

Entwicklung eines Dokumentenerkennungsdienstes

Bachelor Thesis

Submitted by Patrick Richter

In fulfilment of the requirements for the degree
Bachelor of Science
To be awarded by the
Fontys University of Applied Sciences

Moers, 11. Januar 2021

Information Page

Fontys Hogeschool Techniek en Logistiek
Postbus 141, 5900 AC Venlo

Project Plan for Bachelor Thesis Project

Name of student: Patrick Richter
Student number: 3162869
Course: Informatics - Software Engineering
Period: September 2020 - January 2021

Company name: HMM Deutschland GmbH
Address: Eurotec-Ring 10
Postcode, City: 47445, Moers
Country: Deutschland

Company coach: Reiner Riksen
Email: riksen@hmmdeutschland.de
University coach: Frank van Gennip
Email: f.vangennip@fontys.nl

Examinator: Jan Jacobs
External domain expert: Thijs Dorssers

Number of words: 10933
Non-disclosure agreement: No

Statement of Authenticity

I, the undersigned, hereby certify that I have compiled and written the attached document / piece of work and the underlying work without assistance from anyone except the specifically assigned academic supervisors and examiners. This work is solely my own, and I am solely responsible for the content, organization, and making of this document / piece of work.

I hereby acknowledge that I have read the instructions for preparation and submission of documents / pieces of work provided by my course / my academic institution, and I understand that this document / piece of work will not be accepted for evaluation or for the award of academic credits if it is determined that it has not been prepared in compliance with those instructions and this statement of authenticity.

I further certify that I did not commit plagiarism, did neither take over nor paraphrase (digital or printed, translated or original) material (e.g. ideas, data, pieces of text, figures, diagrams, tables, recordings, videos, code, ...) produced by others without correct and complete citation and correct and complete reference of the source(s). I understand that this document / piece of work and the underlying work will not be accepted for evaluation or for the award of academic credits if it is determined that it embodies plagiarism.

Name: Patrick Richter
Student Number: 3162869
Place/Date: Moers, 11. Januar 2021

Signature:

Abstract

Deutsch

Dieser Bericht beschreibt die Entwicklung eines Dokumentenerkennungsdienstes. Ziel des Dienstes ist die Erkennung von ärztlichen Muster 4 Verordnungen für eine Erweiterung des Produktes DeTou-
ro der HMM Deutschland GmbH. Es wurde eine detaillierte Analyse ausgeführt, auf Basis welcher eine Reihe von Anforderungen definiert wurden.

Um ein passendes Framework für die Dokumentenerkennung zu identifizieren wurde eine ausführliche Untersuchung durchgeführt. Es wurde, neben einer Reihe von fertigen Diensten, auch die Option einer eigenen Open Source Implementation, auf Basis von Objekt- und Texterkennungsdiensten, evaluiert. Die Open Source Alternative konnte keine nutzbaren Ergebnisse produzieren. Der von Microsoft, in der Cloud Plattform Azure, angebotene Dienst Form Recognizer wurde bei dieser Untersuchung als am besten geeigneten identifiziert. Dementsprechend wurde eine Empfehlung für die Nutzung dieses Dienstes ausgesprochen.

In der nachfolgenden Design-Phase wurde der Prozessablauf und der Aufbau geplant sowie die Wahl der Authentifizierungsstrategie erläutert. Die Implementation beschreibt im Detail wie die Build Vorgänge, mit Hilfe von Docker, durchgeführt werden, sowie die Einrichtung von Continuous-Integration- und Continuous-Deployment-Pipelines auf Basis, der von GitLab zur Verfügung gestellten, Features. Anschließend werden die Implementation der API und der Konvertierung der Erkennungsergebnisse beschrieben. Zusätzlich wird auch ein Prototyp zur Transformation von Fotos in Scan-artige Bilder erläutert.

English

This report describes the development of a document recognition service. Goal of the service is the recognition of “Muster 4” prescriptions, as an extension for the HMM Deutschland GmbH product DeTouro. A detailed analysis has been conducted, based on which a number of requirements have been defined.

To identify a suitable framework for the document recognition, a complex research has been conducted. Next to a number of premade services, an open source alternative, based on object and text recognition services, is also evaluated. The open source alternative failed to produce meaningful results. The Form Recognizer service, offered by Microsoft as part of the Azure platform, has been identified as most suitable. Based on that the service has been recommended for the implementation.

During the design phase the process, as well as the structure, of the service have been planned. Additionally, the authentication strategy is also explained. The implementation part describes the setup of Docker, as well as the configuration of Continuous-Integration- and Continuous-Deployment-Pipelines, based on GitLab. After that the implementation of the API and the conversion of the recognition results is described. Additional further explanations for a prototype, for converting photos to scan-like images, have been provided.

Inhaltsverzeichnis

Information	II
Statement of Authenticity	III
Abstract	IV
Inhaltsverzeichnis	VI
Abbildungsverzeichnis	IX
Tabellenverzeichnis	X
Akronyme	XI
Glossar	XII
1 Einleitung	1
1.1 Hintergrund	1
1.2 Firma	1
1.3 Motivation	1
1.4 Kontext	2
1.5 Ziele	2
1.6 Scope	4
1.6.1 Out of scope	5
2 Projekt	6
2.1 Stakeholder	6
2.2 Ansatz	7
2.3 Planung	8
2.4 Qualitätskontrolle	8
3 Analyse	10
3.1 Ist-Situation	10

3.2	Soll-Situation	10
3.3	Integration	11
3.4	Abnahmekriterien	11
3.5	Risiko Analyse	13
4	Empfehlung	15
4.1	Metriken	15
4.1.1	Effektivität	15
4.1.2	Preis	16
4.1.3	Datenschutz	16
4.1.4	Technische Anforderungen	17
4.2	Testdaten	17
4.3	Verfügbare Optionen	17
4.4	Untersuchung	18
4.4.1	Amazon Textract	18
4.4.2	Microsoft Form Recognizer	19
4.4.3	Google Document AI	19
4.4.4	Open Source	20
4.5	Ergebnisse	23
5	Design	25
5.1	Prozessablauf	25
5.2	Aufbau	25
5.3	Authentifizierung	27
6	Implementation	29
6.1	Docker	29
6.1.1	Dockerfile	29
6.1.2	Healthchecks	29
6.2	CI/CD-Pipelines	30
6.3	Analyse-Endpunkt	30
6.4	Konvertierung	31

6.5	Pre-Processing	33
7	Zusammenfassung	37
7.1	Bewertung	37
7.2	Ausblick	37
A	Anforderungen	39
B	Untersuchungsergebnisse	40
C	Klassendiagramm	44

Abbildungsverzeichnis

1.1	Beispiel einer ausgefüllten Muster 4 Verordnung	3
1.2	DeTouro Prozesse zur Abwicklung von Krankenfahrten	4
2.1	Power-Interest-Diagramm	6
2.2	Ausschnitt aus dem Jira Board	7
3.1	Definition der Metriken (Cook und Ramadas, 2020)	12
4.1	Durch RetinaNet erkannte Label	22
5.1	Sequenz-Diagramm für den Ablauf des API-Aufrufs	26
6.1	Konfiguration für die Test-Stage der CI-Pipeline	31
6.2	Quellcode für den Muster 4 Analyse Endpunkt	32
6.3	Quellcode für die Konvertierung von Boolean Wertefeldern	33
6.4	Quellcode für die Transformation eines Bildes einer Verordnung	35
6.5	Foto einer Muster 4 Verordnung vor und nach dem Transformieren	36
C.1	Klassendiagramm (Zur Übersichtlichkeit wurden die Eigenschaften von Muster4Result ausgeblendet)	44

Tabellenverzeichnis

1.1	Endprodukte	4
2.1	Stakeholder	6
2.2	Aufgabenplanung	9
3.1	Abnahmekriterien für den Dokumentenerkennungsdienst	13
4.1	Metriken für die Bewertung der Effektivität der Dokumentenerkennungs-Frameworks	16
4.2	Verfügbare Optionen	18
4.3	Ergebnisse verschiedener Objekterkennungssysteme bei Einsatz auf dem COCO Datensatz	21
4.4	Simple HTR Handschrifterkennungsergebnisse	23
4.5	Preise für die Nutzung der Dokumentenerkennungsdienste	24
A.1	Funktionale Anforderungen an den Dienst	39
A.2	Nicht-Funktionale Anforderungen an den Dienst	39
B.1	Ergebnisse der einzelnen Frameworks bei der Erkennung von handschriftlich ausgefüllten Muster 4 Verordnungen	40
B.2	Ergebnisse der einzelnen Frameworks bei der Erkennung von maschinell ausgefüllten Muster 4 Verordnungen	40
B.3	Amazon Textract Handschrifterkennungsergebnisse	41
B.4	Microsoft Form Recognizer Handschrifterkennungsergebnisse	42
B.5	Google Document AI Handschrifterkennungsergebnisse	43

Akronyme

AWS Amazon Web Services.

CD Continous Deployment.

CER Character Error Rate.

CI Continous Integration.

KTA Krankentransportanfrage.

OCR Optical-Character-Recognition.

RBAC Role Based Acces Control.

SSO Single Sign-On.

Glossar

CD Continus Deployment beschreibt das automatisierte Vorbereiten und ausspielen eines Releases.

CI Continus Integration beschreibt automatische Build Vorgänge nachdem Pushen von Änderungen in einem Repository.

Confidence Die Confidence gibt den Grad an Vertrauen in ein Erkennungsergebnis an.

DeTouro Ein Produkt der HMM Deutschland GmbH zur Vergabe von Krankenfahrten.

Fahrdienstleister Erbringer von Fahrleistungen, z.B. Taxi Unternehmen.

Kostenträger Trägt die Kosten einer Verordneten Leistung, meistens Krankenkassen.

Leistungserbringer Erbringer von Leistungen, z.B. Therapeuten, Apotheken oder Ärzte.

RBAC Role Based Access Control ist ein Model für die Steuerung von Zugriff auf Systemressourcen durch an Nutzer vergebene Rollen.

SDK Ein Software Development Kit ist eine Sammlung von funktionen die oft als verbindendes Element für Fremdsysteme genutzt wird.

Single Sign-On Single Sign-On erlaubt Nutzern sich mit einem Login bei mehreren Systemen anzumelden.

1. Einleitung

In diesem Kapitel werden die Hintergründe des Projektes aufgeführt. Die Motivation wird geschildert und der Kontext erklärt. Zudem werden auch die Ziele und der Umfang des Projektes beschrieben.

1.1 Hintergrund

Als Abschluss des Software Engineering Studienganges an der Fontys International University of Applied Sciences, muss eine Bachelor Arbeit verfasst werden. Diese wird auf Basis eines Projektes, diesem Bericht und einer Präsentation bewertet und entscheidet über die Vergabe des Bachelor Diploms. Ziel dieser Arbeit ist, dass die Studenten ihre Fähigkeiten unter Beweisstellen und so ihre Eignung zum Erhalt des Diploms beweisen. Ich schreibe diese Arbeit bei der HMM Deutschland GmbH im Winter Semester 2020/2021.

1.2 Firma

Die HMM Deutschland GmbH ist ein Anbieter für diverse Software Lösungen für das Gesundheitswesen. Zu den Kunden zählen, neben Krankenkassen als Kostenträger, auch diverse Leistungserbringer wie z.B. Arztpraxen, Physiotherapeuten und Krankenfahrdienste. Die angebotenen Produkte erlauben, unter anderem, die digitale Verarbeitung von Patientendaten und eine vereinfachte digitale Abrechnung zwischen Leistungserbringern und Kostenträgern.

Die Firmenzentrale befindet sich in Moers, eine Außenstelle in Berlin. Insgesamt werden ca. 150 Mitarbeiter, primär am Standort Moers, beschäftigt. Diese sind zuständig für die Entwicklung, das Testen, Hosting und die Vermarktung der Produkte, sowie den Kundensupport für diese.

Für den Rahmen meiner Bachelor Arbeit bin ich im Team LEOS eingesetzt, welches für die Entwicklung und Wartung von Systemen, für die Leistungserbringer, zuständig ist.

1.3 Motivation

Im deutschen Gesundheitssystem werden ärztliche Verordnungen auf standardisierten Vordrucken handschriftlich oder maschinell erstellt, wobei der handschriftlich erstellte Anteil, mit ca. 40%, knapp in der Minderheit liegt. Der Patient muss diese dann entweder an seinen Kostenträger, zur Genehmigung, weiterleiten oder er wendet sich direkt an einen Leistungserbringer zur Inanspruchnahme der verordneten Leistungen. Auch dieser muss jedoch die Verordnung, zum Zwecke der Abrechnung, an den entsprechenden Kostenträger weiterleiten.

Die Übertragung und Verarbeitung der Verordnungen, zu und bei den Kostenträgern, erfolgt zunehmend digital, da eine digitalisierte Form viele Vorteile im Bereich der Übertragung, Bearbeitung

und Archivierung bietet. Der dadurch verursachte Technologiebruch, von papierbasierten zu digitalen Verordnungen, bringt einigen Hürden mit sich. Obwohl es von verschiedenen Seiten, auch der HMM Deutschland GmbH, Projekte gibt, um Verordnungen vollständig digital zu managen, vom ausstellenden Arzt, über die Kostenträger bis zum Leistungserbringer, werden diese Systeme es nicht schaffen alle papierbasierten Verordnungen, in der nahen Zukunft, abzulösen. Es besteht deshalb, zumindest kurz- bis mittelfristig, der Bedarf für ein System, welches das digitalisieren von papierbasierten Verordnungen automatisiert bzw. teilautomatisiert abwickeln kann.

1.4 Kontext

Zur automatisierten Digitalisierung von Verordnungen soll ein System entwickelt werden, welches die “Verordnung einer Krankenfahrt” (Abbildung 1.1), auch unter der Bezeichnung “Muster 4 Verordnung” bekannt, erkennen kann. Zu diesem Zweck wurde ein Projekt in Leben gerufen, welches im Rahmen der Bachelorarbeit durchgeführt werden soll.

Der Bedarf für ein System zur Erkennung von diesen Muster 4 Verordnungen, stammt aus der Zielsetzung eine neue Erweiterung für DeTouro, ein Produkt der HMM Deutschland GmbH, zu entwickeln. DeTouro ist eine Plattform die es Kostenträgern ermöglicht Krankenfahrten, als Krankentransportanfragen (KTA), auszuschreiben (Abbildung 1.2a). Fahrdienstleister können dann Gebote für diese abgeben. Der Gewinner wird durch den Kostenträger ausgewählt und anschließend der Leistungserbringer mit der Durchführung der Krankenfahrten beauftragt.

Dieses Gebots-Verfahren ist langfristig und deshalb nur bei Fahrten umsetzbar die bereits lange im Voraus bekannt sind. Zudem müssen nicht alle Verordnungen durch die Kostenträger genehmigt werden. Um DeTouro mit der Fähigkeit auszustatten auch diese, nicht genehmigungspflichtigen und kurzfristigen, Fahrten abwickeln zu können, soll eine Erweiterung erstellt werden, welche es den Fahrdienstleistern erlaubt die Fahrten über DeTouro abzurechnen, indem sie nach abgeschlossenen Krankenfahrten die Verordnung und Fahrtennachweise über DeTouro an denn zuständigen Kostenträger, in den allermeisten Fällen die Krankenkasse des Patienten, weiterleiten (Abbildung 1.2b).

An diesem Punkt setzt das Projekt an, denn für die Verarbeitung der Verordnung muss ihr Inhalt digitalisiert werden. Um diese Aufgabe nicht den Fahrdienstleistern, durch manuelles abtippen, aufzubürden, soll ein System entwickelt werden das die Inhalte der Verordnung automatisch auslesen kann und so die Fahrdienstleister entlastet. Ein solches System hat nicht nur das Potential die Verarbeitungszeit zu verkürzen, sondern kann auch die Fehlerquote reduzieren.

1.5 Ziele

Als Ziel des Projektes soll ein Dienst zur Verfügung gestellt werden, welcher dazu in der Lage ist, mit hoher Präzision, den Inhalt einer Verordnung zu analysieren und für eine weitere Verarbeitung verfügbar zu machen. Um einen Reibungslosen Einsatz in den Systemen der HMM Deutschland GmbH zu ermöglichen, muss der Dienst über Docker einsetzbar sein und soll Fähigkeiten zur Überwachung des eigenen Status besitzen.

