
| Name: | Cal | Vorname: | Baran Ibrahim |
| Matrikel-Nr.: | 768960 | Studiengang: | Softwaretechnik / Medieninformatik |

Hiermit versichern wir, Glock, Nicolai und Cal, Baran Ibrahim dass wir die vorliegende Studienarbeit mit dem Titel KI gestützte Archivierung indexierter PDFs selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebene Literatur und Hilfsmittel verwendet habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

24.06.2025

Datum



Unterschrift

24.06.2025

Datum



Unterschrift