Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Asaad Askar

Effizienzsteigerung eines Datenannotationsprozesses: Analyse von Optimierungspotenzialen einer bestehenden Toolkette und Implementierung ausgewählter Maßnahmen

Fakultät Technik und Informatik
Studiendepartment Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Asaad Askar

Effizienzsteigerung eines Datenannotationsprozesses: Analyse von Optimierungspotenzialen einer bestehenden Toolkette und Implementierung ausgewählter Maßnahmen

Betreuender Prüfer: Prof. Dr. Heike Neumann
Beteuer: Dipl.-Ing. Jan Schloßhauer

Eingereicht am: 1. Januar 23

Asaad Askar

Thema der Arbeit

Effizienzsteigerung eines Datenannotationsprozesses: Analyse von Optimierungspotenzialen einer bestehenden Toolkette und Implementierung ausgewählter Maßnahmen

Stichworte

Tool-Anpassung, Labeling-Prozess-Optimierung, Datenannotation, Prozesskette, XML-Format, Export- Importfunktionen, Teamkommunikation, Datenvisualisierung, Docker Netzwerk

Kurzzusammenfassung

Im Rahmen dieser Arbeit wurde das fortschrittliche Labeling-Tool (CVAT) angepasst, um ein altes Tool zu ersetzen und den Datenannotationsprozess durch neue Funktionen wie Export-Importfunktionen, Datenvisualisierung sowie ein Benachrichtigungssystem anzupassen und zu optimieren. Die umfassende Analyse und praktische Erprobung bestätigten, dass CVAT die Effizienz und Qualität in Teamarbeitsumgebungen erheblich verbessert und das alte Tool erfolgreich ersetzen kann.

Asaad Askar

Title of the paper

Efficiency Improvement of a Data Annotation Process: Analysis of Optimization Potentials of an Existing Toolchain and Implementation of Selected Measures

Keywords

Tool Customization, Labeling Process Optimization, Data Annotation, Process Chain, XML Format, Export and import functions, Team Communication, Data Visualization, Docker Network

Abstract

In the context of this work, the advanced labeling tool (CVAT) was adapted to replace an older tool and enhance and optimize the data annotation process with new features such as export-import functions, data visualization, and a notification system. Comprehensive analysis and practical testing confirmed that CVAT significantly improves efficiency and quality in team work environments and can successfully replace the old tool.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

Abkürzungsverzechnis

CVAT Computer Vision Annotation Tool

XML Extensible Markup Language

UI User Interface

W3C World Wide Web Consortium

HTTP Hypertext Transfer Protocol

URL Uniform Resource LocatorURLs

JSON JavaScript Object Notation

API Application Programming Interface

SQL Structured Query Language

MVC Model-View-Controller

MVT Model-View-Template

OLAP Online analytical processing

DBMS Database Management Systems

KPI Key Performance Indicators

IP Internet Protokoll

UML Unified Modeling Language

CSRF Cross-Site Request Forgery

Glossar

.NET ist eine Softwareplattform und ein Framework. 8

API ist eine Sammlung von definierten Regeln und Protokollen, die es verschiedenen Softwareanwendungen ermöglichen, miteinander zu kommunizieren und auf die Funktionen und Daten der anderen Anwendung zuzugreifen. APIs fungieren als Schnittstellen, die den Informationsaustausch und die Interaktion zwischen unterschiedlichen Programmen ermöglichen. Sie ermöglichen es Entwicklern, Dienste und Funktionen in ihren Anwendungen zu integrieren, ohne den gesamten Quellcode neu schreiben zu müssen. APIs sind entscheidend für die Entwicklung von Softwareanwendungen und die nahtlose Verbindung von verschiedenen Diensten und Systemen. 11, 26, 31

Attribut ist eine spezifische Eigenschaft oder Information, die mit einem Objekt, einer Entität oder einem Datenelement verbunden ist. Attribute dienen dazu, zusätzliche Informationen über das betreffende Objekt oder die betreffende Entität bereitzustellen und sie genauer zu beschreiben. 3, 6, 10, 17, 18, 29, 35, 36

C ist eine General-Purpose-Programmiersprache. 8

DAT-Format ist ein allgemeines Dateiformat, das zur Speicherung von Daten verwendet wird. Es ist nicht standardisiert und sein Inhalt variiert je nach der Anwendung, die es erzeugt. DAT-Dateien können sowohl binäre als auch Textinformationen enthalten. 7, 21, 23

Datenbank ist eine strukturierte Sammlung von Informationen oder Daten, die elektronisch gespeichert und verwaltet werden. Datenbanken dienen dazu, Daten effizient zu organisieren, zu speichern, abzurufen und zu verarbeiten. Sie bestehen aus Tabellen oder Dateien, in denen Daten in Zeilen und Spalten organisiert sind. Jede Zeile in einer Datenbanktabelle repräsentiert einen Datensatz, während die Spalten die verschiedenen Attribute oder Felder dieses Datensatzes darstellen [1]. 9, 10, 15, 21, 30, 36, 37

Framework ist eine vorgefertigte Struktur oder ein vorgefertigter Satz von Tools, Bibliotheken und Regeln, der entwickelt wurde, um die Entwicklung von Anwendungen, Websites oder Software zu erleichtern. 8

HTML steht für "Hypertext Markup Language" und ist die Standardzeichensprache zur Erstellung von Webseiten und Webanwendungen. HTML wird verwendet, um die Struktur und den Inhalt einer Webseite zu definieren, indem es Elemente und Tags verwendet, um Text, Bilder, Links und andere Medienelemente auf einer Webseite zu organisieren und darzustellen. 9, 10, 31, 32, 41, 49, 50

Java ist eine Programmiersprache und Computing-Plattform. 15

Node.js ist eine Open-Source-Laufzeitumgebung, die auf der JavaScript-Plattform von Google Chrome basiert. 8

NumPy ist eine leistungsstarke Python-Bibliothek, die für numerische Berechnungen und Datenverarbeitung entwickelt wurde. 8

OpenCV ist eine riesige Open-Source-Bibliothek für Computer Vision, maschinelles Lernen und Bildverarbeitung. 8

Panel ist Grundbaustein in Grafana, bestehend aus einer Abfrage und einer Visualisierung. Kann innerhalb eines Dashboards verschoben und in der Größe geändert werden. 15

PyPI steht für "Python Package Index" und ist ein Repository, in dem Python-Softwarepakete gehostet und von Entwicklern aus der Python-Community geteilt werden. 8

Sequenzdiagramm ist ein grafisches Werkzeug in der Softwareentwicklung, das die Reihenfolge der Interaktionen zwischen Akteuren oder Systemkomponenten (wie Benutzern und Modulen) zeigt. Vertikale Linien repräsentieren die Teilnehmer, während horizontale Pfeile die Kommunikation zwischen ihnen darstellen. 21, 23, 27

SQLite ist eine leichte, in sich geschlossene SQL-Datenbank, die ohne separaten Server funktioniert und in zahlreichen Anwendungen und Geräten integriert ist. 7, 8, 21–23

Vector ist ein Open-Source-Tool zur Observability-Datenverarbeitung, das das Erfassen, Anreichern und Weiterleiten von Protokollen, Metriken und Traces ermöglicht. Vector zeichnet sich durch hohe Effizienz und Anpassbarkeit aus und wird zur Optimierung von Datenflüssen sowie zur Kostensenkung in Unternehmen eingesetzt [2]. 30, 36, 37

Visualisierung ist eine grafische Darstellung der Abfrageergebnisse. 15

Yandex ist ein Technologieunternehmen, das eine Vielzahl von Internetdiensten und Softwareprodukten anbietet. 15

1 Einleitung

1.1 Motivation

In dieser Bachelorarbeit wird die Optimierung des Prozesses der Annotation von Daten in einer bestehenden Toolkette oder Prozesskette (3.1) untersucht. Die Toolkette ermöglicht die Durchführung von Annotationen für visuelle Daten, ist jedoch mit einigen Herausforderungen verbunden. Im Rahmen dieser Arbeit werden die bestehenden Abläufe und Prozesse analysiert, um Optimierungspotentiale zu identifizieren.

Die Motivation für diese Bachelorarbeit liegt in der Notwendigkeit, die Leistungsfähigkeit und Skalierbarkeit von Labeling-Tools zu untersuchen. Zwei Labeling-Tools, nämlich ein bereits etabliertes oder alters Labeling-Tool und ein neues Web-Labeling-Tool CVAT, stehen im Mittelpunkt der Untersuchung. Durch die praktische Anwendung beider Tools und die Durchführung der Labeling-Prozesskette wird einen tiefen Einblick in ihre Funktionalitäten und Anwendungsgebiete gewinnen. Beide Tools werden aktiv für den Annotationsprozess genutzt, um eine umfassende Vergleichsanalyse durchzuführen.

Das Hauptziel ist es, nicht nur die technischen Aspekte dieser Tools zu verstehen, sondern auch herauszufinden, wie sie in realen Anwendungen eingesetzt werden können. Das Ziel ist es auch, die Prozesskette von der Datenaufbereitung über bis hin zur Modellierung sachlich und logisch zu verstehen und zu optimieren. Insbesondere wird angestrebt, das CVAT so anzupassen, dass es das bestehende Labeling-Tool erfolgreich ersetzen kann.

1.2 Ziel der Arbeit

Das Hauptanliegen dieser Bachelorarbeit liegt darin, ein umfassendes Verständnis der Funktionen und Anwendungsmöglichkeiten der zwei Labeling-Tools zu erlangen. Konkret sollen die folgenden Ziele erreicht werden:

- **Untersuchung und Vergleich von Labeling-Tools:** Die eingehende Untersuchung und der Vergleich der zwei Labeling-Tools sollen dazu dienen, die Stärken und Schwächen im Hinblick auf die Datenkennzeichnung zu identifizieren. Dies umfasst die Evaluierung der Benutzerfreundlichkeit, der Leistung und der Flexibilität der Tools.
- **Praktische Anwendung der Labeling-Prozesskette:** Die vollständige Durchführung der Prozesskette von der Datenaufbereitung über das Labeling bis hin zur Modellentwicklung hat zum Ziel, die Integration beider Labeling-Tools in realen Szenarien zu verstehen.
- **Anpassung des CVATs:** Das CVAT wird so angepasst, dass es das bereits altes Labeling-Tool erfolgreich ersetzen kann. Dies beinhaltet die Implementierung von erforderlichen Funktionen und die Gewährleistung einer nahtlosen Integration in den Labeling-Prozess.
- **Qualitätsverbesserung und Effizienzsteigerung:** Die Arbeit strebt an, die Qualität der annotierten Daten zu verbessern und den Labeling-Prozess insgesamt effizienter zu gestalten. Dies wird durch die Anwendung von Best Practices und Optimierungen in den Labeling-Tools erreicht.
- **Beitrag zur Forschung und Anwendung:** Das Ziel ist es, nicht nur zu einem besseren Verständnis von Labeling-Tools beizutragen, sondern auch praktische Erkenntnisse für die Anwendung von Computer Vision und maschinellem Lernen in verschiedenen Anwendungsbereichen bereitzustellen.

1.3 Das Unternehmen

Intenta ist auf den Gebieten der Bildverarbeitung, Datenfusion sowie der Erkennung von Objekten und Personen tätig. Das Unternehmen entwickelt und vertreibt Produktlinien, die auf dem Intenta S2000 Sensor basieren. Seit der Gründung im Jahr 2011 verzeichnet das Unternehmen kontinuierliches Wachstum und setzt auf ein engagiertes Team, das selbst anspruchsvollste Aufgaben bewältigen kann. Das Leistungsspektrum von Intenta umfasst die Entwicklung und Portierung von Algorithmen, prozessgerechte Softwareentwicklung, Bildverarbeitung und -analyse, modellbasierte Schätzverfahren, Multisensor-Datenfusion sowie die Erkennung und Verfolgung von Objekten. Intenta ist nach DIN EN ISO 9001 zertifiziert und legt einen Fokus auf Kundenorientierung und Qualität. Die verschiedenen Abteilungen von Intenta, darunter Stereosicht, Algorithmen zur Objekt- und Mustererkennung, Algorithmen zur Verhaltensanalyse, Verfolgung, Entwicklung maßgeschneiderter Lösungen und die Abteilung für intelligente Sensoren, tragen zur Vielfalt der angebotenen Dienstleistungen bei [3].

2 Technische Grundlagen

Die dargestellten Fachgrundlagen bilden das solide Fundament, auf dem die Arbeit aufbaut. Sie dienen als Leitfaden zum Verständnis der nachfolgenden Kapitel in der Gesamtheit.

2.1 Labeln

Im Rahmen der Arbeit soll man ein Verständnis dafür haben, wie das manuelle Labeln geht, da das Labeln eine zentrale Aufgabe in beiden Tools ist.

Das manuelle Labeln oder Markieren, auch als "manuelle Annotation" bekannt, ist ein entscheidender Schritt bei der Vorbereitung von Daten für den Einsatz in maschinellen Lernsystemen, insbesondere in Computer Vision-Anwendungen. Dabei werden semantische Informationen, Kategorien oder Attribute manuell den Datenelementen zugeordnet, darunter Bilder, Videos, Texte und andere Arten von Eingabedaten. Das manuelle Markieren kann von menschlichen Annotatoren durchgeführt werden, um die Daten für Trainings- und Testzwecke vorzubereiten.

Eine der Hauptaufgaben beim manuellen Labeln besteht darin, Daten, wie beispielsweise Bildern, Kategorien oder Klassifikationen zuzuweisen. Dies könnte die Identifizierung von Objekten wie Auto, Hund, Flugzeug oder Person in Bildern oder die Zuordnung von Texten zu bestimmten Themen umfassen.
Neben der reinen Klassifizierung können beim manuellen Labeln auch spezifische Attribute oder Merkmale zu Daten hinzugefügt werden. Dies kann Informationen wie Farben, Größen, Formen oder beliebige zusätzliche Details umfassen.

2.2 Webanwendung

Eine Webanwendung ist weitaus komplexer als eine einfache Webseite. Im Kern handelt es sich um eine Client/Server-Anwendung, die auf dem Zusammenspiel zwischen einem Client und einem Server basiert [4].

Der Client ist die Nutzerseite einer Anwendung. Typischerweise handelt es sich dabei um einen Webbrowser, der auf dem Computer, Tablet-PC oder Mobilgerät des Benutzers läuft. Aufgabe des Clients ist es, Anfragen an die Webanwendung zusenden und die Informationen, die vom Server zurückgegeben werden, anzuzeigen.
Der Server ist der Hauptcomputer oder eine Gruppe von Computern, auf denen die Webanwendung gehostet wird. Der Server kümmert sich um die Bearbeitung der Anfragen des Clients, wie die Abbildung 2.1. Das bedeutet, er nimmt die Anfragen entgegen, führt die notwendigen Operationen aus, um die angeforderten Daten zu generieren, und sendet dann die Antwort an den Client zurück.

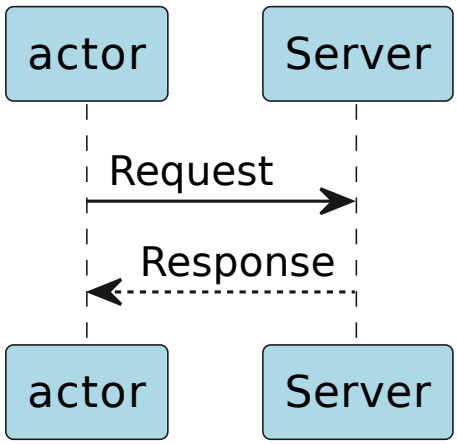

Abbildung 2.1: Client/Server–Architektur

Die Kommunikation zwischen Client- und Serverprogrammen erfolgt über Befehlszeileninteraktionen, die als grundlegende Form dienen. Dieses Konzept ist in Internetkommunikationsprotokollen weitverbreitet und ermöglicht es auch einfachen, auf Befehlszeilen basierenden Benutzeroberflächen, TCP/IP-Dienste zu nutzen [4].
Nachdem die Verbindung zum Server erfolgreich hergestellt wurde, gibt der Client Befehle zeilenweise ein. Es gibt sowohl einzelne Befehlszeilen als auch Blockbefehle. Ein Blockbefehl beginnt mit einer besonderen Zeile, die den Beginn des Befehls kennzeichnet, und endet mit einer weiteren Zeile, die das Ende des Befehls markiert. Der Server antwortet auf jeden Befehl normalerweise mit einer Zeile, die einen Antwortcode enthält [4].
Ein zustandsbehaftetes Protokoll erlaubt es, mehrere Befehle in einer Anfrage zu senden, und

der Server muss den Zustand der Verbindung über alle aufeinanderfolgenden Befehle hinweg aufrechterhalten [4].
Im Gegensatz dazu ist HTTP ein zustandsloses Protokoll. Eine HTTP-Anfrage enthält normalerweise einen einzelnen Blockbefehl und führt zu einer einzelnen Antwort [4].