Anfangs soll dieser Dienst ausschließlich für DeTouro eingesetzt werden, d.h. es müssen lediglich

<p>Zuzahlungspflicht <input checked="" type="checkbox"/></p> <p>Zuzahlungsfrei <input type="checkbox"/></p>	<p>Krankenkasse bzw. Kostenträger KR De-Touvo</p> <p>Name, Vorname des Versicherten Ast, Peter geb. am 31.12.69</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">Kostenträgerkennung 887654321</td> <td style="width: 33%;">Versicherten-Nr. A123456778</td> <td style="width: 33%;">Status 12691</td> </tr> <tr> <td>Betriebsstätten-Nr. 200133500</td> <td>Arzt-Nr. 123447749</td> <td>Datum 13.05.20</td> </tr> </table>	Kostenträgerkennung 887654321	Versicherten-Nr. A123456778	Status 12691	Betriebsstätten-Nr. 200133500	Arzt-Nr. 123447749	Datum 13.05.20	<h3 style="text-align: center;">Verordnung einer Krankenbeförderung 4</h3> <p><input checked="" type="checkbox"/> Unfall, Unfallfolge</p> <p><input type="checkbox"/> Arbeitsunfall, Berufskrankheit</p> <p><input type="checkbox"/> Versorgungsleiden (z.B. BVG)</p> <p><input checked="" type="checkbox"/> Hinfahrt <input checked="" type="checkbox"/> Rückfahrt</p>		
Kostenträgerkennung 887654321	Versicherten-Nr. A123456778	Status 12691								
Betriebsstätten-Nr. 200133500	Arzt-Nr. 123447749	Datum 13.05.20								
<h4>1. Grund der Beförderung</h4> <p>Genehmigungsfreie Fahrten</p> <p>a) <input type="checkbox"/> voll-/teilstationäre Krankenhausbehandlung <input checked="" type="checkbox"/> vor-/nachstationäre Behandlung</p> <p>b) <input type="checkbox"/> ambulante Behandlung bei Merkzeichen „aG“, „Bl“, „H“, Pflegegrad 3 mit dauerhafter Mobilitätsbeeinträchtigung, Pflegegrad 4 oder 5 nur Taxi/Mietwagen (Fahrt mit KTW ist unter f) zu verordnen)</p> <p>c) <input type="checkbox"/> anderer Grund, z.B. Fahrten zu Hospizen: _____</p> <p>Genehmigungspflichtige Fahrten zu ambulanten Behandlungen (vor Fahrtantritt der Krankenkasse vorzulegen)</p> <p>d) <input type="checkbox"/> hochfrequente Behandlung (Dialyse, onkol. Chemo- oder Strahlentherapie) <input type="checkbox"/> vergleichbarer Ausnahmefall (Begründung unter 4. erforderlich)</p> <p>e) <input type="checkbox"/> dauerhafte Mobilitätsbeeinträchtigung vergleichbar mit b) und Behandlungsdauer mindestens 6 Monate (Begründung unter 4. erforderlich)</p> <p>f) <input type="checkbox"/> anderer Grund für Fahrt mit KTW, z.B. fachgerechtes Lagern, Tragen, Heben erforderlich (Begründung unter 3. und ggf. 4. erforderlich)</p>										
<h4>2. Behandlungstag/Behandlungsfrequenz und nächsterreichbare, geeignete Behandlungsstätte</h4> <p>vom/am 270520 / <input type="checkbox"/> x pro Woche, bis voraussichtlich 271120</p> <p>Behandlungsstätte (Name, Ort) St. Josef Krankenhaus Moers</p>										
<h4>3. Art und Ausstattung der Beförderung</h4> <table style="width: 100%;"> <tr> <td><input checked="" type="checkbox"/> Taxi/Mietwagen:</td> <td><input type="checkbox"/> Rollstuhl</td> </tr> <tr> <td><input type="checkbox"/> KTW, da medizinisch-fachliche Betreuung und/oder Einrichtung notwendig ist wegen _____</td> <td><input type="checkbox"/> Tragestuhl</td> </tr> <tr> <td></td> <td><input type="checkbox"/> liegend</td> </tr> <tr> <td><input type="checkbox"/> RTW <input type="checkbox"/> NAW/NEF <input type="checkbox"/> andere _____</td> <td></td> </tr> </table>			<input checked="" type="checkbox"/> Taxi/Mietwagen:	<input type="checkbox"/> Rollstuhl	<input type="checkbox"/> KTW, da medizinisch-fachliche Betreuung und/oder Einrichtung notwendig ist wegen _____	<input type="checkbox"/> Tragestuhl		<input type="checkbox"/> liegend	<input type="checkbox"/> RTW <input type="checkbox"/> NAW/NEF <input type="checkbox"/> andere _____	
<input checked="" type="checkbox"/> Taxi/Mietwagen:	<input type="checkbox"/> Rollstuhl									
<input type="checkbox"/> KTW, da medizinisch-fachliche Betreuung und/oder Einrichtung notwendig ist wegen _____	<input type="checkbox"/> Tragestuhl									
	<input type="checkbox"/> liegend									
<input type="checkbox"/> RTW <input type="checkbox"/> NAW/NEF <input type="checkbox"/> andere _____										
<h4>4. Begründung/Sonstiges</h4> <p>(z. B. Datum Aufnahme Krankenhaus, Gewicht bei Schwergewichtstransport, Wartezeit, Gemeinschaftsfahrt, Ortsangabe, wenn Beförderung nicht von/zur Wohnung stattfindet)</p> <p>2 Fahrten</p>		<div style="border: 1px solid black; padding: 10px; text-align: center;">  <p>Vertragsarztstempel / Unterschrift des Arztes</p> </div> <p style="text-align: right; font-size: small;">Muster 4 (7.2020)</p>								

Abbildung 1.1: Beispiel einer ausgefüllten Muster 4 Verordnung

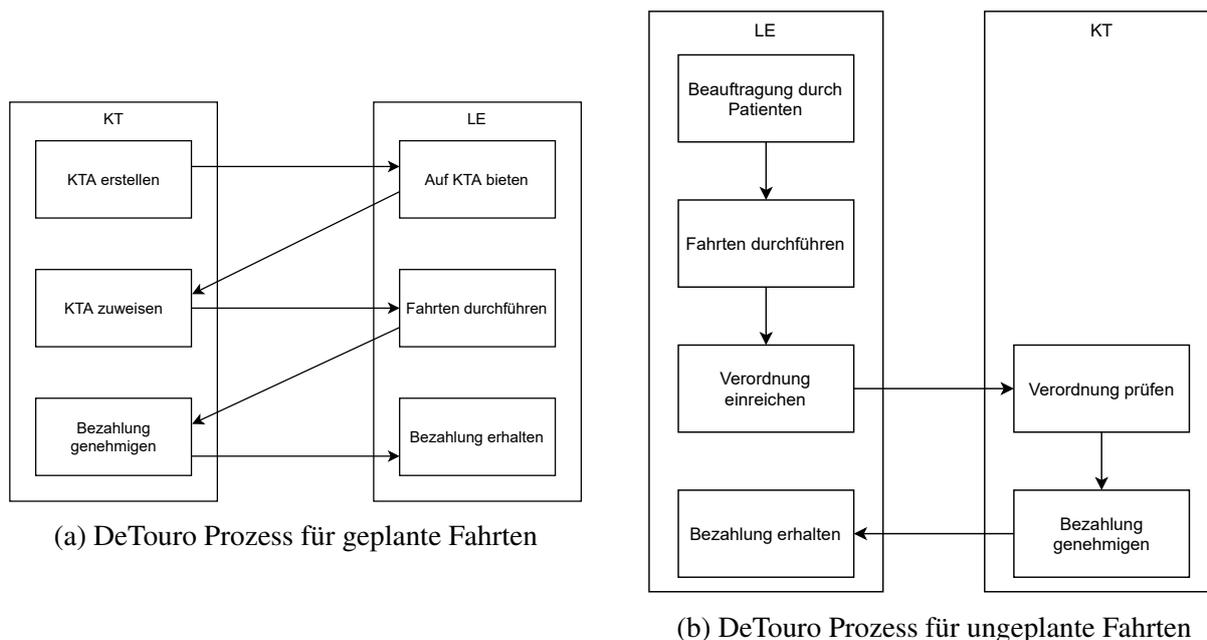


Abbildung 1.2: DeTouro Prozesse zur Abwicklung von Krankenfahrten

Muster 4 Verordnungen erkannt werden können. Jedoch ist der Dienst so zu Implementieren, dass eine zukünftige Erweiterung, für die Erkennung anderer Verordnungen, problemlos möglich ist. Zu diesem Zweck soll auch eine ausführliche Dokumentation für die Benutzung und den technischen Aufbau verfügbar gemacht, sowie eine Reihe von automatisierten Tests erstellt werden, um zukünftige Weiterentwicklungen zu unterstützen und um Wartungsarbeiten zu vereinfachen. Des weiteren muss eine Umgebung auf dem Firmeneigenen GitLab-Server eingerichtet werden, in welcher Continuous Integration (CI)- und Continuous Deployment (CD)-Pipelines konfiguriert werden sollen. Die erwarteten Endprodukte sind auch in Tabelle 1.1 noch einmal zusammengefasst.

Ergebnis	Beschreibung
Dokumentenerkennungsdienst	Als Microservice ausgeführter Dienst zur Erkennung von Muster 4 Verordnungen
Test-Suite	Automatische Tests zur Überprüfung der Funktionalität des Dienstes
Dokumentation	Technische Dokumentation über die Nutzung und den Aufbau des Dienstes
Entwicklungsumgebung	GitLab Server und CI/CD Pipelines

Tabelle 1.1: Endprodukte

1.6 Scope

Der Scope umfasst die Erstellung eines Dokumentenerkennungsdienstes. Dieser Dienst soll als Microservice eingesetzt werden können und sämtliche Kommunikation über ein API abwickeln. Zu-

sätzlich zum Dienst, sollen auch für die Entwicklung benötigte Konfigurationen erstellt werden. Ebenso soll eine Dokumentation des Dienstes, welche sowohl die Nutzung der API, und damit des Dienstes, erläutert, als auch die, für die langfristige Unterstützung, Wartung und Weiterentwicklung benötigten, Informationen zum Technischen Aufbau des Dienstes enthält, verfasst werden. Ebenfalls sollen entsprechende automatisch durchführbare Tests erstellt werden, welche die Qualität des Dienstes, nicht nur während der ersten Entwicklung sondern auch für kommende Erweiterungen oder Wartungsarbeiten, sicherstellen sollen.

Der Dienst soll für die Erkennung der Muster 4 Verordnung optimiert werden, sollte in der Zukunft aber auch für andere Verordnungen und Dokumente erweiterbar sein.

1.6.1 Out of scope

Eine Optimierung des Dienstes für die Erkennung anderer Verordnungen oder Dokumente ist nicht Teil der Aufgabe und fällt außerhalb des Scopes. Auch die Integration des Dienstes in DeTouro oder ein anderes Produkt oder Dienst der HMM Deutschland GmbH oder anderen Organisationen ist nicht Teil des Scopes.

2. Projekt

Dieses Kapitel enthält Informationen über den Aufbau des Projektes, den Ansatz sowie die Planung. Auch die Mechanismen zur Sicherstellung der Qualität werden erläutert.

2.1 Stakeholder

Das Projekt hat mehrere Stakeholder die an dem Projekt und seinem Ausgang Interesse haben. Diese können der Tabelle 2.1 entnommen werden.

Stakeholder	Beschreibung
Betriebs-Betreuer	Bewertet das Produkt
Prüfungskomitee	Bewertet die Ergebnisse und Arbeitsweise
Product Owner	Produkt Verantwortlicher
HMM Deutschland GmbH	Arbeitgeber und Besitzer des Produktes
Fahrdienstleister	Nutzer des Dienstes

Tabelle 2.1: Stakeholder

Eine Einstufung der einzelnen Stakeholder, nach deren Interesse und Befugnissen, kann dem Power-Interest-Diagramm in Abbildung 2.1 entnommen werden.

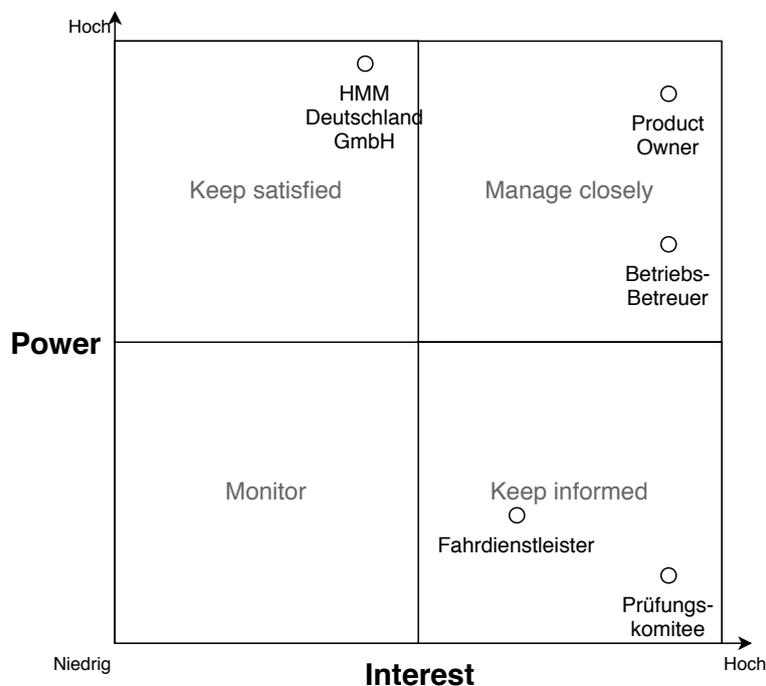


Abbildung 2.1: Power-Interest-Diagramm

2.2 Ansatz

Um den Entwicklungsprozess zu managen wird von dem Projektmanagement Framework Kanban Gebrauch gemacht. Kanban implementiert das *Agile Manifest* und fokussiert sich auf kontinuierliche Entwicklung Individueller Aufgaben (Radigan, 2020). Es wurde ausgewählt, da bereits die Entwicklung vieler Produkte der HMM Deutschland GmbH, unter anderem auch DeTouro, mit ihm gemanagt werden.

Um Kanban effektiv einsetzen zu können ist ein Board, zur Nachverfolgung der einzelnen Aufgaben, zwingend notwendig. In diesem Projekt erfüllt diese Aufgabe Jira, welches von der HMM Deutschland GmbH für das Projektmanagement eingesetzt wird. In Jira können nicht nur Vorgänge erstellt und nachverfolgt, sondern auch Boards erstellt werden die den Status der einzelnen Aufgaben widerspiegeln (Abbildung 2.2). Zusätzlich finden tägliche Stand-Up Meetings statt, in denen der aktuelle Fortschritt kommuniziert wird.

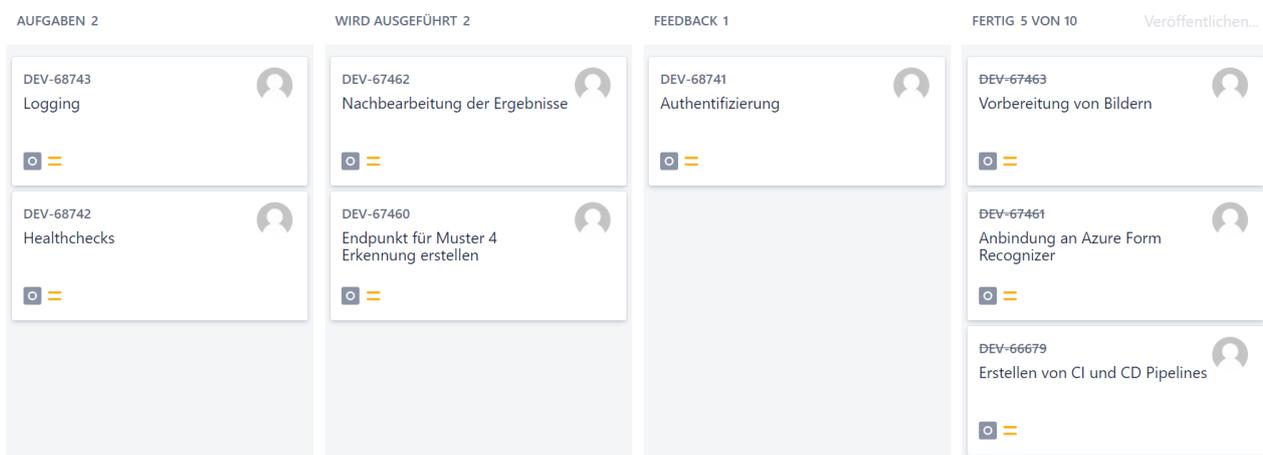


Abbildung 2.2: Ausschnitt aus dem Jira Board

Während der Implementationsphase wird Git für die Quellcodeverwaltung genutzt. Da das Git-Repository auf einem GitLab Server erstellt wird, werden auch die GitLab eigenen Features zur Erstellung von CI- und CD-Pipelines genutzt. Diese erlauben auf gepushte Commits zu reagieren und Aktionen durchzuführen. Im Falle von CI werden ein automatischer Buildvorgang gestartet und etwaige Tests ausgeführt. Eine CD-Pipeline ermöglicht es die neue Version des Programms in einer entsprechenden Umgebung auszuführen. Für den Zeitraum der Entwicklung wird der Dienst lediglich in einer Entwicklungsumgebung ausgeführt werden, es besteht jedoch auch die Möglichkeit Test- und Produktivumgebungen, welche für Abnahmetests und den Einsatz für den Endkunden vorgesehen sind, als Ziele zu definieren.

