HAW HAMBURG

BACHELOR THESIS
Anton Müller

Entwurf und Entwicklung einer Anwendung zur KI-assistierten Beschriftung von Bildern für Maschinelles Lernen

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Anton Müller

Entwurf und Entwicklung einer Anwendung zur KI-assistierten Beschriftung von Bildern für Maschinelles Lernen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Marina Tropmann-Frick

Eingereicht am: 15. Juli 2022

Anton Müller

Thema der Arbeit

Entwurf und Entwicklung einer Anwendung zur KI-assistierten Beschriftung von Bildern für Maschinelles Lernen

Stichworte

Maschinelles Lernen, Serverless, Objekterkennung, Datenbeschriftung, Google Cloud

Kurzzusammenfassung

Diese Arbeit beschreibt den Entwurf und die Entwicklung einer Anwendung, die dabei hilft, Datensätze mit Bildern zu beschriften. Dazu werden Vorschläge von einer KI generiert, welche unter der Verwendung von Objekterkennung mit der YOLO Architektur erzeugt werden. Die Umsetzung des Systems wird mit Serverless Computing auf der Google Cloud realisiert. Um die Leistungsfähigkeit des System zu testen, wird ein Versuch mit einem Beispielszenario durchgeführt.

Anton Müller

Title of Thesis

Design and implementation of an AI-assisted image labeling application for machine learning.

Keywords

Machine Learning, Serverless, Object detection, Data labeling, Google Cloud

Abstract

This thesis describes the design and development of an application that helps to label datasets with images. For this purpose, suggestions are generated by an AI using object detection with the YOLO architecture. The system will be implemented using Serverless computing on the Google Cloud. In order to test the performance of the system, an experiment with an example scenario will be performed.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Problemstellung

Künstliche Intelligenz ist ein in der heutigen Zeit immer wichtiger werdender Bereich, welcher für viele verschiedene Anwendungszwecke, wie zum Beispiel autonomes Fahren, Gesichtserkennung oder auch in der Medizin verwendet werden kann.
Die ständig zunehmenden Mengen an Daten führen dazu, dass mithilfe des Maschinellen Lernens bessere Modelle erstellt werden können. Um mit Maschinellem Lernen Modelle zu trainieren, ist es in den meisten Fällen notwendig einen großen beschrifteten Datensatz zu haben. Da man oftmals keinen fertig beschrifteten Datensatz finden wird, der genau auf seinen Anwendungszweck passt, oder dieser aus eigenen gesammelten Daten besteht, ist das Beschriften von Daten eine notwendige Aufgabe.

Die Aufgabe, diese Daten zu beschriften, muss zum großen Teil per Hand gemacht werden und kann je nach Größe des Datensatzes viel Zeit in Anspruch nehmen. Vor allem im Bereich der Bilderkennung, insbesondere Object detection, ist es aufwendig, für jedes Bild genaue Bounding Boxes, um alle Objekte zu ziehen. Außerdem hängt die Qualität des später trainierten Modells von der Qualität der Beschriftungen ab. So hätte zum Beispiel eine einzelne Person 19 Jahre gebraucht, um den ImageNet Datensatz mit über 14 Millionen Bildern alleine zu beschriften (Sager u. a., 2021).
Diesen Prozess zu optimieren und zu beschleunigen, kann viel Geld und Arbeitsaufwand sparen. Viele der vorhandenen Lösungen, die versuchen, bei diesem Problem zu helfen, sind aber noch nicht sonderlich ausgereift oder richten sich oft nur an sehr große Projekte und Unternehmen.

1.2 Zielsetzung

Das Ziel dieser Bachelorarbeit ist es, eine Anwendung mithilfe von Serverless Computing zu entwerfen und umzusetzen, die diesen Prozess wenigstens teilweise automatisieren kann und dabei hilft, schneller Datensätze von Bildern für Maschinelles Lernen vorzubereiten. Die Anwendung soll hierfür einen Assistenten bieten, welcher dabei hilft, Bilder für die Objekterkennung zu beschriften. Dazu sollen bereits vorhandene Modelle verwendet werden, um für jedes Bild Vorschläge zu generieren, wie die Bounding Boxes platziert sein könnten. Diese Vorschläge müssen dann nur noch manuell überprüft und wenn nötig angepasst werden. Um diese Anpassungen durchzuführen, können mehrere Personen gleichzeitig an einem Datensatz arbeiten. Des Weiteren soll es möglich sein, die beschrifteten Daten in einem gängigen Format zu exportieren, so dass diese direkt verwendet werden können.

1.3 Struktur der Arbeit

Im Kapitel 2 dieser Arbeit geht es zunächst um die Grundlagen, wie Begrifflichkeiten und Konzepte, die für die anderen Kapitel wichtig sind. Zudem werden dort bereits bestehende Lösungen untersucht und verglichen. Danach geht es in Kapitel 3 um die Analyse der Anforderungen an das zu entwickelnde System. Diese werden unter anderem in Form von User Stories und Anwendungsfällen dargestellt. Kapitel 4 zeigt den Entwurf der Architektur für das zu entwickelnde System, damit dieses alle Anforderungen erfüllen kann. In Kapitel 5 wird dann anhand dieses Entwurfes gezeigt, wie das System implementiert wurde. Um das ganze abzuschließen, gibt es in Kapitel 6 eine Evaluierung, ob die Anforderung erfüllt und die Anwendung bei der Problemstellung hilft.

2 Grundlagen

In diesem Kapitel geht es um die Grundlagen, also verschiedene Begrifflichkeiten und Konzepte, welche für den Rest der Arbeit wichtig sind. Dazu gehören die hier verwendeten Technologien, sowie die notwendigen Themen aus dem Bereich des Maschinellen Lernens, die für Objekterkennung benötigt werden. Zudem werden vorhandene Lösungen und Technologien in diesem Bereich untersucht und verglichen.

2.1 Serverless

Durch die stärker werdende Bewegung von der großen monolithischen Anwendung, hin zu eher kleineren Microservice Einheiten, wird auch das Thema Serverless Computing durchgehend relevanter. Trotz der relativen Neuheit von Serverless, ist es aktuell ein bei vielen beliebtes und schnell an Popularität wachsendes Konzept (Baldini u. a., 2017).
Erste Ideen, die sich mit dem Thema Serverless befassten, gab es in 2012, unter anderem gehörte dazu, das von iron.io ins Leben gerufene Produkt IronWorker. Bekannt geworden ist der Begriff Serverless aber erst durch Amazon, als diese 2014 ihr erstes Serverless Produkt, mit dem Namen Lambda, veröffentlicht haben. Lambda ist ein Functions as a Service Dienst (Owens, 2018).

Das Ziel von Serverless Computing ist es, den Fokus auf die Entwicklung der Anwendung zu bringen. Der Name soll nicht ausdrücken, dass es keine Server mehr gibt, sondern nur, dass die grundlegenden Server und Infrastrukturen nicht mehr selber verwaltet werden müssen. Um dies zu ermöglichen, wird auf Cloud Anbieter gesetzt, welche diese Aufgaben vollständig übernehmen (Owens, 2018). Abgerechnet werden Serverless Dienste normalerweise nach dem Pay-As-You-Go Prinzip, so dass nur für die tatsächlich verwendeten Ressourcen bezahlt werden muss. Der Hauptunterschied zu dem ähnlichen Konzept, Platform as a Service, ist die bessere und automatische Skalierung der Anwendung.

Es gibt verschiedene gebräuchliche Anwendungsfälle, bei welchen sich besonders die Nutzung von Serverless Computing anbietet. Allgemein vorteilhaft sind Aufgaben, die unregelmäßig ausgeführt werden und oder auf Ereignisse reagieren. Zu den beliebtesten Anwendungen von Serverless gehören unter anderem (Jonas u. a., 2019):

- Web APIs und Dienste.
- Datenverarbeitung und Transformation (ETL).
- Integration von Dittanbieterdiensten

Natürlich bringt die Nutzung von Serverless Computing auch einige Nachteile mit sich, bei denen es abzuwägen gilt, ob die Nutzung sinnvoll ist. Durch intensive Verwendung eines Cloud Anbieters, ist man sehr stark von deren Diensten abhängig. Die abgegebene Kontrolle kann positive Effekte haben, aber durchaus auch Probleme, in Form von verschiedenen Einschränkungen und Limitierungen der Plattform, mit sich bringen.
Jeder Anbieter hat diverse Produkte, die unterschiedlich funktionieren, so ist ein nachträglicher Wechsel des Anbieters möglicherweise nur schwer zu realisieren und mit viel Aufwand verbunden.
Grundlegend gibt es zwei Kategorien, wie Serverless verwendet werden kann: Backend as a Service und Functions as a Service (Roberts, 2016).
Backend as a Service (BaaS) beschreibt die Nutzung von Drittanbieterdiensten, um Logik und verschiedene Komponenten der Anwendung zu übernehmen. Hierzu gehören unter anderem Datenbanken und Authentifizierungsdienste, wie zum Beispiel Auth0.

2.1.1 Functions as a Service

Bei Functions as a Service (FaaS), wird die Anwendungslogik selber geschrieben und in kleine Funktionen unterteilt, wobei jede Funktion möglichst nur eine Aufgabe übernimmt. Es werden hier keine Server selber verwaltet, sondern der geschriebene Code wird einfach hochgeladen und der Cloud Anbieter übernimmt dann den Rest. Ausgelöst werden Funktionen meistens durch Ereignisse, dazu zählen hochgeladene Dateien, Veränderungen von Daten in einer Datenbank, aber auch eine Auslösung durch einen HTTP Aufruf ist möglich.
Diese Funktionen haben bestimmte Eigenschaften:

Zustandslos: Um die automatische Skalierung der Funktionen zu gewährleisten, dürfen diese keinen Zustand speichern. Benötigte Daten und Zustände müssen extern, zum Beispiel in eine Datenbank oder in einem Objektspeicher, gesichert werden. Durch das ständige hoch-und runterskalieren der Funktionen, würde jeder Zustand schnell wieder verloren gehen oder ungültig werden.
Skalierung: Funktionen werden automatisch, je nachdem wie viele Ressourcen benötigt werden, skaliert. Sollten also aktuell keine Ressourcen benötigt werden, dann gibt es auch keine laufenden Funktionen.
Ressourcen Limits: Die Ressourcen, die Funktionen zur Verfügung stehen, sind oftmals sehr begrenzt. Vor allem CPU und Arbeitsspeicher haben geringe Limits. Aber auch die maximale Ausführungszeit einer Funktion kann eine Einschränkung bei der Entwicklung sein. Sehr rechenintensive, langlebige Aufgaben sind eher nicht für FaaS geeignet.
Cold Starts: Cold Starts treten auf, wenn nicht genügend Funktionen verfügbar sind und eine neue Instanz erstellt werden muss. Dies kann vor allem bei unregelmäßigem Aufrufen zu langen Wartezeiten führen, bevor eine Funktion verwendet werden kann.

Kommunikation zwischen verschiedenen Funktionen kann entweder synchron oder asynchron erfolgen. Bei der synchronen Kommunikation ruft eine Funktion direkt eine andere auf und wartet dann auf eine Antwort. Bei der asynchronen Kommunikation wird von einer Funktion ein Ereignis erstellt, welches wiederum andere Funktionen auslösen kann. Synchrone Kommunikation ist vor allem im Serverless Bereich oftmals problematisch. Es ist sehr ungünstig, wenn Funktionen längere Wartezeiten haben, in denen sie untätig sind. Durch die nutzungsbasierte Abrechnung werden dann während der Wartezeit zwei Funktionen berechnet, obwohl nur eine arbeitet (Kratzke, 2021).

2.2 Google Cloud

Google Cloud gehört mit Amazon Web Services und Microsoft Azure zu den größten Cloud Computing Anbietern, die es gibt. In der Google Cloud werden eine Vielzahl an verschiedenen Produkten für die unterschiedlichsten Anwendungszwecke angeboten. Zu diesen Dienste gehören unter anderem Infrastruktur, Datenbanken und Softwarelösungen.

2.2.1 Firestore

Bei traditionellen nicht Serverless Datenbanken gibt es festgelegte verfügbare Ressourcen, die manuell hoch- und runterskaliert werden müssen.
Serverless Datenbanken übernehmen diesen Prozess für einen. Der benötigte Speicher und die benötigte Abfragekapazität wird automatisch verwaltet. In einem System, in dem der Großteil der Dienste Serverless sind, ist es wichtig darauf zu achten, dass, zum Beispiel, automatisch skalierende Funktionen nicht aus Versehen weniger gut skalierende Teile des Systems lahmlegen.

Google Cloud Firestore bietet eine solche Lösung an. Firestore ist eine Serverless NoSQL Datenbank, dass bedeutet, sie basiert wie der Großteil der Serverless Angebote auf dem Pay-As-You-Go Prinzip. Dadurch skaliert die Datenbank automatisch mit, je nachdem, wie viel Speicherplatz und Anfragen benötigt werden. In Firestore werden Daten in Form von Dokumenten gespeichert, diese Dokumente sind einfache Key/Value Paare. Durch Sammlungen können mehrere Dokumente zusammen organisiert werden, wobei jedes Dokument einer Sammlung angehören muss (Google, 2022e).

2.2.2 PubSub

Google Cloud PubSub ist ein Messaging Dienst für die asynchrone Kommunikation zwischen Anwendungen.
PubSub besteht grundlegend aus dem Publisher und dem Subscriber. Der Publisher hat die Möglichkeit, Nachrichten in ein bestimmtes Thema zu schreiben. Diese Nachrichten können dann von Subscribern über sogenannte Abonnements gelesen und verarbeitet werden. Dies ermöglicht eine robustere asynchrone Kommunikation zwischen Diensten, bei welcher nicht auf Nachrichten gewartet werden muss, sondern auf diese als Ereignisse reagiert werden kann (Google, 2022c).