2.3 Die verwendeten Tools

Diese Arbeit bietet einen Überblick über verschiedene Tools und Technologien, die zur Unterstützung spezifischer Forschungs- und Entwicklungssegmente dienen. Diese werden in diesem Kapitel ausführlich erötert.

2.3.1 CVAT

CVAT ist eine webbasierte Plattform, die als Open-Source-Tool entwickelt wurde und von Intel stammt. Es ermöglicht die Annotation von Bildern und Videos, um Daten für Anwendungen im Bereich der Computer Vision vorzubereiten. Durch seine Funktionen hat das Tool eine breite Anwendung im Bereich des maschinellen Lernens und der Computer Vision gefunden.[5].

Das Tool ermöglicht das Annotieren einer Vielzahl von Objekten mit unterschiedlichen Attributen, was in der Computervision von zentraler Bedeutung ist. Es bietet eine Reihe von Funktionen, die den Prozess der Annotation vereinfachen, darunter eine benutzerfreundliche Schnittstelle. Im Folgenden werden die wichtigsten Funktionen von dem Tool vorgestellt:

2.3.1.1 Annotationstools

Es gibt viele Annotationstools in dem Tool, um die Anmerkungen von Objekten in Bilder oder Videos vorsehen zu können. Dazu gehören Bounding-Boxen, Polygone, Punkte sowie Textannotationen. Durch diese Vielseitigkeit ist es möglich, eine exakte Markierung diverser Objekte und Muster zu erzielen [6].

Die Arbeitsfläche vom Tool, wie in der Abbildung (2.2) dargestellt, bietet eine Vielzahl von Annotationstools auf der linken Seite des Bildschirms. In der Abbildung wurden Bounding-Boxen, Rechteck und Tagsannotation-Werkzeuge verwendet, um Objekte im Bild zu markieren und zu beschreiben. Ein entscheidendes Werkzeug für diese Entwicklungsarbeit ist die Tagsannotation, mit der spezifische Informationen zum Bild hinzugefügt werden können. In diesem Beispiel wird die Tagsannotation verwendet, um die Tageszeit zu beschreiben. Diese Tagsannotation

enthält das Attribut “Tageszeit”, das zwei Zustände aufweist: Tag und Nacht. Die Auswahl des Zustands erfolgt einfach über die rechte Seite im Bereich Details des Attributs “Tageszeit”.

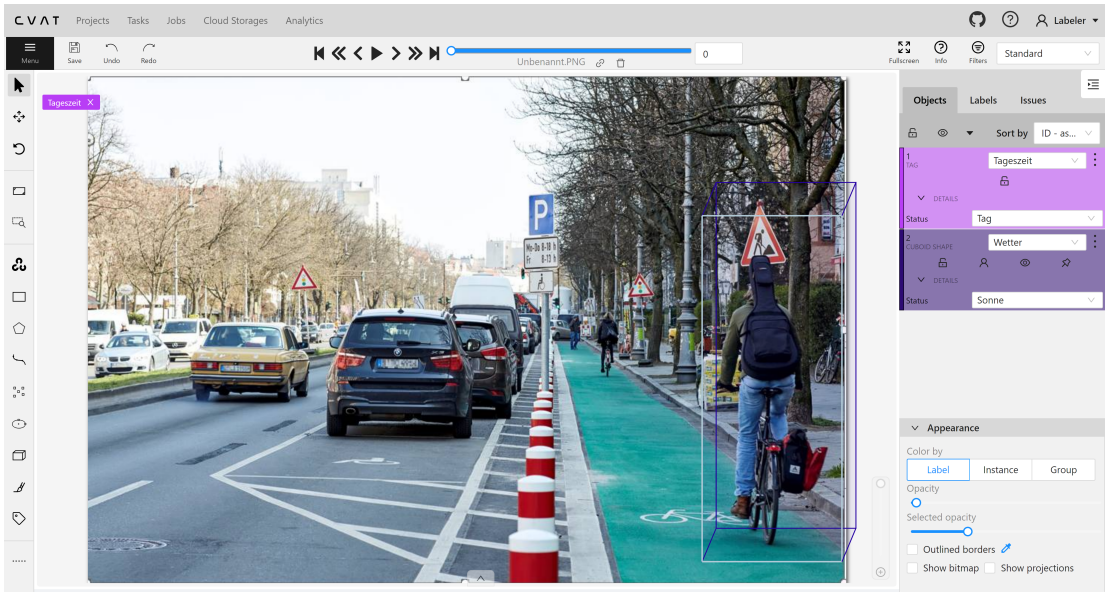

Abbildung 2.2: Arbeitsfläche von CVAT

2.3.1.2 Datenimport und -export

Das Tool bietet Unterstützung für eine Vielzahl von Dateiformaten beim Import und Export von Bild- und Videodateien. Die Dateiformate sind verschiedene Formate: XML, JSON… mit verschiedenen Schemas. Diese Flexibilität erleichtert die nahtlose Integration in bestehende Datenquellen und Arbeitsabläufe [7].

2.3.1.3 Überprüfungsmodus

In dem Tool steht nach der Annotation eine Funktion zur Verfügung, die es ermöglicht, die Anmerkungen von einem Gutachter überprüfen und kommentieren zu lassen, um eventuelle Fehler zu identifizieren [8].

2.3.2 Das alte Tool

Das alte Tool ist eine Windows-basierte Plattform, entwickelt von Intenta (1.3). Es handelt sich um ein 3D-Tool, dessen Hauptaufgabe das Labeling ist. Es ermöglicht die Annotation von

Bildern und Videos. Um mit dem Labeling zu beginnen, müssen sowohl das DAT-Format als auch das SQLite-Format für die Videodatei in das Tool importiert werden. Die annotierten Daten können ausschließlich als SQLite-Format exportiert werden .

Abbildung 2.3: Arbeitsfläche vom alten Tool

Das Tool verfügt über zwei Modus: den Labeling-Modus und den Review-Modus. Die Benutzeroberfläche dieses Tools ist komplex gestalte. Sie besteht aus einem Bündel von mehreren interaktiven Elementen und Fenstern. Diese Fenster können verschiedene Funktionen und Informationen enthalten und anzeigen, wie z. B. Symbolleisten, Menüs, eine Vorschau von 3D-Objekten, Eigenschaftsfelder und so weiter. Alle diese Fenster und Elemente müssen nach Bedarf platziert und angepasst werden, damit der Nutzer effizient arbeiten kann.

2.3.3 Python

Python ist eine interpretierte Programmiersprache, die objektorientierte Prinzipien unterstützt und eine interaktive Nutzung ermöglicht. Sie zeichnet sich durch eine dynamische Typisierung und die Fähigkeit aus, komplexe Datenstrukturen zu verwalten. Die Sprache unterstützt verschiedene Programmierstile, darunter objektorientierte, prozedurale und funktionale Ansätze. Die Syntax von Python ist auf Klarheit und Lesbarkeit ausgerichtet, was die Sprache für ein breites Spektrum von Anwendungen geeignet macht. Python kann Schnittstellen zu vielen

Systemfunktionen und Bibliotheken bereitstellen und erlaubt die Anbindung an mehrere grafische Benutzeroberflächen. Weiterhin kann sie mit C oder C++ für spezifische Anforderungen erweitert werden und eignet sich als Skriptsprache für erweiterbare Anwendungen.
Python ist plattformunabhängig und funktioniert auf verschiedenen Betriebssystemen, einschließlich Unix-Varianten, macOS und Windows[9].

Python bietet eine umfassende Standardbibliothek für gängige Programmieraufgaben und wird ergänzt durch eine Vielzahl von Drittanbieter-Bibliotheken für spezialisierte Anwendungen, wie Bildverarbeitung mit OpenCV oder numerische Berechnungen mit NumPy und PyPI. [10].

2.3.4 JavaScript

JavaScript ist eine dynamische Programmiersprache, die anfangs für dynamische Webseitengestaltung konzipiert wurde und sich mittlerweile auf verschiedene Plattformen ausdehnt. Ihre Anwendungsbereiche umfassen clientseitiges Scripting in Webbrowsern, serverseitige Entwicklung mit Technologien wie .NET und Node.js, sowie die Erstellung von Desktop- und mobilen Anwendungen und das Scripting auf Befehlszeilenebene [11].
JavaScripts Flexibilität und Skalierbarkeit fördern eine schnelle Einarbeitung und Innovation. Dennoch ist es empfehlenswert, sich mit den spezifischen Mustern und Paradigmen vertraut zu machen, um die Möglichkeiten der Sprache voll auszuschöpfen [11].

2.3.5 Django-Framework

In CVAT wurde Django-Framework ursprünglich in der Anwendung für das Backend verwendet.

Django ist ein (2.3.3) Python-basiertes Web-Framework auf High-Level-Ebene, das eine schnelle und effiziente Erstellung von Webanwendungen ermöglicht. Django bietet integrierte Funktionen für alle Anforderungen, einschließlich der Django-Administrationsschnittstelle und der Standarddatenbank SQLite3. Der Aufbau des Textes ist logisch und ermöglicht einen reibungslosen Informationsfluss mit kausalen Zusammenhängen zwischen den Aussagen [12].

Außerdem ist die Django-Struktur hochgradig und basiert auf etablierte MVC-Muster, das wird auch als MVT bezeichnet. Django übernimmt selbst die Arbeit des Controllers innerhalb des MVC und diese Arbeit wird durch die Verwendung von Templates erledigt [13].

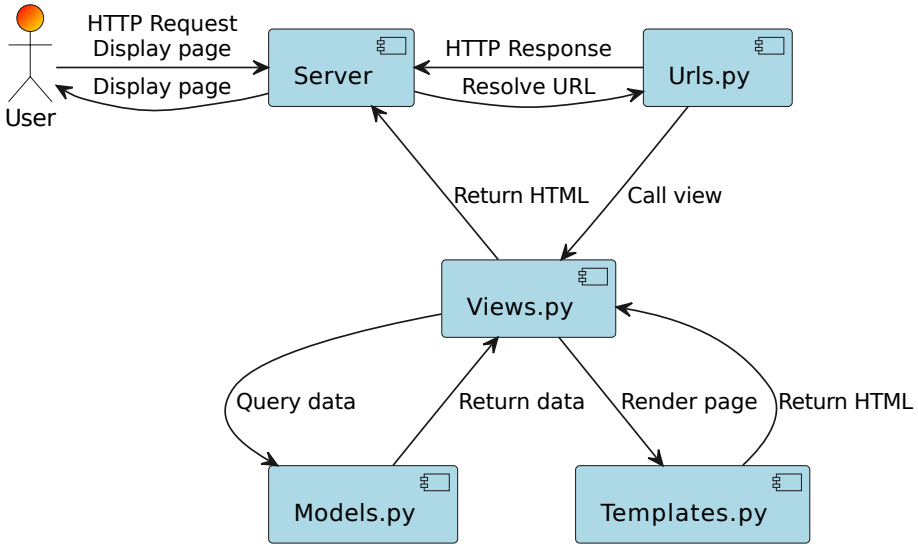

Abbildung 2.4: Django Webanwendungsarchitektur:

Die Abbildung (2.4) zeigt den Arbeitsablauf von Django. Eine Benutzeranfrage wird zunächst vom Webserver an Django weitergeleitet. Django leitet die Anfrage anhand der URL-Konfiguration in der "**urls.py**"-Datei an die entsprechende Ansicht weiter. Die Ansicht, definiert in "**views.py**", verarbeitet die Anfrage und interagiert bei Bedarf mit dem Datenmodell, um Informationen zu holen oder zu aktualisieren. Anschließend wird das Ergebnis in einem Template gerendert, was typischerweise eine HTML-Seite generiert. Die vollständige Antwort, die nun die angeforderten Daten in formatierter Form enthält, wird schließlich vom Server an den Benutzer zurückgeschickt.

Diese Architektur ermöglicht die Trennung von Datenbanklogik und Geschäftslogik in einer Django-Anwendung. Im Folgenden werden die Schlüsselkomponenten und Prinzipien der Django-Architektur erläutert:

2.3.5.1 Modell (Model)

Das Modell in Django bildet die Datenbank ab und definiert die Struktur und den Zugriff auf die Daten. Jedes Modell wird normalerweise als eine Python-Klasse definiert, die Unterklassen

“django.db.models.Model” aufweist. Jedes Attribut des Modells stellt ein Datenbankfeld dar [14].

2.3.5.2 Ansicht (View)

Die Ansicht in Django ist eine Python-Funktion, die die Verantwortlichkeit hat, die HTTP-Anfragen entgegenzunehmen und die HTTP-Antworten zurückzugeben. Die Ansichten definieren, wie die Daten aus der Datenbank abgerufen, verarbeitet und an Templates übergeben werden [15].

2.3.5.3 URL

Die Ansicht-Funktionen können genutzt werden, um HTTP-Anfragen effizienter zu verarbeiten, indem jede Ressource einzeln verarbeitet wird. Im Gegensatz dazu können URLs über eine einzelne Funktion gemeinsam verarbeitet werden. Ansicht-Funktionen sind in der Lage, jede Ressource individuell zu verarbeiten, indem sie die URLs einzeln verarbeiten. Des Weiteren können Ansicht-Funktionen Daten von einem URL-Mapper empfangen, welcher spezifische Literalzeichenfolgen oder Zeichen in einer URL abgleicht und diese als Daten weiterleitet [16].

2.3.5.4 Template

Das Template ist verantwortlich für die Darstellung und Präsentation von Daten auf der Webseite. Es handelt sich um eine textbasierte Datei, die das HTML-Layout und die Darstellung der Webseite definiert. Django nutzt eine spezielle Template-Sprache, die es erlaubt, dynamische Inhalte auf HTML-Seiten zu integrieren [17].

2.3.6 React-Framework

In CVAT wurde React ursprünglich in der Anwendung für die Benutzerfläche verwendet.

React ist ein (2.3.4) JavaScript-Framework, das anfangs bei Facebook im Jahr 2013 entwickelt wurde, um die Erstellung dynamischer Benutzeroberflächen zu vereinfachen, insbesondere in Situationen, in denen sich Daten häufig ändern. Die Hauptziele bei der Entwicklung von React waren die Wartbarkeit und Skalierbarkeit, insbesondere für große Anwendungen wie bei Facebook. Ursprünglich im Werbebereich von Facebook entstanden, bot das Framework eine alternative Architektur zum damals üblichen MVC-Ansatz [18].

2.3.7 Webhooks

Der Datenaustausch in der vernetzten Systemen und Anwendungen zwischen verschiedenen Plattformen ist notwendig. Um in Echtzeit auf Veränderungen und Ereignisse reagieren zu können, sind Webhooks in der Welt der Softwareentwicklung und Integration unverzichtbar geworden. Es ist von großer Bedeutung, das Konzept der Webhooks zu verstehen, da es eines Teil der Arbeit ist.

Webhooks können als eine Art von Schnittstelle betrachtet werden, die durch bestimmte Ereignisse ausgelöst wird, anstatt auf Anfragen zu reagieren. Sie ermöglichen es Anwendungen, automatisch Nachrichten oder Informationen an andere Anwendungen zu senden, wenn ein bestimmtes Ereignis auftritt. Um es einfacher auszudrücken: Ein Webhook funktioniert wie ein automatisches Benachrichtigungssystem zwischen verschiedenen Anwendungen über ein spezifisches Ereignis. Sobald eine Veränderung in einer Anwendung stattfindet, übermittelt der Webhook diese Information sofort an eine andere Anwendung [19].
In einigen Kontexten werden Webhooks auch als "Reverse APIs" bezeichnet. Dieser Begriff spiegelt wider, dass bei Webhooks die sendende Anwendung die Kommunikation initiiert, im Gegensatz zu einer typischen API, bei der die empfangende Anwendung die Anfrage stellt [19].