Zusätzlich wird ein Git Workflow genutzt, welcher das Repository in die Branches “master” und “develop” aufteilt, sowie Gruppen von “feature”, “release” und “hotfix” Branches definiert. Der “master” Branch ist der aktuell in der Produktionsumgebung eingesetzte Stand des Repository, während die Entwicklung im “develop” Branch stattfindet. Einzelne Features werden in “feature” Branches entwickelt und nach “develop” gemerged. Wenn Änderungen von “develop” nach “master” übernommen werden sollen, wird ein neuer “release” Branch erstellt in dem die Änderungen unabhängig getestet werden können. Nach der Abnahme der Änderungen, kann dieser Branch nach “master” gemerged werden. Falls im laufenden System Probleme entdeckt werden, deren Lösung nicht bis zum

nächsten Release warten kann, kann ein “hotfix” Branch von “master” erstellt werden, in dem das Problem behoben wird und welcher, ohne einen “release” Branch, nach “master” zurück gemerged werden kann. Dieser Workflow wurde von Driessen (2010) entwickelt und hat sich als Standard, bei vielen Software Projekten, etabliert.

2.3 Planung

Die Aufgaben lassen sich in die vier Phasen Analyse, Empfehlung, Design und Implementation aufspalten. Eine Übersicht der Aufgaben und Teilaufgaben sowie der entsprechenden Deadlines ist Tabelle 2.2 zu entnehmen.

Die Analyse findet zuerst statt und soll bis zum 18.09.2020 abgeschlossen sein. In diesem Zeitrahmen müssen die Anforderungen an das System identifiziert werden. Anschließend folgt die Empfehlung, welche auf Basis einer ausführlichen Untersuchung, der am Markt verfügbaren Dokumentenerkennungsdiensten, gegeben wird. Dafür werden Bewertungskriterien definiert sowie Testdaten vorbereitet und eine Vorauswahl verfügbarer Angebote erstellt. Diese vorbereitenden Aufgaben sollen bis zum 02.10.2020 durchgeführt werden, damit nachfolgend mit der Untersuchung fortgefahren werden kann. Diese Untersuchung, auf Basis der erstellten Kriterien, der vorausgewählten Angebote soll bis zum 30.10.2020, mit dem formulieren einer Empfehlung zur Nutzung eines Dienstes, abgeschlossen sein.

In der Design-Phase werden entscheidende Fragen der Implementation geklärt, sowie eine Reihe von Artefakten erstellt, anhand welcher das System in der nachfolgenden Implementationsphase implementiert werden kann und sie soll bis zum 13.11.2020 abgeschlossen sein. Die anschließende Implementations-Phase beginnt mit der Einrichtung eines Git-Repository auf dem firmeneigenen GitLab-Server, sowie dem einrichten von CI- und CD-Pipelines. Die vorbereitenden Maßnahmen sollen bis zum 20.11.2020 abgeschlossen sein, damit mit der Implementation des eigentlichen Dienstes begonnen werden kann. Diese umfasst nicht nur das erstellen und einrichten des Dienstes und der zugehörigen API, sondern auch das fortlaufende erstellen und updaten von Tests und verfassen von Dokumentationen. Diese arbeiten, und damit die Implementation, sollen bis zum 18.12.2020 beendet werden.

2.4 Qualitätskontrolle

Um eine Hohe Qualität und technische Robustheit des Dienstes zu Gewährleisten, werden verschiedene Maßnahmen ergriffen. Die erste dieser Maßnahme ist der, in Kapitel 2.2 bereits beschriebene, Gitflow sowie die CI- und CD-Pipelines. Diese sorgen für eine saubere Trennung von neuen Änderungen und bereits funktionierenden Komponenten und stellen die Fehlerfreiheit durch die automatischen Builds sicher.

Des weiteren wird eine Testsuite erstellt, um die implementierten Komponenten auf verschiedenen Ebenen zu testen und ihre korrekte Funktionsweise, auch in Ausnahmefällen, zu gewährleisten. Die Erstellung von automatisierten Tests ermöglicht es, unter anderem, die Ausführung in die CI-Pipelines zu integrieren und eine regelmäßige Ausführung der Tests, nach jedem neuen Commit, zu gewährleisten.

Nr.	Aufgabe	Teilaufgaben	Deadlines
1	<i>Analyse</i>	1. IST-SOLL Analyse	18.09.2020
		2. Abnahmekriterien erstellen	18.09.2020
		3. Risikoanalyse durchführen	18.09.2020
2	<i>Empfehlung</i>	1. Bewertungskriterien identifizieren	25.09.2020
		2. Passende Frameworks auswählen	25.09.2020
		3. Testdaten erstellen	02.10.2020
		4. Untersuchung der gewählten Frameworks anhand der identifizierten Kriterien	30.10.2020
3	<i>Design</i>	1. Prozessablauf definieren	13.11.2020
		2. Aufbau festlegen	13.11.2020
		3. Authentifizierungsschema wählen	13.11.2020
4	<i>Implementation</i>	1. Git-Repository einrichten	20.11.2020
		2. CI-/CD-Pipelines konfigurieren	20.11.2020
		3. Dienst implementieren	18.12.2020

Tabelle 2.2: Aufgabenplanung

Neben den Automatisierten Tests wird der geschriebene Programmcode auch durch andere Entwickler überprüft. Dieser Prozess unterstützt bei der Identifizierung von Fehlern oder Problemen, die durch die Tests möglicherweise nicht aufgedeckt wurden. Zudem wird dadurch die Leserlichkeit des Programmcodes ebenfalls überprüft und kann gegebenenfalls frühzeitig verbessert werden.

3. Analyse

Dieses Kapitel widmet sich der Analyse des Projektes. Es werden die Ist- und Soll-Situationen identifiziert, Abnahmekriterien erstellt und eine Risiko Analyse durchgeführt.

3.1 Ist-Situation

Wie in Kapitel 1.4 beschrieben, bietet De-Touro nicht nur die Möglichkeit geplante Krankenfahrten zu vermitteln, sondern soll in Zukunft auch Funktionen für die Verwaltung und Abrechnung ungeplanter Fahrten zur Verfügung stellen. Eine ungeplante Fahrt kann stattfinden, wenn die Versicherte Person ihre Verordnung nicht durch den zuständigen Kostenträger, meistens die Krankenkasse, genehmigen lassen muss. In diesem Fall ist der Patient freigestellt nach eigenem Ermessen einen Fahrdienstleister mit der Durchführung der verordneten Leistungen zu beauftragen. Der beauftragte Fahrdienstleister erhält, für die Durchführung der Fahrten, die Verordnung vom Versicherten, mit welcher die Bezahlung beim zuständigen Kostenträger beantragt werden kann. Zu diesem Zweck muss der Fahrdienstleister die Verordnung an den jeweiligen Kostenträger übertragen.

Um eine effiziente, automatische und schnelle Abwicklung der Bezahlung zu gewährleisten, ist es nötig dass die Daten der Verordnung digitalisiert werden. Dazu muss der Fahrdienstleister aktuell die Verordnung einscannen und anschließend die Daten aus dem Dokument in eine Eingabemaske übertragen. Es werden sowohl die übertragenden Daten als auch ein Bild der Verordnung benötigt, da sowohl die elementaren Daten für die Verarbeitung als auch eine Kopie des Originals für die Bestätigung benötigt werden.

Die Übertragung der Daten aus der Verordnung muss aktuell händisch durch die Fahrdienstleister durchgeführt werden. Diese Arbeit ist jedoch Zeitaufwendig und Fehleranfällig. Fehler in der Eingabe der Daten können die Abwicklung des gesamten Prozesses gefährden und benötigen häufig größere Maße an manuellem Aufwand zur Korrektur.

3.2 Soll-Situation

Mit der neuen Erweiterung steht DeTouro die Möglichkeit zur Verfügung, dem Fahrdienstleister die aufwendige und langwierige Kommunikation mit den Kostenträgern abzunehmen und so den gesamten Prozess, für die Abrechnung einer durchgeführten Krankenfahrt, ab dem Hochladen der Verordnung, vollautomatisch durchführen zu lassen. Dazu muss der Fahrdienstleister lediglich die Verordnung in DeTouro hochladen, alle weiteren Schritte werden selbstständig und automatisch durchgeführt. So wird der zuständige Kostenträger aus den angegebenen Daten ausgelesen und die Verordnung an diesen weitergeleitet. Zusätzlich wird ihm die durchgeführte Leistung zu Gunsten des Fahrdienstleisters in Rechnung gestellt. Nachdem der Kostenträger die Angaben überprüft hat, wird dem Fahrdienstleister das Geld ausbezahlt und der Prozess ist abgeschlossen.

Um dieses Vision für die neue DeTouro Erweiterung zu realisieren, muss ein System entwickelt und eingesetzt werden, mit dem die Inhalte der Verordnung automatisch ausgelesen werden können. Mit diesem System wird die mühselige Arbeit, die Dokumenteninhalte abzutippen, überflüssig und der

Fahrdienstleister so entlastet. Idealerweise kann der Dienst auch Fotos von Verordnungen analysieren. Damit hätten die Fahrdienstleister die Möglichkeit vorort die Verordnung zu fotografieren und sofort hochzuladen.

Es kann nicht ausgeschlossen werden, dass alle Verordnungen korrekt und vollständig ausgelesen werden. Deshalb ist ein zusätzlicher Überprüfungsschritt notwendig. Um hier so wenig Belastung wie möglich für die Fahrdienstleister, durch diesen Schritt, zu erzeugen, sollen ihnen zusätzliche Systeme bei der Kontrolle unterstützen. Diese sollen darauf hinweisen, ob ein Wert unsicher oder sogar falsch ist. Dies kann durch den Abgleich mit erwarteten Inhalten sowie dem Confidence Wert erreicht werden.

3.3 Integration

Der Dienst ist als Erweiterung für De-Touro konzipiert, soll aber auch, wie bereits beschrieben, auf andere Szenarien anwendbar sein. Aufgrund dieser Anforderung wird der Dienst als Microservice realisiert und über eine REST-Schnittstelle angesprochen. Dieser flexible Implementationsansatz, auf Plattform-übergreifenden Standards basierend, ist gegenüber Problemen der Integration gut abgesichert, solange die API korrekt dokumentiert ist und Nutzer diese Dokumentation für die Implementation eines Clients nutzen. Dennoch muss sichergestellt werden, dass der Dienst korrekt zur Verfügung gestellt wird und Zugriff auf benötigte Ressourcen hat. Auch die Nutzer müssen Zugriff auf den Dienst haben.

Auch wenn DeTouro noch keine Microservice Architektur hat, so gibt es dennoch ein Netzwerk verschiedener, miteinander interagierender, Komponenten, in welches der Dienst eingebunden werden muss. Für die langfristige Nutzung, auch in Hinblick auf das Ziel, den Dienst für die Erkennung von anderen Verordnungen außerhalb von DeTouro einzusetzen, ist es jedoch ratsam, den Dienst in die produktübergreifende Microservice-Architektur, die firmenintern bereits für andere Produkte genutzt wird, zu integrieren. Dies erlaubt allen Anwendungen, die Teil dieser Architektur sind, auf den Dienst zuzugreifen. Damit DeTouro auch weiterhin Zugriff auf den Dienst hat, wäre es jedoch auch nötig, DeTouro ebenfalls in diese Architektur zu integrieren.

3.4 Abnahmekriterien

Damit das Projekt als erfolgreich gewertet werden kann, muss der Dienst eine Reihe von Anforderungen erfüllen. Die definierten Anforderungen können Anhang A entnommen werden.

Die nicht-funktionalen Anforderungen beziehen sich auf eine Reihe von Kriterien, die nicht Teil der klassischen Implementation sind. So muss der Dienst Docker kompatibel sein und muss einige Datenschutzauflagen erfüllen. Da die Muster 4 Verordnungen sensible Patientendaten enthalten, dürfen diese, ohne vorherige Zustimmung aller Beteiligten, nicht an Dritte übertragen werden. Deshalb sollen die Abbildungen der Verordnungen weder langfristig gespeichert, noch an Dritte übertragen werden.

Die funktionalen Anforderungen, auf der anderen Seite, definieren Kriterien, die die Implementation zu erfüllen hat. Zwei große Teilbereiche sind die Effektivität bei der Erkennung und Segmentierung

der einzelnen Wertfelder der Muster 4 Verordnung, und die Erkennung der Inhalte dieser. Wie bereits beschrieben, werden ca. 60% der Verordnungen maschinell ausgefüllt, die verbleibenden 40% jedoch, werden noch handschriftlich erstellt. Aus diesem Grund ist es nötig, dass der finale Dienst sowohl maschinell als auch handschriftlich verfasste Texte verarbeiten kann. Zudem enthält die Muster 4 Verordnung, wie in Abbildung 1.1 erkennbar, eine Vielzahl von Kontrollkästchen, deshalb wird auch die Erkennung dieser bewertet. Insgesamt umfasst die Muster 4 Verordnung 41 Felder. In 17 davon werden Texte oder Zahlen eingetragen, 23 bestehen aus Kontrollkästchen und das letzte verbleibende Feld ist für den Stempel und die Unterschrift des verschreibenden Arztes oder der Praxis vorgesehen.

Zusätzlich gibt es noch weitere Anforderungen an die benötigte Zeit, sowie die Fähigkeit Muster 4 Verordnungen zu erkennen. Eine zusätzliche, optionale, Anforderung ist die Möglichkeit zur Erkennung von Verordnungen auf Fotos.

Auf Basis dieser Anforderungen wurden Kriterien definiert, welche für die Bewertung des Dokumentenerkennungsdienstes eingesetzt werden. Abhängig von der bewerteten Kategorie, wird eins von zwei verschiedenen Systemen für die Quantifizierung der Erkennungsergebnisse genutzt. Bei dem ersten System handelt es sich um eine Verbindung der beiden Metriken Präzision (Precision) und Trefferquote (Recall) (Abbildung 3.1). Die Präzision wird mit den *richtig positiven* Erkennungen über der Summe der *richtig* und *falsch positiven* Erkennungen berechnet. So gibt dieser Wert die Wahrscheinlichkeit an, mit der eine Erkennung korrekt ist. Die Trefferquote, wiederum, wird mit den *richtig positiven* Erkennungen über der Summe der *richtig positiven* und *falsch negativen* Erkennungen berechnet und gibt somit die Wahrscheinlichkeit an, dass eine vorhandene Klasse erkannt wird.

$$\begin{aligned} \text{Präzision} &= \frac{RP}{RP + FP} & RP &= \text{richtig positiv} \\ \text{Trefferquote} &= \frac{RP}{RP + FN} & FP &= \text{falsch positiv} \\ & & RN &= \text{richtig negativ} \\ & & FN &= \text{falsch negativ} \end{aligned}$$

Abbildung 3.1: Definition der Metriken (Cook und Ramadas, 2020)

Grundsätzlich neigt ein System mit einer hohen Präzision, also einer großen Menge an *richtig positiven* Erkennungen, dazu eine niedrige Trefferquote, und damit viele *falsch negative* Erkennungen, zu haben. Auf der anderen Seite neigen Systeme mit einer hohen Trefferquote, also wenigen *falsch negativen* Erkennungen, eine niedrige Präzision, und damit viele *falsch positive* Erkennungen, zu haben. Ein ideales System maximiert beide Werte, jedoch wird im gegebenen Anwendungsfall der Präzision Vorrang eingeräumt, damit die Zahl der fehlerhaften Erkennungen im Ergebnis minimiert wird und damit auch die Chance, dass die Fehler nicht durch die menschliche Überprüfung aufgedeckt werden und so in den Prozess Einzug finden.

