HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Projektdokumentation

Weiterentwicklung eines kamerabasierten Smart-Home-Überwachungssystems der Gesichtserkennung

Referent:	Prof. Dr. Elmar Cochlovius
Korreferent:	Judith Jakob
Vorgelegt am:	16.07.2019
Vorgelegt von:	Janine Merz
	Cherubin Nsingi
	Jack Pham
	Valentin Reichenecker
	Michael Zimmermann

Abstract

Die vorliegende Arbeit ist eine Projektdokumentation des Projekts Weiterentwicklung eines kamerabasierten Smart-Home-Überwachungssystems der Gesichtserkennung. Die Hauptaufgabe des Projekts lag darin, ein bestehendes System, welches Kameradaten, hauptsächlich aber Bilddaten, verarbeitet und auf Bewegungen untersucht, mit einer Gesichtsidentifikation auszustatten. Das System wurde um das Open-Source Framework OpenVINO und die benötigten Funktionalitäten erweitert, das Frontend aktualisiert und aufbereitet und es wurde eine Datenbank angelegt, welche das Verwalten von Daten erleichtern soll. Generell ist die Arbeit am Projekt gut verlaufen und neben einem lauffähigen Produkt haben alle Projektmitglieder an wertvollen Erfahrungen in der Arbeit in Projekten bekommen. Das Projekt konnte zwar umgesetzt werden, jedoch mussten viele Funktionalitäten aufgrund von Zeitdruck gestrichen werden. Dies bietet wiederrum zukünftigen Semesterprojekten Möglichkeiten das System zu verbessern und zu erweitern.

Inhaltsverzeichnis

Abstract III
Inhaltsverzeichnis V
Abbildungsverzeichnis IX
Tabellenverzeichnis XI
1 Einleitung 1
2 Projektmanagement 3
2.1 Planung 3
2.2 Rollenverteilung 4
2.3 Anforderungsspezifikation 4
2.4 Gantt Chart 7
3 Backend 9
3.1 Konzept 9
3.2 Verwendete Bibliotheken 9
3.2.1 OpenCV 9
3.2.2 OpenVINO 9
3.2.3 SciPy 10
3.2.4 Flask 10
3.3 Bildverarbeitung 10
3.4 Ausgabestream 11
3.5 Datenverwaltung 12
3.6 Installation 12
3.6.1 Anpassen der verwendeten Vergleichsgesichter 13
3.6.2 Start des Backends 13
4 Frontend 15
4.1 Verwendete Versionen 15

4.2 Installationen für das Frontend 15
4.2.1 Kommandos zur Installation aller benötigten Komponenten 16
4.3 Weitere Installationen für Angular 17
4.3.1 Animationen Framework 17
4.3.2 Forms-API 17
4.4 Frontend starten 17
4.5 Die neuen Komponenten 18
4.5.1 mode-selection 18
4.5.2 fr-event-log 19
4.5.3 person-formular 19
4.5.4 mainpage 20
4.6 Model-Interfaces 21
4.6.1 person 21
4.6.2 event-log-entry 21
4.6.3 person.service 21
4.6.4 Settings 22
4.6.5 Komponenten-Service-Diagramm 22
5 Datenbank 25
5.1 Aufbau 25
5.1.1 Person 26
5.1.2 Picture 26
5.1.3 Notification 26
5.1.4 PersonNotification 27
5.1.5 Command 27
5.1.6 PersonCommand 27
5.1.7 Camera 27
5.2 Schnittstelle im Backend 28

5.3 Installation der Datenbank 29
6 Tests 31
6.1 Unit-Tests 31
6.1.1 Frontend 31
6.1.2 Datenbank-Interface 31
6.2 Manuelle Tests 31
7 Fazit 33
8 Ausblick 35
Eidesstattliche Erklärung 37
Anhang 39

Abbildungsverzeichnis

Abbildung 1: Projektstrukturplan 6
Abbildung 2: Erweitertes Komponenten-Service-Diagramm 23
Abbildung 3: Datenbank ERM 25
Abbildung 4: Übersicht der Datenbankschnittstelle 28

Tabellenverzeichnis

Tabelle 1: Settingserweiterung durch zwei neue Parameter 22

1 Einleitung

Das Thema unseres Semesterprojekts ist ,,Die Weiterentwicklung eines kamerabasierten Smart-Home-Überwachungssystems der Gesichtserkennung. Dieses Semesterprojekt stützt sich auf das vorherige Projekt ,,Smart-Home-Überwachungssystem für eine IP-Kamera“. Unser Team besteht zum einen aus 3 Studenten der Allgemeinen Informatik (AIN) und zum anderen aus 2 Studenten der IT-Produktmanagement (ITP). Während des Semesterprojektes waren für die Betreuung die Dozenten Herr Prof. Dr. Elmar Cochlovius und Frau Judith Jakob zuständig.