2.2.3 Storage

Google Cloud Storage ist ein skalierbarer Objektspeicher, mit welchem beliebige Daten in beliebiger Größe gespeichert werden können.
Die Daten, die in Cloud Storage gespeichert werden, heißen Objekte, diese Objekte sind Dateien mit einem beliebigen Format. Des Weiteren gibt es Buckets, welche als Container

dienen, um Objekte in ihnen zu speichern.
Innerhalb der Buckets kann es eine verzeichnisartige Struktur geben, mit welcher die einzelnen Objekte geordnet werden können.
Je nachdem, für welchen Anwendungszweck Cloud Storage verwendet wird, gibt es unterschiedliche Speicherklassen, welche Vor-und Nachteile mit sich bringen. Die Speicherklassen bestimmen den Preis, welcher für den verwendeten Speicherplatz, sowie verschiedene Operationen anfällt. Bei den Speicherklassen, mit einem niedrigen Preis pro GB, gibt es eine Mindestaufbewahrungszeit und höhere Kosten, wenn Objekte frühzeitig heruntergeladen werden.
Ein weiteres wichtiges Feature von Cloud Storage ist die Objektversionsverwaltung, mit welcher verschiedene Versionen eines Objektes gleichzeitig aufbewahrt und verwaltet werden können. Zudem ist es möglich, mit Lebenszyklen Regeln festzulegen, wann Objekte automatisch gelöscht werden sollen (Google, 2022d).

2.2.4 Identity Platform

Eine sichere Möglichkeit für Nutzer, sich in seinem System zu registrieren und anzumelden, ist heutzutage die Grundlage einer fast jeden Anwendung. Dabei gibt es unzählige Dinge, die bei der Umsetzung einer solchen Nutzerverwaltung beachtet werden müssen.
Die Google Cloud Identity Platform übernimmt diese Aufgabe und ermöglicht eine vollständig verwaltete Identitäts- und Zugriffsverwaltung für die eigenen Anwendungen. Identity Platform übernimmt dabei alle Aufgaben, die man von einer Nutzerverwaltung erwarten würde. Dazu gehört das Registrieren von neuen Nutzern, sowie das Anmelden bestehender Nutzer. Identity Platform bietet Registrierung über E-Mail und Passwort und über Identity Provider, wie zum Beispiel Google, Twitter und Facebook (Google, 2022b).

2.2.5 API Gateway

Google Cloud API Gateway ist ein Dienst, zum Verwalten und Bereitstellen von APIs. Mit dem API Gateway können eingehende HTTP Anfragen einfach an Serverless Google Cloud Backends, wie zum Beispiel Functions, weitergeleitet werden.
API Gateways sind ein wichtiger Bestandteil für Serverless Architekturen, denn durch sie gibt es einen zentralen Zugangspunkt zu den verschiedenen Backend Diensten.Dies

ermöglicht zudem, dass alle Endpunkte zusammen an einem Ort überwacht und analysiert werden können.
Dabei hilft das API Gateway auch bei der Sicherheit, indem es nur authentifizierte Anfragen zu den Endpunkten durchlässt. Die mitgesendeten Anmeldeinformationen, werden dabei direkt validiert und überprüft, so dass diese allgemeine Authentifizierungslogik nicht in jedem Backend durchgeführt werden muss. API Gateway unterstützt für die Authentifizierung unter anderem auch Identity Platform. Des Weiteren kann das API Gateway die Backends auch durch Rate Limits vor Überlastung schützen, so werden nur eine bestimmte Anzahl an Anfragen nach verschiedenen Kriterien zugelassen (Google, 2022a).
Die API Konfigurationen werden mit der OpenAPI 2.0-Spezifikation erstellt. Mithilfe von dieser OpenAPI Spezifikation lassen sich REST APIs in einem gängigen Format beschreiben.

2.3 Maschinelles Lernen

Die Grundlage des Maschinellen Lernens ist es, durch verschiedene Arten von Trainingsvorgängen, aus Daten und Erfahrungen, zu lernen.
Aus dem Ergebnis eines solchen Trainingsvorgangs können dann Modelle entstehen, welche dazu verwendet werden können, um Muster in Datensätzen zu finden und Vorhersagen zu treffen. Grundsätzlich gibt es zwei Möglichkeiten, den Lernvorgang zu gestalten: überwachtes Lernen und unüberwachtes Lernen.
Bei dem überwachten Lernen sind im Gegensatz zu dem unüberwachten Lernen, sowohl die Eingabe- als auch die Ausgabedaten bekannt und werden beide für das Training verwendet (Otte, 2019, S. 230). Das heißt, die Ergebnisse der Vorhersage, müssen bei dieser Lernmethode vorher bekannt sein. Der gewählte Lernalgorithmus versucht dann, mit diesen Daten, Zusammenhänge und Muster zwischen der Ein- und Ausgabe zu erlernen. Die für diese Arbeit wichtige und relevante Methode ist das überwachte Lernen.

2.3.1 Object detection

Object detection hat heutzutage eine große Anzahl an Anwendungsmöglichkeiten und wird für viele Aufgaben in der Künstlichen Intelligenz benötigt. Egal, ob für das auto-

nome Fahren, Videoüberwachung oder Ähnliches, das Erkennen von Objekten, wie zum Beispiel Personen, Autos und Straßenschilder auf Bildern, ist wichtiger Bestandteil der Computer Vision.
Bei der Object detection ist das Ziel, Objekte auf einem Bild zu erkennen und zu lokalisieren. Objekte werden im Normalfall durch rechteckige Bounding Boxes beschrieben. Diese definieren, wo sich ein Objekt befindet und wie groß dieses ist. Zusätzlich ist dieser Bounding Box eine Klasse zugeordnet, die aussagt, um welche Art von Objekt es sich handelt.
Um ein Neuronales Netzwerk für die Object detection zu trainieren, wird das Überwachte Lernen verwendet. Dazu werden zuerst beschriftete Bilder benötigt. Auf diesen müssen manuell die Bounding Boxes eingetragen werden. Diese Bounging Boxes, bestehen aus den Koordinaten, also Eckpunkten, der Rechtecke, sowie der Klasse, dargestellt als Zahl. Mit diesen beschrifteten Daten wird dann ein sogenanntes Convolutional Neural Network (CNN) trainiert. Als Eingabe werden die Bilder verwendet und als Ausgabe die Beschriftungen, wie die Vorhersagen aussehen sollen.

Convulutional Neural Networks sind eine Art von Neuronalem Netzwerk, welches oft Verwendung bei der Objekterkennung in Bilder findet. Grundsätzlich besteht dieses aus drei Teilen: Convolutional Layer, Pooling Layer und Fully Connected Layer. Bei den Convolutional Layern, wird das Lernen über einen Filter durchgeführt, welcher über die Eingabe geht und sogenannte Feature Maps erstellt. Durch Pooling kann dann im nächsten Schritt die Größe verringert werden, bzw. irrelevante Features entfernt werden. Diese Anordnung von verschiedenen Convolutional Layern und Pooling Layern, werden bei den meisten Netzwerken mehrmals hintereinander wiederholt. Am Ende gibt es üblicherweise noch ein Fully Connected Layer (O'Shea und Nash, 2015).

2.3.2 YOLO

YOLO steht für You Only Look Once und ist ein Algorithmus für die Objekterkennung. Besonders gut ist dieser geeignet für die Object Detection in Echtzeit. YOLO gehört zu den sogenannten One-Stage Detectors, bei denen im Gegensatz zu Two-Stage Detectors nicht zuerst das Bild in mehrere Regionen aufgeteilt wird, um diese dann zu klassifizieren. Bei One-Stage Detectors werden die Bounding Boxes und Klassen in einem Durchlauf mit dem Bild vorhergesagt, dies sorgt dafür, dass die Geschwindigkeit dieser Algorithmen erhöht ist (Redmon u. a., 2015).

2.4 Bestehende Lösungen

Durch die Bedeutsamkeit von Künstlicher Intelligenz in den letzten Jahren, ist auch das Thema der Beschriftung von Bildern relevanter geworden. Die Anzahl an Softwarelösungen und wissenschaftlichen Arbeiten ist in diesem Bereich stark gestiegen. In diesem Abschnitt werden daher nur einige der bereits bestehenden Lösungen, für das Beschriften von Bilder, untersucht.

2.4.1 LabelMe

LabelMe ist eines der ältesten und bekanntesten Werkzeuge, um Bilder online zu beschriften. Es bietet eine einfache grafische Oberfläche, um mithilfe von Rechtecken und Polygonen Beschriftungen auf Bildern hinzufügen zu können. Das Ziel dieses Projekts war es, eine einheitliche Anwendung zu bieten, die auf jeder Plattform funktioniert und mit welcher beschriftete Datensätze geteilt werden können (Russell u. a., 2007).

2.4.2 Roboflow

Roboflow ist eine Plattform, die eine Vielzahl an Tools für Computer Vision bietet. Aktuell hat diese den wahrscheinlich größten Funktionsumfang, der im Laufe dieser Arbeit untersuchten Lösungen.
Auch hier wird unter anderem die Idee verfolgt, bereits trainierte Modelle zu verwenden, um das Beschriften zu beschleunigen. Die durch das Modell erzeugten Vorhersagen sind aber oftmals noch unvollständig und benötigen viele Anpassungen. Zudem bietet Roboflow neben der kostenlosen Version nur Angebote, ab einem Festpreis von 1000 Euro monatlich und aufwärts, an (Roboflow, 2022).

2.4.3 CVAT

CVAT (Computer Vision Annotation Tool) ist eine Open Source Software, welche unter anderem auch für das Beschriften von Bildern verwendet werden kann. Zusätzlich ist es bei dieser Software auch möglich, mit bestimmten Modellen, Vorschläge die bei der Beschriftung helfen sollen, zu generieren. Normalerweise muss diese manuell auf eigener Hardware installiert werden, es gibt aber auch als Online Version, eine Demo, mit welcher CVAT ausprobiert werden kann (OpenVINO, 2022).

2.4.4 Verwandte Arbeiten

Die Idee den Vorgang des Beschriftens zu teilautomatisieren ist nicht neu und es gibt einige Arbeiten, die sich damit beschäftigen, wie eine Zusammenarbeit von Mensch und KI hierbei behilflich sein kann. Bereits in der Veröffentlichung zu LabelMe gab es den Vorschlag, eine KI, Vorhersagen auf Bildern treffen zu lassen, die dann nur noch angepasst werden müssen (Russell u. a., 2007).
In Haider und Michahelles (2021) und Desmond u. a. (2021) wird untersucht, wie das Aufzeigen von KI generierten Vorschlägen, die Beschriftung von Daten optimieren kann. Beide Arbeiten kommen zu dem Ergebnis, dass sowohl die Genauigkeit, als auch die Geschwindigkeit, durch die Verwendung von vorbeschrifteten Daten, verbessert wird. Auch Zhang u. a. (2008) zeigt diese Art von Verbesserung, unter der Verwendung von Active Learning.

3 Anforderungen

In diesem Kapitel wird eine Anforderungsanalyse des zu entwickelnden Systems durchgeführt. Dazu gehören die funktionalen und nichtfunktionalen Anforderungen, gefolgt von den User Stories und einer Beschreibung der wichtigsten Anwendungsfälle im System.

3.1 Stakeholder

> Ein Stakeholder eines Systems ist eine Person oder Organisation, die (direkt oder indirekt) Einfluss auf die Anforderungen des betrachteten Systems hat. (Pohl und Rupp, 2015)

Bevor die genaueren Anforderungen an das System festgelegt werden, ist es wichtig einen Überblick zu haben, welche Personen oder Organisationen die Anwendung später verwenden werden oder ein Interesse an dieser haben. Zudem muss herausgefunden werden, welche Wünsche und Erwartungen bei den einzelnen Stakeholdern beachtet werden müssen.

Die hier herausgearbeiteten Stakeholder sind hauptsächlich die späteren Kunden der Anwendung. Die Kunden können sowohl Organisationen sein, welche ihre Datensätze beschriften wollen, als auch einzelne Personen, wie zum Beispiel Studenten, die für ein Projekt eigene Daten beschriften müssen. Um zu verstehen, was die Kunden von der Anwendung erwarten, wurden diese je nach Interesse in Gruppen unterteilt, wobei eine Person durchaus alle Interessen vertreten kann, wenn diese alleine an einem Datensatz arbeitet.

Datensatzverwalter
Datensatzverwalter sind Personen oder Organisationen, welche einen Datensatz erstellen

und verwalten. Diese Stakeholder müssen nicht selbst an der Beschriftung teilnehmen, können aber jederzeit Mitarbeiter hinzufügen. Sie haben also nicht unbedingt ein so umfangreiches Wissen im Bereich Maschinelles Lernen.

Mitarbeiter
Mitarbeiter sind Personen, die an der Beschriftung eines Datensatzes mitwirken. Sie können eine grafische Oberfläche nutzen, um sich die Bilder und Beschriftungen anzuschauen und diese anzupassen. Für die Mitarbeiter ist es wichtig, eine übersichtliche und einfache Möglichkeit zu haben, diese Aufgaben durchzuführen. Mitarbeiter haben mindestens Grundkenntnisse über Objekterkennung und worauf bei der Beschriftung von Daten geachtet werden muss.

Data Scientist
Data Scientist sind Personen, die den beschrifteten Datensatz für Maschinelles Lernen verwenden wollen, um damit ein Modell für die Objekterkennung zu trainieren. Sie können auch an der Beschriftung teilnehmen, aber ihr Hauptinteresse liegt an der Verwendung des Datensatzes. Ihr Wissen im Bereich Maschinelles Lernen und Objekterkennung ist fortgeschritten. Sie wissen, wie Modelle mit den Daten trainiert werden können. Mithilfe der Exportierungsfunktion können Data Scientists den Datensatz einfach herunterladen. Wichtig dabei ist, dass der Export ein gängiges Format hat und somit direkt von entsprechenden Machine Learning Frameworks, wie zum Beispiel Tensorflow, verwendet werden kann.

3.2 Funktionale Anforderungen

Funktionale Anforderungen werden verwendet, um festzulegen, welche Funktionalität das zu entwickelnde System besitzen soll. Sie sind die Eigenschaften, die zur Lösung des Problems und somit zum Erreichen des Ziels benötigt werden (Pohl und Rupp, 2015).

Damit die Anwendung gut nutzbar ist und ihren Zweck erfüllt, müssen einige Funktionalitäten gegeben sein. Eine Person kann sich in der Anwendung anmelden. Nach der Anmeldung gibt es die Möglichkeit, einen neuen Datensatz zu erstellen. Für diesen Datensatz können dann neue Bilder hochgeladen werden. Wenn diese Bilder fertig hochgeladen

und verarbeitet sind, werden sie in dem Datensatz angezeigt und können beschriftet werden. Für die Beschriftung der Bilder kann ein Nutzer neue Bounding Boxes auf dem Bild zeichnen oder vorhandene anpassen und löschen. Nachdem alle Bilder fertig beschriftet wurden, kann der Datensatz exportiert und die exportierte Version heruntergeladen werden.