Die Webhooks und APIs ermöglichen die Weitergabe von Daten zwischen zwei Systemen oder Anwendungen, aber die beider unterscheiden sich mit folgenden Beschreibung:

Feature	Beschreibung
API	Anfragen-basiert, wird aktiviert, wenn Anfragen von einer Drittanwendung kommen. Der Nutzer sendet und fordert Daten an.
Webhook	Ereignis-basiert, wird ausgeführt, wenn ein spezifisches Ereignis in der Quellanwendung auftritt. Daten werden automatisch gesendet.

Tabelle 2.1: Unterschied zwischen API und Webhook [19]

Das Diagramm in der Abbildung (2.5) stellt den Prozess dar, bei dem ein Client sich für Webhooks bei einem Server anmeldet und wie anschließend Ereignisse zwischen ihnen kommuniziert werden:

- **Registrierung für Webhooks:** Der Client schickt eine Anfrage an den Server, um sich für Webhooks zu registrieren. In dieser Anfrage sind Informationen zur Authentifizierung enthalten, die dem Client erlauben, Webhooks zu empfangen.

- **Post-Anfrage mit Daten:** Wenn ein Ereignis auftritt, sendet der Server eine POST-Anfrage an die Webhook-URL des Clients. Diese Anfrage beinhaltet die Ereignisdaten.
- **Ereignisverarbeitung und Bestätigung:**
  - Bei erfolgreicher Zustellung validiert und verarbeitet der Client die Daten und sendet eine Bestätigung zurück an den Server.
  - Bei einer fehlerhaften Zustellung sendet der Client eine Fehlermeldung zurück.
- **Weitere Ereignisse:** Der Client bleibt für den Empfang und die Verarbeitung weiterer Ereignisse bereit, die vom Server gesendet werden.

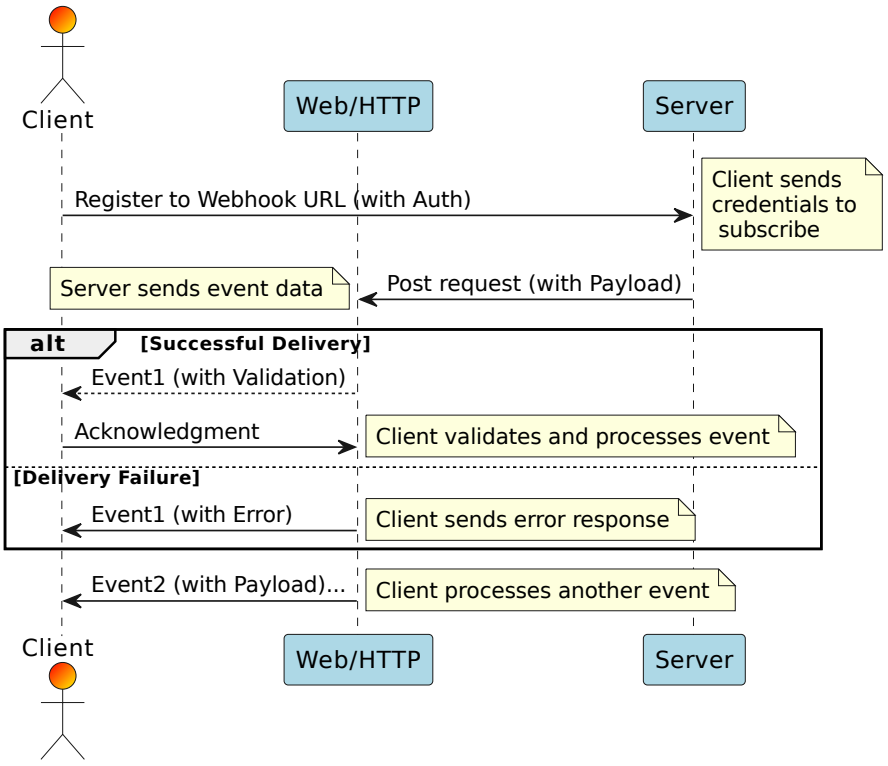

Abbildung 2.5: Webhooks-Kommunikationsfluss im Client-Server

2.3.8 Docker

Docker ist ein Tool, das die automatisierte Bereitstellung von Anwendungen in Containern ermöglicht. Es basiert auf dem Konzept der Containerisierung, bei dem jede Anwendung in

einem separaten Container ausgeführt wird. Diese Container sind standardisierte Softwarepakete, die alle erforderlichen Abhängigkeiten und Konfigurationen enthalten [20].

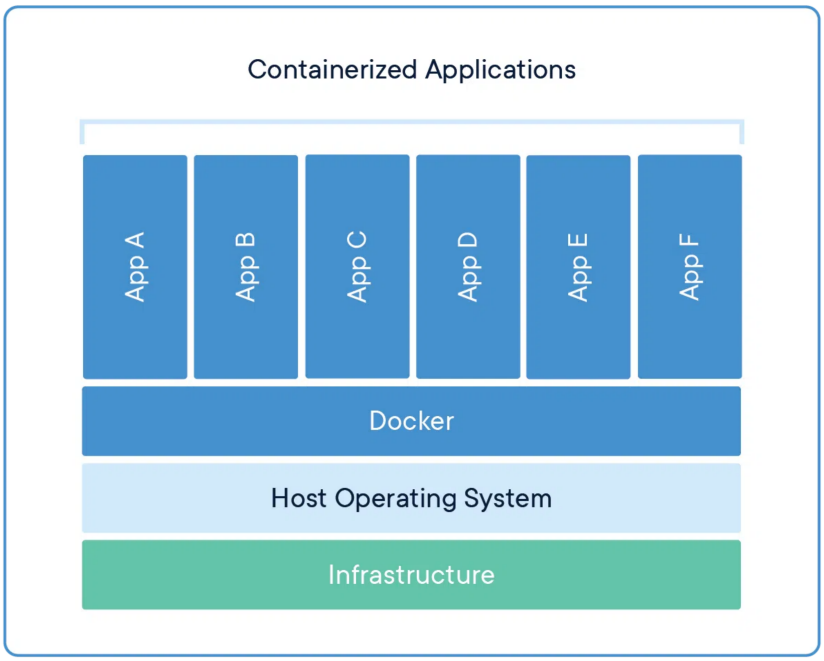

Abbildung 2.6: Containerisierte Anwendungen in Docker [20]

Docker-Images, die als Vorlagen für Container dienen, können in einem zentralen Repository gespeichert und von Entwicklern genutzt werden. Werkzeuge wie Dockerfile und Docker-Compose erleichtern die Erstellung und Verwaltung dieser Container [20].

2.3.8.1 Dockerfile

Bei einer Dockerfile handelt es sich um eine Textdokument, die eine Schritt-für-Schritt-Abfolge von Anweisungen für die Erstellung eines Docker-Images enthält. Diese Anweisungen reichen von der Auswahl des Basis-Images über die Installation von Softwarepaketen bis hin zur Konfiguration von Umgebungsvariablen. Dockerfiles ermöglichen die Automatisierung des

Erstellungsprozesses und die Reproduzierbarkeit von Images unabhängig von der Entwicklungs- oder Produktionsumgebung [21].

2.3.8.2 Docker-Compose

Docker-Compose kann mit dem Multi-Container-Anwendungen definiert und verwaltet werden. Mit Docker-Compose kann eine einzige Konfigurationsdatei erstellt werden, die alle Container, Netzwerke und Volumen für Ihre Anwendung definiert. Dies erleichtert die Zusammenstellung und Orchestrierung von Containern, insbesondere in komplexen Anwendungsszenarien, in denen mehrere Container zusammenarbeiten müssen [22].

2.3.8.3 Docker-Network

In der Container-Technologie ermöglichen Container-Netzwerke die Kommunikation zwischen Containern sowie die Verbindung zu anderen Arbeitslasten außerhalb des Docker-Ökosystems. Container haben keine eigene Kenntnis über die Netzwerkdetails, an die sie angeschlossen sind. Sie interagieren über eine Netzwerkschnittstelle, die mit IP-Adresse und weiteren Netzwerkparametern konfiguriert ist. Standardmäßig sind Container an ein Netzwerk angebunden, es sei denn, sie sind explizit auf den ”None“-Netzwerktreiber eingestellt [23].

Docker stellt verschiedene Netzwerktreiber wie “Bridge”, “Host”, “None” und “Overlay” bereit, um vielfältige Netzwerkkonfigurationen und -anforderungen zu erfüllen. Diese Treiber ermöglichen es Containern innerhalb von Docker, miteinander zu kommunizieren und sich an unterschiedliche Netzwerkumgebungen anzupassen. In der vorliegenden Arbeit wurde speziell der Bridge-Netzwerktreiber eingesetzt [24].

“Bridge”-Netzwerke verwalten ihren eigenen IP-Adressbereich, d. h. jedem Container in einem Bridge-Netzwerk wird eine eigene IP-Adresse zugewiesen. Mithilfe dieser eindeutigen IP-Adressen ist es den Containern möglich, miteinander zu kommunizieren. Die Zuweisung eigener IP-Adressen oder die Verwendung von Hostnamen ermöglicht es Containern in einem “Bridge”-Netzwerk, nahtlos miteinander zu interagieren. Dies vereinfacht die Konfiguration und das Management von Container-Anwendungen erheblich und erleichtert die Entwicklung von Anwendungen, die in Containern ausgeführt werden [25].

2.3.9 Grafana-Dashboard

Grafana ist ein Open-Source-Tool, das entwickelt wurde, um Daten zu überwachen und visuell darzustellen. Es erlaubt Benutzern die Erstellung personalisierter Dashboards, die Daten aus verschiedenen Quellen sammeln und aufbereiten. Diese Dashboards bieten einen umfassenden Überblick über die Daten und erleichtern das Verständnis komplexer Systeme und Abläufe. [26].

Ein Dashboard ist eine Zusammenstellung aus einem oder mehreren Panels, die in einer oder mehreren Zeilen organisiert und strukturiert sind. Grafana stellt eine breite Palette vordefinierter Panels zur Verfügung, die es Nutzern erleichtern, präzise Abfragen zu erstellen und die Visualisierung an die individuellen Anforderungen anzupassen. Diese Flexibilität ermöglicht die Erstellung maßgeschneiderter Dashboards, die spezifischen Anforderungen gerecht werden. Jedes Panel kann nahtlos mit Daten aus jeder zuvor konfigurierten Datenquelle in Grafana interagieren [27].

2.3.10 ClickHouse

ClickHouse ist eine Datenbank für Analytik, die als Open-Source-System entwickelt wurde und darauf abzielt, große Datenmengen schnell zu verarbeiten und zu analysieren. Ursprünglich von Yandex entwickelt, hat es sich zu einem etablierten Open-Source-Projekt mit einer wachsenden Gemeinschaft von Entwicklern und Unternehmen auf der ganzen Welt entwickelt [28].

ClickHouse ist ein SQL-basiertes DBMS mit einem spaltenorientierten Speicheransatz. Dies bedeutet, dass Daten in Spalten statt in Zeilen gespeichert werden. Es wurde speziell für die OLAP konzipiert. Als Open-Source-System bietet es die Möglichkeit, große Datenmengen zu analysieren und ist darauf ausgelegt, komplexe Abfragen zeitnah zu bearbeiten und für die Beantwortung einer bestimmten Frage werden nur wenige Spalten ausgewählt [29].

Die Abbildung (2.7) zeigt, dass die ClickHouse sich gut in verschiedene Datenverarbeitungsszenarien integrieren lässt. Nach der Einrichtung von ClickHouse werden die Datenbanken und Tabellen erstellt, um die Daten zu speichern. Daten können aus verschiedenen Quellen in ClickHouse geladen werden. Die Datenbank kann große Datensätze in Echtzeit verarbeiten. ClickHouse ermöglicht eine reibungslose Zusammenarbeit mit verschiedenen Programmiersprachen wie Python (2.3.3), Java, und vielen anderen. Es gibt spezielle Bibliotheken und Treiber, die es ermöglichen, von diesen Sprachen aus auf ClickHouse-Daten zuzugreifen. Dadurch kann

ClickHouse nahtlos in bestehende Anwendungen und Dienste integriert werden.
Diese Tools bieten die Möglichkeit zur Erstellung attraktiver Grafiken, Diagramme und Berichte, um nützliche Erkenntnisse aus den Daten zu gewinnen und sie für Entscheidungsträger verständlich zu präsentieren.

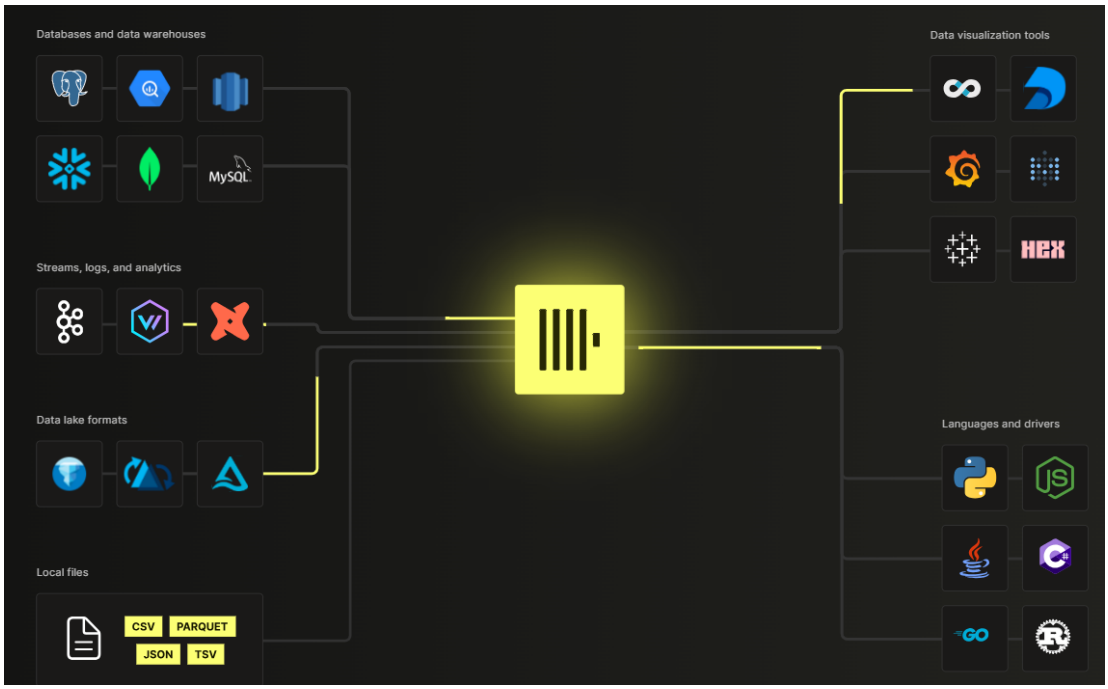

Abbildung 2.7: ClickHouse-Übersicht [30]

2.3.11 XML-Format

Das XML ist ein vielseitiges Instrument, das unabhängig von spezifischer Software oder Hardware sowohl die Speicherung als auch den Datentransport ermöglicht.
Ein charakteristisches Merkmal von diesem Format ist seine Selbstbeschreibungsfähigkeit, mit der Inhalt und Struktur explizit beschrieben werden können. XML -Dokumente können Informationen strukturiert und verständlich präsentieren, was den Austausch und die Interpretation von Daten wesentlich erleichtert. Es sei angemerkt, dass XML als vom W3C empfohlene Technologie etabliert ist, was seine außerordentliche Bedeutung für die Welt der Informationstechnologie unterstreicht [31].
Die nachfolgende Darstellung in der Abbildung (2.8) zeigt die Syntaxregeln für das Verfassen unterschiedlicher Arten von Markup und Text in einem XML -Dokument.