Vorausgehend von einer Überwachungskamera mit Motion Capture (Bewegungserkennung), welches uns von dem vorherigen Projektteam schon bereitgestellt wurde, besteht unsere Aufgabe nun die Kamera mit weiteren Funktionen zu erweitern. Die Funktionen beinhalten unter anderem die Gesichtserkennung und die Gesichtsidentifikation. Die IP Kamera soll in der Lage sein, Gesichter sowohl von bekannten als auch von unbekannten Personen identifizieren zu können.

Zum Abgleich der Gesichter wurde intern eine Galerie in der Datenbank erstellt. Die Namen der identifizierten Personen sollen im Kamerastream erscheinen, während unidentifizierte Personen nur mit einer ID versehen werden. Sollte sich eine unautorisierte Person im Blickwinkel der Kamera befinden, wird der Nutzer per E-Mail benachrichtigt.

Selbstverständlich ist das Semesterprojekt so ausgerichtet, sodass das System am "Tag der Informatik" vorführbar sein muss. Die Einteilung unseres Teams am besagten Tag soll aus einem Repräsentanten der Allgemeinen Informatik (AIN) als auch einem IT-Produktmanagers (ITP) bestehen. Für den Fall, dass etwaige organisatorische Fragen bezüglich unseres Projektes auf-

kommen sollten, wäre hierfür der richtige Ansprechpartner der IT Produktmanagers zu nennen. Auf der anderen Seite können die Allgemeinen Informatiker technische Fragen ausführlicher beantworten.

Unser Ziel ist es, ein funktionsfähiges System zu erstellen, welches im Smart Home Labor simuliert werden kann. Die Durchführung unseres Semesterprojekts schließt mit einer Präsentation am Tag der Informatik ab.

2 Projektmanagement

2.1 Planung

Zur Durchführung unseres Projekts, wurde die Methode des agilen Projektmanagements Scrum verwendet. Es wurden intern regelmäßige Treffen mit der Gruppe vereinbart. Das Treffen wurde in wöchentlichen Sprints an jedem Dienstag um 13:15 Uhr abgehalten. In diesen Treffen geht es darum festzulegen, welche Aufgaben bis zum nächsten Termin abgearbeitet werden müssen. Außerdem wurde sich darauf verständigt, dass jedes Teammitglied ein Statusbericht abgibt. Dies soll einen allen Projektmitgliedern den Überblick über unseren Projektstand verschaffen.

In unserem Treffen haben die Gruppenmitglieder die Möglichkeit, ihre eigene Meinung zu den Ergebnissen der anderen Mitglieder zu äußern und Verbesserungsvorschläge zu unterbreiten. Nebenbei wurde ein Protokoll geführt, damit wichtige Informationen sowie Diskussionen und Erkenntnisse festgehalten werden können. Zusätzlich wurde vermerkt, welche Personen zu diesen Terminen anwesend waren und welche nicht.

Das Smart Home Labor wurde uns sowohl für die Erledigung unserer Aufgaben, als auch für regelmäßigen Treffen mit unseren Betreuern zur Verfügung gestellt.

Für die Ablage von Dokumenten wie (z.B. Protokolle, Power Point Präsentationen, Grafiken und Diagramme) als auch Aufgabenergebnisse der Projektmitglieder offenzulegen, haben wir Google Drive genutzt. Wir sehen darin den Vorteil, dass somit Änderungen und Verbesserungen von allen Teammitgliedern individuell vorgenommen werden können. Die Webmail Adressen der Projektbeteiligten wurden ebenfalls verwendet, um Informationen auszutauschen. Des Weiteren wurde als Mittel für die Kommunikation WhatsApp genutzt, um detaillierte Informationen bezüglich des Projektes zu besprechen.

Zusätzlich wurden alle zwei Wochen ein Treffen mit den Betreuern (Herr Prof. Dr. Elmar Cochlovius, Frau Judith Jakob) vereinbart, in welchem wir eine

kleine Präsentation über unseren Projektfortschritt hielten. Im Anschluss haben wir ein direktes Feedback von unseren Betreuern erhalten. Das Feedback hat uns gezeigt, an welchen Stellen wir Verbesserungen, egal ob organisatorischer, technischer oder persönlicher Art, vornehmen müssen.

Um zunächst einen Überblick über die technischen Hintergründe des Projekts zu bekommen, haben sich die Gruppenmitglieder in die Technologien OpenVINO, OpenCV, Python und AngularJS eingearbeitet. Der Hauptkern des Projekts nimmt dabei die Software Implementierung ein.