Aus dem diesem Anwendungsszenario ergeben sich folgende funktionale Anforderungen.

Anforderung 1 Es soll die Möglichkeit bestehen, sich mit einer Email Adresse und einem Passwort bei der Anwendung anzumelden

Anforderung 2 Angemeldete Benutzer sollen die Möglichkeit haben, neue Datensätze zu erstellen

Anforderung 3 Für einen bestimmten Datensatz sollen neue Bilder hinzugefügt und hochgeladen werden

Anforderung 4 Mehrere Personen sollen an einem Datensatz zusammen arbeiten können

Anforderung 5 Hochgeladene Bilder sollen automatisch gespeichert werden

Anforderung 6 Nach dem Hochladen sollen Bilder automatisch von einem bereits trainierten Modell beschriftet werden

Anforderung 7 Die Beschriftungen der Bilder sollen in einer Datenbank gespeichert werden

Anforderung 8 Die Datensätze sollen in einer Datenbank gespeichert werden

Anforderung 9 Die Beschriftungen der Bilder sollen überprüft und angepasst werden können

Anforderung 10 Die Datensätze und beschrifteten Bilder sollen auf einer Grafischen Oberfläche zu sehen sein

Anforderung 11 Fertig beschriftete Datensätze sollen exportiert werden können

Anforderung 12 Die exportierten Versionen der Datensätze sollen gespeichert werden

Anforderung 13 Diese exportierten Versionen sollen dann heruntergeladen werden können

3.3 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen sind unter anderem Qualitätsanforderungen an das System, die nicht direkt etwas mit der Funktionalität zu tun haben. Hier werden Qualitätsmerkmale, die das System besitzen soll, festgelegt (Pohl und Rupp, 2015).

1. **Sicherheit**
   Sicherheit ist eine wichtige Grundlage für das System. Die Daten der Nutzer sollten vor unautorisiertem Zugriff geschützt sein. Zudem werden viele Bilder hochgeladen und gespeichert. Auch bei diesen muss sichergestellt werden, dass dort kein ungewollter Zugriff stattfindet.

2. **Verfügbarkeit**
   Das System sollte eine hohe Verfügbarkeit haben. Damit die Anwendung nutzbar ist, ist es wichtig, dass alle Dienste jederzeit erreichbar sind. Mit einer zu geringen Verfügbarkeit bringt auch die beste Funktionalität nichts, denn, wenn das System sehr häufige Ausfälle hat, erfüllt es keinen seiner Zwecke.

3. **Skalierbarkeit**
   Das System soll sich an die benötigte Leistung anpassen können, je nachdem, ob gerade viele oder wenig Nutzer aktiv sind. Zudem soll es in der Lage sein, Leistungsspitzen, also wenn auf einmal viele Nutzer gleichzeitig aktiv sind, zu unterstützen. Natürlich ist es auch umgekehrt wichtig, dass im Falle einer geringen Benutzung keine unnötigen Ressourcen bereitstehen und bezahlt werden.

4. **Benutzbarkeit**
   Die Benutzbarkeit eines Systems ist wichtig und sollte bei dem Entwurf und bei der Entwicklung nicht aus dem Auge verloren gehen. Die Bedienung der Anwendung soll einfach und unkompliziert möglich sein. Im Frontend sollte darauf geachtet werden, dass die Nutzung ohne Vorkenntnisse der Anwendung möglich ist, lediglich die Grundlagen von Maschinellem Lernen und Objekterkennung sind für eine sinnvolle Bedienung vorausgesetzt.

3.4 User Stories

User Stories zeigen konkret einzelne Anforderungen und Funktionen auf, die von Stakeholdern bei der Anwendung erwartet werden.
Geschrieben werden diese User Stories aus der Sicht der Stakeholder und zeigen auf, welche Aufgabe dieser ausführen möchte.
Sie haben das Format: *Als Stakeholder möchte ich Funktion.*

Allgemein

- Als Nutzer möchte ich mich anmelden können
- Als Nutzer möchte ich neue Datensätze erstellen können
- Als Nutzer möchte ich meine Datensätze sehen können

Datensatzverwalter

- Als Datensatzverwalter möchte ich meine Datensätze löschen können
- Als Datensatzverwalter möchte ich meine Datensätze bearbeiten können
- Als Datensatzverwalter möchte ich neue Bilder zu meinem Datensatz hinzufügen können
- Als Datensatzverwalter möchte ich andere Nutzer zu meinem Datensatz hinzufügen können
- Als Datensatzverwalter möchte ich andere Nutzer von meinem Datensatz entfernen können.
- Als Datensatzverwalter möchte ich sehen können, welche Nutzer dem Datensatz hinzugefügt sind

Mitarbeiter

- Als Mitarbeiter möchte ich mir die Bilder in einem Datensatz ansehen können
- Als Mitarbeiter möchte ich mir die Beschriftungen in einem Datensatz ansehen können
- Als Mitarbeiter möchte ich Beschriftungen bearbeiten können

- Als Mitarbeiter möchte ich Beschriftungen hinzufügen können
- Als Mitarbeiter möchte ich Beschriftungen löschen können
- Als Mitarbeiter möchte ich neue Bilder hinzufügen können

Data Scientist

- Als Data Scientist möchte ich einen Datensatz exportieren können
- Als Data Scientist möchte ich einen exportierten Datensatz herunterladen können

3.5 Anwendungsfälle

Anwendungsfälle oder auch Use Cases genannt beschreiben beispielhafte Abläufe der Nutzung des Systems. (Pohl und Rupp, 2015, S. 63)
Dabei wird beschrieben, wie das System und die Akteure sich unter verschiedenen Bedingungen verhalten. Dargestellt werden diese Anwendungsfälle in Anwendungsfalldiagrammen oder in textuellen Anwendungsfallbeschreibungen. Hier werden die wichtigsten Anwendungsfälle der Anwendung dargestellt und beschrieben.

3.5.1 Anwendungsfalldiagramm

Abbildung 3.1 zeigt das Anwendungsfalldiagramm.

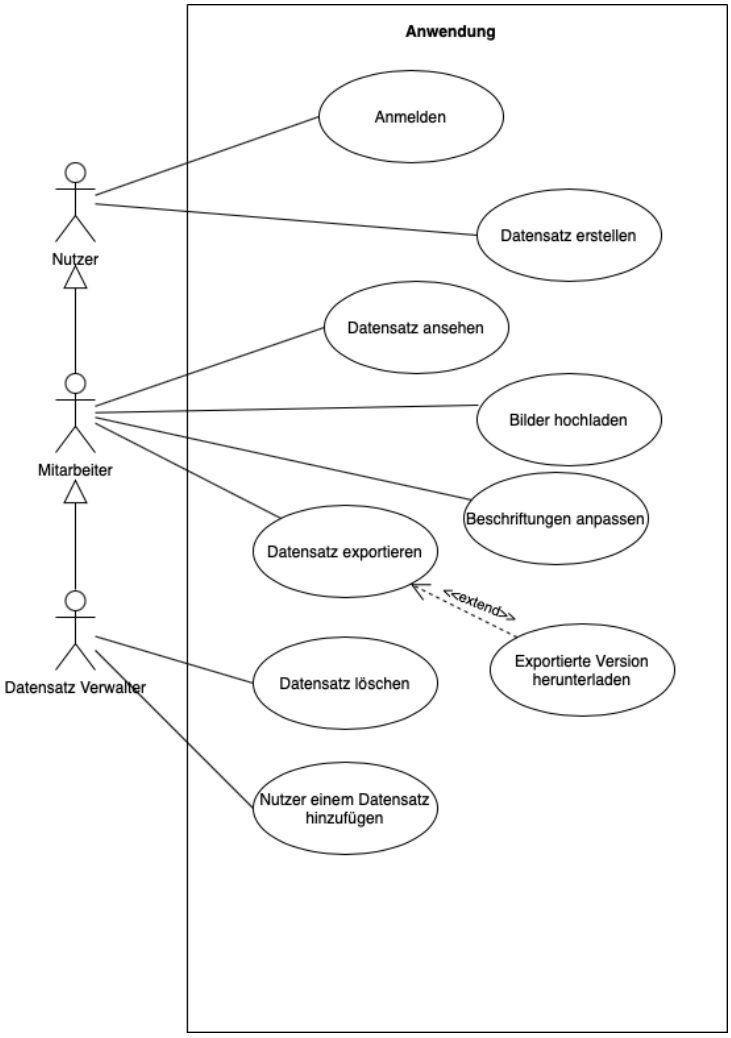


Abbildung 3.1: Anwendungsfalldiagramm

3.5.2 Use Cases

Use-Case 1: Datensatz erstellen
Akteure:

- Nutzer

Beschreibung:
Ein Nutzer erstellt einen neuen Datensatz, um dann dort Bilder beschriften zu können
Vorbedingung:

- Der Nutzer ist im System angemeldet

Nachbedingung:

- Ein neuer Datensatz wurde angelegt

Hauptszenario:

1. Datensatz erstellen wird ausgewählt
2. Die Informationen für den neuen Datensatz werden eingegeben
3. Die Daten für den neuen Datensatz werden übermittelt und validiert
4. Der neue Datensatz wird erstellt
5. Der neue Datensatz wird angezeigt

Fehlerfälle:
3a Die Validierung ist fehlgeschlagen oder der Datensatz existiert

- Die Erstellung wird abgebrochen und der Fehler angezeigt

Use-Case 2: Datensatz ansehen
Akteure:

- Datensatzverwalter
- Mitarbeiter

Beschreibung:
Ein Nutzer schaut sich einen seiner Datensätze oder einen Datensatz, bei welchem er mitarbeitet an.
Vorbedingung:

- Der Nutzer ist im System angemeldet
- Der Datensatz existiert

Nachbedingung:

- Der Datensatz wurde angezeigt

Hauptszenario:

1. Es wird ein Datensatz ausgewählt, der angeschaut werden möchte

2. Die Anfrage wird vom System überprüft
3. Der Datensatz wird angezeigt

Use-Case 3: Datensatz exportieren
Akteure:

- Datensatzverwalter
- Mitarbeiter

Beschreibung:
Ein Nutzer exportiert einen fertigen Datensatz und lädt ihn dann herunter
Vorbedingung:

- Der Nutzer ist im System angemeldet
- Der Datensatz existiert
- Der Datensatz ist fertig beschriftet

Nachbedingung:

- Der exportierte Datensatz kann heruntergeladen werden

Hauptszenario:

1. Es wird ein Datensatz ausgewählt, der exportiert werden soll.
2. Datensatz exportieren wird ausgewählt und die Anfrage wird an das System übermittelt
3. Das System sammelt alle Bilder und Beschriftungen und startet die Exportgenerierung
4. Aus dem gesammelten Daten erzeugt das System ein neues Format und speichert dieses
5. Der Status des Exportvorgangs wird angezeigt

Fehlerfälle:
3a. Bei der Generierung der Exportversion ist ein Fehler unterlaufen

- Die Erstellung wird abgebrochen

Use-Case 4: Nutzer hinzufügen
Akteure:

- Datensatzverwalter

Beschreibung:
Ein Datensatzverwalter fügt einen anderen Nutzer als Mitarbeiter zu einem seiner Datensätze hinzu
Vorbedingung:

- Der Datensatzverwalter ist im System angemeldet
- Der Datensatz existiert
- Der neue Nutzer existiert

Nachbedingung:

- Der neue Mitarbeiter hat Zugriff auf den Datensatz

Hauptszenario:

1. Es wird ein Datensatz ausgewählt
2. Die E-Mail Adresse des neuen Nutzers wird angegeben
3. Das System validiert die Anfrage
4. Der neue Nutzer wird als Mitarbeiter hinzugefügt und bekommt die notwendigen Zugriffsrechte

Fehlerfälle:
3a. Der Nutzer existiert nicht oder ist bereits Mitarbeiter

- Der Vorgang wird abgebrochen

Use-Case 5: Anmelden
Akteure:

- Nutzer

Beschreibung:
Ein Nutzer meldet sich im System an
Nachbedingung:

- Der Nutzer ist nun im System angemeldet

Hauptszenario:

1. Es wird eine Anmeldemethode ausgewählt
2. Die Schritte der jeweiligen Anmeldemethode werden ausgeführt
3. Die Daten werden an den Authentifiezierungsdienst übermittelt
4. Der Nutzer wird im System angemeldet

Fehlerfälle:
3a. Die Anmeldung ist fehlgeschlagen

- Die Anmeldung wird abgebrochen

Use-Case 6: Bild hochladen
Akteure:

- Datensatzverwalter
- Mitarbeiter

Beschreibung:
Ein Nutzer fügt ein oder mehrere Bilder zu einem Datensatz hinzu
Vorbedingung:

- Der Nutzer ist im System angemeldet
- Der Datensatz existiert

Nachbedingung:

- Das Bild wurde erfolgreich hochgeladen und beschriftet

Hauptszenario:

1. Es werden ein oder mehrere Bilder ausgewählt, die hochgeladen werden sollen
2. Die Bilder werden zu dem System hochgeladen
3. Das System verarbeitet und speichert die Bilder
4. Für jedes hochgeladene Bild wird automatisch eine vorläufige Beschriftung generiert und gespeichert

Fehlerfälle:
3a. Die Bilder konnten nicht gespeichert werden oder hatten ein falsches Format

- Der Hochladevorgang wird abgebrochen

4a. Bei der Erstellung der Beschriftungen ist ein Fehler aufgetreten

- Der Beschriftungsvorgang für dieses Bild wird abgebrochen

Use-Case 7: Bild beschriften
Akteure:

- Datensatzverwalter
- Mitarbeiter

Beschreibung:
Ein Nutzer passt die Beschriftungen von einem hochgeladenen Bild an
Vorbedingung:

- Der Nutzer ist im System angemeldet
- Der Datensatz existiert

Nachbedingung:

- Die neue Beschriftung für das Bild wurde gespeichert

Hauptszenario:

1. Es wird ein Bild ausgewählt
2. Es werden vorhandene Bounding Boxes auf dem Bild angepasst oder neue hinzugefügt
3. Wenn die Bounding Box fertig bearbeitet ist, wird sie vom System gespeichert

3.6 Datenmodell

Das Datenmodell zeigt unabhängig der verwendeten Datenbank alle zu dem System gehörigen Entitäten und wie diese zueinander stehen.