Diese beiden Werte lassen sich nur bestimmen, wenn die Erkennungsergebnisse sich mit den vier Klassen *richtig positiv*, *richtig negativ*, *falsch positiv* und *falsch negativ* beschreiben lassen. Wenn diese Klassen nicht zugewiesen werden können ist eine Berechnung der Werte nicht möglich, deshalb wird für diese Fälle ein alternatives Bewertungssystem genutzt.

Dieses System ist ein einfacher Score, welches angibt wie viele der möglichen Erkennungen korrekt

sind. Im gegebenen Anwendungsfall wird diese Metrik für die Text- und Zeichenerkennung eingesetzt, da eine Klassifizierung nach dem obigen Schema mit hohem Aufwand verbunden wäre und zudem, im vorliegenden Fall, über wenig zusätzliche Aussagekraft verfügt.

Insgesamt sind sieben Kriterien definiert (Tabelle 3.1). Diese beziehen sich auf die Präzision und Trefferquote bei der Feld Segmentierung, der Score beim Auslesen von maschinell ausgefüllten Textfeldern, der Score beim Auslesen handschriftlich ausgefüllter Felder und der Präzision und Trefferquote bei der Erkennung von Kontrollkästchen und ihrem Status. Das siebte Kriterium beschreibt die maximale Verarbeitungsdauer. Diese wird vom Empfangen einer Verordnung am Dienst bis zum Senden des Ergebnisses gemessen, etwaige Zeitverluste durch die Übertragungsgeschwindigkeit sind nicht beeinflussbar und werden deshalb auch nicht berücksichtigt. Die Werte für diese Kriterien basieren auf den oben genannten funktionalen Anforderungen.

Kriterium	Mindestwert
Feld Segmentierung (Präzision)	0,95
Feld Segmentierung (Trefferquote)	0,9
Texterkennung (Score)	0,94
Handschrifterkennung (Score)	0,5
Erkennung von Kontrollkästchen (Präzision)	0,95
Erkennung von Kontrollkästchen (Trefferquote)	0,9
Verarbeitungsgeschwindigkeit	max. 60 Sekunden

Tabelle 3.1: Abnahmekriterien für den Dokumentenerkennungsdienst

3.5 Risiko Analyse

Um den Erfolg des Projektes zu gewährleisten müssen eventuelle Risiken identifiziert und klassifiziert werden, damit mit diesen adäquat umgegangen werden kann. Nachfolgend werden die einzelnen Risiken im Detail analysiert.

Das Zentrale Risiko, dem sich das Projekt ausgesetzt sieht, ist ein eventueller Defizit im Bereich der Dokumentenerkennung bei den verfügbaren Frameworks und Diensten. Dieser Defizit kann sich in vielen Bereichen ausdrücken, zu nennen wären der Preis oder die Erfüllung von Datenschutzkriterien. Der wohl gefährdetste Bereich ist jedoch die Effektivität der Dokumentenerkennung, d.h. die Unzulänglichkeit der Frameworks bei der Erkennung von Muster 4 Verordnungen. Da im Bereich der Dokumentenerkennung jedoch bereits seit vielen Jahren an funktionalen und zuverlässigen Lösungen gearbeitet wird, ist ein Eintreten dieses Falles eher Unwahrscheinlich. Auch etwaige Datenschutzkonflikte sollte aufgrund der großen Marktvielfalt kein übermäßiges Risiko darstellen. Lediglich im Bereich des Preises besteht ein reelles Risiko, dass Lösungen, die den Datenschutz wahren und eine gute Effektivität liefern, den finanziellen Rahmen übersteigen.

Falls es sich herausstellen sollte, dass zum aktuellen Zeitpunkt, keine, frei auf dem Markt verfügbare, Lösung alle Anforderungen erfüllt, besteht primär die Möglichkeit ein eigenes und unabhängiges Dokumentenerkennungssystem zu entwickeln. Unter Zuhilfenahme von Open Source Technologien

könnte so eine Alternative entwickelt werden. In diesem Fall ist jedoch damit zu rechnen, dass, aufgrund des erheblichen Mehraufwands, die Zeitplanung nicht eingehalten werden kann.

Des weiteren besteht das Risiko der nicht Akzeptanz durch die Nutzer. Diese könnte aus langen Ladezeiten, unzuverlässigen Ergebnissen oder umständlichen Prozessen resultieren. Um dieses Risiko zu mitigieren ist es nötig die Belastung der Nutzer so gering wie Möglich zu halten. Dafür wird ein schnelles System mit einer hohen Präzision benötigt. Des weiteren wird ein intuitiver und unterstützender Workflow entwickelt werden müssen.

4. Empfehlung

Um einen Dienst, der dazu in der Lage ist, mit hoher Präzision, die Inhalte von Muster 4 Verordnungen zu erkennen, im vorgegebenen Zeitrahmen zu entwickeln und die anderen Anforderungen, des Projektes, zu erfüllen ist es notwendig eine solide Basis, in der Form eines Dokumentenerkennungs-Frameworks zu nutzen. Das identifizieren des, für dieses Projekt, am besten geeignetsten Frameworks, ist Aufgabe dieses Kapitels.

4.1 Metriken

Bevor mögliche Kandidaten untersucht werden können ist es notwendig eine Reihe von Metriken festzulegen, nach welchen die Frameworks bewertet werden können. Diese sind auf Basis der Abnahmekriterien und Anforderungen aus Kapitel 3.4 definiert. Die drei zentralen Bereiche sind die Effektivität, der Preis und die Einhaltung der Datenschutzrichtlinien. Des weiteren müssen auch die Nicht-Funktionalen Anforderungen durch das Framework erfüllt werden können.

4.1.1 Effektivität

Der Bereich der Effektivität beschreibt die Fähigkeit des Frameworks bei der Dokumentenerkennung. Dieser Bereich lässt sich in die Metriken Feld Segmentierung, Texterkennung, Handschrifterkennung sowie der Erkennung von besonderen Strukturen wie Tabellen oder Kontrollkästchen aufteilen. Da der zu analysierende Bereich der Muster 4 Verordnung keine Tabellen enthält, spielt die Erkennung dieser eine untergeordnete Rolle, sollte aber, in Hinblick auf zukünftige Erweiterbarkeit, trotzdem beachtet werden. Auch die Geschwindigkeit mit der die Erkennung durchgeführt wird, wird hier betrachtet.

Insgesamt wurden acht verschiedene Metriken zur Messung der Effektivität definiert (Tabelle 4.1). Die Metriken für die Feld Segmentierung und die Kontrollkästchen Erkennung wurden ohne Anpassung aus der Definition der Abnahmekriterien übernommen. Für die Metriken zur Erkennung von maschinellen Texten und Handschriften, gibt es jedoch einige Änderungen, da für jeden Bereich zwei Metriken definiert wurden. Die erste beschreibt den Score bei der Erkennung des Inhalts eines ganzen Feldes. Die andere gibt die Character Error Rate (CER) für die Erkennung an. Die CER beschreibt die Zahl der Fehler pro erkannten Zeichen (Alvermann, 2019), ein niedriger Wert ist also besser. In diesem Fall wird die CER über die gesamte Verordnung berechnet. Während die erste Metrik für die Abnahmekriterien und Endnutzer relevanter ist, da sie angibt wie viele Felder ohne Korrektur übernommen werden können, gibt die Zweite einen besseren Eindruck über die tatsächliche Leistungsfähigkeit der Texterkennung.

Die letzte Metrik dieses Bereiches ist die Geschwindigkeit. Hier wird die durchschnittliche Verarbeitungszeit einer Muster 4 Verordnung gemessen. Die nötige Verarbeitungsgeschwindigkeit ist durch die Anforderungen auf 60 Sekunden begrenzt.

Metrik	Beschreibung
Feld Segmentierung (Präzision)	Präzision bei der Segmentierung einzelner Felder
Feld Segmentierung (Trefferquote)	Trefferquote bei der Segmentierung einzelner Felder
Texterkennung	CER bei der Erkennung einzelner maschinell erstellter Zeichen
Texterkennung pro Feld	Score bei der Erkennung eines ganzen maschinell ausgefüllten Feldes
Handschrifterkennung	CER bei der Erkennung einzelner handschriftlich erstellter Zeichen
Handschrifterkennung pro Feld	Score bei der Erkennung eines ganzen handschriftlich ausgefüllten Feldes
Erkennung von Kontrollkästchen (Präzision)	Präzision bei der Erkennung des Status von Kontrollkästchen
Erkennung von Kontrollkästchen (Trefferquote)	Trefferquote bei der Erkennung des Status von Kontrollkästchen
Geschwindigkeit	Durchschnittliche Dauer der Verarbeitung

Tabelle 4.1: Metriken für die Bewertung der Effektivität der Dokumentenerkennungs-Frameworks

4.1.2 Preis

Das Zentrale Entscheidungskriterium ist zwar die Effektivität, jedoch darf auch der Preis, für die Erkennung, nicht außer acht gelassen werden. Grundsätzlich kann dabei in zwei verschiedene Abrechnungsarten unterschieden werden. Auf der einen Seite sogenannte “Pay-as-you-go” Abonnements, bei denen der Preis von der Nutzung abhängig ist. Auf der anderen Seite Vertragsmodelle bei denen im Voraus Nutzungsrechte erworben werden. In diesem Fall hängt der Preis nicht, oder nur über Preisklassen, von der Nutzungsmenge ab.

Für den Preisvergleich wird angenommen das insgesamt 1000 Verordnungen jeden Monat verarbeitet werden müssen. Die Preise werden soweit Möglich in € pro Verordnung angegeben. Für Preise die nur in US-\$ angegeben werden, wird ein Umrechnungsfaktor von 0,85, auf Basis der Wechselkurse im Herbst 2020, angenommen.

4.1.3 Datenschutz

Wie in den Anforderungen beschrieben, ist die Erfüllung der extrem strengen Datenschutzauflagen unabdingbar. Aus diesem Grund ist die Datenschutz-Metrik als Ausschlusskriterium definiert. Sobald ein Framework die Anforderungen nicht erfüllen kann, kann es nicht eingesetzt werden.

4.1.4 Technische Anforderungen

Zusätzlich muss das Framework dazu in der Lage sein die Technischen Anforderungen, unter anderem definiert durch die nicht-Funktionalen Anforderungen, zu erfüllen. Zum einen muss das Framework in einer Server Umgebung über Docker einsetzbar sein. Ebenfalls muss die Möglichkeit bestehen, über eine API, durch andere Programme und Dienste auf die Fähigkeiten zuzugreifen. Abschließend sollte die Anwendung entweder in .NET, Java oder Python implementiert werden können, dafür wird ggf. ein entsprechendes SDK benötigt.

4.2 Testdaten

Um die Aussagekraft und Fairness der Untersuchung zu sichern, werden eine Reihe von Testdaten erstellt, welche für die Evaluierung aller Kandidaten genutzt werden.

Diese Testdaten bestehen aus einer Reihe von, sowohl handschriftlich als auch maschinell ausgefüllten, Muster 4 Verordnungen. Es ist zu beachten das es sich bei den eingetragenen Informationen um Dummy-Daten handelt, es werden also keine echten Patientendaten genutzt und damit auch nicht gefährdet. Die genutzten Daten sind dennoch von Realen Daten, in Bezug auf die Struktur, praktisch nicht zu unterscheiden, weshalb sich die Ergebnisse, bei der Erkennung von diesen Testdaten, auch auf echte Verordnungen anwenden lassen.

Insgesamt wurden jeweils 20 handschriftlich und maschinell ausgefüllte Verordnungen vorbereitet. Die handschriftlichen Verordnungen wurden erstellt, indem Vordrucke ausgedruckt und anschließend handschriftlich ausgefüllt wurden. Für die Erstellung der maschinell ausgefüllten Verordnungen wurde Adobe Acrobat Reader (Adobe, 2020) genutzt um die Vordrucke auszufüllen und diese anschließend auszudrucken. Alle Verordnungen wurden daraufhin erneut eingescannt. Dieser Prozess, am realen Workflow orientiert, erzeugt Daten die eine hohe Ähnlichkeit mit den erwarteten realen Daten haben. Falls nötig besteht jederzeit die Möglichkeit weitere Testdaten auf diese Art und Weise zu erstellen.

4.3 Verfügbare Optionen

Bevor eine Detaillierte Untersuchung durchgeführt werden kann, muss eine Vorauswahl für mögliche Kandidaten getroffen werden. Nach einer anfänglichen Marktrecherche wurden drei KI-basierte Systeme für die Dokumentenerkennung identifiziert. Bei allen dreien handelt es sich um neue Angebote der drei größten Anbietern von Cloud Dienstleistungen, Amazon Web Services (AWS), Microsoft Azure und Google Cloud Platform. Die Dienste werden damit beworben eine große Bandbreite von strukturierten Dokumenten auslesen zu können.

Zusätzlich wurden auch mehrere Open Source Alternativen erkannt, welche, auch wenn sie nicht für die Dokumentenerkennung optimiert sind, für den Einsatzzweck nutzbar scheinen. Alle ausgewählten Optionen können Tabelle 4.2 entnommen werden.

Option
Amazon Textract
Microsoft Form Recognizer
Google Document AI
Open Source

Tabelle 4.2: Verfügbare Optionen

4.4 Untersuchung

Nachfolgend wird die Untersuchung der einzelnen Optionen beschrieben und einige Besonderheiten erläutert.

Für jedes Framework wurden, mithilfe von Demoversionen, die Testdaten analysiert und die Ergebnisse notiert. Anschließend wurden diese Daten ausgewertet um die definierten Metriken zu berechnen. Für die Bestimmung des Preises wurden die offiziellen Preisangaben auf den angenommenen Fall angewendet. Um die Erfüllung des Datenschutzes und der technischen Anforderungen zu prüfen wurden sowohl die Dokumentationen als auch Datenschutzvereinbarungen studiert.

Die Ergebnisse für die Metriken der Effektivität können Anhang B entnommen werden.

4.4.1 Amazon Textract

Amazon Textract, als Dokumentenerkennungsdienst des Marktführenden Cloud Anbieters AWS (Dignan, 2020), wird mit großer Flexibilität und der Fähigkeit zum Auslesen von Formularen beworben.

Die Untersuchung wurde mithilfe einer kostenlosen Demo Version des Dienstes durchgeführt. Da der Dienst, auch in der Vollversion, nicht weiter angepasst oder optimiert werden kann, gibt diese Version einen guten Einblick in die Leistungsfähigkeit von Amazon Textract.

Die Feld Segmentierung hat sich als unzuverlässig herausgestellt. Zudem wird die Verarbeitung der Erkennungsergebnisse durch die Verwendung von erkannten Feldtiteln erschwert, da diese nicht immer korrekt und gleich erkannt werden. Zudem beschreiben einige Feldtitel zwei Wertfelder, wie z.B. "anderer Grund", in der Verordnung unter "1c" zu finden. Hier muss die Option mit einem Kontrollkästchen markiert und zusätzlich Text in ein Textfeld eingetragen werden. Für beide Wertfelder existiert jedoch nur ein Titel. Diese Komplexe Feldstruktur kann Amazon Textract nicht verarbeiten.

Bei der Untersuchung der Inhaltserkennung ist aufgefallen, dass die Erkennung von Handschriften stark fehleranfällig ist (Tabelle B.3). Aufgrund der Tatsache das Textract, laut eigenen Angaben, lediglich Englisch unterstützt, ist anzunehmen, dass ein besseres Ergebnis möglich wäre, wenn Dokumente in Englischer Sprache analysiert werden würden. Diese Annahme wird dadurch unterstützt das die Erkennung von Zahlen und Ziffern wesentlich bessere Ergebnisse liefert als die von Texten. Die Erkennung von maschinellen Texten wiederum lieferte bessere Ergebnisse. Hier wurden fast alle Textfelder korrekt erkannt. Die Kontrollkästchen wurden alle fehlerfrei ausgelesen.

Bei der Untersuchung hinsichtlich des Datenschutzes ist aufgefallen, dass der Dienst ausschließlich

in der Cloud auf AWS Servern zur Verfügung gestellt wird. Obwohl die Einhaltung von DSGVO und anderen Datenschutzstandards versichert wird, ist ein Einsatz in dieser Form nicht mit den strengen Datenschutzvereinbarungen denen DeTouro unterliegt kompatibel.

Für Amazon Textract werden eine Reihe von verschiedenen Client Bibliotheken, mit Unterstützung für unter anderem .NET, Java und Python, angeboten. Damit besteht die Möglichkeit ein Programm mit einer der geforderten Technologien zu entwickeln und so die technischen Anforderungen zu erfüllen.