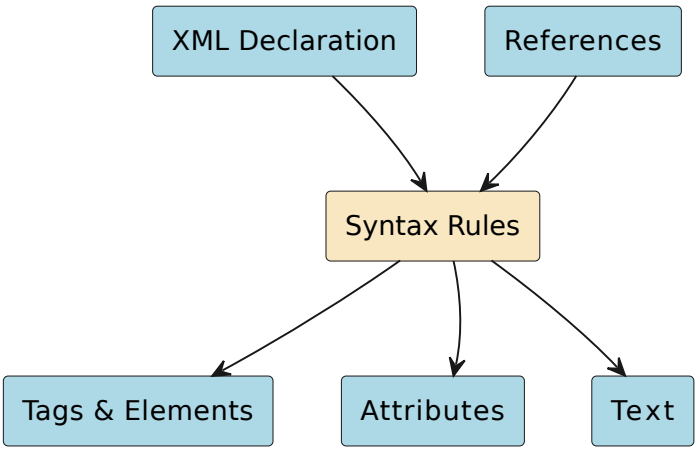

Abbildung 2.8: Syntaxregeln in XML [31]

Die XML-Deklaration ist eine optionale, aber häufig verwendete Zeichenfolge am Anfang eines XML-Dokuments zur Bereitstellung von Informationen über das Dokument und zur Identifizierung des Dokuments für Verarbeitungsanwendungen. Die Deklaration beginnt in der Regel mit <?xml gefolgt von verschiedenen Attribute und endet mit?> [31].

```
<?xml version="1.0" encoding="UTF-8"?>
```

Listing 2.1: XML-Deklaration

XML verwendet Tages, um Elemente zu definieren. Ein Tag besteht aus einem öffentlichen Tag <Element> und einem schließenden Tag <Element>.

```
<element>....</element>
```

Listing 2.2: Tags-Definition in XML

Ein Element kann andere Elemente enthalten und bildet eine hierarchische Struktur. Es gibt ein Wurzelelement, das alle anderen Elemente im Dokument umschließt.

```
<element>
   <sentence>...</sentence>
   <token>...</token>
</element>
```

Listing 2.3: Tags-Struktur in XML

Elemente können Attribute haben, die zusätzliche Informationen über das Element liefern. Attribute werden im öffnenden Tag eines Elements angegeben. Für die
Ein Element kann Textinhalt zwischen seinen öffnenden und schließenden Tags enthalten.

```
<token id="12" stem="tale" id="15">....</token>
```

Listing 2.4: Tags-Struktur in XML

3 Aufgabenstellung

In diesem Kapitel wird erläutert, welche Aufgaben zu bewältigen sind. Zudem wird die bestehende Prozesskette darstellt.

3.1 Prozesskette

Die Abbildung (3.1) veranschaulicht die aktuelle Prozesskette, die im alten Label-Tool stets durchgeführt wurde, um einen erfolgreichen Labeling-Prozess zu erreichen..

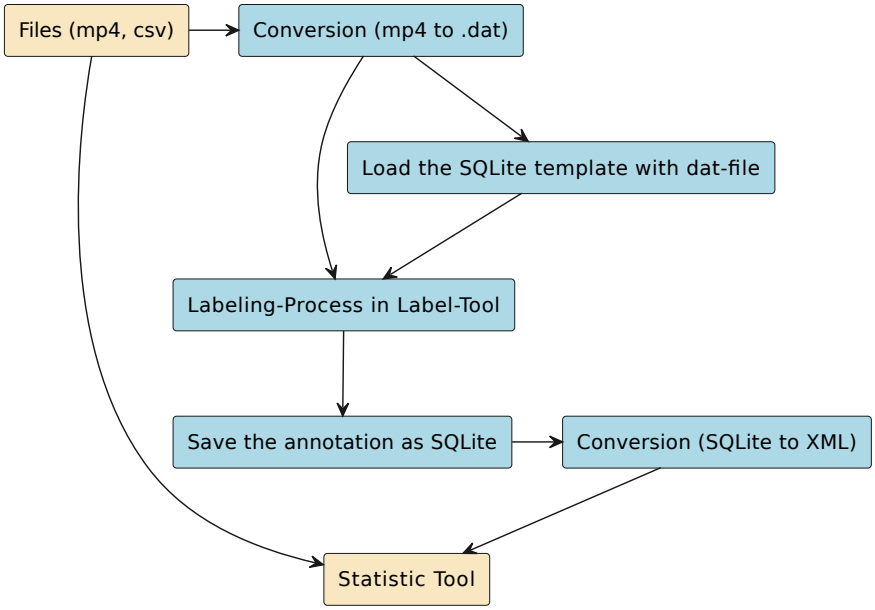

Abbildung 3.1: Die Prozesskette für den Ist-Stand

Die Prozesse, die in Hellblau dargestellt sind, repräsentieren die Aktionen innerhalb des Arbeitsablaufs von dem Labeling-Prozess, die in dem alten Tool stets durchgeführt ist. Der Statistikprozess, hervorgehoben in Hellbraun, bleibt jedoch unverändert, da er in einem externen Tool durchgeführt wird.

3.2 Aufgabenstellung

Im Folgenden werden die zentralen Aufgaben dieser Arbeit dargelegt:

- **Ist-Stand-Analyse:**
  - Analyse und Verständnis der bestehenden Prozesskette (3.1), gefolgt von der Erstellung UML-Diagramme.
  - Praktische Durchführung mit dem alten Tool (2.3.2). Dabei soll zunächst verstanden und erlernt werden, wie das Labeling korrekt im Tool umgesetzt wird.
- **Soll-Stand-Definition:**
  - Realisierung des Labeling-Prozesses mit demselben Beispiel aus der Ist-Stand-Analyse in dem neuen Tool CVAT (2.3.1), zwecks Ermittlung der speziellen Anforderungen für die Entwicklung.
- **Anforderungsdefinition:**
  - Festlegung der notwendigen Anpassungen in CVAT zur Verbesserung des Tools und Integration einer neuen Prozesskette, die zu identischen Ergebnissen wie die derzeitige Prozesskette führt.
  - Festlegung möglicher Features in CVAT zur Verbesserung des Labeling-Prozesses
- **Konzeptentwicklung:**
  - Erstellung eines detaillierten Konzepts für jede definierte Anforderung.
- **Implementierung:**
  - Technische Umsetzung der definierten Anforderungen in CVAT .
- **Testphase:**
  - Überprüfung der implementierten Funktionen.
  - Durchführung von Vergleichstests des Labeling-Prozesses in CVAT , um sicherzustellen, dass die Ergebnisse mit dem alten Tool übereinstimmen.

4 Ist-Stand-Analyse

In diesem Abschnitt wird analysiert, wie die Prozesskette gemäß der Abbildung (3.1) im alten Tool durchgeführt wird. Nach der Durchführung des Labelings für die bestehende Prozesskette wurden die Schritte des Labeling-Prozesses in einem Sequenzdiagramm dargestellt.

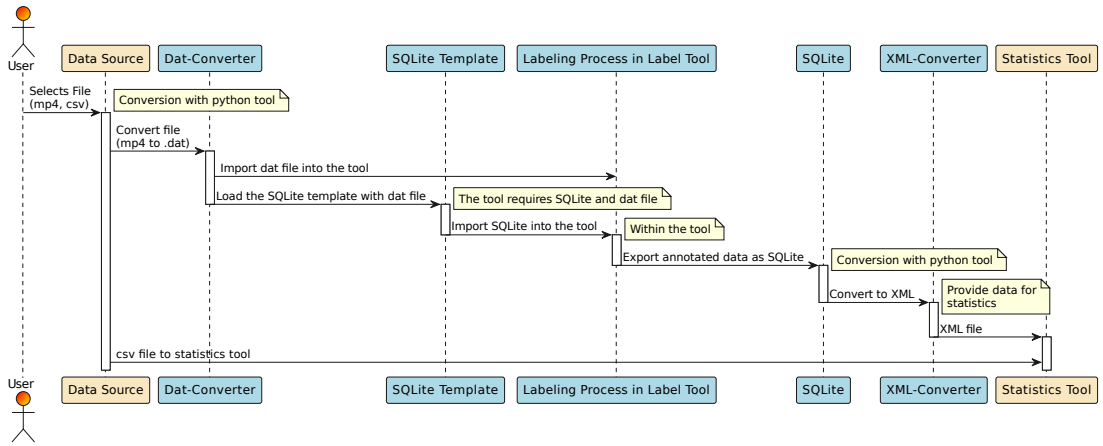

Abbildung 4.1: Sequenzdiagramm des Labeling-Prozesses im alten Tool

Das Prozessdiagramm in der Abbildung (4.1) veranschaulicht einen mehrstufigen Arbeitsablauf, der dem Labeling-Prozess vorausgeht. Der Prozess beginnt damit, dass der Nutzer eine CSV- und MP4-Datei auswählt, die als Eingabe dient. Zunächst wird eine eingehende MP4-Datei mit einem externen Tool in ein DAT-Format konvertiert, um sie für die weiteren Schritte vorzubereiten. Nachdem die Konvertierung erfolgreich abgeschlossen wurde, wird die Dat-Datei zusammen mit einer SQLite-Vorlage in eine SQLite-Datenbank eingefügt. Dieser Schritt ist erforderlich, um die Daten für den Labeling-Prozess im Label-Tool entsprechend zu strukturieren.

Im nächsten Schritt, der den eigentlichen Kern des Prozesses darstellt, werden sowohl die SQLite-Vorlage als auch die Dat-Datei in dem Labeling-Tool geladen. Innerhalb dieses Tools findet der Labeling-Prozess statt, bei dem die eingeführte Datei annotiert und dann die annotierte Datei überprüft werden. Die Ergebnisse dieser Annotation werden in der SQLite-Format ge-

speichert. Dies ermöglicht eine systematische Erfassung und Verwaltung der annotierten Daten.

Nach dem Abschluss des Labeling-Prozesses wird die SQLite-Datei, die nun die annotierten Daten enthält, in ein XML-Format konvertiert. Diese Konvertierung ist ein entscheidender Schritt, da diese Format ein weitverbreitetes Format für den Datenaustausch ist, das von vielen Anwendungen und Skripten verarbeitet werden kann. Insbesondere wird das XML-Datei zusammen mit der CSV-Datei benötigt, um spezielle Skripte auszuführen, die Statistiken über die annotierten Daten generieren. Diese Statistiken sind wichtig, um Einblicke in die Daten zu gewinnen und den Labeling-Prozess zu bewerten.

Die Ergebnisse aus der tatsächlichen Anwendung des Labelings im alten Tool dienen als Vergleichsgrundlage während der Testphase, um die Qualität der Labeling-Ergebnisse in CVAT nach den vorgenommenen Anpassungen zu bewerten.

5 Soll-Stand-Definition

In diesem Abschnitt wird die Prozesskette gemäß der Abbildung (3.1) realisiert. Nach der Einrichtung des neuen Tools und Durchführung des Labelings für das gleiche Beispiel aus der Ist-Stand-Analyse im neuen Tool wurden die Schritte des Labeling-Prozesses auch in einem Sequenzdiagramm in der Abbildung (5.1) dargestellt.

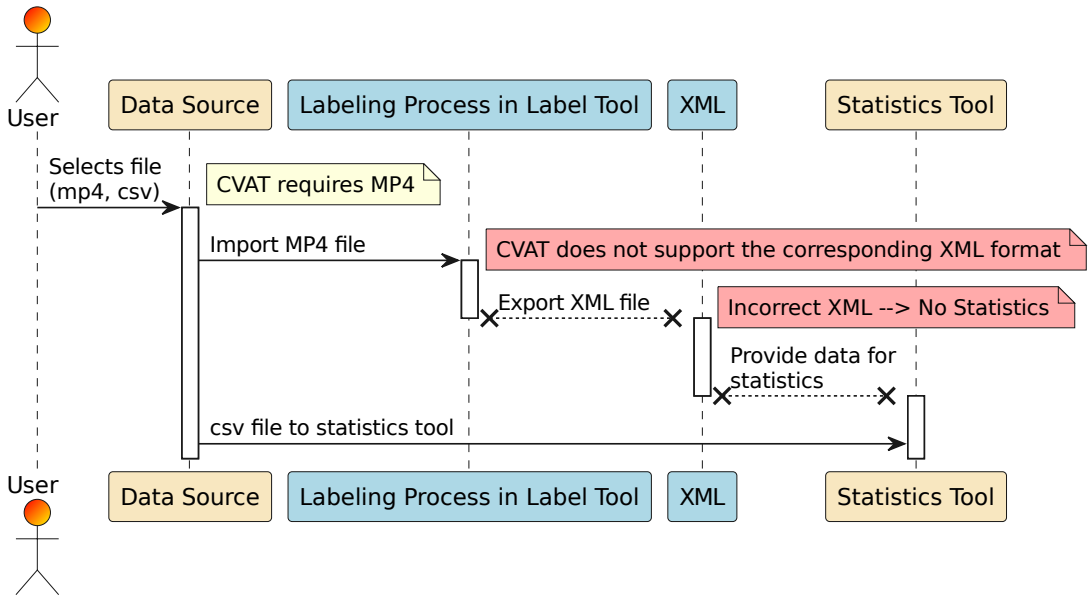

Abbildung 5.1: Sequenzdiagramm des Labeling-Prozesses im neuen Tool

Die aktuelle Prozesskette wurde in CVAT realisiert. Die Schritte des Labeling-Prozesses im neuen Tool sind einfacher und weniger aufwendig als im alten Tool. Dies wird durch den Vergleich der Sequenzdiagramme von CVAT (5.1) und dem alten Tool (4.1) deutlich. Die Konvertierung von MP4 in das DAT-Format und anschließend in SQLite, die vor dem Labeling erforderlich war, ist in dem neuen Tool nicht notwendig, da dieses Tool MP4 und andere gängige Formate für Videos und Bilder unterstützt. Nutzer können somit Dateien direkt in das neue Tool importieren.

Darüber hinaus sind die Labeling-Abläufe wie Annotation und “Review” im neuen Tool effizienter, was teilweise auf die verbesserte Benutzeroberfläche (UI) von CVAT zurückzuführen ist. Nach Abschluss des Labeling-Prozesses in CVAT tritt jedoch eine Einschränkung zutage: Das Tool unterstützt nicht das XML-Format, welches vom Statistiktool für die Auswertung der Annotationen benötigt wird. Dies hat zur Folge, dass der Auswertungsprozess nicht durchgeführt werden kann.

Deutlich wird im Sequenzdiagramm (5.1), welche Anpassungen in dem neuen Tool vorgenommen werden müssen, um den gesamten Prozessablauf erfolgreich zu gestalten. Die notwendigen Anpassungen und Ergänzungen werden im nächsten Kapitel über die Anforderungen definiert.

6 Anforderungen

In diesem Kapitel werden die Anforderungen definiert, die im Rahmen dieser Bachelorarbeit zu erfüllen sind. Eine klare Definition dieser Anforderungen bildet die Grundlage, um ein erfolgreiches Konzept entwickeln und umsetzen zu können. Die Anforderungen sind nach der Analyse des Ist-Standes und der Definition des Soll-Standes definiert. Durch die folgenden und definierten Anforderungen soll das neue Tool angepasst oder erweitert werden:

- **Exportfunktion:** In der aktuellen Prozesskette existieren Auswertungsskripte, die spezifische Formate für annotierte Daten benötigen. Das neue Tool unterstützt dieses erforderliche Format nicht, was eine vollständige Durchführung der Prozesskette im neuen Tool verhindert. Daher ergibt sich die Anforderung, das Tool so anzupassen, dass es mit dem benötigten Datenformat kompatibel wird. Dies ist wesentlich, um eine reibungslose Integration in die bestehende Prozesskette zu ermöglichen und die Funktionalität des Tools vollständig zu sichern.

- **Importfunktion:** Es ist erforderlich, dass das aktualisierte Tool eine konsistente Formatunterstützung für sowohl den Import als auch den Export von Daten bietet. Das bedeutet, dass das Tool in der Lage sein muss, Daten im selben spezifizierten Format zu verarbeiten, unabhängig davon, ob es sich um einen Import- oder Exportvorgang handelt.

- **Analyse und -Überwachung:** Sind bestehende Funktionen im neuen Tool, die es ermöglichen, bestimmte Ereignisse und Benutzerinformationen sowie die für Aufgaben aufgewendete Zeit zu visualisieren. Obwohl diese Funktionen nicht direkt Teil der Prozesskette sind, stellen sie wichtige neue Features dar. Eine geplante Erweiterung dieser Funktionen zielt darauf ab, zusätzliche Informationen über Nutzer und Aufgaben bereitzustellen. Dies umfasst die Anzeige der E-Mail-Adressen der Benutzer und, neben den Aufgaben-IDs, auch die Namen der Aufgaben sowie weitere relevante Details.