User sind die Benutzer des Systems und haben mindestens eine Email und eine ID.
Datensätze werden im System durch die Entität **Dataset** dargestellt. **User** können diese Datensätze erstellen oder als Mitglied hinzugefügt werden. Zudem kann ein Datensatz mehrere Klassen besitzen. Diese beschreiben, welche Objekte auf den Bildern gefunden werden können.
Image sind die Bilder, die zu einem Datensatz gehören. Diese können wiederum verschiedene Beschriftungen haben.
Label beinhalten Informationen über die konkrete BoundingBox des Objekts und den zugehörigen Namen der Klasse.
Export besitzt eine ID und einen Status und dient dazu, die exportierten Versionen eines Datensatzes abzubilden.

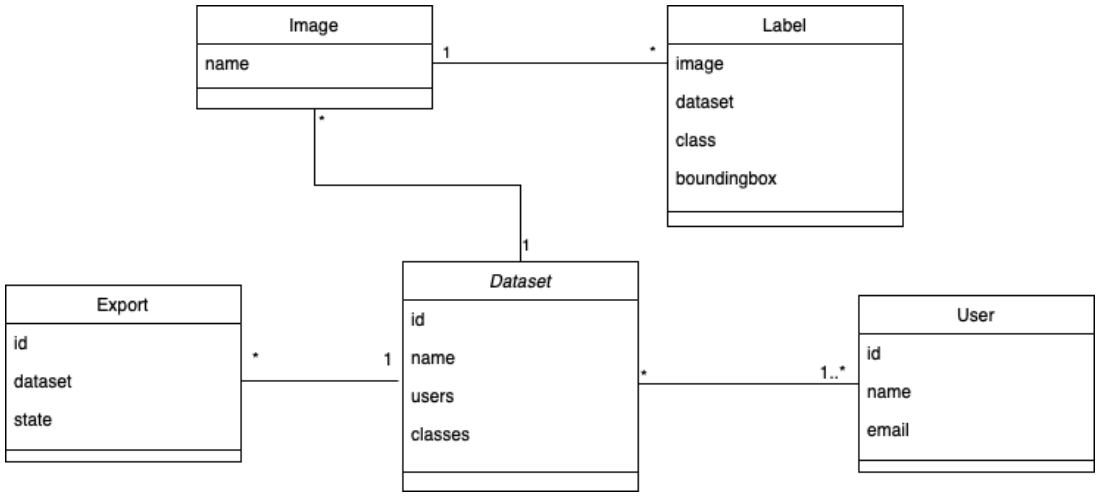


Abbildung 3.2: Datenmodell des Systems

4 Entwurf

In diesem Kapitel geht es um die Architektur des Systems. Um diese zu veranschaulichen, werden Architektursichten verwendet. Außerdem werden hier wichtige Technologieentscheidungen erläutert.

4.1 Architektursichten

Eine einzelne Sicht auf ein komplexes System reicht oftmals nicht aus, um dieses vollständig im Detail zu beschreiben. Außerdem ist nicht jede Person an den gleichen Informationen interessiert. Aus diesem Grund werden verschiedene Sichten verwendet, um verschiedene Perspektiven und Informationsbedürfnisse der Architektur abzudecken.
Die vier wichtigsten Sichten sind: Kontextabgrenzung, Bausteinsicht, Laufzeitsicht und Verteilungssicht.

4.1.1 Kontextabgreznung

Kontextabgrenzung ist die gröbste Sicht auf das System. Sie zeigt eine Übersicht des Systems über den Zusammenhang mit seinem Umfeld. Dazu gehören die Schnittstellen zu den Nachbarsystemen, Interaktionen mit Stakeholdern sowie die wesentliche umgebende Infrastruktur (Starke, 2020, S. 156).

Auf Abbildung 4.1 ist diese Übersicht mit der Anwendung und dem externen Authentifizierungsdienst zu sehen.

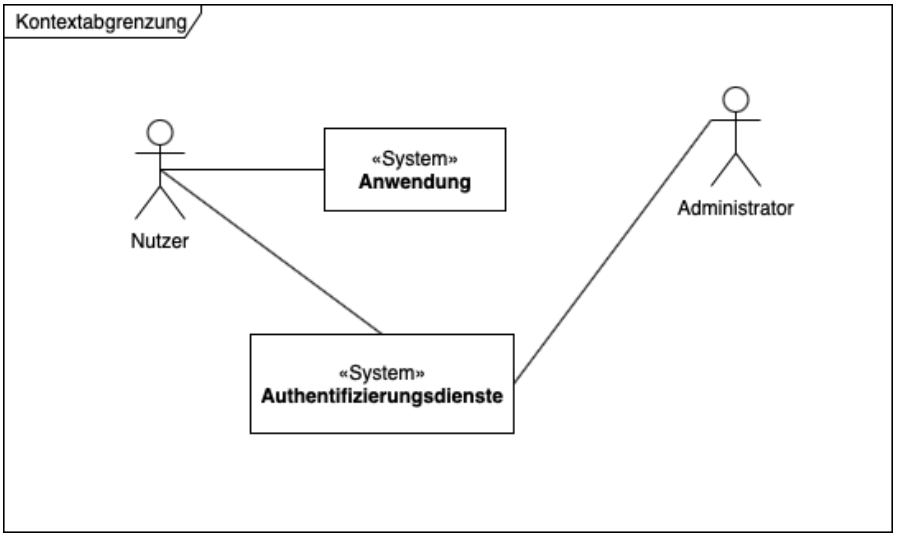


Abbildung 4.1: Kontextabgreznung

4.1.2 Bausteinsicht

In der Bausteinsicht wird genauer darauf eingegangen, wie das eigene System im Detail strukturiert und aufgebaut ist. In dieser Sicht werden alle Bausteine und Komponenten des Systems beschrieben, sowie deren Schnittstellen und Abhängigkeiten.
Auf der Abbildung 4.2 ist eine detailliertere Sicht auf die Anwendung aus der Kontextabgrenzung zu sehen. Es ist dort der grundlegende Aufbau der Backend Architektur der Anwendung, in Form von verschiedenen Komponenten, die miteinander interagieren, zu sehen. Zudem sind externe Schnittstellen zu Datenbanken, Datenspeicherung und Authentifizierungsdiensten sichtbar.

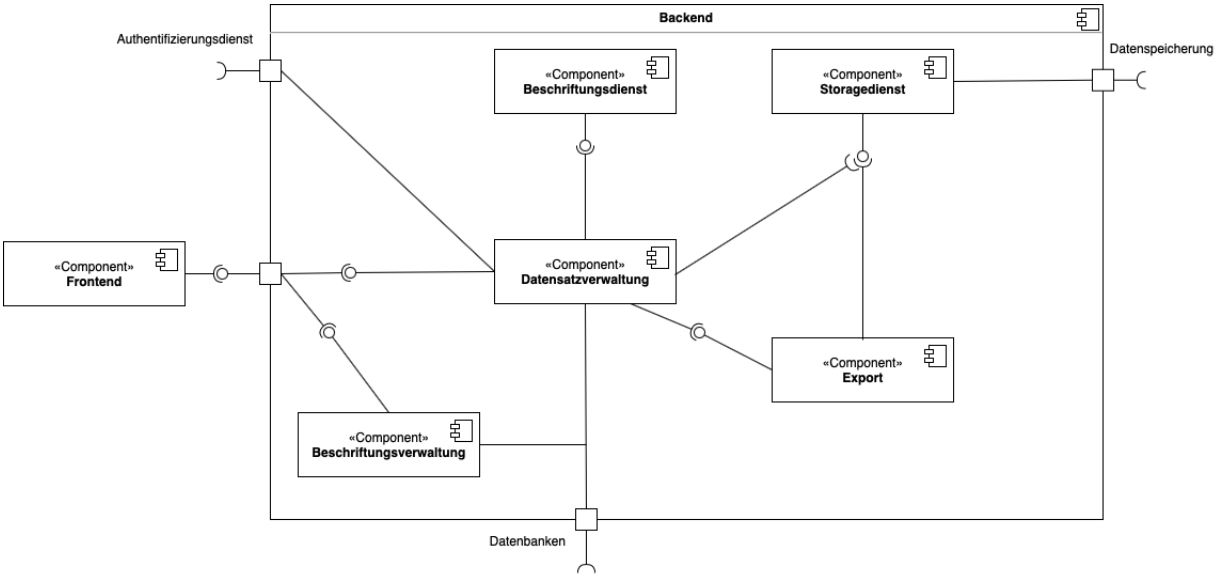


Abbildung 4.2: Bausteinsicht

4.1.3 Google Architektur

Das Google Cloud Architektur Diagramm zeigt, wie die Infrastruktur der Anwendung aufgebaut ist und wo die einzelnen Bausteine verteilt sind. Zudem ist dort zu sehen, welche Dienste verwendet werden, wo Daten gespeichert werden und wie die Kommunikation abläuft.

Die Abbildung 4.3 zeigt eine Übersicht des Systems mit einer Serverless Architektur auf der Google Cloud und wie der Nutzer damit interagieren kann. Den Einstiegspunkt in das System stellt das API Gateway dar. Dieses ist vor den HTTP Funktionen und leitet die eingehenden Anfragen von einem Nutzer intern an die richtige Funktion weiter. Das API Gateway übernimmt außerdem auch die Aufgabe der Authentifizierung der Nutzer, indem es eingehende Anfragen bei der Identity Platform verifiziert. Anmeldungen und Registrierungen laufen über die Identity Platform. Damit ein Nutzer eine authentifizierte Anfrage an das API Gateway senden kann, muss sich dieser zuerst bei der Identity Platform anmelden. Diese speichert alle Nutzer der Anwendung und stellt bei erfolgreicher Anmeldung ein Token aus. Mithilfe von diesem kann dann eine Anfrage an das API Gateway gesendet werden, so ist sichergestellt, dass nur angemeldete Nutzer auf das System zugreifen können.

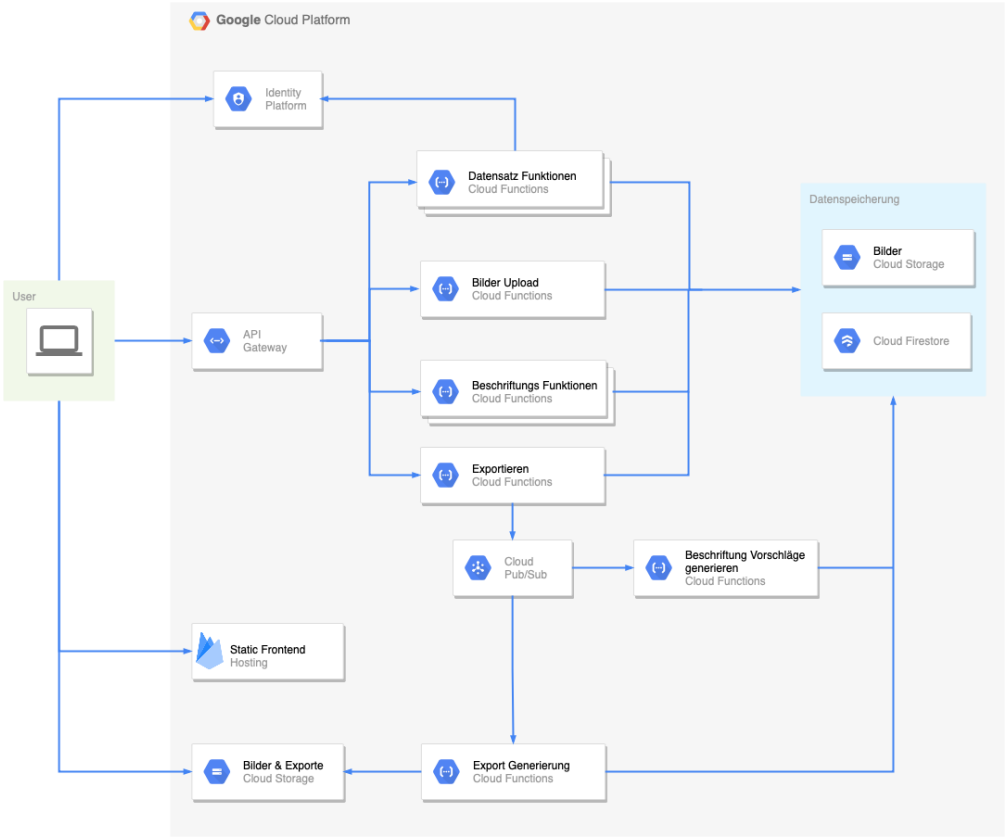


Abbildung 4.3: Google Cloud Architekturdiagramm

Die Funktionen hinter dem API Gateway sind teilweise im Sinne der Übersicht zusammengefasst. Datensatz- und Beschriftungsfunktionen beschreiben, so zum Beispiel, alle Funktionen für die Standard CRUD (Create Read Update Delete) Operationen auf dem jeweiligen Datentyp.
Die Speicherung der Datensätze und zugehörigen Beschriftungen erfolgt in Cloud Firestore. Bilder sowie die exportierten Datensätze werden in Cloud Storage Buckets gespeichert. Auf die in den Buckets gespeicherten Bilder kann der Nutzer mit den richtigen Berechtigungen direkt zugreifen, so dass diese dem Nutzer beim Beschriften angezeigt werden können.
Bilder werden, von einer Cloud Function entgegengenommen, verarbeitet und dann in einem Cloud Storage Bucket gespeichert. Nachdem ein Bild vollständig in Cloud Storage hochgeladen wurde, entsteht automatisch ein Ereignis, welches die Beschriftung Vorschläge generieren Funktion auslöst. Diese Funktion nutzt das neue Bild und generiert dann dafür Bounding Boxes und speichert diese in Firestore.

Die Kommunikation der Funktionen erfolgt über Cloud PubSub. Das heißt, dass eine Funktion eine Nachricht in einem Thema veröffentlicht, wodurch eine andere Funktion ausgelöst werden kann. Verwendet wird diese Kommunikation für die Auslösung der Export-Generierungsfunktion. Da diese, je nach Größe des Datensatzes, durchaus etwas länger brauchen kann, wird sie als langlebige Hintergrundaufgabe entworfen. Diese Art der Trennung von langlebigen Funktionen ist ein gebräuchliches Muster in Serverless Computing (Katzer, 2020, S. 41). Die Generierung ist ein ETL (Extract Transform Load) Prozess und speichert die erzeugten Daten nach Verarbeitung wieder in einem Cloud Storage Bucket, in welchem die Nutzer direkt Zugriff haben.