4.4.2 Microsoft Form Recognizer

Als Teil der Cloud Platform Azure bietet Microsoft den Dienst Form Recognizer an. Dieser Dienst wird, wie Amazon Textract, damit beworben in der Lage zu sein Formulare auszulesen. Im Gegensatz zu Textract wird jedoch ein Dienst angeboten, der zuerst für die gewünschten Dokumente trainiert werden muss. Dieses Training kann entweder "Supervised" oder "Unsupervised" stattfinden, das heißt es können sowohl Gelabelte als auch nicht Gelabelte Trainingsdaten genutzt werden. Unabhängig vom Training, wird durch dieses benutzerdefinierte Model garantiert, dass die identifizierten Felder im Erkennungsergebnis immer unter dem gleichen Label zu finden sind. Für das trainieren wird ein eigenes Tool zur Verfügung gestellt, mit dem die Trainingsdaten auch gelabelt werden können. Für die Erprobung wurde ein Modell durch "Supervised learning" trainiert. Dadurch konnten auch komplexe Feldstrukturen abgebildet, sowie eigene Label für die einzelnen Felder vergeben werden.

Die Feldsegmentierung lieferte sehr viel versprechende Ergebnisse. Dank des Benutzerdefinierten Models, wurden auch komplexe Felder richtig erkannt. Wie Tabelle B.4 zu entnehmen, lieferte die Handschrifterkennung akzeptable Ergebnisse, die jedoch immer noch von Fehlern durchzogen sind. Die Erkennung von maschinellen Texten und Kontrollkästchen, im Gegensatz dazu, war praktisch fehlerfrei.

Auch wenn Form Recognizer, wie auch Amazon Textract, eigentlich als Cloud Dienst zur Verfügung gestellt wird, so wird dennoch ein alternativ Betrieb auf eigener Hardware ermöglicht. Damit besteht die Möglichkeit diesen Dienst Datenschutzkonform einzusetzen.

Wie auch bereits für Amazon Textract, stehen auch für Form Recognizer eine Reihe von verschiedenen Client Bibliotheken, mit Unterstützung für .NET, Java, Python und andere Sprachen, zur Verfügung.

4.4.3 Google Document AI

Google bietet, als dritter großer Cloud Anbieter, ebenfalls einen Dokumentenerkennungsdienst an. Der Dienst, vermarktet unter dem Namen Document AI, wird, wie auch schon seine beiden Konkurrenten, mit der Fähigkeit beworben komplexe Dokumente und Formulare analysieren zu können. Aktuell befindet sich der Dienst noch im Beta Stadium, kann jedoch bereits genutzt werden.

Die Untersuchung wurde, ähnlich wie bereits bei Amazon Textract, mit einer Online zur Verfügung gestellten Demo durchgeführt. Die Feldsegmentierung ist nicht zufriedenstellend durchgeführt worden. Viele Felder wurden nicht erkannt, andere wurden nicht korrekt voneinander getrennt. Berück-

sichtigt man dies bei der Analyse der Handschrifterkennung, indem man grob fehlerhaft segmentierte Felder nicht für die Bewertung der Texterkennung einbezieht, fallen diese Ergebnisse sehr gut aus (Tabelle B.5). Maschinelle Texte und Kontrollkästchen wurden, wie oben, ohne größere Fehler, größtenteils korrekt erkannt.

Ähnlich wie bei Amazon Textract, wird auch Document AI nur als Cloud Dienst angeboten und kann deshalb nicht auf eigener Hardware betrieben werden. Aus diesem Grund ist auch hier ein Datenschutz konformer Einsatz nicht möglich.

Auch wenn für Document AI Client Bibliotheken für Java und Python zur Verfügung stehen, gibt es keine .NET Bibliothek. Damit ist eine Implementation in der bevorzugten Technologie, nicht oder nur schwierig möglich.

4.4.4 Open Source

In den vergangenen Jahren gab es viele Fortschritte im Bereich des Machine Learning und der Künstlichen Intelligenz. Viele dieser Fortschritte sind von der Akademischen Welt vorangetrieben worden und stehen deshalb unter Open Source Lizenzen zur Verfügung. Aus diesem Grund gibt es heute eine breite Vielfalt von Lösungen die für die Dokumentenerkennung, mit voller Kontrolle über alle Systembestandteile, eingesetzt werden könnten.

Grundsätzlich spaltet sich die Erkennung von strukturierten Dokumenten in zwei Bereiche auf. Zuerst werden die Wertfelder segmentiert und anschließend deren Inhalt ausgelesen. Dementsprechend müssen auch zwei unterschiedliche Systeme entwickelt und miteinander kombiniert werden. Eine Möglichkeit besteht in der Verwendung des Objekterkennungs-Framework YOLOv3 und dem Optical-Character-Recognition (OCR) Dienst Tesseract. Yun-An u. a. (2019) haben mit diesem Ansatz ein System für die Dokumentenerkennung entwickelt und gute Ergebnisse erzielen können. YOLOv3, wobei YOLO für "You only look once" steht, ist ein Objekterkennungsdienst, der für seine hohe Geschwindigkeit bekannt ist und mit besonderem Augenmerk auf die Echtzeiterkennung entwickelt wird. Da eine Echtzeiterkennung, wie beschrieben, nicht notwendig ist und Geschwindigkeit meistens auf Kosten der Präzision erreicht wird, wurden auch alternative Dienste überprüft.

Objekt Segmentierung

Im Bereich der Objekt Segmentierung gibt es grundsätzlich zwei Ansätze. Beim ersten handelt es sich um sogenannte "Region-Based detectors", bei diesen werden die Positionen von Objekten zuerst ermittelt und anschließend klassifiziert (Ren u. a., 2017). Im Gegensatz dazu werden bei den sogenannten "Single Shot detectors" beide Schritte gleichzeitig durchgeführt (Liu u. a., 2016). Im Vergleich sind "Single Shot Detectors" häufig schneller, während "Region-Based Detectors" eine höhere Präzision erreichen. Da im gegebenen Use-Case die Verarbeitungsgeschwindigkeit nur eine untergeordnete Rolle spielt, sind für das Projekt die "Region-Based Detectors" besonders interessant, die Auswahl wird jedoch dadurch jedoch nicht eingeschränkt.

Folgende Systeme wurden für eine detaillierte Evaluierung herangezogen:

- Faster R-CNN

- SSD
- RetinaNet
- YOLOv3

Vergleich Um zu vermeiden für jedes der ausgewählten Systeme eigene Tests durchführen zu müssen, werden die Ergebnisse anderer Arbeiten ausgewertet und so die Auswahl auf die vielversprechendste Option begrenzt. Die Leistungsfähigkeit der einzelnen Systeme wird anhand der Ergebnisse bei einem Einsatz auf dem “COCO” Datensatz bestimmt. Dieser Datensatz ist eine umfangreiche und anspruchsvolle Sammlung von Bildern, die für den Zweck der Entwicklung und des Vergleichens von Objekterkennungs-Systemen zusammengestellt wurde und hat sich in der jüngeren Vergangenheit zu einem de-facto Standard in der Objekterkennung entwickelt. Während ein Datensatz für die Objekterkennung nicht zwangsläufig repräsentative Ergebnisse für einen anderen Anwendungsfall, wie die Dokumentenerkennung, liefert, kann den Ergebnissen dennoch eine Tendenz, in Bezug auf die Leistungsfähigkeit in Relation zu den anderen Optionen, entnommen werden.

Die Ergebnisse für “YOLOv3” sind der Arbeit von Redmon und Farhadi (2018) entnommen, in welcher sie über die Verbesserungen von “YOLOv3” gegenüber vorangegangenen Versionen berichten. Bei diesem System handelt es sich um einen “Single Shot detector”. Die Ergebnisse für “RetinaNet”, ebenfalls ein “Single shot detector”, sind der Arbeit von Lin u. a. (2020) entnommen, welche sich mit der Entwicklung des Systems auseinandersetzt. Für die Ergebnisse von “SSD” wurde Liu u. a. (2016) herangezogen. In ihrer Arbeit beschrieben sie die Entwicklung und Ergebnisse dieses “Single shot detectors”. Für die letzte Option, “Faster R-CNN”, werden die Ergebnisse der Arbeit von Ren u. a. (2017) entnommen, welche die Optimierung bestehender “Region-based detectors” beschreibt.

Die Ergebnisse werden in Tabelle 4.3 aufgeführt. Es ist zu beachten, dass für die Ergebnisse die Definitionen von COCO (2020) genutzt werden. Die relevante Metrik ist “AP”, die anderen beiden sind ebenfalls in der Definition aufgeführt, jedoch handelt es sich bei der “AP” Metrik um die jüngste und strikteste, welche die “Average Precision”, also durchschnittliche Präzision, der Systeme misst. Unabhängig davon hat “RetinaNet” in allen Metriken die besten Ergebnisse erzielt. Da die Ergebnisse für “Faster R-CNN” vor der Einführung des “AP” Standards erarbeitet worden sind, existieren in der, als Quelle genutzten Arbeit, keine Ergebnisse für diese Metrik.

System	AP	AP ₅₀	AP ₇₅
YOLOv3	33,0	57,9	34,4
RetinaNet	40,8	61,1	44,1
SSD	28,8	48,5	30,3
Faster R-CNN	-	42,7	21,9

Tabelle 4.3: Ergebnisse verschiedener Objekterkennungssysteme bei Einsatz auf dem COCO Datensatz

Inwieweit die Ergebnisse, die bei der Erkennung von Objekten in Bildern erstellt worden sind, Aussagekraft über die Fähigkeiten bei der Erkennung von Wertefeldern in Dokumenten haben, ist weder eindeutig zu klären, noch Teil dieser Arbeit. Dennoch ist davon auszugehen, dass ein besseres

<input checked="" type="checkbox"/> Zuzahlungsbeitrag	Krankenkasse bzw. Kostenträger KK De-Touro		Verordnungsart
<input type="checkbox"/> Zuzahlungsfrei	Name, Vorname des Versicherten Möller, Barbara		
		geb. am 15.06.1984	<input type="checkbox"/> Un
	Kostenträgerkennung 987654321	Versicherten-Nr. Y117980540	<input type="checkbox"/> Art
		Status 1184C	<input checked="" type="checkbox"/> Ver
	Betriebsstätten-Nr. 985629906	Arzt-Nr. 302602970	<input checked="" type="checkbox"/> Hir
		Datum 17.09.2020	
1. Grund der Beförderung			
Genehmigungsfreie Fahrten			

Abbildung 4.1: Durch RetinaNet erkannte Label

Ergebnis in der klassischen Objekterkennung auch eine bessere Leistungsfähigkeit in der Dokumentenerkennung bedeutet. Aus diesem Grund wird für das, im nächsten Schritt folgende, Experiment "RetinaNet" genutzt.

Experiment Für das Experiment wird ein Prototyp, auf Basis von "RetinaNet" (Tsung-Yi Lin und He, 2020), für die Erkennung von Muster 4 Verordnungen entwickelt. Um den Aufwand für das Experiment zu reduzieren, und so größere Datenmengen testen zu können, wurde lediglich der Kopf der Verordnung ausgelesen. Dadurch wird die Zahl der benötigten Label von 40 auf 9 reduziert. Für das Labelling der Daten wurde auf das Tool "VoTT" (Microsoft, 2020) zurückgegriffen. So wurden insgesamt alle 40 Verordnungen des Testdatensets beschriftet. Anschließend wurden, mit der Online Plattform "Roboflow" (Roboflow, 2020), die Daten augmentiert und so von 40 auf 120 Datensätzen erweitert. Dabei wurden die Daten, mithilfe der zur Verfügung gestellten Tools, mutiert. Es wurde Körnung hinzugefügt, die Sättigung angepasst und die Bilder zufällig, um maximal 1° in beliebige Richtungen, geschert. Anschließend wurden 100 der Datensätze genutzt um ein Model zu trainieren, während die verbleibenden 20 Datensätze zum Testen des Modells genutzt wurden.

Die durch das Experiment generierten Ergebnisse sind enttäuschend. Zu keinem Zeitpunkt gelang es "RetinaNet" eine Confidence von über 20% zu erreichen. Wie zu erwarten sind keine Label korrekt platziert worden. Unerklärlicher Weise, konzentrieren sich die Labels mit der höchsten Confidence in der rechten unteren Ecke (Abbildung 4.1), auch wenn die korrekte Position des Feldes auf der anderen Seite des Datenblocks liegt. Auch nach einiger Anpassung des Modells, unter anderem durch Variationen bei der Vorbereitung durch Änderung der Skalierung, konnten die Ergebnisse nicht signifikant verbessert werden.

Texterkennung

Für die Erkennung maschineller Texte wurden zwei Systeme evaluiert. Auf der einen Seite Tesseract OCR (Google, 2019), auf der anderen GOCR (Schulenburg, 2018). Beide Programme sind weit verbreitete und bekannte Open Source Texterkennungsdienste, welche bereits seit einigen Jahren erfolgreich eingesetzt werden. Wie zu erwarten wurden die maschinellen Texte von beiden Diensten problemlos und praktisch fehlerfrei erkannt.

Da beide Dienste nicht für die Handschrifterkennung optimiert sind, wurde für diese Aufgabe “Simple HTR” (Harald Scheidl, 2019) genutzt. Die erzielten Ergebnisse sind praktisch unbrauchbar (Tabelle 4.4).

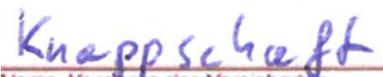
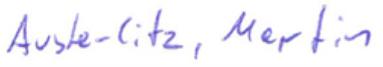
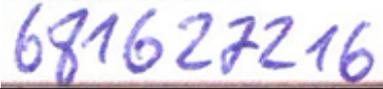
	Original	Erkannter Wert	Tatsächlicher Wert
1		#t	Knappschaft
2		ue-ctmertin	Austerlitz, Martin
7		b9GLE	681627216
9		5.B.19	5.8.19

Tabelle 4.4: Simple HTR Handschrifterkennungsergebnisse

4.5 Ergebnisse

Um die Untersuchung abzuschließen werden nun die Ergebnisse analysiert. Die detaillierten Ergebnisse können dem Anhang B entnommen werden.

Die von Amazon Textract und Google Document AI erzielten Ergebnisse bei der Feldsegmentierung sind unzureichend, da sie die geforderten Anforderungen nicht erfüllen können. Lediglich Microsoft Form Recognizer hat, in dieser Metrik, sehr robuste Ergebnisse liefern können. Was auffällt, ist das die Segmentierung von maschinell ausgefüllten Verordnungen, bei allen Kandidaten, etwas besser ausgefallen ist.

Bei der Handschrifterkennung hat Google Document AI die besten Ergebnisse erzielen können. Es ist zu beachten das für die CER-Metrik ein niedriger Wert besser ist. Lediglich 6% aller Zeichen wurden fehlerhaft erkannt. Jedoch konnten nur knapp die Hälfte aller Felder, in ihrer Gesamtheit, korrekt erkannt werden. Die Ergebnisse von Microsoft Form Recognizer sind etwas schlechter ausgefallen, es wurden jedoch immer noch knapp 90% aller Zeichen korrekt erkannt. Amazon Textract liefert erneut die schlechteste Performance ab und konnte lediglich 14% aller Felder korrekt auslesen.

Die Erkennung von maschinellen Texten wurde von allen praktisch Fehlerfrei durchgeführt. Obwohl die CER aller Dienste 0 war, konnten dennoch nicht alle Felder korrekt erkannt werden. Dies lässt

sich damit erklären, das vereinzelt Trennzeichen zwischen Feldern, fälschlicherweise, als Buchstaben oder Zahlen identifiziert wurden.

Die Erkennung von Kontrollkästchen lief praktisch ohne Fehler ab. Lediglich Google Document AI, konnte einige handschriftlich ausgefüllte Kontrollkästchen nicht korrekt identifizieren.

Beim betrachten der benötigten Zeit, ist festzustellen das alle Dienste ca. 10 Sekunden pro Verordnung benötigen und damit das Limit von 60 Sekunden weit unterschreiten. Der schnellste Dienst war jedoch Amazon Textract, welcher als einziger das 10 Sekunden Ziel unterschreiten konnte.

In Bezug auf den Preis ist Tabelle 4.5 zu entnehmen das Microsoft Form Recognizer für das angenommene Szenario, mit 42,17€ pro 1000 Verordnungen, die günstigste Option ist.