- **Benachrichtigungssystem:** Eine bestehende Funktion im neuen Tool, die Webhooks verwendet, um JSON-Daten bei spezifischen Ereignissen zu senden. Dazu gehören Ereignisse wie Statusänderungen von Aufgaben oder die Zuweisung von Aufgaben an

Personen, insbesondere wenn mehrere Personen an derselben Aufgabe arbeiten. Die Funktion ist nicht Teil der Prozesskette, aber sie bietet erhebliche Neuerungen durch folgende geplante Erweiterungen:

- Erweiterung der Webhook-Senderfunktionalität. Ziel ist es, die Übermittlung zusätzlicher wichtiger Informationen zu verbessern. Dazu zählen bspw. der Name von Aufgaben oder Projekten sowie weitere relevante Details, die die Nutzer bei der Gruppenarbeit helfen.
- Das Tool ist nicht in der Lage, die gesendeten Daten zu empfangen, was die direkte Kommunikation zwischen den Benutzern einschränkt. Das Ziel ist es, sowohl einen Datenempfänger als auch einen Nachrichtenmanager für die Datensendung zu integrieren. Das System sollte fähig sein, Daten zu empfangen und Nachrichten zu versenden, ohne auf externe APIs angewiesen zu sein.

7 Konzept

Dieses Kapitel konzentriert sich auf die Entwicklung des Konzeptentwurfs sowie auf die Begründung für die Auswahl spezifischer Maßnahmen zur Weiterentwicklung von CVAT.

7.1 Konzept zur Anpassung der Export- und Importfunktionen

Ziel ist es, die Export- und Importfunktion in CVAT so zu erweitern, dass sie ein XML-Format unterstützen, welches mit dem alten Tool kompatibel ist. Diese Anpassung wird die Kompatibilität mit bestehenden Skripten ermöglichen, die zur Statistikauswertung der exportierten Daten genutzt werden (Siehe oben die Prozesskette 3.1).

Als Lösungsansatz wird vorgelegt, ein neues Skript für das geforderte Format in Containerformat von dem Tool zu integrieren. Zudem soll die Benutzeroberfläche des Tools angepasst werden, um den Nutzern die Auswahl des neuen Formats während des Import- oder Exportvorgangs zu ermöglichen. Im Rahmen dieses neuen Format-Skripts werden sowohl die Export- als auch die Importfunktion so implementiert, dass Nutzer die Option haben, das neue Format bei ihren Import- oder Exportaktivitäten auszuwählen.

In dem dargestellten Sequenzdiagramm (7.1) wird der Ablauf der Interaktionen zwischen Benutzer, Benutzeroberfläche, Backend-System und dem neu entwickelten Format-Skript dargestellt, um den Export- oder Importvorgang zu veranschaulichen.

Der Prozess beginnt mit dem Benutzer, der über die Benutzeroberfläche des Tools entscheidet, ob ein Export- oder Importvorgang durchgeführt werden soll. Nachdem der Benutzer diese Entscheidung getroffen hat, wählt er anschließend das gewünschte Format aus dem Containerformat für den jeweiligen Vorgang aus.

Diese Informationen - sowohl die Wahl zwischen Export und Import als auch das ausgewählte Format - werden von der Benutzeroberfläche an das Backend weitergeleitet. Das Backend

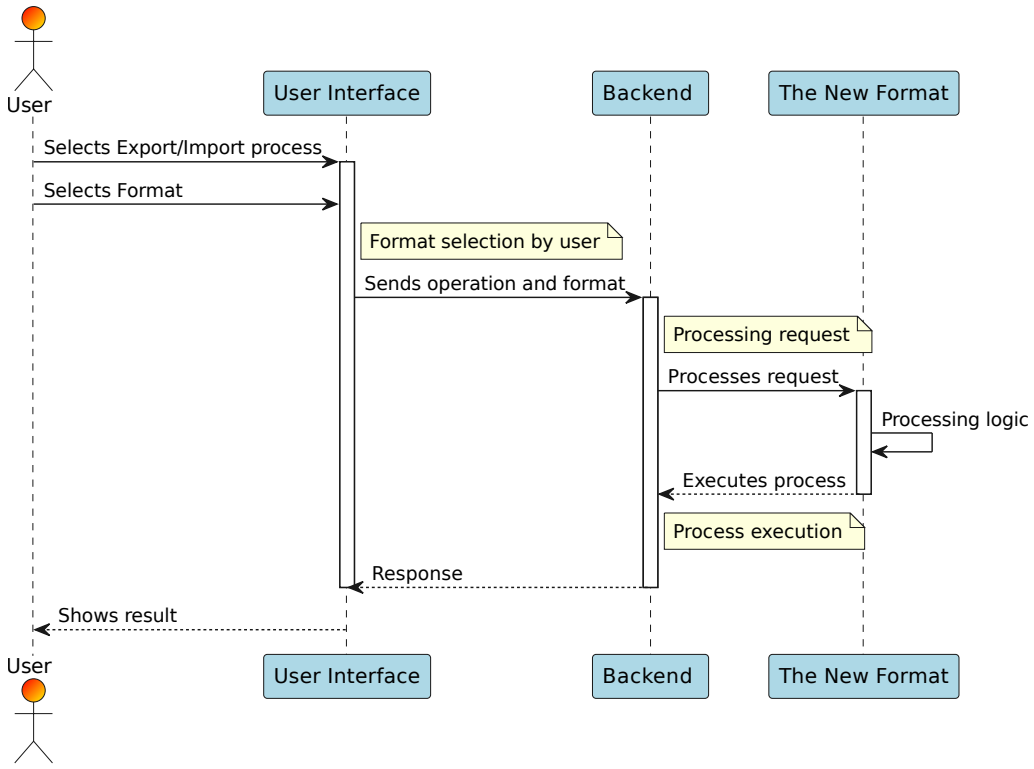

Abbildung 7.1: Sequenzdiagramm zur Verarbeitung des neuen Formats

wiederum fungiert als Schnittstelle zum neuen Format-Skript, welches speziell für die Verarbeitung des ausgewählten Formats konzipiert wurde.

Nach dem Abschluss dieses Prozesses wird die Ergebnisse an das Backend zurückgeschickt. Das Backend verarbeitet diese Informationen und leitet sie schließlich als Rückmeldung an die Benutzeroberfläche weiter. Diese Rückmeldung informiert den Benutzer über den Erfolg oder eventuelle Probleme während des Export- oder Importvorgang.

Da das neue Format auf XML basiert, ist die Berücksichtigung des zugehörigen XML-Schemas erforderlich. Zunächst wird untersucht, welches Schema des alten Tools verwendet, um dieses anschließend in das neue Format von neuem Tool zu integrieren. Nach dem Exportprozess sollen die annotierten Daten in der festgelegten Struktur ausgegeben, wie in Auflistung (7.1) dargestellt.

```
<?xml version="1.0" encoding="utf-8"?>
<video filename="Name der Datei.xml">
  <frame-labeling>
    <value class="Enum name" start="Frame number" end="Frame number" id="
    Status of the frame finding"/>
    <!-- Weitere value-Elemente mit Attributen class, start, end und id -->
    <value class="Enum name" start="Frame number" end="Frame number" id="
    Status of the frame finding"/>
  </frame-labeling>
</video>
```

Listing 7.1: Das XML-Schema

Das in der obigen Auflistung dargestellte Schema dient als strukturelle Basis von Annotationsdaten. Es beginnt mit einer Deklaration, die die XML-Version und Codierung festlegt, um universelle Lesbarkeit zu sichern. Das Hauptelement <video> enthält alle Annotationsdaten und spezifiziert über das Attribut "filename" den Dateinamen. Innerhalb dieses Rahmens organisiert das Element <frame-labeling> die Annotationen, die als <value>-Elemente erfasst werden. Jedes <value>-Element ist mit einer Reihe von Attributen ausgestattet, die wesentliche Informationen über die Annotierung bereitstellen: "class" kennzeichnet den Name der Aufzählung von der Fehler-Status, "start" und "end" definieren den Anfangs- und Endpunkt der Annotierung innerhalb des Video-Streams anhand der Frame-Nummern, und "id" bietet den Zustand oder Fehler des Frame-Findings.

7.2 Konzept zur Erweiterung der Analyse und -Überwachungsfunktionen

Die Erweiterung der Analyse- und Überwachungsfunktionen stellt ein wichtiges Feature in aktualisierten CVAT dar, das maßgeblich zur Steigerung der Effizienz und Effektivität der Nutzer beiträgt. Diese Funktionen ermöglichen es, den Arbeitsaufwand und die Geschwindigkeit der Nutzer zu erfassen und zu analysieren, was wiederum die Optimierung der Arbeitsprozesse erleichtert. Durch gezielte Überwachung kann die Qualität der Annotationen sichergestellt und Arbeit der Nutzer kontinuierlich überprüft werden. Zudem sind diese Funktionen unerlässlich für das Projektmanagement, da sie es erlauben, den Fortschritt der Projekte genau zu überwachen, fristgerechte Lieferungen zu sichern und letztendlich die Projektziele zu erreichen.

Um die Analyse- und Überwachungsfunktionen zu erweitern, ist eine Anpassung des Ba-

ckends vorgesehen. Diese Anpassung umfasst die Erweiterung der Datenerfassung, sodass neben den bisherigen Informationen wie Aufgaben-IDs und Nutzer-IDs zusätzliche Daten erfasst werden, etwa die Namen der Aufgaben und die E-Mail-Adressen der Benutzer. Diese zusätzlichen Daten sollen das Verständnis der Nutzerinteraktionen und des Projektfortschritts verbessern und somit zu einer umfassenderen Analyse beitragen.
Die abgebildete Architektur (7.2) verdeutlicht die nahtlose Integration verschiedener Technologien - von dem CVAT-UI über das Backend, Vector, ClickHouse (2.3.10) bis hin zu Grafana (2.3.9), um ein leistungsstarkes System für Analyse und Überwachung zu schaffen.

Abbildung 7.2: Architektur zur Analyse und -Überwachungsfunktion [32]

Das Backend wird diese erweiterten Daten in die bestehende Ereignisbehandlung integrieren, sodass sie durch Vector verarbeitet und an die Datenbank ClickHouse weitergeleitet werden können. ClickHouse dient hierbei als zentrale Datenquelle, die die erweiterten Daten speichert und bereitstellt. Schließlich wird Grafana genutzt, um aus diesen reichhaltigen Datenquellen aussagekräftige Visualisierungen zu erstellen. Diese sollen die neu integrierten Informationen aufgreifen und in Form von Dashboards darstellen.

7.3 Konzept zur Erweiterung des Benachrichtigungssystems

Das automatische Benachrichtigungssystem im neuen Tool wird durch Webhooks (2.3.7) ermöglicht, welche den Labeling-Prozess, eine typische Teamarbeit bestehend aus Annotieren und dem Überprüfen von Annotationen, effizient unterstützen. Die parallele Arbeit an verschiedenen Aufgaben erfordert eine effektive Kommunikation, um alle Teammitglieder über den Fortschritt und Statusänderungen informiert zu halten, ohne dass eine ständige manuelle Prüfung notwendig ist. Ebenso wird der Überprüfer automatisch informiert, sobald er eine Aufgabe zur Fehlerkontrolle zugewiesen bekommt. Dieses System trägt maßgeblich zur Beschleunigung des Gesamtprozesses bei.

Eine mögliche Lösung für das automatisierte Benachrichtigungssystem wäre der Einsatz von Anwendungen wie Zapier, Ngrok oder Make von Microsoft, die es ermöglichen, Daten

von Webhooks zu empfangen und als E-Mails an Nutzer zu versenden. Jedoch entspricht diese Herangehensweise nicht der Anforderung, da explizit gefordert wird, keine externen APIs zu verwenden.

Innerhalb des neuen Tools wird eine innovative Lösung für das Benachrichtigungssystem erfunden, welche auf einem intern entwickelten Webhook-Empfänger basiert. Diese Lösung, realisiert als eigenständige Django-Anwendung für Webhook-Receiver, ermöglicht den Empfang und die Analyse von Daten, die vom Webhook-Sender des Tools übermittelt werden. Nach der Datenanalyse erfolgt die Erstellung einer E-Mail-Benachrichtigung mittels einer HTML-Vorlage. Der abschließende Versand an die Nutzer geschieht durch den integrierten Nachrichtenmanager, welcher die Benachrichtigungen direkt an die spezifizierten E-Mail-Adressen der Nutzer sendet.

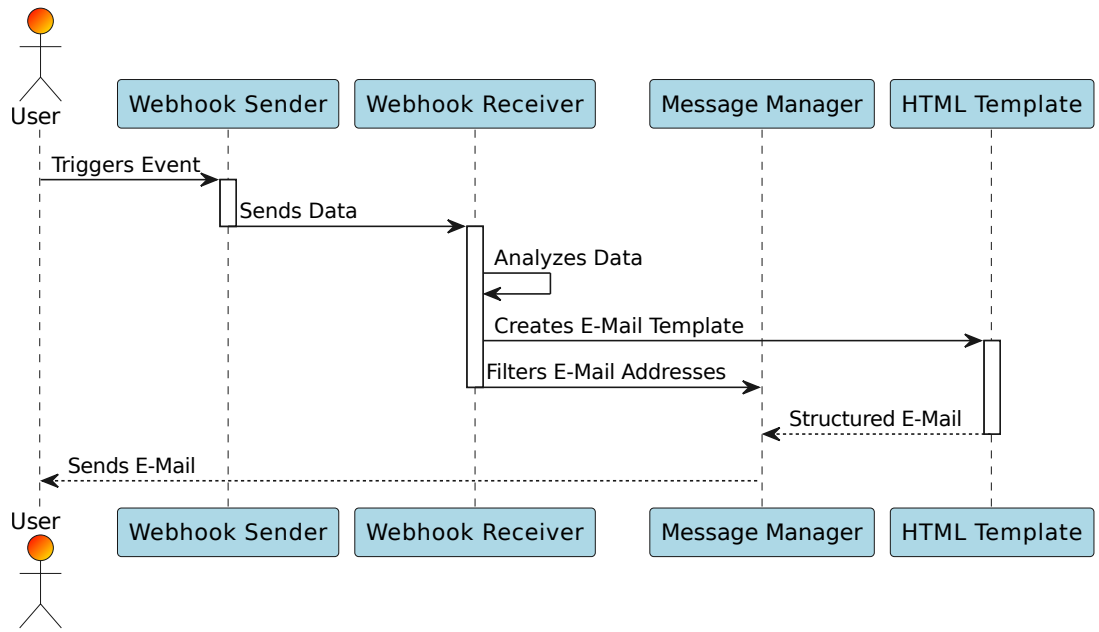

Abbildung 7.3: Konzept zur Erweiterung des Benachrichtigungssystems

Der aktuelle Webhook-Sender übermittelt nicht alle relevanten Informationen, wie zum Beispiel die E-Mail-Adressen aller Mitarbeiter, die gemeinsam an einer Aufgabe arbeiten. In dem Tool wird eine Organisation erstellt, die es mehreren Nutzern ermöglicht, an derselben Aufgabe zu arbeiten. In einem solchen Szenario sollte der Ersteller die E-Mail-Adressen der betreffenden Mitarbeiter angeben können. Diese E-Mail-Adressen sollten dann durch den Webhook-Sender

mit anderen relevanten Details versendet werden können.

Die Aufgabe der HTML-Vorlage im Kontext eines Webhook-Receivers ist es, die empfangenen JSON-Daten in ein benutzerfreundliches Format umzuwandeln, das per E-Mail versendet werden kann. Anstatt dem Nutzer rohe JSON-Daten zu präsentieren, die schwer zu lesen und zu interpretieren sein können, ermöglicht die HTML-Vorlage die Strukturierung und Formatierung dieser Daten in einer deutlich ansprechenden Weise.

8 Implementierung

In der bevorstehenden Implementierungsphase wird das Konzept zur Erweiterung und Optimierung von CVAT auf wesentliche Funktionsbereiche angewendet: die Export- und Importmechanismen sowie Analyse- und Überwachungsfunktionen und das Benachrichtigungssystem.

8.1 Implementierung des neuen Datenformats für Export und Import

Die Implementierung des neuen Formats und dessen Integration in den bestehenden Formate-Container wird im Folgenden erläutert. Zunächst wurde ein eindeutiger Name für das neue Format festgelegt, der auf der Benutzeroberfläche neben den Bezeichnungen der bereits vorhandenen Formate angezeigt wird. Für dieses Format wurde der Name “Intenta” und die Version “1.0” ausgewählt. Dieser Name erscheint für die Benutzer in der Benutzeroberfläche und dient der Unterscheidung von früheren Formatversionen.