Das Frontend ist eine Single Page Application bestehend aus statischen Dateien, die von Firebase Hosting bereitgestellt werden.

4.1.4 Laufzeitsicht

Die Laufzeitsicht zeigt, wie sich einzelne Komponenten und Bausteine zur Laufzeit des Systems verhalten und wie diese miteinander interagieren.
Konkret kann mit dieser Sicht der Ablauf von bestimmten Use-Cases beschrieben werden (Starke, 2020, S. 168).
Das erste Sequenzdiagramm auf Abbildung 4.4 zeigt den Ablauf des Registrierungs- und Login- Prozesses. Dieser wird dort beispielhaft anhand des Google Identitätsanbieters aufgezeigt.
Nachdem ein Nutzer Google als Anmeldemethode ausgewählt hat, wird dieser weitergeleitet, um sich bei Google mit einem Google Account anzumelden. Ist dies erfolgreich, erhält er ein OAuth Token von Google für seinen Account. Dieses Token kann dann verwendet werden, um eine Anfrage an die Identity Platform zu senden. Diese verifiziert dieses Token und liefert ein idToken zurück, für die Identität des Nutzers in der Anwendung. Dieses idToken ist ein Base64 kodiertes JWT Token, mit Informationen der Nutzer ID und der E-Mail Adresse.
Für die anderen Sequenzdiagramme ist vorausgesetzt, dass der Nutzer den Anmeldeschritt durchgeführt hat und im Besitz eines gültigen idTokens ist. Da der Vorgang des Sendens des idTokens an das Gateway und die Verifizierung von diesem bereits erklärt wurde und immer gleich ist, wird dieser nicht explizit in den Diagrammen dargestellt.
Abbildung 4.5 gibt einen genaueren Überblick über den Ablauf eines Bilduploads mit

den dazugehörigen Komponenten.
Auf der Abbildung 4.6 ist der Anwendungsfall Datensatz exportieren zu sehen. Es ist dort genau zu sehen, aus welchen Quellen, welche Daten geladen und benötigt werden.
Das Sequenzdiagramm 4.7 zeigt, wie ein neuer Nutzer, mit Berechtigungen, zu einem Datensatz hinzugefügt wird. Nach der Überprüfung, ob der Nutzer in der Anwendung registriert ist, wird dieser in der Datenbank und in Cloud Storage als Mitglied des Datensatzes eingetragen, um die benötigten Berechtigungen zu erhalten.

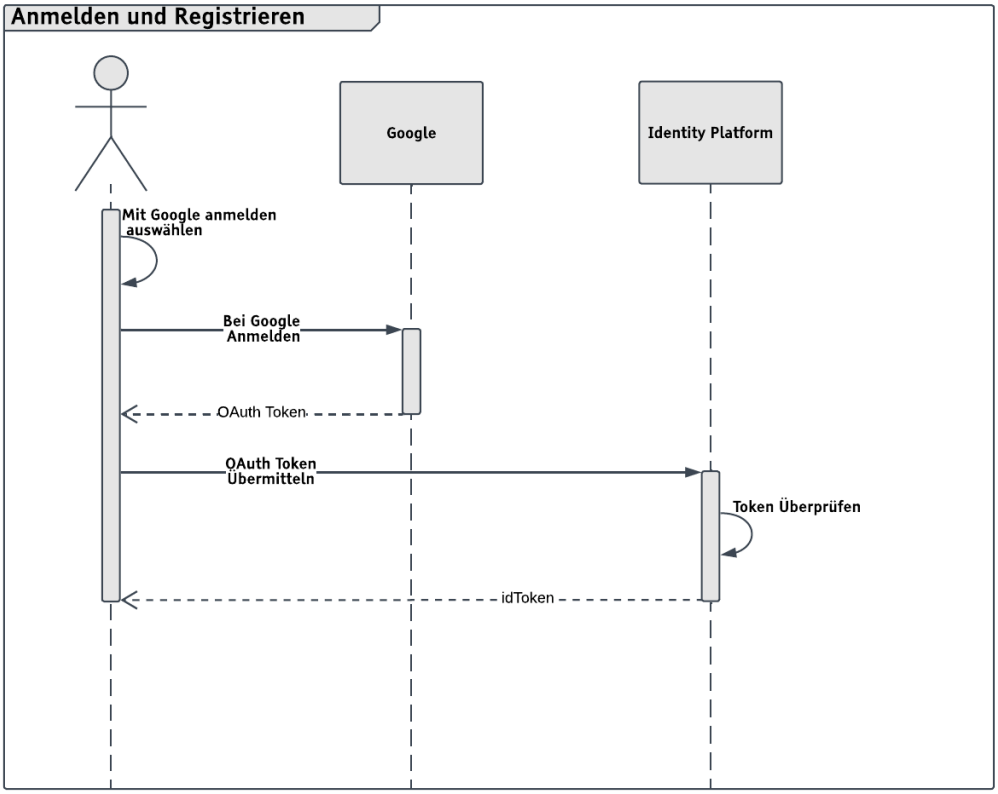


Abbildung 4.4: Anmelden und Registrieren

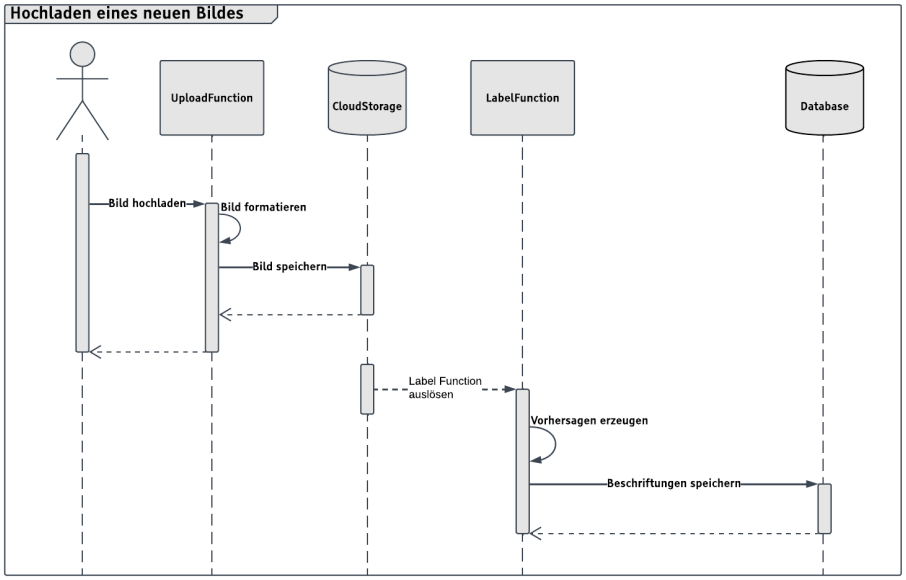


Abbildung 4.5: Hochladen eines Bildes

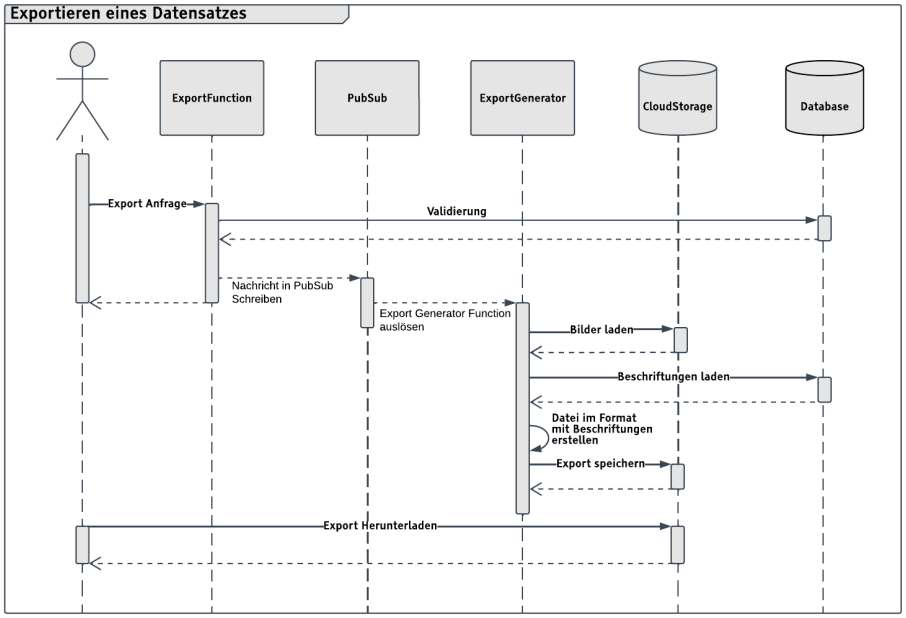


Abbildung 4.6: Exportieren eines Datensatzes

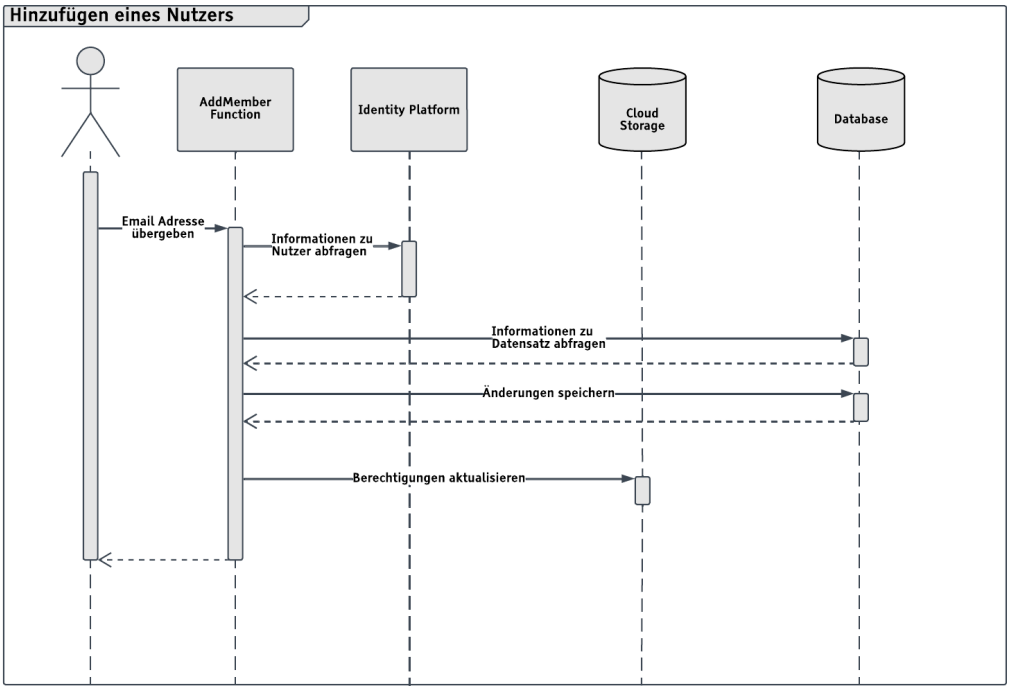


Abbildung 4.7: Hinzufügen eines Nutzers

4.2 Technologieentscheidungen

4.2.1 NodeJS, JavaScript und TypeScript

Für den Großteil des Backends der Anwendung wurde NodeJS verwendet. NodeJS ist eine asynchrone, Event-basierte Laufzeitumgebung und kann für die serverseitige Programmierung von Webanwendungen mit JavaScript verwendet werden (OpenJS Foundation, 2022). Mithilfe von NodeJS ist es möglich, mit vergleichsweise wenig Aufwand performante und skalierbare Systeme zu entwickeln.
Als Programmiersprache wurde TypeScript[1] verwendet. TypeScript basiert auf Javascript und fügt zusätzliche Syntax und Typisierung hinzu. Als Vorteil ergibt sich daraus, dass viele Fehler bereits während der Programmierung gefunden werden können und nicht erst bei dem Ausführen des Programms auftreten. Aus TypeScript Code wird bei der Kompilierung normaler Javascript Code erzeugt.

[1] https://www.typescriptlang.org/

4.2.2 VueJS

Für die Umsetzung des Frontends wird das Javascript Framework Vue.js benutzt. Vue.js hilft bei der Entwicklung von grafischen Oberflächen und bietet durch sein deklaratives und reaktives Programmiermodell eine gute Grundlage, um Single-Page-Anwendungen umzusetzen (You, 2022).

4.2.3 PyTorch

PyTorch[2] ist ein für Python geschriebenes Framework, welches Unterstützung bei der Programmierung im Bereich Maschinelles Lernen bietet. Vor allem für das Erstellen von Neuronalen Netzen ist PyTorch gut geeignet.
Hier wird das Framework und Python im Zusammenhang mit YOLOv5 verwendet, um die initiale Objekterkennung zu realisieren.

4.2.4 Annotorious

Annotorious[3] ist eine JavaScript Bibliothek zum Beschriften von Bildern.
Diese Bibliothek bietet eine grafische Oberfläche, mithilfe welcher es möglich ist, Bounding Boxes in Form von Rechtecken einfach auf ein Bild zu zeichnen und zu bearbeiten.

4.3 REST API

Die Anwendung bietet verschiedene REST Endpunkte an, welche verwendet werden können, um mit dem System zu interagieren. Unterteilt sind diese in die Kategorien: datasets, annotations und exports, welche dann die Endpunkte für die Operationen mit der jeweiligen Ressource ermöglichen. Informationen wie zum Beispiel, um welchen Datensatz es sich handelt, oder welche Daten ein neuer Mitarbeiter des Datensatzes hat, werden über die Parameter und den Body der HTTP Anfrage, an das Backend übergeben.
Abbildung 4.8 zeigt alle REST Schnittstellen für die Verwaltung von Datensätzen.
Abbildung 4.9 zeigt alle REST Schnittstellen für die Verwaltung von Beschriftungen auf Datensätzen.

[2] https://pytorch.org/
[3] https://recogito.github.io/annotorious/

Auf Abbildung 4.10 sind die Schnittstellen zum Erstellen und Auflisten der Exporte zu sehen.