2.2 Rollenverteilung

Es wurden jedem Projektbeteiligten bestimmte Rollen zugewiesen, auf Basis der bereits vorhandenen Kompetenzen. Dadurch entstanden Konstrukte, in welchen die Studierenden des Studiengangs IT-Produktmanagement sich mit den organisatorischen Aufgaben auseinandergesetzt haben, während die Studierenden des Studiengangs Allgemeine Informatik sich mit dem technischen Aspekt des Projekts beschäftigt haben Die Rollen wurden so vergeben, dass jede Person für mindestens 4 verschiedene Bereich verantwortlich ist. Zu jeder Rolle gibt es zwei Personen, einen Hauptverantwortlichen und im Falle dessen Ausfalls einen Stellvertreter. Die Rollenverteilung ist im Anhang unter C zu finden.

2.3 Anforderungsspezifikation

Eine Anforderungsspezifikation beschreibt alle funktionalen Anforderungen, welche für den Erfolg der Anwendung umzusetzen sind. Die Anforderungen für das Projekt wurden nach dem ersten Kick Off Meeting von dem Betreuer in Auftrag gegeben. Die Anforderungen welches das System erfüllen soll sind folgende:

1. Gesichtserkennung

Die Software soll in der Lage sein in einem Video-Stream Gesichter zu erkennen. Erkannte Gesichter sollen entsprechend hervorgehoben werden.

2. Intervall für Video-Stream (Erweiterung zu Gesichtserkennung)

Beim Video-Stream kann das gewünschte Intervall festgelegt werden.

3. Gesichtsidentifikation

Erkennt die Software ein Gesicht, so soll ein Abgleich mit Bildern in einer internen Galerie abgeglichen werden. Dadurch soll die Software in der Lage sein, Gesichter zu identifizieren und einer Person zuzuordnen.

4. Ablage von optimierten Dateien in der Datenbank (Erweiterung zu Gesichtsidentifikation)

Es soll die Möglichkeit gegeben sein, optimierte Dateien für die Gesichtsidentifikation in der Datenbank abzulegen.

5. Schnittstelle zum bestehenden System (PIPCO)

Es soll eine Schnittstelle bzw. eine Kommunikationsmöglichkeit mit dem bestehendem System PIPCO entwickelt werden. Zunächst ist lediglich die Gesichtserkennung aktiv. Wird ein Gesicht erkannt, jedoch schlägt die Identifikation fehl handelt es sich um ein unbekanntes Gesicht und die Bewegungserkennung wird hinzugeschaltet.

6. Darstellung der Gesichtserkennung im Browser

Ein Gesicht, welches erkannt wird, soll im Browser durch einen über das Originalbild gelegten Rahmen kenntlich gemacht werden. Ist die Identifikation erfolgreich, so soll dies über die zusätzliche Anzeige der Initialen der identifizierten Person geschehen.

7. Hinzufügen von neuen Gesichtern für die Identifikation

Über das Web-Interface sollen Bilder hochgeladen werden können, welche in der Galerie für einen Abgleich für eine Identifikation abgelegt werden. Zusätzlich sollen auch Name/Initialen und Status/Rolle eingegeben werden können (z.B. vertrauenswürdig, Professor, Student).

8. E-Mail-Benachrichtigung bei erkannten aber nicht identifizierbaren Gesichtern

Handelt es sich bei einem Gesicht um ein nicht Identifizierbares, so soll eine E-Mail-Benachrichtigung an einen E-Mail Verteiler gesendet werden.

9. Logging

Generell soll bei jedem Schritt der Gesichtserkennung (Erkennung → Identifikation) ein Logging-Eintrag erstellt werden. Zusätzlich soll über gesendete Emails, gespeicherte Bilder und anderen Ereignissen informiert werden.

10. Bilder speichern

Bei einem nicht identifizierbaren Gesicht soll ein Bild des Gesichts abgespeichert werden.