Angebot	Preis (€/1000 Seiten)
Amazon Textract	53,13€
Microsoft Form Recognizer	42,17€
Google Document AI	55,02€

Tabelle 4.5: Preise für die Nutzung der Dokumentenerkennungsdienste

Wie bereits festgestellt, lassen sich weder Amazon Textract noch Google Document AI auf eigener Hardware betreiben. Damit können beide NFR-04 (Tabelle A.2) nicht erfüllen, was wiederum einen Datenschutzkonformen Einsatz dieser Angebote ausschließt.

Die untersuchte Open Source Alternative konnte keine relevanten Ergebnisse erzeugen. Es ist jedoch davon auszugehen, dass ein wettbewerbsfähiges System entwickelt werden könnte, wenn mehr Ressourcen, allem voran mehr Zeit, zur Verfügung stehen würden. In der aktuellen Situation ist diese Option jedoch nicht umzusetzen.

Wenn man alle Ergebnisse zusammennimmt, ist Microsoft Form Recognizer der eindeutige Sieger dieses Vergleichs. Der Dienst hat in allen Metriken gute Ergebnisse erzielen können. Im Bereich der Geschwindigkeit ist er nur knapp hinter Amazon Textract. Zusätzlich ist er am günstigsten einzusetzen und kann sowohl Datenschutz- als auch technischen Anforderungen erfüllen. Lediglich die Handschrifterkennung kann die Anforderung nach einem Score von mindestens 50%, mit nur 27% nicht erfüllen. Mit einer CER von 0,12 werden jedoch ca. 88% aller Zeichen korrekt erkannt.

Aus diesem Grund wird für die Implementation eines Dokumentenerkennungsdienstes die Nutzung von Microsoft Form Recognizer als Basis empfohlen.

5. Design

Um eine erfolgreiche und Zielführende Implementation durchführen zu können, ist es notwendig vorher einige entscheidende Fragen zu beantworten sowie Teile des Systems zu entwerfen. Dieses Kapitel widmet sich diesen Arbeitsschritten und beschreibt im Detail den Aufbau des Dienstes sowie den Prozessablauf und die zu nutzende Authentifizierung.

5.1 Prozessablauf

Um eine strukturierte und planbare Implementation zu ermöglichen, muss zuerst der Ablauf der Erkennungsprozesses geplant werden. Zu diesem Zweck wurde ein Sequenz-Diagramm erstellt (Abbildung 5.1). Insgesamt gibt es vier Komponenten und Akteure. Der Client, ein Dienst oder eine Anwendung die, auf Anfrage eines Endnutzers den Dienst nutzt, sendet eine Anfrage mit einer beiliegenden Abbildung einer Verordnung. Daraufhin wird vom Dienst die Berechtigung der Aufrufenden Instanz, durch eine Authentifizierungs-Komponente (Kapitel 5.3), überprüft, bevor mit der eigentlichen Verarbeitung fortgefahren wird.

Der erste Schritt der Verarbeitung besteht im Pre-Processing. Hier werden einige vorbereitenden Schritte durchgeführt. So wird zum Beispiel der Dateityp des Bildes überprüft. Anschließend wird die Verordnung an den Form Recognizer Service gesendet, welcher die Daten aus der Verordnung, mithilfe des vorher trainierten Models, ausliest. Die Form Recognizer API führt die Analyse als sogenannten "long running job" durch, das bedeutet, dass das Ergebnis größere Vorbereitungszeit benötigt und nicht als Antwort auf die Anfrage gesendet wird. Stattdessen muss das Ergebnis durch einen Ressourcen-Identifikator abgefragt werden. Da die Analyse einer Muster 4 Verordnung jedoch nur wenige Sekunden in Anspruch nimmt und auch eine Einbindung in andere Dienste so einfacher ist, wurde entschieden die API des Dienstes konventionell auszuführen und das Ergebnis direkt als Antwort auf die Anfrage zurückzugeben. Um dies realisieren zu können, muss der Status der Analyse fortlaufend bei der Form Recognizer API abgefragt werden.

Nachdem die Analyse der Verordnung, durch den Form Recognizer, angeschlossen wurde, müssen die Ergebnisse noch aufbereitet werden. Diese Nachbearbeitung umfasst das Überführen der Erkennungsergebnisse in ein Standardisiertes und an eine Muster 4 Verordnung angelegtes Datenformat. Dieses kann dann an den Client zurückgegeben werden, welcher die Daten schnell und einfach aus dem standardisierten Format auslesen kann.

5.2 Aufbau

Da DeTouro mit Hilfe von ASP.NET implementiert wurde und auch Microsoft Azure über ein umfangreiches und aktuelles SDK für .NET verfügt, wurde entschieden den Dienst ebenfalls in ASP.NET zu implementieren. Im Sinne von Paradigmen der Objekt Orientierten Programmierung, wurde die Funktionalität des Dokumentenerkennungsdienstes auf verschiedene Komponenten aufgeteilt. Die gesamte Struktur ist Anhang C zu entnehmen. Die Zentrale Komponente ist der "Muster4Controller". Diese Klasse definiert den API Endpunkt für die Dokumentenerkennung und enthält

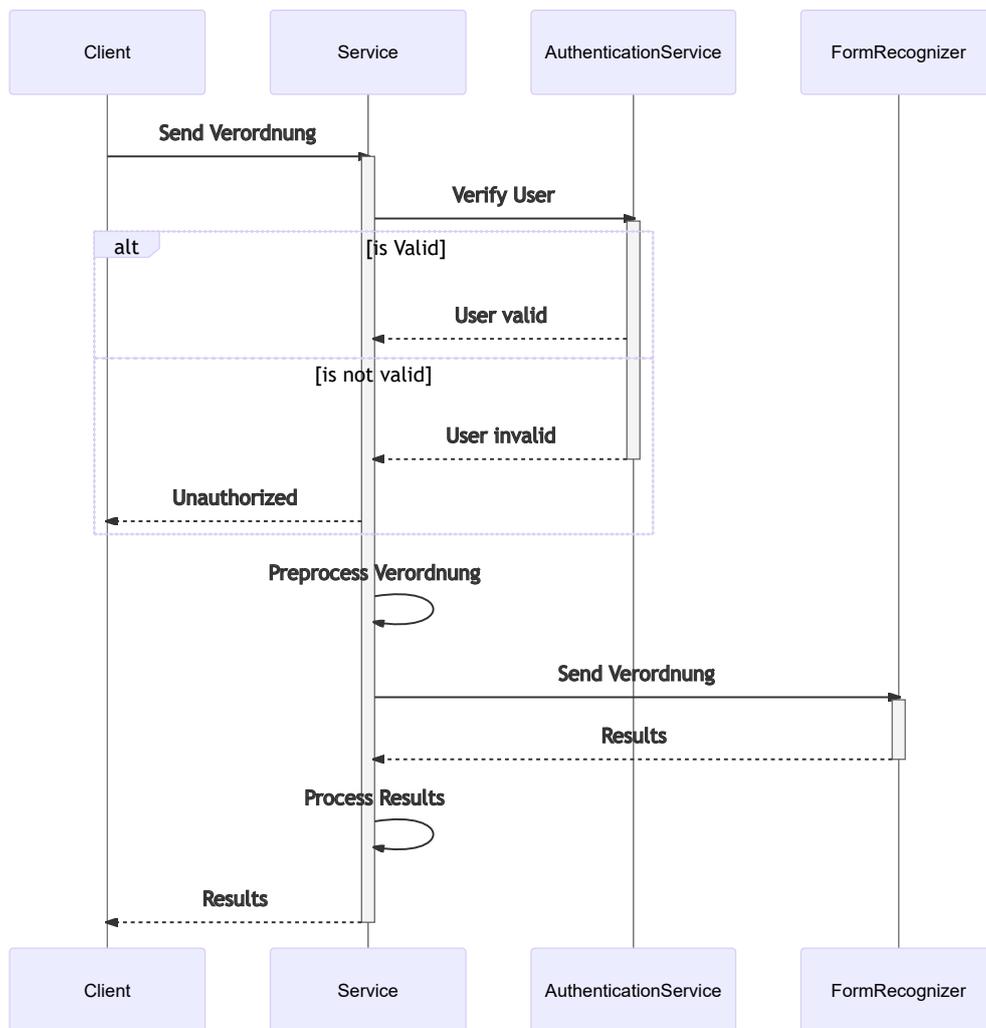


Abbildung 5.1: Sequenz-Diagramm für den Ablauf des API-Aufrufs

seine Logik.

Für die Nachbearbeitung der Ergebnisse steht, mit dem “Muster4Converter”, eine weitere Klasse zur Verfügung. Diese implementiert die Abstrakte Klasse “ResultConverter”. Die Funktionalität wurde aufgespalten damit generische Funktionen, wie z.B. die Übertragung von Daten einzelner Wertefelder, und spezifische Funktionalität, wie die Übertragung einer ganzen Verordnung, voneinander getrennt sind. Durch diese Trennung können mögliche zukünftige Erweiterungen, für andere Verordnungen, die “ResultConverter” Klasse implementieren und so auf bereits vorhandene Logik zugreifen, während lediglich, die für die neuen Dokumente, spezifischen Funktionalitäten implementiert werden müssen.

Der Rückgabewert für den Dokumentenanalyse-Endpunkt ist “RecognitionResponse”. Diese Klasse enthält neben dem Ergebnis, auch zusätzliche Metadaten, wie z.B. die benötigte Zeit. Das Ergebnis wird als abstrakte Klasse “Result” angegeben, welche als Verbindungselement fungiert, damit verschiedene Arten von Erkennungsergebnissen in den Rückgabewert eingefügt werden können, ohne weitere Konfigurationsanpassungen an anderen Klassen vornehmen zu müssen. Für die Muster 4 Verordnung existiert die “Muster4Result” Klasse, welche, mit 40 Eigenschaften, alle Felder der Muster 4 Verordnung enthält. Zur Übersicht wurden diese Eigenschaften im angehängten Klassen-Diagramm jedoch ausgeblendet. Die einzelnen Eigenschaften sind jedoch alle vom Typ “RecognitionResult”, welches neben der rohen Erkennung und dem, in das erwartete Datenformat konvertierte, Erkennungsergebnis auch die Confidence enthält. Die Nutzung einer einzelnen Klasse für verschiedene Datentypen wird durch die Nutzung von Generics ermöglicht.

Zusätzlich zu den Komponenten der Business Logik, existieren noch weitere technische Komponenten. Neben den von ASP.NET für die Konfiguration und das ausführen des Programms benötigten “Program” und “Startup” Klassen, wurden noch die “Config” und “Authentication” Klassen hinzugefügt. Aufgabe der “Config” Klasse ist das zur Verfügung stellen der Programmkonfiguration aus den Konfigurationseinstellungen, die entweder über Konfigurationsdateien oder Umgebungsvariable festgelegt werden können. Diese Konfiguration umfasst Einstellungen für das Logging, die Verbindung mit dem Form Recognizer Dienst und der Zugriffskontrolle. Die “Authentication” Klasse beinhaltet Logik zum Authentifizieren der Nutzer, und wird mit Hilfe von ASP.NET Funktionalitäten als Middleware bei jeder API Anfrage aufgerufen.

5.3 Authentifizierung

Da der Dienst lediglich von anderen Diensten aufgerufen werden soll, ist eine zur Verfügung Stellung der API für den Endnutzer nicht notwendig. Aus diesem Grund kann der Dienst im internen Netzwerk, ohne Zugriff auf externe Ressourcen, eingesetzt werden. Ein API Gateway sowie Firewalls stellen die Integrität dieses Netzwerkes sicher, indem sie alle Nutzer Authentifizieren und unberechtigte Zugriffe abweisen. Dennoch kann nicht davon abgesehen werden eine eigene Zugriffskontrolle für den Dienst einzurichten.

Nach Rücksprachen mit dem Leitenden Entwickler, ist die Entscheidung getroffen worden eine Authentifizierung durch API Keys einzusetzen. In diesem System werden Zugriffsschlüssel für einen Dienst erstellt und verteilt. Die Nutzer geben diese Schlüssel bei ihren Anfragen mit an und können so authentifiziert werden. Dieses System wird häufig für die Absicherung von APIs, primär wenn diese von anderen Programmen genutzt werden sollen, verwendet. Ein Nachteil besteht in der schwa-

chen Sicherheit dieses Systems, da eine Übertragung des Schlüssels unverschlüsselt erfolgt, bzw. die Verschlüsselung auf anderen Ebenen stattfinden muss. Aus diesem Grund ist dieses System anfällig gegen Angriffe bei denen der Schlüssel aus einer Anfrage ausgelesen und anschließend durch eine nicht berechnete Person genutzt wird. Dieses Risiko wird jedoch, wie oben bereits beschrieben, durch die Positionierung des Dienstes im internen Firmennetzwerk größtenteils mitigiert.

Eine Alternative bestünde in der Verwendung des OAuth Standards, welcher nicht nur eine einfache Zugriffskontrolle, sondern auch Role Based Access Control (RBAC) und Single Sign-On (SSO) ermöglicht. Die Vorteile dieses Standards sind eine Zentrale Verwaltung von Zugriffsrechten, sowie die Möglichkeit auch den Ursprung einzelner Anfragen zurückzuverfolgen. Gleichzeitig benötigt dieses System jedoch auch einen zentralen Authentifizierungsserver, welcher die Anmeldung und Zugriffskontrolle abwickelt. Da ein solcher Dienst aktuell nicht für den Teilbereich DeTouro eingerichtet ist, wurde sich gegen dieses System entschieden. Es besteht jedoch die Möglichkeit, in Zukunft die Authentifizierung über einen solchen Dienst abzuwickeln. Die Rekonfiguration der Anwendung, ist durch den flexiblen Aufbau von ASP.NET, ohne größere Schwierigkeiten und ohne Einfluss auf die API möglich.

6. Implementation

Dieses Kapitel widmet sich der Implementation des Dokumentenerkennungsdienstes. Im Detail werden beschrieben wie auf den Form Recognizer Dienst zugegriffen wird und wie die API und unterliegende Struktur aufgebaut ist.

6.1 Docker

Damit der Dienst flexibel einsetzbar ist, besteht eine Anforderung darin ihn Docker kompatibel zu machen. Hierfür wurde Dockerfiles erstellt und die Verwendung von Healthchecks konfiguriert. Docker bietet die Möglichkeit Programme zu Kapseln um so einen Reibungslosen Einsatz auf Zielsystemen, ohne Konflikte mit dem Zielsystem selbst oder anderen Diensten, zu gewährleisten. Dazu wird, von einer Docker Runtime, auf Basis einer Dockerfile, das gewünschte Programm erstellt. Das Ergebnis dieses Vorgangs ist ein Image, welches von jeder Container Runtime, unabhängig vom Basis System, genutzt werden kann um einen Container zu starten, welcher wiederum das gewünschte Programm ausführt. Zusätzlich können diese Images jedoch auch in Container Registries hochgeladen und so verteilt werden. Der große Vorteil von Docker und anderen Container Runtimes ist, dass die Abhängigkeiten zum erstellen und ausführen des Dienstes durch die Runtime für jeden Container separat verwaltet werden können, ohne mit dem Basissystem oder anderen Programmen oder Diensten zu interagieren oder in Konflikt zu geraten.

6.1.1 Dockerfile

Für den Dienst werden zwei verschiedene Dockerfiles benötigt. Auf der einen Seite eine zum bauen des Dienstes für einen potentiellen Einsatz, auf der anderen Seite ein weiteres zum durchführen der Tests.

Das Dockerfile für den Build-Vorgang des Dienstes, enthält mehrere Anweisungen die das Programm zuerst in einen Container kopieren und anschließend erstellen und ausführbar machen. Um die finale Größe des Image so weit wie möglich zu reduzieren, wird die Anwendung in einem separaten Container gebaut und nur das kompilierte Programm in das finale Image übertragen.

Das Dockerfile für die Testausführung enthält neben den Befehlen um das Projekt, zusammen mit den Tests zu Bauen, Anweisungen zum durchführen der Tests. Der dabei erstellte Testreport wird, für eine mögliche Analyse, im Container gespeichert. Beide Dockerfiles werden als Teil der CI-Pipelines automatisch ausgeführt.

6.1.2 Healthchecks

Mit aktivierten Healthchecks hat die Docker Umgebung die Möglichkeit die Gesundheit des Containers abzufragen. Da es sich um einen Microservice mit niedriger Komplexität und wenigen Abhängigkeiten handelt, werden lediglich einfache Healthchecks genutzt. Diese fügen einen weiteren

API Endpunkt hinzu, der, ohne weitere Logik, mit einem HTTP Status Code 200 antwortet. Solange die Docker Runtime diesen Endpunkt aufrufen kann und ein positives Ergebnis bekommt, kann angenommen werden das auch die anderen Teile der API bereit sind Anfragen entgegen zu nehmen. Andernfalls müsste der Container neugestartet werden, was vollautomatisch und mit minimaler Downtime, von der Docker Runtime, sofern Konfiguriert, durchgeführt werden kann.