Abbildung 4.8: API Endpunkte für die Datensätze

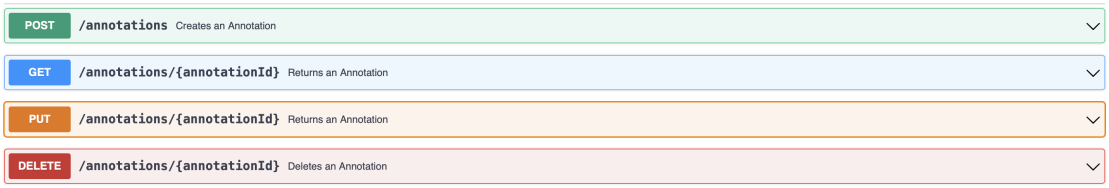


Abbildung 4.9: API Endpunkte für die Beschriftungen

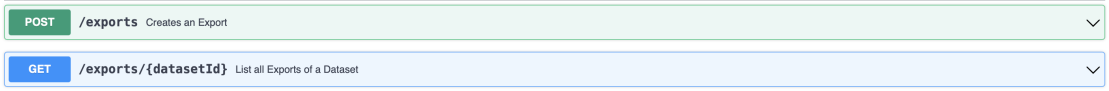


Abbildung 4.10: API Endpunkte für die Exporte

5 Implementierung

Im folgenden Kapitel wird genauer auf die Implementierung der Anwendung eingegangen. Dafür werden die im Entwurf entwickelten Konzepte und Architekturen für die Realisierung verwendet.

5.1 Google Dienste

5.1.1 Konfiguration Identity Platform

Die Konfiguration der Identity Platform ist relativ simpel und es muss standardmäßig nicht viel eingestellt werden. Das Wichtigste dabei ist das Festlegen der Identitätsanbieter. Für die Anwendung wurden insgesamt zwei Anbieter konfiguriert: E-Mail/Passwort und Google. Bei der E-Mail/Passwort Anmeldemethode können sich Nutzer ganz normal mit einer E-Mail und einem Passwort bei der Anmeldung registrieren und sich damit auch wieder anmelden. Bei Google können sie sich über ein vorhandenes Google Konto, mithilfe von OAuth, anmelden.

5.1.2 Konfiguration API Gateway

Der wichtigste Schritt bei der Konfiguration des API Gateways ist die Erstellung einer Swagger Definition. Diese muss mindestens Informationen zu allen verfügbaren Endpunkten und mit welcher HTTP Methode diese aufgerufen werden, enthalten. Der folgende Ausschnitt zeigt, anhand von der Route zum Erstellen eines Datensatzes, wie die Routendefinitionen bei Swagger aussehen müssen.

```
1 /datasets:
2     post:
3       summary: creates a dataset
4       x-google-backend:
```

```
5         address: https://europe-west3-labelapp-339118.cloudfunctions.net/
    createDataset
6       consumes:
7       - application/json
8       produces:
9       - application/json
10      parameters:
11      - in: body
12        name: body
13        description: Dataset object that needs to be added
14        required: true
15        schema:
16          $ref: '#/definitions/Dataset'
17      responses:
18        201:
19          description: Dataset created
20          schema:
21            $ref: '#/definitions/Dataset'
22        400:
23          description: Invalid input
```

Listing 5.1: Swagger Route

Am Anfang wird die HTTP Methode festgelegt, in diesem Fall ist das post. Danach kommen weitere Informationen, welche Art von Daten dieser Endpunkt erzeugt und verbraucht. Bei parameters können verschiedene Eingabemöglichkeiten festgelegt werden. Im Fall von post befinden sich die Daten im body der Anfrage. Am Ende gibt es die Möglichkeit, anzugeben, was für Antwortmöglichkeiten diese Route hat. Eine für die Google Cloud spezifische Angabe, die hier hinzugefügt werden muss, ist x-google-backend. Dies legt den internen Google Dienst fest, an welchen das Gateway die Anfrage weiterleiten soll.
Um das API Gateway dazu zu bringen, dass es die Anfragen authentifiziert, muss ein weiterer Konfigurationsabschnitt hinzugefügt werden. In diesen sogenannten *security-Definitions* werden Informationen zu dem verwendeten OAuth Anbieter, hier Identity Platform, eingetragen. Ist dieser Abschnitt ausgefüllt, kann das API Gateway bei jeder Anfrage automatisch bei Identity Platform überprüfen, ob der Nutzer angemeldet ist.

Eine Schwierigkeit, die es zum Zeitpunkt der Erstellung des API Gateways gab, ist die Einstellung von CORS. Standardmäßig verhindert der Browser aus Sicherheitsgründen

Javascript daran, HTTP Anfragen an externe Domains zu senden. Für viele Anwendungen wird dies jedoch benötigt, deshalb ist es möglich mithilfe von CORS Regeln festzulegen, welche Cross-Origin Anfragen erlaubt sein sollen. Da es bei dem API Gateway noch keine Einstellung gibt, die dies allgemein erlaubt, muss dafür bei jeder Route eine weitere Defintion für eine OPTIONS Anfrage hinzugefügt werden.

5.2 Allgemeine Struktur

5.2.1 Realisierung der Functions

Um die Funktionen mit Node.js und TypeScript umzusetzen, wurde das offiziell von Google veröffentlichte Functions Framework, für Node.js[1], verwendet. Dieses bietet Möglichkeiten zum Bereitstellen, Lokalen Ausführen und Testen von Funktionen.
Google Cloud Functions verwenden das Framework Express.js[2], um eingehende Anfragen zu verarbeiten.

5.2.2 Middleware

Middleware sind in Express.js Funktionen, die während einer Anfrage, vor dem Hauptteil der Anwendungslogik ausgeführt werden. Mithilfe dieser Middleware können wiederverwendbare Komponenten erstellt werden, um einheitliche Logik bei bestimmten Routen, oder in diesem Fall Cloud Functions, umzusetzen.
Verwendet werden Middlewares in der Anwendung unter anderem, für ein einheitliches Error Handling, um Eingaben zu validieren und um bestimmte allgemeine HTTP Header zu setzen. Des Weiteren gibt es eine Middleware, die verwendet werden kann, um bestimmte Berechtigungen auf einer Route durchzusetzen. Dazu wird festgelegt, welche Art von Berechtigung für diese Route benötigt wird und die Middleware überprüft dann, ob der angemeldete Nutzer die für die Aktion erforderliche Rolle hat. Zum Beispiel dürfen nur Datensatzverwalter einen Datensatz löschen und nicht jeder Mitarbeiter.

[1] https://github.com/GoogleCloudPlatform/functions-framework-nodejs
[2] https://expressjs.com/de/

5.3 Rechte Verwaltung / Autorisierung

Die Aufgabe der Authentifizierung wird nur vom API Gateway geregelt. Die Funktionen sind in der Google Cloud so konfiguriert, dass diese nicht direkt von Nutzern aufgerufen werden können. Nur das API Gateway hat die Berechtigungen, die einzelnen HTTP Funktionen auszulösen. Dadurch ist sichergestellt, dass kein unangemeldeter Zugriff auf die Funktionen erfolgen kann. Das API Gateway fügt nach erfolgreicher Authentifizierung den Header *X-Apigateway-Api-Userinfo* der Anfrage hinzu. Dieser kann von den Funktionen dann verwendet werden, um Informationen zu dem angemeldeten Nutzer zu erfahren, von dem die Anfrage stammt. Die wichtigsten Daten sind die Nutzer ID und die E-Mail Adresse. Da das Identity and Access Management (IAM) von Google direkte Anfragen an die Funktionen blockiert, ist sichergestellt, dass dieser Header auch wirklich vom API Gateway stammt.
Die genaueren Berechtigungen (Autorisierung) übernehmen dann die Funktionen selber, um genauer entscheiden zu können, wann ein Nutzer auf welche Ressource wie zugreifen kann.
Damit Nutzer Zugriff auf die Bilder eines Datensatzes haben, werden diese in Form von Metadaten den Objekten in Cloud Storage hinzugefügt. Mithilfe von Sicherheitsregeln kann dann beim Zugriff überprüft werden, ob der Nutzer der jeweiligen Anfrage in diesen Metadaten enthalten ist. Dazu werden die Daten, aus dem von Identity Platform erstellten idToken, mit einfachen Bedingungen ausgewertet.

```
1 service firebase.storage {
2   match /b/{bucket}/o {
3     match /{dataset}/{allPaths=**} {
4       allow list: if true;
5       allow read: if request.auth.token.email
6                   in resource.metadata.values();
7     }
8   }
9 }
```

Listing 5.2: Sicherheitsregeln

Wenn ein Nutzer bei einem Datensatz hinzugefügt oder entfernt wird, müssen die Berechtigungen in Cloud Storage auch aktualisiert werden. Diese Vorgehensweise ist zwar

etwas umständlich, sorgt aber dafür, dass die Berechtigungen sehr genau auf Datensatz Ebene eingestellt werden können.

5.4 Node.js Streams

Für viele, der hier umgesetzten Konzepte, wie Bilder Upload und Export Generierung, wurden Node.js Streams verwendet. Streams sind ein wichtiger, grundlegender Bestandteil von Node.js und finden unter anderem Verwendung beim Lesen/Schreiben von Dateien, Netzwerkkommunikation und allgemeinem Informationsaustausch (OpenJS Foundation, 2019). Vor allem größere Mengen an Daten können auch mit geringem Arbeitsspeicher durch Streams effizient bearbeitet werden. Die Daten werden hier nicht alle auf einmal gelesen, sondern immer Stück für Stück.

5.5 Bilder Upload

Der erste Schritt, wenn die Bilder hochgeladen werden, ist die Generierung eines neuen Namens. Dieser Vorgang ist wichtig, um ungewollte Zeichen aus dem Namen zu entfernen und Namensduplikate zu vermeiden. Die neuen Namen bestehen aus zufällig generierten Zeichenketten, mit der originalen Dateiendung des Bildes.
Vor der Speicherung des Bildes in Cloud Storage, wird die Größe dieses auf 640x640 Pixel verändert. 640x640 ist eine Standardgröße für viele Objekterkennungsmodelle.

```
1     const fileBlob = this.imageFile(fileName);
2     const { writeStream } = createStorageWriteStream(
3         fileBlob,
4         permissionFromDataset(this.dataset)
5     );
6     const sharpStream = createResizeStream();
7
8     file.pipe(sharpStream)
9     sharpStream.pipe(writeStream);
```

Listing 5.3: Upload eines Bildes

Der Codeabschnitt zeigt grundlegend diesen Vorgang. Die Bilder werden über den Content-Type multipart/form-data an das Backend übertragen. Das Hochladen von Dateien nach Cloud Storage funktioniert über die Erstellung von Write Streams. Mithilfe von diesem, können dann Daten direkt in ein Cloud Storage Objekt geschrieben werden. Das hochgeladene Bild muss dadurch nicht in der Funktion zwischengespeichert werden, sondern wird durch eine pipeline direkt weitergeleitet. Durch die Verwendung eines Transformations Streams, welcher sowohl Schreiben als auch Lesen kann, werden die Bilder zwischen der Eingabe und dem Hochladen noch verarbeitet. Zudem werden bei der Erstellung der Datei für das Bild Metadaten hinzugefügt, damit alle Nutzer des Datensatzes Zugriff auf dieses Bild haben.
Es gibt einen Cloud Storage Bucket, in dem alle Bilder gespeichert werden. Um die Bilder bestimmten Datensätzen zuzuordnen, gibt es innerhalb der Buckets Präfixe für die einzelnen Datensätze. Der Pfad zu einem Bild mit dem Namen bild1.jpg, aus dem Datensatz mit der ID UknQZwtfFt8H8YIwscUL, ist: *UknQZwtfFt8H8YIwscUL/bild1.jpg*.

5.6 Datensatz exportieren

Die Export Funktion ist zuständig, für einen bestimmten Datensatz ein Archiv zu erzeugen, mit welchem ein neues Modell trainiert werden kann. Das hier für den Export verwendete Format, ist das Standard Tensorflow object detection Format. Dieses besteht aus einer CSV Datei mit den Informationen zu allen Bildern und deren Beschriftungen.CSV steht für comma seperated values und ist ein einfaches Dateiformat, in welchem in jeder Zeile ein Datensatz, mit Komma getrennten Datenfeldern, steht.
Die Tabelle 5.1 zeigt den Aufbau des hier verwendeten Formats. Die letzten Felder xmin,

filename	width	height	class	xmin	ymin	xmax	ymax
bildname.jpg	640	640	person	390	179	420	279

Tabelle 5.1: Beispiel CSV Datei mit header und einem Datensatz

ymin, xmax und ymax beschreiben die Eckpunkte, (xmin, ymin) und (xmax, ymax), der Bounding Box für die angegebene Klasse.

Für die Erzeugung der Exporte werden zwei Funktionen benötigt, eine als HTTP Funktion, und eine als Hintergrundfunktion, die den Hauptteil der Aufgabe übernimmt. Die HTTP Funktion erzeugt einen Export in der Datenbank und veröffentlicht dann über

Pub/Sub eine JSON Nachricht, in dem Thema createExport, mit der Datensatz ID und der Export ID als Inhalt. Die Hintergrundfunktion kann mit diesen Informationen den Exportprozess starten. Um das Archiv zu erzeugen, muss sie grundlegend vier Arbeitsschritte durchführen:

1. Sammeln aller Beschriftungen aus der Datenbank
2. Sammeln aller Bilder aus Cloud Storage
3. Generierung der CSV Datei
4. Zusammenführung dieser Dateien zu einem ZIP Archiv

Die Transformation und Verarbeitung der Daten wurde umgesetzt, unter der Verwendung von Node.js Streams.

```
1 const archive = archiver('zip', {
2     zlib: { level: 9 },
3     store: true
4 });
5
6 archive.pipe(writeStream);
7
8 this.generateCSV(archive);
9 await this.copyImages(archive);
10
11 archive.finalize();
```

Listing 5.4: Erzeugung des Export Archivs

Bevor die Beschriftungen, in die CSV Datei exportiert werden können, müssen die Bounding Boxes angepasst werden. Es gibt zwei verschiedene Speicherungsmöglichkeiten von Bounding Boxes, wobei das Format, welches bei der automatischen Beschriftung entsteht, nicht das Gleiche ist, welches für den Export benötigt wird. Bei dem hier benötigten Format, besteht die Bounding Box aus dem kleinsten und größten Eckpunkt des Rechtecks. Das in der Datenbank gespeicherte Format besteht jedoch aus dem kleinsten Eckpunkt und der Breite sowie der Höhe des Rechtecks. Für die Umwandlung kann der fehlende größte Eckpunkt, mithilfe der Höhe und Breite folgendermaßen berechnet werden:

$$maxx = minx + Breite$$

$$maxy = miny + Höhe$$

Der Hauptstream ist, wie in dem Codebeispiel zu sehen, der Stream zum Erstellen des ZIP Archivs. Dieser leitet seine Ausgabe an Cloud Storage weiter. Die Erstellung der CSV Datei und die einzelnen Bilder aus dem Storage Bucket, in dem diese gespeichert werden, stellen eigene Streams dar. Um das Archiv zu erstellen, müssen diese einfach nur an den Archiv Stream angehangen werden.
Damit die generierten Dateien nicht für immer bestehen bleiben und Speicherplatz für nicht mehr benötigte Exporte verbraucht wird, müssen diese regelmäßig überprüft und gelöscht werden. Cloud Storage bietet dafür, durch Lebenszyklen von Objekten, eine einfache Lösung für dieses Problem. Bei den Lebenszyklen lassen sich Regeln festlegen, unter welchen Bedingungen, Objekte automatisch gelöscht oder verschoben werden sollen. Die hier verwendete Regel lautet:
7+ Tage, seitdem das Objekt aktualisiert wurde.
Das bedeutet, dass Cloud Storage automatisch Objekte aus diesem Bucket löscht, wenn sie mindestens 7 Tage nicht mehr aktualisiert wurden.