2.4 Projektstrukturplan

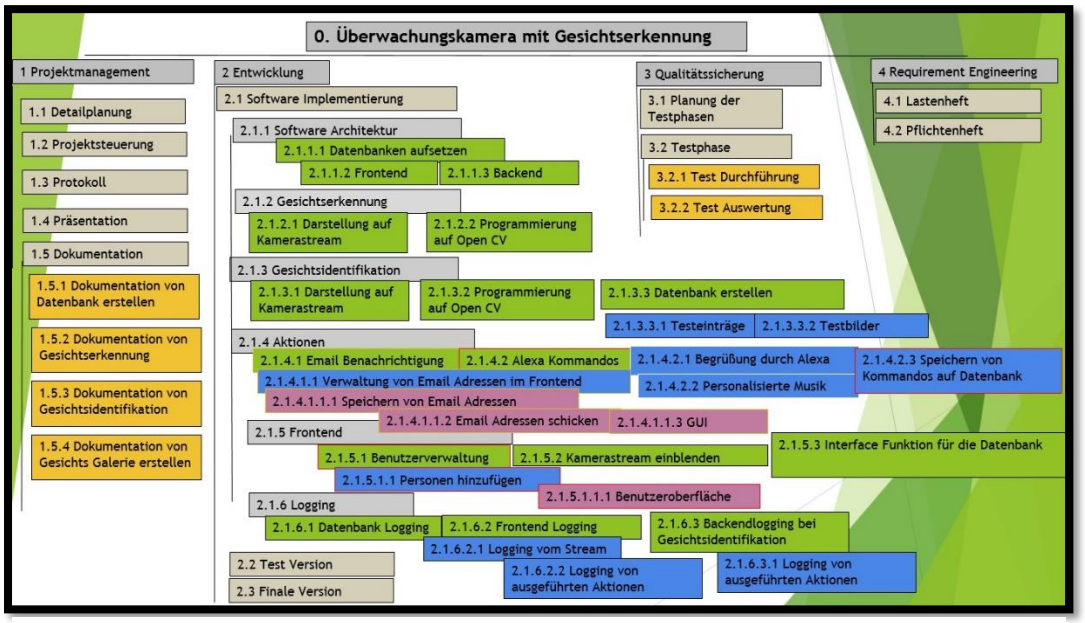


Abbildung 1: Projektstrukturplan

Unser Projektstrukturplan, siehe Abbildung 1, zeigt die Struktur des Projekts mit der wir uns im Laufe des Semesters beschäftigt haben. Unser Seminarprojekt ist in 4 Hauptarbeitspakete aufgeteilt. Zum einen haben wir das Projektmanagement, die Entwicklung, die Qualitätssicherung und das Requirement Engineering.

Die Aufgabeneinteilung wurde je nach dem Studiengang der Projektbeteiligten aufgeteilt. Während die IT Produktmanager (Cherubin Nsingi, Jack Pham) sich mit den Arbeitspaketen Projektmanagement, Qualitätssicherung und das Requirement Engineering, also generell mit Organisatorischen und Projektleitenden Aufgaben beschäftigen, arbeiten die Allgemeinen Informatiker (Janine Merz, Michael Zimmermann, Valentin Reichenecker) an dem technischen Aspekt des Projekts, nämlich dem Entwurf, Konstruktion und Implementierung der Software.

Das Arbeitspaket Projektmanagement setzt sich aus den Teil-Arbeitspaket Detailplanung, Projektsteuerung, Protokoll, Präsentation und Dokumentation zusammen.

Die Entwicklung stellt das umfassendste und aufwendigste Arbeitspaket da. Dieses wird zunächst in die Software Implementierung, die erste Test Version und die finale Version unterteilt. Die Software Implementierung besteht dann weiterhin aus 6 Aufgabenbereichen, die Software Architektur, die Gesichtserkennung, die Gesichtsidentifikation, die Aktionen, das Frontend und das Logging.

Diese Arbeitspakete werden auch wieder in kleine Arbeitspakete zerlegt. Das Arbeitspaket Qualitätssicherung besteht aus den Tätigkeiten der Planung der Testphasen und die Testphase an sich, welches aus der Durchführung des Tests und dem Auswerten der Testergebnisse besteht. Das Arbeitspaket Requirements Engineering setzt sich aus den Lasten- und Pflichtenheft (siehe Anforderungsspezifikation) zusammen.

2.4 Gantt Chart

Um die zeitliche Abfolge von Aktivitäten in grafischer Form unseres Projektplans darstellen zu können, haben wir das Programm Gantt Projekt ausgewählt. Es gibt 8 Arbeitspakete, die sich in Bereichen wie Projektstart, Planungsphase, Software implementieren, Frontend Logging, Neue Personen hinzufügen/ entfernen, Testphase, Endphase und Dokumentation einordnen lassen.

Das Gantt Chart wurde auf Basis des Projektstrukturplans erstellt. Wie aus dem Gantt Chart ersichtlich bzw. zu entnehmen ist, nimmt dabei das Arbeitspaket Software implementieren die meiste Zeit für die Erledigung bzw. Durchführung ein. Für die 2 Arbeitspakete Projektstart und Planungsphase wurde hingegen wenig Zeit benötigt (ca. 1 Woche).

Das Gantt Chart wurde stets mit den Projektfortschritten als auch von den bisherigen Meetings mit unserem Betreuer Herr Cochlovius und Frau Jakob verändert und angepasst.