8.1.1 Integration des neuen Formats in den Format-Container

Der erste Schritt bestand darin, die Funktionalität des Formats in einer separaten Skriptdatei “**Intenta.py**” zu definieren, die in Da Dataset-Manager-App von Django (2.3.5) unter dem Verzeichnis Formats platziert wurde.

Anschließend wurde das neue Format im “**register.py**” des CVAT-Systems importiert, wie die Auflistung (8.1) zeigt, um es im gesamten Anwendungsbereich verfügbar zu machen. Dieser Registrierungsmechanismus stellt sicher, dass das neue Format als Option in der Benutzeroberfläche von CVAT erscheint und den Nutzern zur Auswahl neben den bereits vorhandenen Formaten steht.

```
1 import cvat.apps.dataset_manager.formats.intenta
```

Listing 8.1: Registrierung des neuen Formats

Die Registrierung des neuen Formats erfolgt mittels spezieller Dekoratoren in der "**register.py**"-Datei, die als Schnittstelle für das Hinzufügen von Export- und Importfunktionalitäten dienen. Die Auflistung (8.2) zeigt die Dekoratoren von dem Export.
Dekoratoren sind in Python definierte Konstrukte, die es ermöglichen, zusätzliche Funktionalitäten zu bestehenden Funktionen oder Klassen hinzuzufügen, ohne deren Code zu verändern.

```
1  def exporter(name, version, ext, display_name=None, enabled=True, dimension=
       DimensionType.DIM_2D):
2      assert name not in EXPORT_FORMATS, "Export format '%s' already
       registered" % name
3      def wrap_with_params(f_or_cls):
4          t = _wrap_format(f_or_cls, Exporter,
5              name=name, ext=ext, version=version, display_name=display_name,
6              enabled=enabled, dimension=dimension)
7          key = t.DISPLAY_NAME
8          assert key not in EXPORT_FORMATS, "Export format '%s' already
       registered" % name
9          EXPORT_FORMATS[key] = t
10         return t
11     return wrap_with_params
```

Listing 8.2: Dekoratoren den Export

Im Zusammenhang erlauben die Dekoratoren "exporter" und "importer", eine nahtlose Integration des neuen Formats in das System und wird die verwendet, um den Funktionen oder Klassen, die für den Export und Import zuständig sind, notwendige Informationen hinzuzufügen. Diese Informationen umfassen den eindeutigen Namen des Formats, die Version und die spezifische Dateierweiterung, die vom Format verwendet wird.

8.1.2 Implementierung des neuen Formats

Die beigefügten Codeausschnitte zeigen die Implementierung von wichtigsten Funktionen des neuen Formats, die für die Export- und Importdaten verwendet werden.

8.1.2.1 Exportvorgang

Die Funktionalität zum Exportieren im neuen Format wird über den "exporter"-Dekorator in der "**register.py**"-Datei definiert. Dieser Dekorator markiert die Funktion "_export _images", die für den Export von Annotationsdaten verantwortlich ist. Die Funktion nimmt verschiedene Parameter entgegen, einschließlich der Zieldatei "dst_file", eines temporären Verzeichnisses

“temp_dir” und der Instanzdaten “instance_data”. Diese Instanzdaten können sich auf ein ganzes Projekt oder eine einzelne Aufgabe beziehen und sie tragen die Annotationsdaten.

```
1 @exporter(name='Intenta', ext='ZIP', version='1.0')
2 def _export_images(dst_file, temp_dir, instance_data, save_images=False):
3     #check if project or (task or job)
4     if isinstance(instance_data, ProjectData):
5         _export_project(dst_file, temp_dir, instance_data,
6             anno_callback=dump_as_cvat_annotation, save_images=save_images)
7     else:
8         _export_task_or_job(dst_file, temp_dir, instance_data,
9             anno_callback=dump_as_cvat_annotation, save_images=save_images)
```

Listing 8.3: Exportvorgang für neues Format

Die Exportfunktion überprüft zunächst in der Zeile (4), ob die Daten, die exportiert werden sollen, einem Projekt oder einer Aufgabe zugeordnet sind. Je nachdem wird entweder das gesamte Projekt oder die spezifische Aufgabe zusammen mit den Annotationsdaten exportiert. Die Funktion “dump_as_cvat_annotation” wird als “Callback” für die Annotationen verwendet, und es gibt die Option, Bilder zu speichern. Das neue Format verwendet das ZIP-Format für den Export, wodurch alle relevanten Daten in einer komprimierten Datei vereint werden.

Die Funktion “dump_as_cvat_annotation” spielt eine entscheidende Rolle im Prozess des Umwandelns und Formatierens von Annotationsdaten entsprechend des im Konzept dargestellten XML-Schemas (7.1). Diese Funktion übernimmt die sorgfältige Verarbeitung jeder einzelnen Annotation, um diese für den Export vorzubereiten. Sie durchläuft systematisch alle Frames und deren assoziierte Tags, wobei sie wichtige Informationen extrahiert: die Frame-Nummer für die “start”- und “end”-Attribute, den vom Nutzer angegebenen Aufzählungsnamen für das “class”-Attribut und den ausgewählten Zustand des Frame-Findings für das “id”-Attribut. Diese Informationen werden anschließend in das definierte XML-Format überführt.

8.1.2.2 Importvorgang

Für das neue Format wurde eine spezialisierte Importfunktion entwickelt, die XML- und ZIP-Dateien als Eingabe unterstützt. Diese Funktion wird durch den “importer”-Dekorator Die Funktion beginnt mit der Überprüfung, ob die bereitgestellte Quelldatei “src_file” eine ZIP-Datei ist. Wenn dies der Fall ist, entpackt die Funktion den Inhalt in ein temporäres Verzeichnis “temp_dir”, um auf die einzelnen Dateien zugreifen zu können.

```
@importer(name='Intenta', ext='XML, ZIP', version='1.0')
def _import(src_file, temp_dir, instance_data, load_data_callback=None, **
    kwargs):
  #  slogger.glob.info('__import')
    is_zip = zipfile.is_zipfile(src_file)
    src_file.seek(0)
    if is_zip:
        zipfile.ZipFile(src_file).extractall(temp_dir)

        anno_paths = glob(osp.join(temp_dir, '**', '*.xml'), recursive=True)
          for p in anno_paths:
               load_anno(p, instance_data)
    else:
        load_anno(src_file, instance_data)
```

Listing 8.4: Importvorgang für neues Format

Danach durchsucht sie das temporäre Verzeichnis nach allen XML-Dateien, die enthalten sind. Jede gefundene XML-Datei wird einzeln bearbeitet und die Funktion “load_anno” wird verwendet, um die Annotationsdaten zu importieren. Wenn die Quelldatei keine ZIP-Datei, sondern direkt eine XML-Datei ist, wird unmittelbar auf diese Datei zugegriffen und die Funktion “load_anno” verarbeitet die Annotationsdaten darin.

Die Funktion “load_anno” lädt Annotationsdaten aus XML-Dateien und fügt sie in das System ein. Sie analysiert jedes XML-Element, extrahiert relevante Informationen für jedes Frame, wie die Framenummer aus dem “start”-Attribut, den Aufzählungsnamen aus dem “class”-Attribut und deren Zustände, und integriert diese Daten in die bestehenden CVAT-Annotationen.

8.2 Erweiterung der Analyse und -Überwachungsfunktion

Das folgende Komponentendiagramm (8.1) zeigt die Architektur und den Datenfluss innerhalb der erweiterten Tool-Anwendung, welche durch zusätzliche Backend-Logik, eine integrierte Vector-Konfiguration und die Verbindung zu einer ClickHouse-Datenbank gekennzeichnet ist.

Im Backend wurde die Event-Modelklasse und ihr Serializer erweitert (Siehe die Auflistung 8.5), um zusätzliche Felder für die Überwachung und Analyse von Ereignissen zu erfassen. Die ClientEventsSerializer-Klasse verarbeitet die empfangenen Ereignisdaten, indem er Zeitstempel anpasst und die Daten mit Kontextinformationen anreichert. Die EventsViewSet-Klasse unter-

stützt das Empfangen, Validieren und Serialisieren der Ereignisdaten. Die Vector-Konfiguration

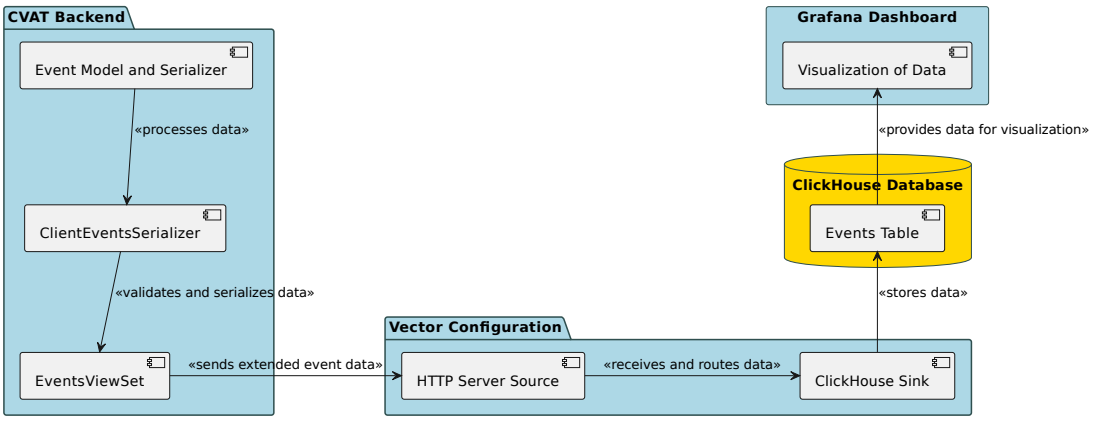

Abbildung 8.1: Die Architektur der Erweiterung des Analyse-Überwachungprozess

wurde angepasst, um einen HTTP-Server als Datenquelle zu etablieren, der neuen Ereignisdaten mit alten Ereignisdaten im JSON-Format empfängt. Die Clickhouse-Sink-Konfiguration wurde hinzugefügt, um diese Daten an die ClickHouse-Datenbank weiterzuleiten, wobei Authentifizierungsparameter und Felder für die Datenübertragung festgelegt wurden.
In der ClickHouse-Datenbank wurde die Ereignistabelle für die neuen Eventdaten aktualisiert, die die neuen Datenfelder aus dem Backend und Vector reflektiert. Die Datenstruktur in ClickHouse speichert die erweiterten Event-Daten und stellt sie für Abfragen und Analysen bereit.

```
            last_timestamp = timestamp
            event['timestamp'] = str((timestamp + time_correction).timestamp
    ())
            event['source'] = 'client'
            event['task_name'] =get_task_name(event)
            event['project_name'] = get_object_name(event)
            event['org_id'] = org_id
            event['org_slug'] = org_slug
            event['user_id'] = request.user.id
            event['user_name'] = request.user.username
            event['user_email'] = request.user.email
        return data
```

Listing 8.5: Auszug aus dem ClientEventsSerializer

Der dargestellte Code-Auszug (8.5) aus dem ClientEventsSerializer zeigt, wie Ereignisdaten um wichtige Informationen ergänzt werden. Im Mittelpunkt der Anreicherung stehen die Anpassung von Zeitstempeln und das Hinzufügen von Namen. Der Zeitstempel jedes Ereignisses wird durch die Berücksichtigung der Zeitdifferenz zwischen dem Senden und Empfangen der Daten korrigiert, was eine genauere Zeitangabe der Ereignisse ermöglicht. Zusätzlich werden die Namen von Aufgaben “task_name” und Projekten “project_name” basierend auf ihren eindeutigen IDs aus den zugehörigen Objekten extrahiert und den Ereignisdaten hinzugefügt. Diese Anreicherung trägt wesentlich dazu bei, die Event-Daten im Kontext des CVAT-Systems detaillierter und aussagekräftiger zu gestalten.

8.3 Erweiterung des Benachrichtigenssystems

Die Optimierung des Benachrichtigungssystems beginnt mit der Anpassung des Webhook-Senders, dann mit der Implementierung eines Webhook-Receivers und eines Nachrichtenmanagers innerhalb der CVAT-Anwendung. Diese Neuerungen ermöglichen eine automatisierte Kommunikation zwischen den Systemkomponenten.

8.4 Anpassung des Sendungssystems

Der Webhook-Sender reagiert auf spezifische Ereignisse innerhalb des Tools. Der folgende Codeabschnitt (8.6) zeigt die verschiedenen Ereignistypen, auf die der Webhook-Sender eingestellt werden kann. Diese Ereignisse sind im Tool bei der Einrichtung eines Webhooks sichtbar und der Nutzer kann auswählen, auf welche Ereignisse der Webhook-Sender reagieren soll.

```
class Events:
    RESOURCES = {
        "project": ["create", "update", "delete"],
        "task": ["create", "update", "delete"],
        "job": ["create", "update", "delete"],
        "issue": ["create", "update", "delete"],
        "comment": ["create", "update", "delete"],
        "organization": ["update", "delete"],
    }
```

Listing 8.6: Klassendefinition der Ereignisse

Zunächst wird eine Organisation erstellt, zu der Nutzer hinzugefügt werden, die gemeinsam an Projekten arbeiten. Innerhalb dieser Organisation können Projekte und Aufgaben definiert werden, für die anschließend Webhooks eingerichtet werden können. Bei Änderungen an einer

Aufgabe oder einem Projekt innerhalb der Organisation – sei es Erstellung, Aktualisierung oder Löschung – reagiert der Webhook-Sender automatisch, indem er JSON-Daten an die bei der Einrichtung des Webhooks angegebene URL für den Endpunkt sendet.
Ereignisse wie "Issue" oder "Comment" treten im Überprüfungsmodus von annotierten Aufgaben auf. Entdeckt ein Reviewer einen Fehler in der Annotation, kann er ein Issue eröffnen und innerhalb dieses Issues Kommentare hinterlassen.
Zuerst wurde erweiter, dass bei allen Ereignissen E-Mail-Adressen aus der erstellten Organisation zusammen mit den Namen des gesamten Teams filtert und dann versendet werden. Diese Erweiterung ist einen kritischen Punkt für den Nachrichtenmanager, der im nächsten Kapitel behandelt wird. Außerdem wurde ergänzt, dass bei Ereignissen, die Aufgaben oder Projekte betreffen, umfassendere Daten wie Namen der Aufgaben und Projekte, sowie Namen der Nutzer, neben den IDs und weitere relevante Informationen übermittelt werden.

Dann wird noch angepasst, dass bei "Issue" und "Comment" zusätzliche Daten gesendet werden, wie etwa die Kommentare des Reviewers, zugehörige und aktuelle Stand der Aufgabe sowie der betreffende Frame. Es wird auch angegeben, wie viele Issues eröffnet sind. Zudem wurde angepasst, dass URLs zur Aufgabe oder zum Projekt sowie zu dem "Issue" und "Comment" gesendet werden können.

```
def build_url(payload):
    # Use base CVAT url
    CVAT_BASE_URL = settings.CVAT_BASE_URL
    # Comment or issue url: CVAT_BASE_URL/tasks/<id>/jobs/<job_id>?frame=<
    image_id>
    if 'comment' in payload['event']:
        url = '{}/tasks/{}/jobs/{}?frame={}'.format(CVAT_BASE_URL,
               payload['comment']['task_id'], payload['comment']['job_id'],
               payload['comment']['frame'])
        return urllib.parse.urlunparse(urllib.parse.urlparse(url))
    if 'issue' in payload['event']:
        url = '{}/tasks/{}/jobs/{}?frame={}'.format(CVAT_BASE_URL,
               payload['issue']['task_id'], payload['issue']['job'],
               payload['issue']['frame'])
        return urllib.parse.urlunparse(urllib.parse.urlparse(url))
    #further ....
```

Listing 8.7: URL-Erzeugungsfunktion

Der Codeausschnitt (8.7) zeigt die "build_url"-Funktion die spezifische URLs für "Comment" und "Issues", Aufgaben und mehr generiert, basierend auf der Aufgaben-ID, der Job-ID und

der Frame-ID wie in der Zeile 4. Diese URLs dienen als direkte Eintrittskarten zu den entsprechenden Bereichen in dem Tool, wodurch ein nahtloser Sprung und eine unmittelbare Ansicht möglich werden. Dieser Sprung kann direkt von erhaltenden Mail durchgeführt.