5.7 Automatisches beschriften

Wie bereits im Entwurf beschrieben, wird die Beschriftung der Bilder von einer Google Cloud Function übernommen.
Um die Performance der Beschriftung zu verbessern, ist sowohl das Modell als auch das benötigte YOLO Repository direkt in der Funktion enthalten, so müssen diese nicht jedes mal heruntergeladen werden. Um das Modell zu verwenden, muss es zuerst mit PyTorch aus einer Datei geladen werden.
Um die Geschwindigkeit zu erhöhen, passiert das, im globalen Kontext der Funktion. Dies bedeutet, dass wenn Funktionen wiederverwendet werden, sie nicht erneut geladen werden müssen.

```
1 def detection(event, context):
2     file_name = event["name"]
3     bucket = event["bucket"]
4 
5     split = file_name.split("/")
6     dataset = split[0]
7     img_name = split[1]
8 
```

```
9       print(f"Detection on {file_name} in {bucket}, parsed {dataset}, {img_name
        }")
10
11      download_image(file_name, bucket, img_name)
12      img = f"/tmp/{img_name}"
13
14      results = model(img)
15
16      for row in results.pandas().xyxy[0].itertuples():
17          print(f"Adding annotation class {row.name} with confidence {row.
        confidence}, bb: ({row.xmin}, {row.ymin}) ({row.xmax}, {row.ymax})")
18
19          add_annotation(dataset, img_name, row)
20
21      os.remove(f"/tmp/{img_name}")
22
23      return 'done'
```

Listing 5.5: Erzeugung der Beschriftungen

In diesem Codebeispiel wird die Methode detection gezeigt, diese wird automatisch ausgeführt, wenn ein Bild hochgeladen wird. Informationen dazu befinden sich in dem event Parameter. Die ID des Datensatzes und der Name des Bildes können aus dem Pfad der Cloud Storage Datei entnommen werden.
Bevor das Modell die Vorhersage zu dem Bild treffen kann, muss dieses zuerst von Google Cloud Storage in ein temporäres Verzeichnis, in der Cloud Function, geladen werden.
Wenn Bild und Modell geladen sind, wird mit PyTorch die Inferenz ausgeführt. Zusätzlich wird der Datensatz mit der zugehörigen ID aus der Datenbank geladen, dadurch kann überprüft werden, ob die Klasse der Vorhersage in dem Datensatz vorhanden ist. Dieser Schritt ist wichtig, da ansonsten alle 80 in dem Modell vorhandenen Klassen für die Beschriftung verwendet werden würden. Wenn aber zum Beispiel nur Personen beschriftet werden sollen, könnten die unnötigen Beschriftungen zu mehr Arbeit führen. Wenn die vorhergesagte Klasse benötigt wird, dann wird die Beschriftung direkt in die Firestore Datenbank geschrieben. Am Ende wird aufgeräumt und das Bild wird aus dem temporären Verzeichnis gelöscht.

5.7.1 Modell Auswahl

Als Basis für die erzeugten Vorhersagen wird ein bereits vortrainiertes Modell verwendet. Dieses wurde auf einem sehr großen Datensatz mit vielen verschiedenen Klassen trainiert, um in möglichst vielen Bereichen anwendbar zu sein.
Bei YOLOv5 gibt es grundlegend fünf vortrainierte Modelle (YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x), die verwendet werden können. Der Unterschied zwischen diesen, liegt in der Größe und Komplexität des Modells. Der Vorteil bei der Nutzung eines größeren Modells ist, dass dort die Genauigkeit höher ist. Die bessere Genauigkeit führt aber auch dazu, dass sich die Inferenzzeit wesentlich erhöht. So ist zum Beispiel das größte Modell mehr als doppelt so langsam wie das mittlere. Die Unterschiede der Geschwindigkeit lassen sich auf Abbildung 5.1 gut sehen.

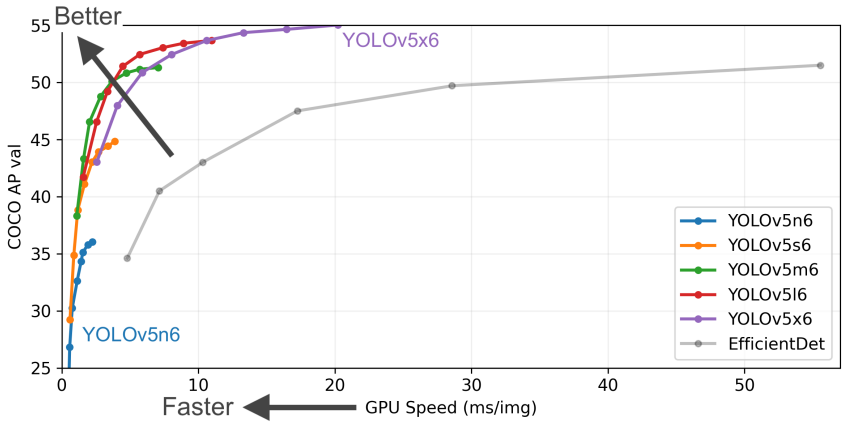


Abbildung 5.1: YOLOv5 Modell Vergleich (Ultralytics, 2020)

Für eine Serverless Umgebung, in der es nur begrenzte Ressourcen gibt, welche im Millisekunden Bereich berechnet werden, müssen diese Unterschiede natürlich beachtet werden. Das Modell muss gut genug die Objekte erkennen und die Vorhersagen trotzdem möglichst schnell erzeugen. Für diese Aufgabe scheint das YOLOv5s Modell am besten geeignet, dieses ist relativ klein und schnell, erkennt aber trotzdem den Großteil der Objekte zuverlässig. Die Genauigkeit des Modells YOLOv5m ist zwar etwas höher, macht aber vermutlich nicht den Geschwindigkeitsunterschied wieder gut.

Um zu überprüfen, welches Modell am sinnvollsten ist, wurden einige Tests durchgeführt. Die Tabelle 5.2 zeigt Messungen von den Modellen s, m und l, wie sich diese auf die Per-

Modell	Ausführungszeit	Speicherauslastung
YOLOv5s	1,1s	650MB
YOLOv5m	1,8s	890MB
YOLOv5l	2,4s	1.2GB

Tabelle 5.2: Übersicht der Modelle mit durchschnittlicher Ausführungszeit und Speicherauslastung

formance und Ressourcennutzung der Beschriftungsfunktion auswirken. Es ist dort zu sehen, wie lange die Funktion durchschnittlich für die Generierung der Beschriftungen benötigt und wie viel RAM dabei benötigt wird.
Google bietet einen Preisrechner für Cloud Produkte an, so lässt sich anhand dieser Daten ungefähr schätzen, wie die Erhöhung der Ausführungszeit und Speicherauslastung den Preis beeinträchtigt.
Folgende Kosten würden ungefähr bei der Erzeugung von 10.000.000 Beschriftungen entstehen (Google, 2022f):
YOLOv5s: 207 Euro
YOLOv5m: 373 Euro
YOLOv5l: 997 Euro

Es lässt sich an diesem Beispiel gut sehen, wie wichtig es ist, bei Serverless Computing, auch auf kleinere Performance Unterschiede zu achten. Der mehr als vierfache Preis, der Nutzung von YOLOv5l über YOLOv5s, ist den eher geringen Vorteil nicht wert.

5.8 Optimierung

Auf Optimierung, der als Backend verwendeten Funktionen, muss während der Implementierung immer wieder geachtet werden, denn diesen stehen teilweise nur sehr limitierte Ressourcen zur Verfügung. Zudem lohnt es sich hier, wegen des Pay-As-You-Go Prinzips, besonders darauf zu achten, dass keine unnötigen Ressourcen verschwendet werden.
Google Cloud bietet in Coe und Skoviera (2020) einige best practices für das Schreiben von Node.js Funktionen. Der erste wichtige Schritt zur Verbesserung der Performance, in Cloud Functions ist, diese möglichst kleinzuhalten und so wenige Abhängigkeiten wie möglich zu verwenden. Bei den Cold Starts spielt vor allem die Größe und Anzahl der verwendeten externen Abhängigkeiten und Bibliotheken eine große Rolle.

5.9 Frontend

Das Frontend ist relativ simpel gehalten und ermöglicht eine einfache Interaktion mit dem Backend der Anwendung. Um die grafische Oberfläche zu gestalten, wurde Bootstrap[3] verwendet. Das Frontend wurde mithilfe von Vue.js als Single Page Application aufgebaut und kommuniziert mithilfe der im Entwurf beschriebenen REST Schnittstelle mit dem Backend.
Mithilfe der Annotorious Bibliothek können dann auf Bilder Bounding Boxes gezeichnet werden, welche nach Vollendung automatisch an das Backend übermittelt werden.

5.10 Testen

Für die Anwendung wurden natürlich auch Tests erstellt, um sicherzustellen, dass keine unerwarteten Fehler auftreten.
Es gibt sowohl Unit Tests, um einzelne Komponenten zu testen, als auch Integrationstests, um die Zusammenarbeit verschiedener Komponenten zu verifizieren. Bei den Tests werden die verwendeten Google Dienste mit Mocks ersetzt. Dabei werden diese Mocks anstelle der echten Objekte verwendet, um diese externen Dienste in der Testumgebung zu simulieren.

5.11 CI/CD

Um den Entwicklungs- und Bereitstellungsprozess zu automatisieren, wurde eine einfache CI-/CD-Pipeline, mithilfe von Cloud Build, erstellt.
Dazu muss eine Datei mit dem Namen cloudbuild.yaml vorhanden sein. Die Pipeline besteht grundlegend aus drei Schritten: build, test und deploy. Im ersten Schritt build, werden alle benötigten Abhängigkeiten installiert. Im zweiten Schritt werden alle erstellen Tests ausgeführt und der Vorgang wird abgebrochen, wenn ein Test fehlschlägt. Im letzten Schritt werden dann die Funktionen bereitgestellt.
Diese ganzen Schritte werden automatisch von Cloud Build ausgeführt, sobald über Git Änderungen hochgeladen werden.

[3] https://getbootstrap.com/

6 Evaluierung

In diesem Kapitel wird die entwickelte Anwendung evaluiert. Es wird geschaut, ob alle geplanten Anforderungen umgesetzt wurden und inwiefern diese den Prozess des Beschriftens optimieren können.

6.1 Anforderungen erfüllt

In diesem Abschnitt wird überprüft, ob alle funktionalen und nichtfunktionalen Anforderungen erfüllt worden sind.

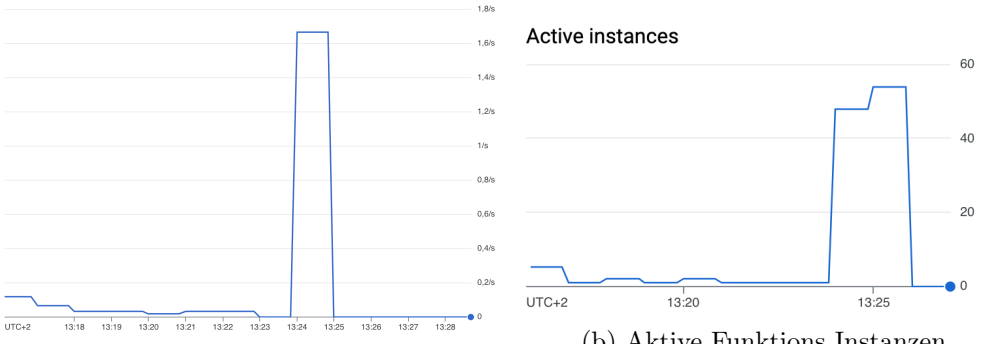
6.1.1 Funktionale Anforderungen

Im Rahmen der Anforderungsanalyse wurden 13 verschiedene funktionale Anforderungen an das System festgelegt. Diese 13 Anforderungen konnten bei der Implementierung umgesetzt werden. Dabei haben Testfälle geholfen, die Funktionalität des Systems sicherzustellen.

6.1.2 Nichtfunktionale Anforderungen

Auch auf die nichtfunktionalen Anforderungen wurde während der Entwicklung geachtet.
Die Verfügbarkeit des Systems hängt hauptsächlich von der Verfügbarkeit der Google Cloud Dienste ab. Durch die Unterteilung des Systems in viele kleine Funktionen wird vorgebeugt, dass ein Fehler in einem Teil des Systems andere nicht direkt beeinträchtigt. Außerdem haben Google Cloud Funktionen den Vorteil, dass wenn neuere Versionen dieser, Fehler hervorrufen, automatisch die letzte fehlerfreie Version, stattdessen verwendet wird.