Inhalte der Arbeitspakete sind:

1. Projektstart: Projektdefinition, Projektziele
2. Planungsphase: Team zusammenstellen, Abstimmung der Meetings, Strukturplanung, Umfeldanalyse, Lastenheft erstellen, Pflichtenheft erstellen
3. Software implementieren: Personen verwalten, Softwarearchitektur, Gesichtserkennung, Programmierung Open CV, Datenbank erstellen, Darstellung im Stream, Gesichtsidentifikation, Darstellung auf Homestream, Aktionen verwalten (E-Mail Benachrichtigung, Verwalten von E-Mail Adressen im Frontend, GUI), Alexa Kommandos (Begrüßung durch Alexa, personalisierte Musik, Speichern von Kommandos im Datenbank), Schnittstelle Bewegungserkennung und Gesichtsidentifikation, Logging, Datenbank Logging, Backendlogging bei Gesichtsidentifikation
4. Frontend Logging: Logging von Stream
5. Neue Personen hinzufügen/entfernen: Benutzeroberfläche, Interface-Funktion für die Datenbank
6. Testphase: Testdurchführung 1 und 2 mit den darauffolgenden Testauswertung 1 und 2
7. Endphase: Präsentation, Protokolle
8. Dokumentation: Dokumentation 1-5

Das Gantt-Diagramm ist im Anhang unter D zu finden.

3 Backend

Der Quellcode ist in einem GitHub-Repository zu finden (https://github.com/goingenrage/PIPCO-Faceident).

3.1 Konzept

Das Backend soll mit der Programmiersprache Python entwickelt werden. Das Backend umfasst das Abgreifen einer Kamera mithilfe OpenCV, die Erkennung von Gesichtern in den zu verarbeitenden Frames und die Wiedererkennung von Personen, die berechtigt sind sich in diesem Raum aufzuhalten. Außerdem sollen Videosequenzen aufgenommen werden, wenn sich eine nicht berechtigte Person zeigt. Zudem wird eine E-Mail-Benachrichtigung an die im Frontend eingetragenen Email-Adressen geschickt.

3.2 Verwendete Bibliotheken

3.2.1 OpenCV

OpenCV ist eine Bibliothek für Bildverarbeitung und maschinelles Sehen. Sie ist unter den Bedingungen der BSD-Lizenz zur freien Verfügung. Einzusetzen ist die Bibliothek in den Programmiersprachen C, C++, Java und Python. Da wir unser Projekt in Python umgesetzt haben, interessiert uns nur der Python Aspekt. OpenCV haben wir in unserem Projekt dahingehen eingesetzt, dass wir Frames von einer Kamera ausgelesen haben und zur Weiterverarbeitung vorbereitet haben. Zudem wurde mit OpenCV eine Videosequenz aufgenommen, wenn sich eine nicht berechtigte Person zeigt.

3.2.2 OpenVINO

OpenVINO ist ein Toolkit von Intel, dass wir für die Gesichtserkennung und Wiedererkennung verwendet haben. OpenVINO steht für Open Visual Inferencing and Neural Network Optimization, wobei wir nur die Inference Engine genutzt haben.

3.2.3 SciPy

SciPy ist eine Python-Bibliothek die mathematische Werkzeuge zur Verfügung stellt. Genutzt haben wir die Funktion „cosine“, welche die Kosinus-Ähnlichkeit zweier Matrizen berechnet.

3.2.4 Flask

Um die Schnittstelle zwischen Frontend und Backend zu bewerkstelligen, haben wir wie im Vorgängerprojekt auf einen Flask-Server gesetzt. Flask ist ein in Python geschriebenes Webframework, das einen Webserver für Testzwecke mitliefert. Dadurch ist es für uns einfach gewesen die Schnittstelle aufzusetzen.

3.3 Bildverarbeitung

Die Inference Engine arbeitet mit vortrainierten Models, welche von OpenVINO zur Verfügung gestellt gibt. Wir benutzen ein Model für die Gesichtserkennung und für die Wiedererkennung. Anfangs müssen zwei Engines erstellt werden und jeweils mit dem richtigen Model initialisiert werden. Dadurch können wir im weiteren Verlauf jedes reinkommende Frame analysieren.

Im Initialisierungsprozess werden alle relevanten Pfadangaben aus einer config-Datei eingelesen um es benutzerfreundlich zu ermöglichen auf verschiedenen Umgebungen das Programm zu starten.

Anfangs wird eine Personengalerie mit Bildern der Personen erstellt, die eine Berechtigung haben. Dafür werden aus der Datenbank alle Bilder der Personen in einen Ordner gespeichert. Anschließend wird ein JSON-File erstellt, welches Namen der Personen speichert und die dazugehörigen Pfadangaben zu den Bildern. Diese werden durch die Inference Engine geschickt und eine Liste mit allen Matrizen der Gesichter erstellt. Anhand dieser Liste kann später abgeglichen werden, ob eine erkannte Person eine Zugangsberechtigung hat oder nicht.

Die mit dem Gesichtserkennungsmodel initialisierte Engine bekommt das komplette Frame von der Kamera und erkennt alle Gesichter. Es liefert Koordinaten der Regionen im Bild, indem sich Gesichter befinden. Anhand dieser Koordinaten kann das Gesicht extrahiert werden und für die Wiedererkennung bereitgestellt werden. Die Inference Engine, die mit dem Wiedererkennungsmodel initialisiert wurde analysiert das Gesicht und liefert eine Matrix mit Gesichtsmerkmalen zurück. Diese Matrix wird mit einer Personengalerie abgeglichen, die nach demselben Schema die Matrizen erstellt hat. Anhand der Kosinus-Ähnlichkeit wird bestimmt, ob es dasselbe Gesicht ist oder nicht.