8.4.1 Implementierung des Empfängersystems und Nachrichtenmanagers

In der Entwicklungsphase wurde der Webhook-Receiver als zentrale Komponente des erweiterten Benachrichtigungssystems entworfen. Dieser agiert als Schnittstelle, die Daten vom Webhook-Sender empfängt und für die weitere Verarbeitung für den Nachrichtenmanager vorbereitet. Die Implementierung beinhaltete folgende wesentliche Schritte: Zunächst wurde innerhalb der Django-Anwendung ein spezifischer Empfangsendpunkt eingerichtet, um POST-Anfragen vom Webhook-Sender anzunehmen.

```
1 from django.urls import path
2 from receiver.views import receive_data
3
4 urlpatterns = [
5 # url endpoint of webhook receiver
6 path('receive_data', receive_data, name='receive_data'),
7 ]
```

Listing 8.8: Der Endpunkt des Webhook-Receivers

Der Codeausschnitt (8.8) legt eine URL-Route innerhalb der Django-App fest, die POST-Anfragen an den “receive_data/” Endpunkt lenkt, wo sie von der importierten “receive_data” View-Funktion verarbeitet werden. Dieser Endpunkt dient speziell dazu, Webhook-Anfragen von dem Webhook-Sender zu akzeptieren und zu verarbeiten.

```
1  @csrf_exempt  # Disables CSRF token for the API request
2  def receive_data(request):
3      if request.method == 'POST':
4          # Load data from the POST request
5          data = json.loads(request.body)
6          # Processing and anaylse data
7          # ...
8          # ...
9          html_template_render(data, request)
10         return JsonResponse({'message': 'Data has been sent successfully.'},
       status=200)
```

Listing 8.9: Auszug aus der View-Funktion des Webhooks-Receivers

Dieser Codeausschnitt (8.9) zeigt, wie die “receive_data’-Funktion in der Django-App entworfen ist, um POST-Anfragen zu verarbeiten. Der CSRF-Schutz wird für diese spezielle API-Anfrage deaktiviert, um eine reibungslose Funktion ohne die Notwendigkeit eines CSRF-Tokens zu ermöglichen. Die Funktion liest die übermittelten Daten, verarbeitet und analysiert sie. Die prozessierten Daten in der Funktion werden am Ende zu der Funktion “html_email_template_render” ermittelt, wie in Zeile 8 dargestellt.

```
1 def html_template_render(webhook_data, request):
2     context = {'data': webhook_data}
3     email_html = render(request, 'receiver/email_template.html',
4                         context).content.decode('utf-8')
5     #Send structured e-mail and webhook data to the mail manager
6     send_mails(email_html, webhook_data)
```

Listing 8.10: Rendern-Funktion für die HTML-Vorlage

Die Funktion “html_email_template_render” wandelt JSON-Daten mithilfe einer HTML-Vorlage in ein benutzerfreundliches E-Mail-Format um. Nach dem Rendern der Daten übergibt sie das formatierte HTML an die Funktion “send_mails”, die es an den Mail-Manager weiterleitet. Dieser filtert die Empfängeradressen aus den Webhook-Daten und versendet die E-Mail an die entsprechenden Nutzer.

```
1  def send_mails(email_html, webhook_data):
2      recipient_mails, event = get_mails_eventdata(webhook_data) #Filter the
       recipient address and event type
3      sender_mail = 'sender@muster-mail.de' # E-Mail of sender
4      email = EmailMessage(
5          f'CVAT {event}', #Title of the event type
6          email_html,
7          sender_mail,
8          recipient_mails,
9      )
10     # Add HTML content to the e-mail message
11     email.content_subtype = "html"
12     # Send e-mail message
13     email.send()
14     return JsonResponse({'message': 'Data successfully received and
       processed. E-mail sent.'})
```

Listing 8.11: Hauptfunktion des den Nachrichtenmanagers

Der obige Code (8.11) zeigt die “send_mails” Funktion, die als Mail-Manager innerhalb des Benachrichtigungssystems dient. Sie verwendet formatiertes HTML und Webhook-Daten, um

E-Mail-Benachrichtigungen zu versenden. Innerhalb der Funktion filtert “get_mails_eventdata” die Empfängeradressen und den Ereignistyp heraus. Anschließend wird mit der EmailMessage-Klasse eine E-Mail erstellt und versendet. Nach dem Versand liefert die Funktion eine Rückmeldung über den erfolgreichen Prozess.

8.4.2 Kommunikation zwischen Benachrichtigungssystemkomponenten

Das Tool funktioniert auf einer Docker-Plattform, wobei jede Komponente in einem separaten Docker-Container ausgeführt wird, so auch der Webhook-Sender. Der Webhook-Sender benötigt eine Ziel-URL, an die er Daten senden kann; jedoch kann eine URL mit Computer-Localhost wie “https://localhost:port/endpoint” nicht verwendet werden, da dies aus der Perspektive des Containers sich selbst bezeichnet, nicht den Host. Also wird der Webhook-Receiver in einem eigenen Docker-Container ausgeführt, damit er mit dem Webhook-Sender kommunizieren kann. Sie kommunizieren sich über interne Netzwerk von Docker, wobei für die Verbindung die IP-Adresse des Receiver-Containers und ein spezifischer Port genutzt werden. So etwa “https://172.18.0.40:port/endpoint”, was die Datenübergabe sicherstellt.

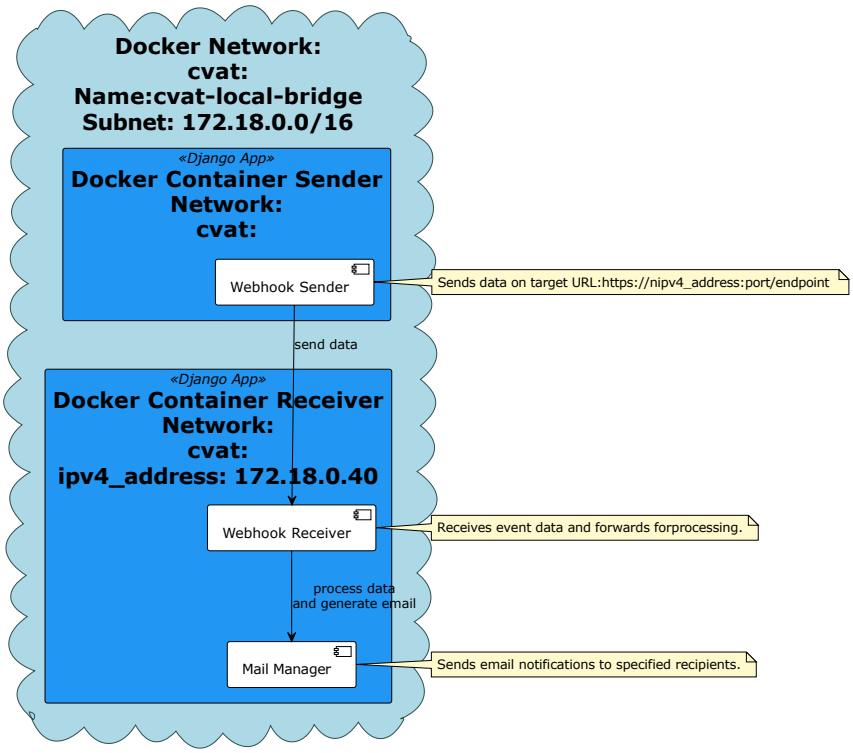

Abbildung 8.2: Übersicht zur Kommunikation zwischen Benarchtigungssytemkomponenten

Die Abbildung (8.2) zeigt, dass Webhook-Sender-Container und Webhook-Receiver-Container gleiche Docker-Netzwerk “cvat” haben. Das Netzwerk ist ein “Bridge”-Netzwerk. Für dieses Netzwerk wurde eine festen Subnetz erstellt und das wird von Docker verwendet, um Netzwerke von Container zu definieren. Für das Subnetz wurde der feste Adressbereich “172.18.0.0/16” festgelegt. Dadurch erhalten alle Container innerhalb dieses Bereichs automatisch definierte IP-Adressen, was die Kommunikation und Interaktion zwischen den Containern innerhalb dieses Subnetzes ermöglicht.

Die IP-Adresse von dem Webhook-Receiver-Container “172.18.0.40” ist auch festgelegt. Die feste IP-Adresse ermöglicht eine konsistente und zuverlässige Konfiguration für Webhook-Ziele wie das Port 8000 und der Endpunkt “receive_data” (Siehe die Abbildung 8.8). Dadurch bleibt die URL “https://172.18.0.40:8000/receive_data” stabil, unabhängig von Neustarts des Docker-Systems. Das automatisiert die Einrichtung von den Webhooks in dem Tool, da die Ziel-URL aufgrund der festen IP-Adresse und des Ports, des Endpunktes nicht geändert werden muss.

9 Testphase

In diesem Kapitel werden die Umsetzungen den Konzepten überprüft, um sicherzustellen, ob das neue Tool funktioniert wie erwartet und in der Lage ist, das alte Tool zu ersetzen. Es wird drei Teste durchgeführt; Zuerst wird die angepasste Export-Import-Funktion geprüft, danach die Analyse- und Überwachungsfunktion. Abschließend erfolgt die Überprüfung der Benachrichtigungssystems.

9.1 Testen der Export-Importfunktion

Nach der Implementierung des neuen Formats wird überprüft, ob dieses in der Benutzeroberfläche exportierbar und importierbar ist und die exportierten Daten vom Statistik-Tool korrekt verarbeitet werden können.

9.2 Verfügbarkeit und Funktionalität des neuen Formats

Das neue Format ist nun erfolgreich in den Format-Container der Benutzeroberfläche integriert und für den Export- und Importprozess verfügbar, wie in der Abbildung (9.1) dargestellt.
Nachdem das Tool in der Lage war, die implementierten Formate zu exportieren, wurden die Annotationsergebnisse für die gelabelte Datei exportiert.

Das Ergebnis wird von dem neuen Format als XML-Datei ausgegeben, die dem im Konzept vorgestellten Schema (Siehe die Auflistung 7.1) entspricht.
Nach dem Export wurden die Annotationsdaten der gleichen gelabelten MP4-Datei gelöscht oder geleert und die exportierten Annotationsdaten erneut in dieselbe Datei importiert und dann exportiert. Das Ergebnis entsprach den ursprünglich exportierten Daten. Dies bestätigt, dass die Importfunktion wie erwartet funktioniert.

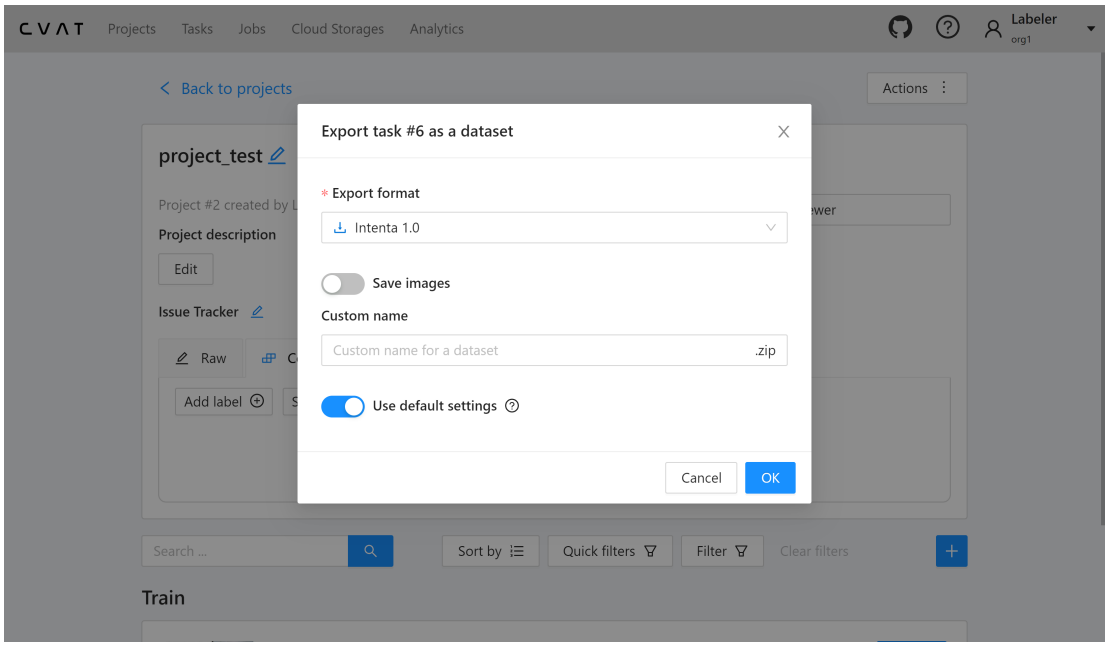

Abbildung 9.1: Das neue Format im Containerformat

9.2.1 Statistik-Tool

Das Statistik-Tool analysiert gelabelten Dateien, indem es die exportierten XML-Daten auswertet. Es berechnet verschiedene KPIs, um die Qualität der Annotationen zu messen.

Das unverändert aus der bestehenden Prozesskette übernommene Statistik-Tool soll in der Lage sein, die vom neuen Tool erzeugten XML-Dateien zu lesen und zu verarbeiten. Das Ziel ist, dass sich die Analyseergebnisse für die vom neuen Tool annotierte Datei mit denen für die vom alten Tool annotierte Datei decken, damit die Kompatibilität der Daten und die Genauigkeit der Analyseergebnisse zwischen den beiden Versionen des Tools sichergestellt wird.
Nachdem das Statistik-Tool für die beiden XML-Dateien gestartet wurde, sind die KPI wie folgt entstanden:

```
KPIs:
TPR: 0.987144 FNR: 0.012856 TPR: 0.039992
```

Listing 9.1: KPIs für die XML-Datei des alten Tool

```
KPIs:
TPR: 0.987143 FNR: 0.012857 TPR: 0.039975
```

Listing 9.2: KPIs für die XML-Datei des neuen Tool

Die Testergebnisse aus dem Statistik-Tool zeigen, dass die KPIs für die mit dem alten Tool und neuen Tool exportierten XML-Dateien nahezu identisch sind. Die beobachtete minimale Abweichung in den KPIs zwischen dem alten und neuen Tool ist wahrscheinlich auf den Bug im alten Tool zurückzuführen, der dazu führte, dass ein zusätzlicher Frame gezählt wurde. Das Ergebnis bestätigt die erfolgreiche Integration des vorgestellten Formats und die Realisierung der Prozesskette im neuen Tool, was dessen Einsatz in bestehenden Arbeitsabläufen ermöglicht und bestätigt auch, dass das neue Tool in der Lage ist, das alte Tool zu ersetzen.

9.2.2 Neue Prozesskette

Nach der Bestätigung der Testergebnisse bezüglich der Export- und Importfunktionalität sowie dem Statistik-Tool, wird das Problem in der Prozesskette in Abbildung (5.1) behoben und die neue Prozesskette von CVAT wie folgend dargestellt.

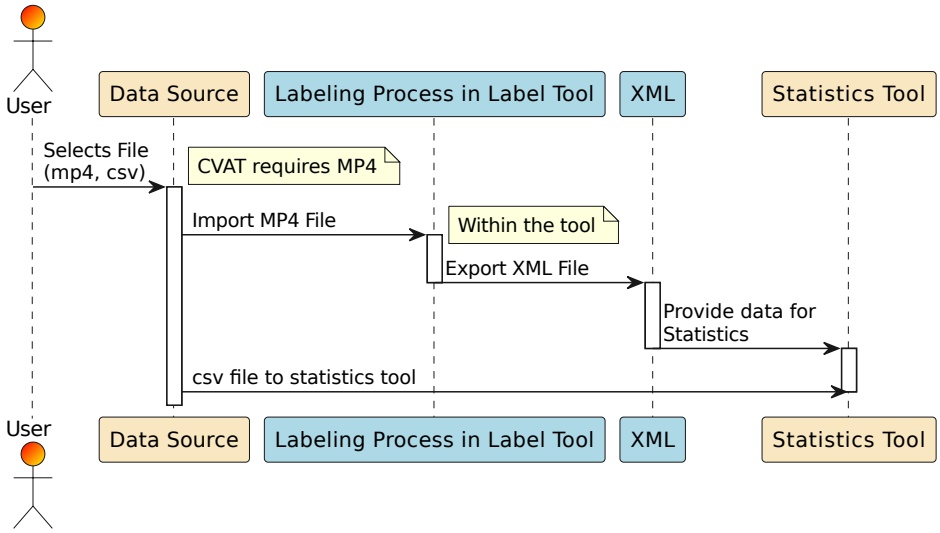

Abbildung 9.2: Sequenzdiagramm des neuen Labeling-Prozesses im neuen Tool

9.3 Testen der Analyse- und Überwachungserweiterungen

Die Funktionen für Analyse und Überwachung wurden erweitert, um e die Darstellung für zusätzlicher relevanten Informationen im Grafana-Dashboard zu ermöglichen.