6.2 CI/CD-Pipelines

Bevor mit der eigentlichen Implementation begonnen werden kann, müssen Git-Repository und CI- und CD-Pipelines eingerichtet werden. Nachdem das Repository erstellt wurde, wurde auch der in Kapitel 2.2 beschriebene Git Workflow eingerichtet.

Für die Konfiguration der Pipelines wurde von den, von GitLab selbst, zur Verfügung gestellten Funktionen Gebrauch gemacht. Diese erlauben das reagieren auf eine Vielzahl von Ereignissen im Repository und können entsprechende Befehle ausführen.

Die CI-Pipeline setzt sich aus insgesamt vier Schritten zusammen. Diese sind der Build, die Tests, das Hochladen des erstellten Programms sowie eine Aufräumphase. Während des Build wird das Programm, mit Hilfe von Docker, erstellt. Durch das definieren der Dockerfile muss an dieser Stelle keine weitere Programm spezifische Konfiguration angegeben werden, was den Prozess einfach und portable macht. Die anschließende Phase führt die Tests für das Programm aus (Abbildung 6.1). Dafür muss zuerst das Programm erstellt werden (Zeilen 5-6). Auch dieser Schritt wird von Docker durchgeführt. Mit der “-t” Option wird dem erstellten Image ein Tag zugewiesen, während die “-f” Option die zu benutzende Dockerfile definiert. Hier ist es nicht möglich auf das Ergebnis des vorherigen Schrittes zuzugreifen, da das Programm zusammen mit den Tests erstellt werden muss, diese jedoch nicht in einer Version enthalten seien sollen die potentiell eingesetzt wird. Anschließend können die Test ausgeführt werden, in dem das erstellte Programm ausgeführt wird (Zeilen 7-8). Die Id, des aus dem Image erstellten Containers, wird dabei zwischengespeichert um sie im nächsten Schritt einzusetzen um die Testergebnisse aus dem Container zu extrahieren (Zeilen 9-10). Diese werden durch die Option “artifacts” in Zeilen 14-16 in GitLab zur Verfügung gestellt.

6.3 Analyse-Endpunkt

Durch die Verwendung von ASP.NET als Framework für den Webservice, ist das erstellen von API Endpunkten einfach. Neben, den vom Framework zu Verfügung gestellten Tools und Fähigkeiten, wird auch auf externe Bibliotheken zurückgegriffen um Zugriff auf zusätzliche Funktionen zu erhalten. Die wohl wichtigste Bibliothek ist “Swashbuckle”, welche es ermöglicht API Endpunkte detaillierter zu beschreiben als mit den ASP.NET Werkzeugen möglich wäre. Zusätzlich kann, auf Basis dieser Beschreibungen, eine OpenAPI Spezifikation erstellt werden, welche als Dokumentation der API dient und genutzt werden kann um API Clients zu erstellen. Die Spezifikation kann ebenfalls als Basis für einen Grafischen API Explorer, welcher die Endpunkte interaktiv beschreibt, genutzt werden.

Da der Dienst als Microservice entwickelt werden soll und aktuell nur die Muster 4 Verordnung Teil des Scopes ist, enthält die API, zum aktuellen Zeitpunkt, lediglich einen einzelnen Endpunkt.

```

1 test:
  image: docker:latest
3  stage: test
  script:
5    - docker build -t form-recognizer-unit-test:$CI_PIPELINE_ID
      -f FormRecognizerUnitTestDockerfile .
7    - container_id=
      $(docker create form-recognizer-unit-test:$CI_PIPELINE_ID)
9    - docker cp $container_id:/app/form-recognizer-unit-test/TestResults
      form-recognizer-unit-test/TestResults
11   - docker rm $container_id
  tags:
13   - docker
  artifacts:
15   paths:
      - form-recognizer-unit-test/TestResults/*.trx
17

```

Abbildung 6.1: Konfiguration für die Test-Stage der CI-Pipeline

Dieser ist für die Erkennung von Muster 4 Verordnungen zuständig und erfüllt seine Aufgabe, indem er die Abbildung einer Verordnung entgegen nimmt, sie an den Form Recognizer weiterleitet, die Erkennungsergebnisse aufbereitet und an den Client sendet (Abbildung 6.2).

In Zeile 6 ist zu erkennen wie der Endpunkt die Verordnung, als Teil eines Formulars, erwartet. Anschließend wird, in Zeilen 9 bis 14, die Datei überprüft um sicherzustellen das sie ein unterstütztes Format hat. Dazu wird eine Liste mit, von Form Recognizer unterstützten, Dateitypen genutzt. Falls der Dateitype nicht unterstützt wird, wird eine entsprechende Fehlermeldung erstellt, der Prozess abgebrochen und der Client informiert.

In Zeilen 16 und 17 werden eine Stoppuhr vorbereitet und die Datei geöffnet. Anschließend wird ein Form Recognizer Client, welcher Teil des Azure SDK ist, initialisiert indem die Konfigurationsparameter aus der Konfiguration geladen und übergeben werden (Zeilen 18-20). Nachdem diese Vorbereitungen abgeschlossen wurden, wird die Datei an den Form Recognizer übertragen (Zeilen 23-26), wonach auf die Fertigstellung der Erkennung gewartet wird (Zeilen 27-28). Die benötigte Zeit für diese beiden Operationen wird dabei durch die vorbereitete Stoppuhr gemessen (Zeilen 22 & 29).

Anschließend wird das Ergebnis in das Rückgabeformat konvertiert (Zeilen 34-35) und an den Client gesendet (Zeilen 38-41).

6.4 Konvertierung

Für die Konvertierung der Erkennungsergebnisse in das, in Kapitel 5.2 definierte, Rückgabeformat, wird eine extra Klasse benötigt. Diese Klasse ist der “Muster4Converter”, welcher die Erkennungen, auf Basis der, durch den Form Recognizer zugewiesenen, Labels durch, Datentyp spezifische, Hilfsfunktionen übersetzen lässt. Eine dieser Hilfsfunktionen ist “CreateResultFromCheckbox”, welche das Erkennungsergebnis eines Kontrollkästchen konvertieren kann (Abbildung 6.3). Dafür werden

```
[Route("upload")]
2 [HttpPost]
  [Authorize]
4 [ProducesResponseType(typeof(Muster4Result), 200)]
  public async Task<IActionResult> UploadTest(
6   [FromForm(Name = "file")] IFormFile file,
   [FromForm(Name = "imageType")] ImageType imageType)
8 {
10  if (!allowedContentTypes.Contains(file.ContentType))
11  {
12     return BadRequest($"Illegal Content-Type! " +
13     "Only following Content-Types are permitted: " +
14     $"{String.Join(", ", allowedContentTypes)}");
15  }
16  var stopwatch = new Stopwatch();
17  var stream = file.OpenReadStream();
18  var client = new FormRecognizerClient(
19     new Uri(configuration.FormRecognizerEndpoint), credential
20  );
21
22  stopwatch.Start();
23  var operation = await client.
24     StartRecognizeCustomFormsAsync(
25     configuration.FormRecognizerModelID, stream
26     );
27  var result = await operation.
28     WaitForCompletionAsync(new TimeSpan(0,0,5));
29  stopwatch.Stop();
30
31  Muster4Result muster4 = null;
32  if (result.Value.Count > 0)
33  {
34     muster4 = muster4Converter.
35     ConvertRecognitionResult(result.Value.First().Fields.Values);
36  }
37
38  return Ok(new RecognitionResponse<Muster4Result>() {
39     Result = muster4,
40     ElapsedTime = stopwatch.Elapsed
41  });
42 }
```

Abbildung 6.2: Quellcode für den Muster 4 Analyse Endpunkt

der tatsächlich erkannte (Zeilen 7-8) und der, in das erwartete Datenformat, konvertierte Wert (Zeile 9) sowie die Confidence (Zeile 10) ausgelesen und für die Erstellung eines “RecognitionResult” Objektes genutzt. Das so erstellte Objekt wird zurückgegeben und von der “Muster4Converter” Klasse genutzt um, zusammen mit den anderen Feldern, ein “Muster4Result” zu bilden.

```
protected virtual RecognitionResult<bool> CreateResultFromCheckbox(  
2   FormField field  
   )  
4 {  
   return new RecognitionResult<bool>  
6   {  
       Value = field.Value.AsSelectionMarkState() ==  
8       SelectionMarkState.Selected,  
       Confidence = field.Confidence,  
10      Raw = field.ValueData.Text  
   };  
12 }
```

Abbildung 6.3: Quellcode für die Konvertierung von Boolean Wertefeldern

6.5 Pre-Processing

Um zu überprüfen ob die optionale Funktionalität, der Analyse von Fotos von Muster 4 Verordnungen, sinnvoll und im Zeitrahmen implementiert und in den Dienst integriert werden kann, wurde ein weiteres Programm zu Testzwecken geschrieben. Die Implementation der Bildmanipulation findet auf Basis von OpenCV sowie dem Kompatibilitätslayer OpenCVSharp, auf Basis der Arbeit und Beschreibungen von Rosebrock (2014), statt.

Die, für die erfolgreiche Konvertierung eines Fotos in ein Scann ähnliches Bild, benötigten Schritte sind folgende:

1. Kanten erkennen
2. Konturen des Dokuments erkennen
3. Perspektivische Transformation durchführen

Bevor mit der Analyse begonnen werden kann, wird das Bild zuerst auf eine feste Größe skaliert und der Skalierungsfaktor gespeichert. Anschließend werden, durch die OpenCV “Canny” Methode, Kanten im Bild erkannt. Diese können für die Kontur Erkennung durch die OpenCV “FindContours” Methode genutzt werden. Nachdem die Kontour des Dokumentes erkannt wurde, werden die vier äußersten Ecken extrahiert, um mit ihnen die Transformation durchzuführen.

Der Programmcode für die Transformation ist Abbildung 6.4 zu entnehmen. Zuerst werden die Ecken, durch eine weitere Methode, im Uhrzeigersinn, beginnend oben links, sortiert (Zeile 7). Dies ist notwendig, damit die Reihenfolge der Ecken konsistent ist und die Transformation korrekt durchgeführt werden kann. Anschließend werden die Punkte wieder auf die Ursprüngliche Größe skaliert,

indem die einzelnen Koordinaten mit dem Skalierungsfaktor, aus dem ersten Schritt, multipliziert werden (Zeilen 8-9).

Nachdem die einzelnen Punkte aus der Liste extrahiert wurden (Zeilen 11-14), werden die Höhe und Breite des Dokuments errechnet (Zeilen 16-22). Dies geschieht indem die jeweiligen Distanzen berechnet werden und, im Falle von leichten Abweichungen, die längere der beiden Kanten angenommen wird.

Anschließend wird auf Basis dieser Werte ein Ziel erstellt (Zeilen 24-30), welche genutzt wird um, mithilfe der OpenCV Methode "GetPerspectiveTransform", eine Transformationsmatrix zu erstellen (Zeile 32). Diese Matrix wird dann, mithilfe der OpenCV Methode "WarpPerspective", genutzt um das Original Bild zu transformieren (Zeile 33). Damit ist die Transformation abgeschlossen. Das Ergebnis einer solchen Transformation kann Abbildung 6.5 entnommen werden.

Ein großes Hindernis in der Integration dieser Funktion, ist die Abhängigkeit von einer OpenCV Installation im ausführenden System. Dies führt zu einem, im Vergleich zum Dienst ohne die Funktion, wesentlich komplexeren Build-Prozess und einem, mehrere Male größeren Docker Image. Der komplexere Build-Prozess steigert auch den Wartungsaufwand. Zusätzlich muss die Robustheit der Erkennung und Transformation noch weiter verbessert werden, denn sobald nicht alle vier Ecken sichtbar sind oder die Verordnung in einem steilen Winkel abgebildet ist, schlägt der Prozess der Kontur- und Ecken-Erkennung, und damit auch die Transformation, fehl. Sowohl aus diesen Gründen, als auch aus Zeitmangel wurde deshalb keine Integration in den Dienst vorgenommen.

```
private static void PerspectiveTransform(
2  Mat image,
   int ratio,
4  Point2f[] corners
)
6 {
   var orderedCorners = OrderCorners(corners);
8  orderedCorners = orderedCorners.
   Select(e => new Point2f(e.X * ratio, e.Y * ratio)).ToArray();
10
   var topLeft = orderedCorners[0];
12  var topRight = orderedCorners[1];
   var bottomLeft = orderedCorners[2];
14  var bottomRight = orderedCorners[3];

16  var widthA = topLeft.DistanceTo(topRight);
   var widthB = bottomLeft.DistanceTo(bottomRight);
18  var maxWidth = (int)Math.Max(widthA, widthB);

20  var heightA = topLeft.DistanceTo(bottomLeft);
   var heightB = topRight.DistanceTo(bottomRight);
22  var maxHeight = (int)Math.Max(heightA, heightB);

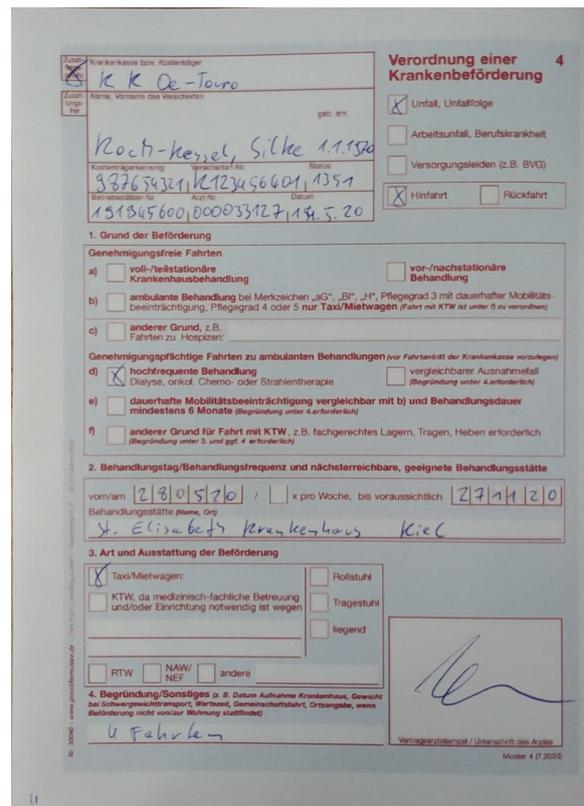
24  var dst = new Point2f[]
   {
26     new Point2f(0, 0),
       new Point2f(maxWidth - 1, 0),
28     new Point2f(0, maxHeight - 1),
       new Point2f(maxWidth - 1, maxHeight - 1)
30  };

32  var m = Cv2.GetPerspectiveTransform(orderedCorners, dst);
   Cv2.WarpPerspective(image, image, m, new Size(maxWidth, maxHeight));
34 }
```

Abbildung 6.4: Quellcode für die Transformation eines Bildes einer Verordnung



(a) Verordnung vor dem Transformieren



(b) Verordnung nach dem Transformieren

Abbildung 6.5: Foto einer Muster 4 Verordnung vor und nach dem Transformieren

7. Zusammenfassung

Dieses Kapitel fasst die Ergebnisse des Projektes zusammen und bewertet sie. Zudem wird ein Ausblick auf den zukünftigen Einsatz und mögliche Weiterentwicklungen gegeben.

7.1 Bewertung

Das Projekt konnte, trotz der widrigen Umstände durch die andauernde Covid 19 Pandemie, ohne größere Schwierigkeiten fertiggestellt werden. Das Arbeiten im Home-Office sowie die erschwerte Kommunikation mit Ansprechpartnern, in sowohl Unternehmen als auch Universität, haben die Arbeit teilweise zwar verkompliziert, jedoch nicht nachhaltig beeinträchtigt.

Die Untersuchung der verschiedenen Frameworks für die Dokumentenerkennung hat viel Zeit in Anspruch genommen, jedoch interessante Ergebnisse produziert. Enttäuschend waren die Leistungen der Open Source Frameworks. Diese Defizite lassen sich wahrscheinlich durch zusätzlichen Aufwand in der Optimierung und mehr Zeit in der Entwicklung beheben, jedoch stand diese Menge an Ressourcen nicht für dieses Projekt zur Verfügung. Die Möglichkeiten dieser Alternativen könnten jedoch in einem separaten Projekt erforscht werden.