Wenn eine Matrix mit einer aus der Gesichtsgalerie übereinstimmt, wird der Name der Person über dem Gesicht auf dem Frame platziert und es werden keine weiteren Aktionen durchgeführt.

Wenn eine Matrix mit keiner der gespeicherten Personen übereinstimmt, wird geprüft, ob es auch mehrere Frames hintereinander als nicht identifizierbare Person erkannt wird um Fehlmeldungen zu vermeiden. Wenn das der Fall ist, wird die Person als nicht erlaubt aber schon einmal gesehen abgespeichert um möglicherweise unterschiedliche Aktionen durchführen zu können.

In jedem Fall kann eine Videoaufnahme gestartet werden, um im Frontend eine Sequenz anzeigen zu können, indem die Person zu sehen ist. Auch kann eine E-Mail-Benachrichtigung verschickt werden. Ein- bzw. Ausschalten steuern Buttons im Frontend.

3.4 Ausgabestream

Die Frames, auf denen die Ausgaben der Gesichtswiedererkennung geschrieben sind sollen auf der Weboberfläche angezeigt werden. Um das zu ermöglichen müssen diese natürlich vom Backend bereitgestellt werden. Hier haben wir uns am Vorgängerprojekt orientiert. Das neueste Frame wird im Modul DataStorage zwischengespeichert. Der Webserver holt sich dieses Frame und gibt es auf der URL-Endung /videostream aus. Das Frontend kann diesen dann darstellen.

Zusätzlich dazu stellt das Backend ein Videostream zur Verfügung, welches den Kamerastream darstellt ohne zusätzliche Informationen.

3.5 Datenverwaltung

Da wir einige Daten haben, welche von verschiedenen Modulen genutzt werden, haben wir wie schon das Vorgängerprojekt die Daten in einem Modul verwaltet, welches als Singleton-Objekt implementiert wurde. Diese Implementierung war weitestgehend vom Vorgängerprojekt schon vorhanden und wir konnten unsere Daten ergänzen. Der Vorteil dieser Datenhaltung mit einem Singleton-Objekt ist es, dass immer dieselben Daten von allen Instanzen oder Modulen genutzt werden. Somit ist es ausgeschlossen, das falsche oder veraltete Daten verwendet werden.

3.6 Installation

Um das entwickelte Backend zu betreiben werden die in 3.2 beschriebenen Bibliotheken gebraucht. Auf der im Anhang befindlichen Daten-DVD befindet sich das Installationsverzeichnis von OpenVINO welches OpenVINO und OpenCV installiert. Zusätzlich befindet sich dort ein Shell-Skript, welches mit sudo-Rechten ausgeführt werden muss. Dieses wird alle Installationsschritte ausführen. Der Nutzer muss damit nur den Installationsanweisungen folgen, die von OpenVINO auf der Konsole ausgegeben werden. Das Skript wird auch die Umgebungsvariablen für OpenVINO richtig setzen. Anschließend wird es die benötigten Komponenten für Python installieren.

Wichtig ist, dass das Skript mit sudo-Rechten ausgeführt wird, denn sonst werden Pfadangaben nicht mehr stimmen und die weitere Installation muss manuell fortgesetzt werden.

Nachdem das Skript durchgelaufen ist, muss durch den Befehl

```
sudo find / -name libcpu_extension*
```

gecheckt werden, ob der ausgegebene Pfad mit dem des im Facerecognition-Backend liegendem config-File übereinstimmt. Sollte dies nicht der Fall sein,

muss es natürlich angepasst werden. Bei allen Testsystemen war es der Fall, dass das Shared-Object mit dem Postfix avx2 benötigt wurde. Falls es zu Problemen damit kommt, soll entweder mit avx512 oder mit sse4 probiert werden.

Im selben config-File können weitere Pfadangaben gesteuert werden, jedoch ist die Vorkonfiguration für einen reibungslosen Betrieb eingestellt.

3.6.1 Anpassen der verwendeten Vergleichsgesichter

In der Standardkonfiguration werden Referenzbilder für die Gesichtswiedererkennung aus der verwendeten Datenbank, die über das Frontend mit Bildern gefüllt werden kann, genommen. Falls dies nicht gewollt wird, muss in dem im Verzeichnis Gesichtswiedererkennung-Backend liegendem config-File der Pfad für die zu verwendende Gesichtsgallery angepasst werden. Dafür die Variable face_gallery mit dem entsprechen Pfad angepasst werden.