Die dargestellte Abbildung (9.3) visualisiert die Benutzeraktivität an spezifischen Daten mittels Diagrammen, in denen die Aktivitäten von zwei unterschiedlichen Benutzern ersichtlich sind. Unterhalb der Diagramme befindet sich eine Tabelle zur Arbeitszeiterfassung. Diese Tabelle beinhaltet sowohl existierende Daten, wie Job- und Projekt-IDs, Benutzer-IDs sowie aufgewendete Zeiten, als auch neu hinzugefügte relevante Informationen. Zu diesen neuen Daten gehören die Namen der ausgeführten Aufgaben und Projekten, die Nutzernamen sowie die E-Mail-Adressen der Nutzer, wodurch eine umfassende Visualisierung der Benutzerinteraktionen erreicht wird. Das Ergebnis der Visualisierung verdeutlicht, dass die Erweiterung der Analyse- und Überwachungsfunktionen erfolgreich implementiert wurde. Weitere Informationen lassen sich durch geringfügige Anpassungen in der Pipeline der Komponenten einbinden, wie in der Abbildung (8.1) dargestellt.

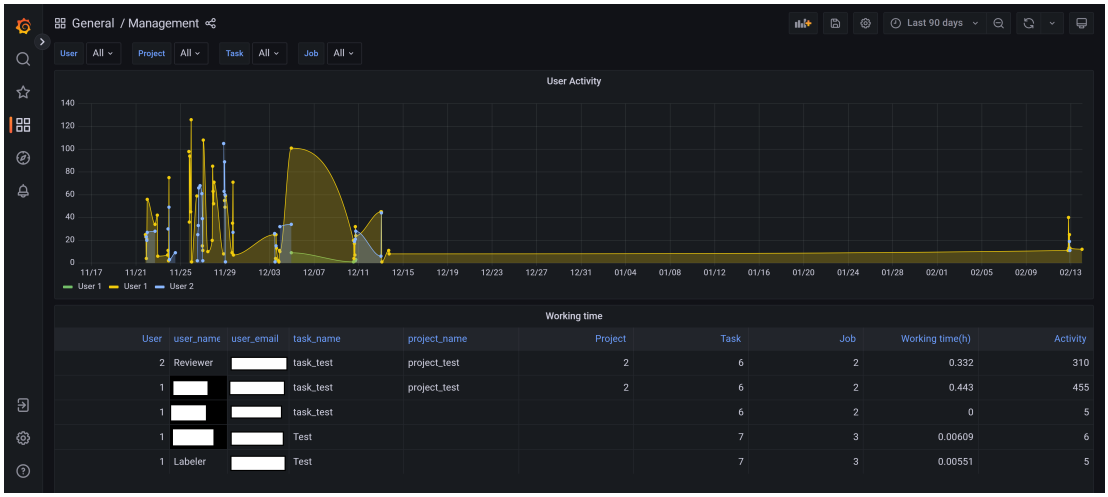

Abbildung 9.3: Übersicht des Managementdashboards von Grafana

9.4 Testen des Benachrichtigungssystems

Um das Benachrichtigungssystem zu vervollständigen, wurden das Sendersystem angepasst und ein Empfängersystem und ein Nachrichtenmanager in das neue Tool eingebaut. Nach

der Implementierung empfängt das System Ereignisdaten an der definierten Ziel-URL “https://172.18.0.40:8000/receive_data” wie folgend:


```
{
   "event":"create:comment",
   "comment":{
      "id":335,
      "issue":167,
      "job_id":2,
      "task_id":6,
      "frame":14,
      "resolved":false,
      "owner":{
         "url":"http://localhost:8080/api/users/2",
         "id":2,
         "username":"Reviewer",
         "first_name":"",
         "last_name":"",
         "email":"reviewername@muster-mail.de"
      },
      "message":"Fehlerhafte Annotation! Das zweite Auto ist bedeckt, nicht das erste",
      "created_date":"2024-02-12T21:00:07.450486Z",
      "updated_date":"2024-02-12T21:00:07.450499Z"
   },
   "sender":{
      "url":"http://localhost:8080/api/users/2",
      "id":2,
      "username":"Reviewer",
      "first_name":"",
      "last_name":"",
      "email":"reviewername@muster-mail.de"
   },
   "webhook_id":3,
   "org_users":{
      "user1":{
         "name":"Labeler",
         "email":"labelername@muster-mail.de"
      },
      "user2":{
         "name":"Reviewer",
         "email":"reviewername@muster-mail.de"
      }
   },
```

```
41    "urls":"http://localhost:8080/tasks/6/jobs/2?frame=14"
42 }
```

Listing 9.3: JSON-Datei für ein Kommentarereignis

Diese Daten im JSON-Format enthalten eine umfassende Darstellung der Ereignisse, die innerhalb des Benachrichtigungssystems generiert werden, wie z.B.: Details über das Kommentar-Ereignis (einschließlich des Kommentars in der Zeile (18), den der Benutzer hinzugefügt hat, Daten über die Person, die den Kommentar hinzugefügt hat, Listen von Nutzern, die an der gleichen Aufgabe sitzen, sowie spezielle Daten zum Aufgaben- und Frame-Kontext, einschließlich Link in der Zeile (41) zum Frame oder Ereignisort,...).
Der Nachrichtenmanager analysiert die Daten und sendet, basierend auf der HTML-Vorlage, eine E-Mail an die Nutzer in der Liste “org_users” au der Zeile (31-40), wobei der Ereignisersteller selbst keine Benachrichtigung erhält. Auf diese Weise kann der Nachrichtenmanager aus der Daten identifizieren, wer der Sender oder Ereignisersteller ist. Schlussendlich erhalten die Nutzer die E-Mail entsprechend, wobei der Ersteller der Aktion ausgenommen ist.

User details

Info	Sender	Comment creator
Username	Reviewer	Reviewer
Email	[redacted]	[redacted]

Comment details

Info	Comment
Comment url	http://localhost:8080/tasks/6/jobs/2?frame=14
Message	Fehlerhafte Annotation! Das zweite Auto ist bedeckt, nicht das erste.
Job Id	2
Task Id	6
Frame	14
Issue Id	167
Issue resolved ?	False
Created date	2024-02-12T21:59:43.195009Z

Abbildung 9.4: Übersicht der strukturierten Nachricht

Die Abbildung zeigt (9.4), wie wichtige Details aus einer JSON-Daten extrahiert und in die HTML-Vorlage eingefügt wurden, um eine personalisierte E-Mail-Benachrichtigung zu erstellen.

9.5 Vergleichende Analyse der Labeling-Tools

Nach der Überprüfung der implementierten Funktionen fand eine Vergleichsanalyse zwischen den beiden Tools statt, mit dem Ziel, die Stärken und Schwächen der Tools herauszustellen. Diese Analyse umfasste die gesamte Prozesskette des Labeling-Vorgangs sowie Aspekte der Teamkommunikation, um die Zusammenarbeit, die Qualität und die Geschwindigkeit der Datenannotation zu bewerten.

Kriterium	**Altes Tool**	**Neues Tool (CVAT)**
Benutzeroberfläche	Grundlegend	Hochmodern und benutzerfreundlich
Labeling-Prozesskette	Datei-Konvertierung nötig	Datei direkt importieren/exportieren
Benachrichtigungssystem	Nicht vorhanden	Vollständig integriert
Teamarbeit	Nicht unterstützt	Unterstützt
Datenvisualisierung	Nicht unterstützt	Umfassende Funktionen

Tabelle 9.1: Vergleich zwischen der Tools nach der Entwicklung

Aus der Vergleichstabelle geht hervor, dass das neue Tool signifikante Vorteile gegenüber dem alten Tool bietet. Dies beginnt bei der modernen und benutzerfreundlichen Oberfläche. Bezüglich der Prozesskette zeigt die Analyse deutlich, dass beim neuen Tool keine Konvertierungen vor dem Labeling-Prozess notwendig sind (siehe Sequenzdiagramme der Prozesskette von dem Labeling-Prozess für das alte Tool (4.1) und neue Tool (9.2), wodurch Zeit und manuelle Schritte eingespart werden. Weiterhin erleichtert das integrierte Benachrichtigungssystem die interne Kommunikation innerhalb des Teams, was besonders vorteilhaft ist, wenn mehrere Nutzer gemeinsam an einem Projekt arbeiten. Dies fördert die Teamarbeit effektiv. Abschließend ermöglicht die Datenvisualisierung eine detaillierte Einsicht in die Arbeitsgeschwindigkeit, was für das Zeitmanagement von großem Nutzen ist.

10 Zusammenfassung und Fazit

Die Optimierung und Vereinfachung von Prozessen, die manuelle Arbeit erfordern, sind besonders bei der Arbeit in großen Teams von großer Bedeutung. Sie ermöglichen oder gewährleisten eine hohe Arbeitsgeschwindigkeit und eine Steigerung der Prozesseffizienz. Im Rahmen dieser Arbeit wurde ein neues Labeling-Tool (CVAT) weiterentwickelt, um ein altes Labeling-Tool zu ersetzen.

Als Erstes wurde eine Untersuchung der Funktionen und Einsatzmöglichkeiten zweier Labeling-Tools durchgeführt, um deren Vorteile und Limitationen bei der Datenannotation zu ergründen. Ein zentraler Punkt der Arbeit vor der Entwicklung lag auf dem praktischen Durchlauf der Labeling-Prozesskette, beginnend mit der Vorbereitung der Daten, über das eigentliche Labeling, bis hin zur Entwicklung von Modellen, um so die Einbindung der Tools in realitätsnahen Anwendungen zu erfassen.

Nach der Durchführung praktischer Anwendungen und einer umfassenden Vergleichsanalyse konzentrierte sich die Arbeit auf die Anpassung und Optimierung des neuen Tools (CVAT), um das alte Labeling-Tool zu ersetzen und den Labeling-Prozess durch die Einführung neuer Funktionen zu vereinfachen. Die Implementierung der definierten Anforderungen umfasste folgende Maßnahmen:

- Integration eines neuen spezifizierten XML-Formats in das neue Tool, den Export- und Importprozess von diesem Format zu ermöglichen.
- Erweiterung der Analyse- und Überwachungsfunktionen, um eine umfassendere Visualisierung von Benutzerdaten, Aufgaben und weiteren relevanten Informationen zu ermöglichen.
- Vervollständigung des Benachrichtigungssystems von CVAT durch die Einführung eines Empfangssystems und eines Nachrichtenmanagers, was die Automatisierung der Kommunikation zwischen den Nutzern des Tools fördert.

Die Funktionalität des neuen Tools nach der Implementierung der Anforderungen wurde der Tools durch die Durchführung identischer Labeling-Prozesse für dieselbe Datei in beiden Tools getestet. Dabei lag der Fokus auf der Qualität der Annotationen im neuen Tool sowie darauf, ob die Ergebnisse der Annotation mit denen des alten Tools übereinstimmen. In einem Teamansatz wurde zudem das Benachrichtigungssystem in Kombination mit der Visualisierung der neuen Daten überprüft. Der Testprozess bestätigte die Fähigkeit des neuen Tools, das alte erfolgreich zu ersetzen, und zeigte auf, wie der Labeling-Prozess durch neue Features, wie das Benachrichtigungssystem und die Datenvisualisierungsfunktionen, die im alten Tool fehlten, vereinfacht wurde.

Literaturverzeichnis

[1] Prof. Dr. Thomas Kudraß. *Taschenbuch Datenbanken*. 2015, S. 20–23. ISBN: 978-3-937514-69-7.

[2] Vector Datadog. *Eeinführung von Vector*. 2019–2024. URL: https://vector.dev/docs/about/what-is-vector/.

[3] Intenta Automotive GmbH. "Über Intenta". In: (2011). URL: https://www.intenta-automotive.de/en/home.html.

[4] John Wiley Sons Ltd. *Web Application Architecture: Principles, protocols and practices*. 2003, S. 5–18. ISBN: 0-471-48656-6.

[5] CVAT.ai. *CVAT-Einfürung*. 2016-202X. URL: https://github.com/opencv/cvat.

[6] CVAT.ai. *CVAT-Annotationstools*. 2016-202X. URL: https://opencv.github.io/cvat/docs/manual/basics/types-of-shapes/.

[7] CVAT.ai. *CVAT-Dateiformats*. 2016-202X. URL: https://opencv.github.io/cvat/docs/manual/advanced/formats/.

[8] CVAT.ai. *CVAT-Review*. 2016-202X. URL: https://opencv.github.io/cvat/docs/manual/advanced/analytics-and-monitoring/manual-qa/.

[9] Python Software Foundation. *Allgemeine Python-FAQ*. 2001–2023. URL: https://docs.python.org/3/faq/general.html.

[10] Mark Summerfield. *Programming in Python 3*. 2008, S. 1. ISBN: 978-0-13-712929-4.

[11] Stoyan Stefanov. *JavaScript Patterns*. 2011, S. 1–3. ISBN: 978-3-89721-598-6.

[12] Django Software Foundation. *Django-overview*. 2005-2023. URL: https://www.djangoproject.com/start/overview/.

[13] Django Software Foundation. *Django-Arbeitsablauf*. 2005-2023. URL: https://docs.djangoproject.com/en/5.0/faq/general/.

[14] Django Software Foundation. *Django-Models*. 2005-2023. URL: https://docs.djangoproject.com/en/4.2/topics/db/models/.

[15] Django Software Foundation. *Django-views*. 2005-2023. URL: https://docs.djangoproject.com/en/4.2/topics/http/views/.

[16] Django Software Foundation. *Django-URLS*. 2005-2023. URL: https://docs.djangoproject.com/en/4.2/topics/http/urls/.

[17] Django Software Foundation. *Django-Template*. 2005-2023. URL: https://docs.djangoproject.com/en/4.2/topics/templates/.

[18] Cory Gackenheimer. *React: Introduction to React*. 2015, S. 1–2. ISBN: 978-1-4842-1245-5.

[19] Webhooks.net. *Webhooks-Defination*. 2023. URL: https://webhook.net/webhooks-vs-apis.

[20] Docker. *Containerisierte Anwendungen*. 2023. URL: https://www.docker.com/resources/what-container/.

[21] Docker. *Dockerfile*. 2023. URL: https://docs.docker.com/engine/reference/builder/.

[22] Docker. *Dockercompose*. 2023. URL: https://docs.docker.com/compose/.

[23] Docker. *Docker-Network*. 2023. URL: https://docs.docker.com/network/.

[24] Docker. *Docker-Network-Drivers*. 2023. URL: https://docs.docker.com/network/drivers/.

[25] Docker. *Docker-Bridge-Network*. 2023. URL: https://docs.docker.com/network/drivers/bridge/.

[26] Grafana Labs. *Grafana-Einführung*. 2023. URL: https://grafana.com/docs/grafana/latest/fundamentals/.

[27] Grafana Labs. *Grafana-Dashboard*. 2023. URL: https://grafana.com/docs/grafana/latest/dashboards/.

[28] Clickhouse. *clickhouse History*. 2016–2023. URL: https://clickhouse.com/docs/en/about-us/history.

[29] Clickhouse. *clickhouse-Fuktionalität*. 2016–2023. URL: https://clickhouse.com/docs/en/intro.

[30] Clickhouse. *clickhouse-Übersicht*. 2016–2023. URL: https://clickhouse.com/.

[31] Mihaela Vela und Hannah Kermes. *XML*. 2017. URL: http://fedora.clarin-d.uni-saarland.de/teaching/Corpus_Linguistics/Tutorial_XML.html.

[32] CVAT.ai. *CVAT-analytics*. 2016-202X. URL: https://opencv.github.io/cvat/docs/administration/advanced/analytics/.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 1. Januar 23 Asaad Askar