Für die Sicherheit wird gesorgt, indem nicht autorisierte Zugriffe auf die Bilder, durch festgelegte Sicherheitsregeln, blockiert werden.
Die Anforderungen, an die Skalierbarkeit des Systems, konnten durch die Nutzung von Serverless Computing gut umgesetzt werden. Alle die hier verwendeten Dienste, können sich automatisch an die Last anpassen. Werden viele Anfragen gleichzeitig an das Backend gesendet, werden temporär so viele Funktionen erzeugt, um die Leistungsspitzen abzudecken.
Vor allem am Upload der Bilder und der automatischen Beschriftung lässt sich die Anpassung an die benötigten Ressourcen gut erkennen. Jedes hochgeladene Bild wird von einer eigenen Instanz der Beschriftungs-Funktion bearbeitet, so dass auch wenn 100 Bilder auf einmal hochgeladen werden, alle in kurzer Zeit und beinahe gleichzeitig beschriftet werden können.
Auf der Abbildung 6.1 ist dieses gut zu erkennen. Links sieht man einen steilen Anstieg, an hochgeladenen Bildern, in kurzer Zeit. Auf der rechten Seite sieht man, wie im gleichen Zeitraum, sich die Anzahl an aktiven Instanzen von Beschriftungs-Funktionen verändert. Die Funktion passt sich direkt an und skaliert innerhalb von Sekunden, von 0 auf 60 Instanzen und wieder zurück auf 0, wenn diese nicht mehr benötigt werden.



(a) Hochgeladene Bilder

(b) Aktive Funktions Instanzen

Abbildung 6.1: Übersicht Skalierung

6.2 Experiment

Um zu überprüfen, ob die entwickelte Anwendung bei der Problemstellung hilft und ihr Ziel erfüllt, wurde ein Versuch mit Beispielbildern durchgeführt. In den beiden Teilen dieses Experiments wurden jeweils 100 Bilder beschriftet. In dem ersten Teil wurden keine

Vorschläge für die Bilder generiert und jedes Objekt musste manuell beschriftet werden. In dem zweiten Ablauf wurden die hochgeladenen Bilder vorher von dem System mit Beschriftungen versehen, so dass diese Beschriftungen hier nur manuell überprüft und gegebenenfalls verbessert werden mussten. Das Ziel dieses Versuchsaufbaus ist, herauszufinden, ob die Vorschläge zu einer vereinfachten und schnelleren Beschriftung der Beispielbilder führen.

Die Tabelle 6.1 zeigt die Ergebnisse der Versuche. Die Dauer zeigt, wie lange die Beschriftung von den 100 Bildern in Minuten und Sekunden gedauert hat. In den anderen Spalten sind Informationen darüber, wie viele Bounding Boxes manuell hinzugefügt, gelöscht und angepasst werden mussten.

Vorbeschriftet	Dauer	Hinzugefügt	Gelöscht	Bearbeitet
Ja	20:53	181	14	10
Nein	48:40	1274	0	0

Tabelle 6.1: Übersicht der Ergebnisse des Experiments

Es ist dort zu erkennen, dass in dem Versuch ohne generierte Beschriftungen, die Dauer mit 48 Minuten und 40 Sekunden mehr als doppelt so lang war, wie in dem Versuch mit KI erzeugten Beschriftungen. Zudem sieht man, dass das bereits trainierte Modell viel Arbeit abnimmt und die Anzahl, der manuell hinzugefügten Bounding Boxes wesentlich höher ist, wenn keine Vorschläge erzeugt werden.

7 Fazit

In dieser Arbeit wurde ein System entwickelt, welches dabei helfen soll, das Beschriften von Bildern für Maschinelles Lernen zu optimieren. Ein vortrainiertes Modell erzeugt dazu Vorschläge, für die einzelnen Bilder. Diese Vorschläge können manuell nach Bedarf überprüft und angepasst werden. Dafür gibt es eine einfache Weboberfläche zur Bedienung und das Backend wurde, unter Verwendung von Serverless Computing Architekturen, umgesetzt.
Zu Beginn wurden Anforderungen festgelegt, damit bei der Entwicklung deutlich wird, welche Funktionen es geben soll. Anhand der Anforderungsanalyse wurde dann im nächsten Schritt ein Entwurf für das System erstellt. Mit diesem Entwurf als Grundlage konnte das System so implementiert werden, dass alle Anforderungen erfüllt wurden.
Die Anwendung wurde anhand eines Beispielszenarios getestet. Bei diesem Szenario wurden, mit und ohne KI Unterstützung, Bilder beschriftet. Auch wenn dieser Versuch nochmal mit mehreren Teilnehmern durchgeführt werden müsste, um genauere Werte zu liefern, haben die Ergebnisse gezeigt, dass die Beschriftungen mit Unterstützung einfacher zu erstellen sind, da signifikant weniger Bounding Boxes manuell hinzugefügt werden müssen. Zudem ist die benötigte Zeit um mehr als die Hälfte reduziert worden.
Die Verwendung von Serverless Computing hat sich insgesamt als geeignet gezeigt und in der Verfügbarkeit und Skalierbarkeit des Systems zu guten Ergebnissen geführt.
Das einzig wahrnehmbare Problem, welches dabei aufgetreten ist, waren die Cold Starts. Dadurch kam es nach längerer Nichtbenutzung der Anwendung zu Wartezeiten, in denen die Funktionen gestartet werden mussten.

7.1 Ausblick

Die Anwendung kann zum aktuellen Stand schon gut dabei helfen, Datensätze für Maschinelles Lernen vorzubereiten. Trotzdem gibt es einige Erweiterungsmöglichkeiten, die in der Zukunft das Potenzial dieser erhöhen könnten.

Die grafische Oberfläche lässt sich zwar einfach bedienen, sollte jedoch durchaus noch in der Benutzbarkeit verbessert werden. Das könnte unter anderem durch das Hinzufügen weiterer Werkzeuge, die bei der Beschriftung helfen, realisiert werden.
Die Anwendung kommt gut mit bekannten Standardobjekten klar. Damit auch neuartige Daten automatisch beschriftet werden können, wäre es von Vorteil, wenn das vorhandene Modell von den manuell erstellten Beschriftungen dazu lernen könnte. Um dies umzusetzen müsste man, nachdem eine gewisse Anzahl an Beschriftungen angepasst wurden, einen Trainingsprozess starten. Dieser würde dann das Standard YOLOv5 Modell um neue Trainingsdaten erweitern, damit die dort vorhandenen Objekte besser erkannt werden.
Andere sinnvolle Erweiterungsmöglichkeiten wären das Hinzufügen von zusätzlichen Export-Formaten und die Möglichkeit, existierende Datensätze zu importieren.

Literaturverzeichnis

[Baldini u. a. 2017] Baldini, Ioana ; Castro, Paul ; Chang, Kerry ; Cheng, Perry ; Fink, Stephen ; Ishakian, Vatche ; Mitchell, Nick ; Muthusamy, Vinod ; Rabbah, Rodric ; Slominski, Aleksander ; Suter, Philippe: *Serverless Computing: Current Trends and Open Problems*. S. 1–20. In: Chaudhary, Sanjay (Hrsg.) ; Somani, Gaurav (Hrsg.) ; Buyya, Rajkumar (Hrsg.): *Research Advances in Cloud Computing*. Singapore : Springer Singapore, 2017. – URL https://doi.org/10.1007/978-981-10-5026-8_1. – ISBN 978-981-10-5026-8

[Coe und Skoviera 2020] Coe, Benjamin ; Skoviera, Martin: *Tips for writing and deploying Node.js apps on Cloud Functions*. 2020. – URL https://cloud.google.com/blog/products/serverless/running-effective-nodejs-apps-on-cloud-functions. – Zugriffsdatum: 2022-04-23

[Desmond u. a. 2021] Desmond, Michael ; Muller, Michael ; Ashktorab, Zahra ; Dugan, Casey ; Duesterwald, Evelyn ; Brimijoin, Kristina ; Finegan-Dollak, Catherine ; Brachman, Michelle ; Sharma, Aabhas ; Joshi, Narendra N. ; Pan, Qian: Increasing the Speed and Accuracy of Data Labeling Through an AI Assisted Interface. In: *26th International Conference on Intelligent User Interfaces*. New York, NY, USA : Association for Computing Machinery, 2021 (IUI '21), S. 392–401. – URL https://doi.org/10.1145/3397481.3450698. – ISBN 9781450380171

[Google 2022a] Google: *Cloud API Gateway*. 2022. – URL https://cloud.google.com/api-gateway/docs/about-api-gateway. – Zugriffsdatum: 2022-03-22

[Google 2022b] Google: *Cloud Identity Platform*. 2022. – URL https://cloud.google.com/identity-platform?hl=de. – Zugriffsdatum: 2022-03-22

[Google 2022c] Google: *Cloud PubSub*. 2022. – URL https://cloud.google.com/pubsub/docs/overview. – Zugriffsdatum: 2022-03-22

[Google 2022d] Google: *Cloud Storage*. 2022. – URL https://cloud.google.com/storage?hl=de. – Zugriffsdatum: 2022-03-22

[Google 2022e] Google: *Firestore*. 2022. – URL https://cloud.google.com/firestore?hl=de. – Zugriffsdatum: 2022-03-22

[Google 2022f] Google: *Google Cloud Pricing Calculator*. 2022. – URL https://cloud.google.com/products/calculator. – Zugriffsdatum: 2022-05-12

[Haider und Michahelles 2021] Haider, Tom ; Michahelles, Florian: Human-Machine Collaboration on Data Annotation of Images by Semi-Automatic Labeling. In: *Mensch Und Computer 2021*. New York, NY, USA : Association for Computing Machinery, 2021 (MuC '21), S. 552–556. – URL https://doi.org/10.1145/3473856.3473993. – ISBN 9781450386456

[Jonas u. a. 2019] Jonas, Eric ; Schleier-Smith, Johann ; Sreekanti, Vikram ; Tsai, Chia-Che ; Khandelwal, Anurag ; Pu, Qifan ; Shankar, Vaishaal ; Carreira, Joao ; Krauth, Karl ; Yadwadkar, Neeraja ; Gonzalez, Joseph E. ; Popa, Raluca A. ; Stoica, Ion ; Patterson, David A.: *Cloud Programming Simplified: A Berkeley View on Serverless Computing*. 2019. – URL https://arxiv.org/abs/1902.03383

[Katzer 2020] Katzer, J.: *Learning Serverless: Design, Develop, and Deploy with Confidence*. O'Reilly Media, Incorporated, 2020. – ISBN 9781492057017

[Kratzke 2021] Kratzke, Nane: *Serverless Architekturen*. 2021. – URL https://splatblob.download/assets/ss2021/cloudarch/unit-08.pdf. – Zugriffsdatum: 2022-04-02

[OpenJS Foundation 2019] OpenJS Foundation: *Node.js Streams*. 2019. – URL https://nodejs.dev/learn/nodejs-streams. – Zugriffsdatum: 2022-04-17

[OpenJS Foundation 2022] OpenJS Foundation: *Node.js*. 2022. – URL https://nodejs.org/en/about/. – Zugriffsdatum: 2022-04-17

[OpenVINO 2022] OpenVINO: *CVAT*. 2022. – URL hhttps://github.com/openvinotoolkit/cvat. – Zugriffsdatum: 2022-05-12

[O'Shea und Nash 2015] O'Shea, Keiron ; Nash, Ryan: An Introduction to Convolutional Neural Networks. In: *CoRR* abs/1511.08458 (2015). – URL http://arxiv.org/abs/1511.08458

[Otte 2019] Otte, Ralf: *Künstliche Intelligenz für Dummies*. 1. Auflage. Weinheim : Wiley, Wiley-VCH Verlag GmbH & Co. KGaA, 2019 (für Dummies). – ISBN 978-3-527-71494-0

[Owens 2018] Owens, Ken: CNCF Serverless Whitepaper v1.0. (2018). – URL https://github.com/cncf/wg-serverless/blob/master/whitepapers/serverless-overview/cncf_serverless_whitepaper_v1.0.pdf. – Zugriffsdatum: 2022-04-02

[Pohl und Rupp 2015] Pohl, Klaus ; Rupp, Chris: *Basiswissen Requirements Engineering*. 4., überarbeitete Auflage. Heidelberg : dpunkt.verlag, 2015. – ISBN 978-3-86491-673-1

[Redmon u. a. 2015] Redmon, Joseph ; Divvala, Santosh ; Girshick, Ross ; Farhadi, Ali: *You Only Look Once: Unified, Real-Time Object Detection*. 2015. – URL https://arxiv.org/abs/1506.02640

[Roberts 2016] Roberts, Mike: Serverless Architectures. (2016). – URL https://martinfowler.com/articles/serverless.html. – Zugriffsdatum: 2022-04-02

[Roboflow 2022] Roboflow: *Roboflow*. 2022. – URL https://roboflow.com/. – Zugriffsdatum: 2022-05-12

[Russell u. a. 2007] Russell, Bryan C. ; Torralba, Antonio ; Murphy, Kevin P. ; Freeman, William T.: LabelMe: A Database and Web-Based Tool for Image Annotation. In: *International Journal of Computer Vision* (2007), Oktober. – URL https://people.csail.mit.edu/brussell/research/AIM-2005-025-new.pdf

[Sager u. a. 2021] Sager, Christoph ; Janiesch, Christian ; Zschech, Patrick: A survey of image labelling for computer vision applications. In: *Journal of Business Analytics* 4 (2021), Nr. 2, S. 91–110. – URL https://doi.org/10.1080/2573234X.2021.1908861

[Starke 2020] Starke, Gernot: *Effektive Software-Architekturen: Ein praktischer Leitfaden*. Hanser, 2020. – ISBN 978-3-446-46589-3 978-3

[Ultralytics 2020] Ultralytics: YOLOv5. (2020). – URL https://github.com/ultralytics/yolov5. – Zugriffsdatum: 2022-05-10

[You 2022] You, Evan: *Vue.js*. 2022. – URL https://vuejs.org/guide/introduction.html. – Zugriffsdatum: 2022-04-17

[Zhang u. a. 2008] Zhang, Lei ; Tong, Yan ; Ji, Qiang: Active Image Labeling and Its Application to Facial Action Labeling. In: Forsyth, David (Hrsg.) ; Torr, Philip (Hrsg.) ; Zisserman, Andrew (Hrsg.): *Computer Vision – ECCV 2008*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, S. 706–719. – ISBN 978-3-540-88688-4

Glossar

Auth0 Auth0 ist eine Authentifizierungs- und Autorisierungsplattform.

CORS Cross-Origin Resource Sharing.

JSON JavaScript Object Notation.

JWT JSON Web Tokens sind ein Standard, um Berechtigungen darzustellen und zu verifizieren.

OAuth OAuth ist ein standardisiertes Protokoll für die sichere Autorisierung.

REST Representational State Transfer ist ein Programmierparadigma für den Austausch von Informationen bei Webdiensten.

Swagger Swagger bietet Werkzeuge, um Webdienste zu entwerfen und dokumentieren.

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original