Die optionale Funktion zur Erkennung von Verordnungen auf Fotos ist erfolgreich demonstriert worden, konnte jedoch nicht in den Dienst übernommen werden. Auch hier gibt es noch Bedarf das System weiter zu optimieren, aber auch diese Arbeit war mit den gegebenen Ressourcen, vorrangig der Zeit, im Anbetracht der niedrigen Priorität, nicht möglich.

Zusammenfassend lässt sich feststellen, dass die Arbeit an dem Projekt gut abgelaufen ist. Die Analyse und Empfehlung, auf Basis der Untersuchung, haben viel Arbeit und Zeit in Anspruch genommen, während die Design und Implementation, gerade durch den beschränkten Funktionsumfang eines Microservice, weniger komplex waren. Dennoch mussten hier in allen Bereichen, vom Einrichten der Entwicklungsumgebung und konfigurieren der Anwendung, bis zum Implementieren des Dienstes und Erstellen der Tests, Arbeiten durchgeführt werden, die nicht nur in den Bereich der Software Entwicklung sondern auch von DevOps und anderen Bereichen fallen.

7.2 Ausblick

Der Dienst kann alle gestellten Anforderungen erfüllen und könnte damit für produktive Zwecke eingesetzt werden. Voraussetzung ist das Ausführen von Form Recognizer auf einem lokalen Server. Jedoch muss ein Einsatz auf eigenen Servern erst von Microsoft freigeschaltet werden.

Damit die Fähigkeiten des Dienstes auch vom Endnutzer in Anspruch genommen werden können, ist es notwendig den Dienst in DeTouro zu integrieren. Dazu müssen entsprechende Erweiterungen durchgeführt werden, die es erlauben Verordnungen aus DeTouro an den Dienst, zur Analyse, zu schicken. Des Weiteren müssen die Erkennungsergebnisse von den Nutzern validiert werden. Hierfür werden neue Eingabemasken und Änderungen am Workflow nötig sein.

Um die Möglichkeiten des Dienstes voll auszuschöpfen, sollten weitere Verordnungen oder Doku-

mente zum Fähigkeitsprofil hinzugefügt werden. Damit diese Fähigkeiten jedoch im vollen Umfang genutzt werden können, ist es notwendig das der Dienst, allgemein verfügbar, in die produktübergreifende Microservice Architektur integriert wird. Dieser Schritt würde auch erfordern das die, bis jetzt unabhängigen, Dienste von DeTouro ebenfalls in diese Architektur integriert werden, da sonst ein Zugriff nicht mehr möglich wäre.

In Zukunft sollte auch darüber nachgedacht werden, die Arbeiten am Modul für die Erkennung von Dokumenten auf Fotos fortzusetzen. Hier besteht jedoch auch die Möglichkeit, speziell in Hinblick auf die Abhängigkeit von OpenCV, eine solche Funktion in einem eigenen Microservice, oder sogar als Teil von eigenen Smartphone Apps, wie z.B. der DeTouro App, zu implementieren. Andernfalls kann auch ein Austausch von OpenCV gegen ein anderes, besser in .NET integriertes, Bildmanipulations-Framework sinnvoll sein.

Zusammenfassend bietet der Dienst eine robuste Grundlage für die Erkennung von medizinischen Verordnungen und anderen Formularen, und damit der Automatisierung und Digitalisierung, welche langfristig und nachhaltig die Ärzte, Leistungserbringer und Kostenträger entlasten können.

A. Anforderungen

Bezeichnung	Beschreibung
FR-01	Der Dienst muss Muster 4 Verordnungen erkennen können
FR-02	Bei der Erkennung von Feldern in der Verordnung, dürfen nicht mehr als zwei Felder falsch erkannt werden.
FR-03	Bei der Erkennung von maschinellen Texten, darf nicht mehr als ein Feld fehlerhaft erkannt werden.
FR-04	Bei der Erkennung von handschriftlichen Texten muss mindestens die Hälfte alle Felder korrekt erkannt werden.
FR-05	Bei der Erkennung von Kontrollkästchen darf nicht mehr als ein Feld fehlerhaft erkannt werden.
FR-06	Der Dienst kann dazu in der Lage sein Verordnungen von Fotos zu erkennen.
FR-07	Die Verarbeitung einer Verordnung darf nicht länger als 60 Sekunden dauern.
FR-08	Die Verarbeitung einer Verordnung kann in unter 10 Sekunden abgeschlossen werden.

Tabelle A.1: Funktionale Anforderungen an den Dienst

Bezeichnung	Beschreibung
NFR-01	Der Dienst muss über Docker eingesetzt werden können.
NFR-02	Der Dienst soll die Docker Gesundheitsüberwachung unterstützen.
NFR-03	Der Dienst darf die Verordnungen nicht permanent speichern.
NFR-04	Zur Erfüllung der Datenschutz Richtlinien muss der gesamte Dienst auf eigener Hardware ausgeführt können.

Tabelle A.2: Nicht-Funktionale Anforderungen an den Dienst

B. Untersuchungsergebnisse

Metrik	Amazon Textract	Microsoft Form Recognizer	Google Document AI
Feldsegmentierung Recall	0,68	0,94	0,76
Feldsegmentierung Precision	1	0,96	0,91
Texterkennung Score	0,14	0,27	0,52
Texterkennung CER	0,31	0,12	0,06
Kontrollkästchen Recall	1	1	0,95
Kontrollkästchen Precision	1	1	1
benötigte Zeit	00:07,8	00:10,8	00:13,7

Tabelle B.1: Ergebnisse der einzelnen Frameworks bei der Erkennung von handschriftlich ausgefüllten Muster 4 Verordnungen

Metrik	Amazon Textract	Microsoft Form Recognizer	Google Document AI
Feldsegmentierung Recall	0,79	0,97	0,80
Feldsegmentierung Precision	0,99	0,99	0,96
Texterkennung Score	0,99	0,99	0,98
Texterkennung CER	0	0	0
Kontrollkästchen Recall	1	1	1
Kontrollkästchen Precision	1	1	1
benötigte Zeit	00:07,6	00:10,5	00:12,2

Tabelle B.2: Ergebnisse der einzelnen Frameworks bei der Erkennung von maschinell ausgefüllten Muster 4 Verordnungen

	Original	Erkannter Wert	Tatsächlicher Wert
1	<u>KK De-Touro</u>	kkc Oe-ouo	KK De-Touro
2	Ast, Peter	n.A.	Ast, Peter
3	31.12.69	31.12.6	31.12.69
4	987654321	397654321	987654321
5	A123456778	\$876543211/12)416776 778	A123456778
6	12691	12691	12691
7	200133500	200133500	200133500
8	<u>123443749</u>	173447251	123443749
9	13.05.20	13.05- 2o	13.05.20
10	<u>270520</u>	27015210 2 2	27.05.20
11	<u>271120</u>	Z H A Z O	27.11.20
12	<u>St. Josef Krankenhaus Moers</u>	St. Joef Meet	St. Josef Krankenhaus Moers
13	<u>2 Fahrten</u>	2	2 Fahrten

Tabelle B.3: Amazon Textract Handschrifterkennungsergebnisse

	Original	Erkannter Wert	Tatsächlicher Wert
1	RR De-Touro	RR De-TourD	KK De-Touro
2	Ast, Peter	Ast, Peter	Ast, Peter
3	31.12.69	37. 12.69	31.12.69
4	987654321	98 76 59 3211	987654321
5	A123456778	A 123456778l	A123456778
6	12691	12637	12691
7	200133500	200 133 500,	200133500
8	123443749	123 49 7 749,	123443749
9	13.05.20	13. 05.20	13.05.20
10	270520	12171015/ 2 0 1 x pro	27.05.20
11	271120	12 1 7 7 1 2 0	27.11.20
12	St. Josef Krankenhaus Moers	St. Josef Ruankey have Moers	St. Josef Krankenhaus Moers
13	2 Fahrten	2. Fehren	2 Fahrten

Tabelle B.4: Microsoft Form Recognizer Handschrifterkennungsergebnisse

	Original	Erkannter Wert	Tatsächlicher Wert
1	KR Oe-Touro	KR Oe-Touro	KK De-Touro
2	Ast, Peter	geb.am Ast, Peter 31.12.69	Ast, Peter
3	31.12.69	geb.am Ast, Peter 31.12.69	31.12.69
4	987654321	3876543211A123456778112631	987654321
5	A123456778	3876543211A123456778112631	A123456778
6	12691	3876543211A123456778112631	12691
7	200133500	n.A.	200133500
8	123443749	n.A.	123443749
9	13.05.20	n.A.	13.05.20
10	270520	270520	27.05.20
11	271120	/27/1120	27.11.20
12	St. Josef Krankenhaus Moers	St. Josef Ruankenhaus Moers	St. Josef Krankenhaus Moers
13	2 Fahrten	n.A.	2 Fahrten

Tabelle B.5: Google Document AI Handschrifterkennungsergebnisse

C. Klassendiagramm

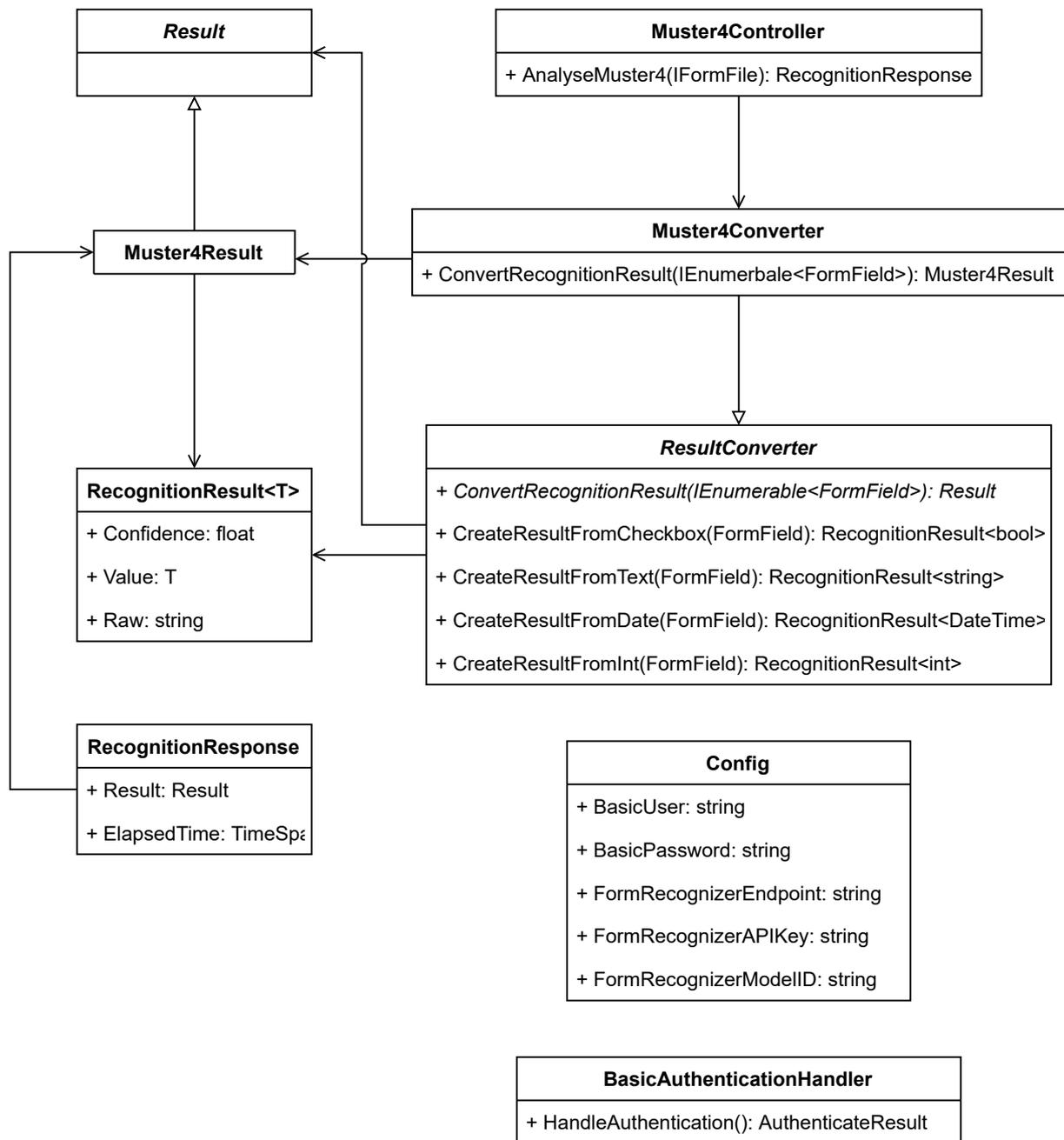


Abbildung C.1: Klassendiagramm (Zur Übersichtlichkeit wurden die Eigenschaften von `Muster4Result` ausgeblendet)

Literatur

- Adobe (2020). *Adobe Acrobat Reader DC*. <https://get.adobe.com/de/reader/>.
- Alvermann, D. (2019). *Word Error Rate & Character Error Rate – How to evaluate a model*. [online] abger. unter <https://rechtsprechung-im-ostseeraum.archiv.uni-greifswald.de/word-error-rate-character-error-rate-how-to-evaluate-a-model/> [besucht am 14. Okt. 2020].
- COCO (2020). *COCO - Common Object in Context*. [online] abger. unter <https://cocodataset.org/#detection-eval> [besucht am 29. Okt. 2020].
- Cook, J. und Ramadas, V. (2020). When to consult precision-recall curves. *Stata Journal*, 20(1), S. 131–148. ISSN: 15368734. DOI: 10.1177/1536867X20909693.
- Dignan, L. (2020). *Top cloud providers in 2020: AWS, Microsoft Azure, and Google Cloud, hybrid, SaaS players*. [online] abger. unter <https://www.zdnet.com/article/the-top-cloud-providers-of-2020-aws-microsoft-azure-google-cloud-hybrid-saas/> [besucht am 29. Sep. 2020].
- Driessen, V. (2010). *A successful Git branching model*. [online] abger. unter <https://nvie.com/posts/a-successful-git-branching-model/> [besucht am 29. Sep. 2020].
- Google (2019). *Tesseract OCR*. <https://github.com/tesseract-ocr/tesseract>.
- Harald Scheidl Nishant Bhandari, R. J. (2019). *Simple HTR*. <https://github.com/githubharald/SimpleHTR>.
- Lin, T. Y., Goyal, P., Girshick, R., He, K. und Dollar, P. (2020). Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2), S. 318–327. ISSN: 19393539. DOI: 10.1109/TPAMI.2018.2858826. arXiv: 1708.02002.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y. und Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9905 LNCS, S. 21–37. ISSN: 16113349. DOI: 10.1007/978-3-319-46448-0_2. arXiv: arXiv:1512.02325v5.
- Microsoft (2020). *VoTT (Visual Object Tagging Tool)*. <https://github.com/Microsoft/VoTT>.
- Radigan, D. (2020). *Kanban - A brief introduction* | Atlassian. [online] abger. unter <https://www.atlassian.com/agile/kanban> [besucht am 29. Sep. 2020].
- Redmon, J. und Farhadi, A. (2018). YOLO v3. *Tech report*, S. 1–6. [online] abger. unter <https://pjreddie.com/media/files/papers/YOLOv3.pdf>.
- Ren, S., He, K., Girshick, R. und Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), S. 1137–1149. ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2577031. arXiv: 1506.01497.

- Roboflow (2020). *Roboflow*. [online] abger. unter <https://app.roboflow.com> [besucht am 28. Okt. 2020].
- Rosebrock, A. (2014). *How to Build a Kick-Ass Mobile Document Scanner in Just 5 Minutes*. [online] abger. unter <https://www.pyimagesearch.com/2014/09/01/build-kick-ass-mobile-document-scanner-just-5-minutes/> [besucht am 4. Jan. 2021].
- Schulenburg, J. (2018). *GOOCR*. [online] abger. unter <https://www-e.ovgu.de/jschulen/ocr/> [besucht am 18. Okt. 2020].
- Tsung-Yi Lin Priya Goyal, R. G. und He, K. (2020). *Keras RetinaNet*. <https://github.com/fizyr/keras-retinanet>.
- Yun-An, Z., Ziheng, P., Hongyan, D. und Guanghan, B. (Dez. 2019). YOLOv3-Tesseract model for improved intelligent form recognition. *Recent Advances in Computer F and Communications*, 13. DOI: 10.2174/2666255813666191204141610.