3.6.2 Start des Backends

Um nach erfolgreicher Installation das Backend zu starten, muss im Verzeichnis Gesichtswiedererkennung-Backend der Befehl

```
python3 Main.py
```

ausgeführt werden. Dies wird den Backendserver und die Bildverarbeitung starten. Anschließend kann sich über das Frontend angemeldet werden.

4 Frontend

Für die Weiterentwicklung des bestehenden Frontends, haben wir wie unsere Vorgänger, mit Angular gearbeitet. Da Angular komponentenbasiert und dies ein großer Vorteil ist, haben wir uns ebenfalls für dieses Framework entschieden.

Der Quellcode für das Frontend ist in einem GitHub-Repository zu finden (https://github.com/goingenrage/PIPCO-Frontend).

4.1 Verwendete Versionen

- OS: Ubuntu 16.04
- Node: 12.6.0
- Npm: 6.9.0
- Angular CLI: 7.3.8
- Visual Studio Code: OS: 1.33.1

4.2 Installationen für das Frontend

Bevor das vorhandene Frontend benutzt sowie erweitert werden kann, müssen einige grundlegende Komponenten installiert werden:

1. **node.js**:ist eine Plattform, um serverseitige JavaScript-Applikationen zu ermöglichen
   cURL: (Client for URLs) Kommandozeilen Programm
2. **Npm:** (Node Package Manager) ist für die Verwaltung der Abhängigkeiten einer Node-Applikation zuständig
   Angular CLI (Command Line Interface) und **karma**
3. **Visual Studio Code** (https://code.visualstudio.com/Download) nicht zwingend erforderlich

Um sich beim Frontend nun anmelden zu können, müssen alle Komponente vom Backend, ebenfalls installiert sein (siehe 3.6).

4.2.1 Kommandos zur Installation aller benötigten Komponenten

1. **node.js und cURL:**

```
sudo apt-get install curl
curl -sL https://deb.nodesource.com/setup_8.x -o no-
desource_setupC.sh
sudo bash nodesource_setup.sh
sudo apt-get install nodejs
```

2. **NPM, Angular CLI, Karma:** (das -g bedeutet hier global)

```
sudo npm install -g npm@latest
npm install -g @angular/cli
npm install --save -dev karma@4.1.0
```

Um zu kontrollieren, ob alles korrekt installiert wurde, überprüft man einfach, die jeweiligen Versionen:

```
node –version
npm –version
ng –version
```

Nach diesen Schritten, muss noch der Owner angepasst werden

```
sudo chown -R $USER:$GROUP ~/.npm
sudo chown -R $USER:$GROUP ~/.config
```

Nun ist es noch wichtig, dass man die IP des Servers in /src/environments/environments.ts bei der backendAdress einträgt.

3. **Visual Studio Code** (https://code.visualstudio.com/Download)
nach Download muss man noch folgendes Kommando in die Konsole eingeben:

```
sudo dpkg -i code_1.33.1-1554971066_amd64.deb
```

4.3 Weitere Installationen für Angular

4.3.1 Animationen Framework

Angular bietet eine geschickte Verbindung zwischen Komponentenzustand und Animationsausführung. Auch wenn Animationen für die Kernfunktionalität nicht unbedingt notwendig sind, wirkt ein Frontend mit Animationen durchaus professioneller als ohne.

Deshalb haben wir für unser PIPCO Projekt die Angular Animationen mit folgendem Befehl installiert:

```
npm install --save @angular/material @angular/cdk
@angular/animations
```

4.3.2 Forms-API

Ebenfalls wird für die Weiterentwicklung des bestehenden Frontends ein Formular benötigt. Für diesen Fall stellt Angular die sogenannte „Forms-API“ bereit. Diese implementiert eine Reihe von Anweisungen und Anbietern für die Kommunikation mit systemeigenen DOM-Elementen (Document Object Model= Schnittstelle zwischen HTML und dynamischem JavaScript) beim Erstellen von Formularen zur Erfassung von Benutzereingaben. Forms-API können mit folgendem Befehl installiert werden:

```
npm install @angular/forms –save
```

4.4 Frontend starten

Bevor man das Frontend starten kann, ist es wichtig, im Backend-Ordner über die „cmd“ folgende Befehle auszuführen:

```
python Webserver.py

python Main.py
```

Nach ausführen der oben genannten Befehle geht man in die Entwicklungsumgebung von Angular und gibt im Angular-Terminal den „`ng serve`“ Befehl ein. Wenn man diese Schritte gemacht hat, öffnet man den Browser und gibt http://localhost:4200 ein. Nun wird das Frontend angezeigt und ein Login mit dem Passwort ist möglich. Nun können alle Funktionen des Frontends benutzt werden.

4.5 Die neuen Komponenten

Das bestehenden PIPCO System soll durch neue Komponenten (Components) erweitert werden. Da die Vorgängergruppe ausführlich auf die Angular-Begriffe wie Components, Services, Guards und Module eingegangen ist, möchten wir an dieser Stelle auf deren Dokumentation verweisen, um Wiederholungen zu vermeiden. Dies ist der Grund, weshalb wir hier nur auf die neu erstellen Komponenten eingehen werden. Zu den neuen Komponenten gehört unter anderem das Logging für die Gesichtserkennung, sowie die Möglichkeit den Kamera-Modus auszuwählen („CamOnly“, „Motion“ und „Face recognition“) und eine Komponente, die es ermöglicht, Personen neu anzulegen und ein Bild des Nutzers, vom Frontend direkt in die Datenbank einzufügen. Im Folgenden werden die Bausteine, die das bestehende Frontend erweitern, vorgestellt.

4.5.1 mode-selection

In der „mode-selection“-Komponente kann der Modus der Kamera eingestellt werden. Hier gibt es drei auswählbare Möglichkeiten, welche durch Radio Buttons ausgewählt werden können:

1. **„Cam only“:** hier wird nur der normale Kamerastream angezeigt.
2. **„Motion detection“**: bei Auswahl dieses Radio Buttons wird auf dem Kamerastream die Bewegungserkennung hinzugeschalten.
3. **„Face recognition“:** hier wird die Gesichtserkennung aktiviert.

Hat man nun einen der drei möglichen Buttons ausgewählt, wird dieser Input durch eine Zahl (CamOnly = 0, Motion detection =1, Face recognition =2) gespeichert.

Wenn sich nun der Radio-Button ändert, wird im sogenannten EventEmitter (der für die Übergabe der Daten an die Oberkomponente zuständig ist) die ausgewählte Zahl weitergegeben und der Modus wird in die Variable „changeMod“ gespeichert und so ändert sich der Modus bzw. die Anzeige im KameraStream. Der ausgewählte Modus wird zudem in den Einstellungen gespeichert, so dass bei dem nächsten Aufruf der Webseite dieselbe Einstellung voreingestellt ist.

4.5.2 fr-event-log

Diese Komponente dient dazu, dem Anwender die aufgezeichneten Gesichter aufzulisten. Das Starten dieser Aufnahme wird per Klick auf den „ON“ Button aktiviert. Dieser „ON“ Button befindet sich in der TitleBarComponent, hier kann man die Aufnahme auch wieder mit „OFF“ deaktivieren. Sobald er aktiviert ist, wird die Gesichtserkennung aktiviert und es speichert das Thumbnail. Es funktioniert so, dass durch ein Event eine Funktion in der MainPageComponent aufgerufen wird. Diese Funktion löst wiederum in der VideoComponent das eigentliche Abspielen des Videos aus. Genau wie in der EventLogComponent implementiert, werden auch in dieser Komponente nicht alle Einträge angezeigt. Es werden immer erst 6 Einträge angezeigt und sobald gescrollt wird, werden dem Nutzer weitere 6 Einträge aufgelistet. So müssen nicht immer alle Daten vom Backend aufgerufen werden. Des Weiteren ist es möglich die Aufnahmen wieder zu löschen.

4.5.3 person-formular

Das person-formular erweitert die SettingsPageComponent. In der neuen Komponente hat man die Möglichkeit über das Frontend Daten direkt in die Datenbank zu speichern. Dies geschieht mithilfe eines Formulars, in welches Daten wie Name, Vorname, Kommentare, sowie Bilder hochgeladen werden

können. Dieses Formular funktioniert so, dass der Wert in der Benutzeroberfläche immer mit dem Domänenmodell aus der Klasse synchronisiert wird. Dies ist möglich, da es sich hier um eine bidirektionale Datenbindung handelt (das sogenannte two-way-binding). Damit können Benutzer Daten von der Komponente zur Ansicht und von der Ansicht zur Komponente austauschen. Diese Datenbindung wird mithilfe einer ngModel-Direktive erreicht. Jeder Input wird vom ngModel verarbeitet, welches Zugriff auf die input-Elemente hat. Um ein Bild hochladen zu können gibt es die Methode onFileSelected(), welche ein Event als Übergabeparameter hat. In diesem Event stehen die Infos, die an die Oberkomponente geschickt werden. Der Upload-Button ist mit einer Funktion versehen, die bei Aufruf das Personen Objekt füllt. In diesem Personen Objekt werden persönliche Daten wie z.B. Name, Vorname und Kommentare, sowie das Bild gespeichert. Das Bild, wird für den Transfer in das Backend mit Base64 asynchron encodiert. Das zuvor erstellte Objekt wird an die Methode addNewPerson() übergeben. Erwähnenswert ist, dass bei der addNewPerson()-Methode ein Observable-Objekt zurückgegeben wird. Wichtig ist hierbei, dass Observables „lazy“ sind, was bedeutet, dass sie erst mit der Veröffentlichung von Werten beginnen, sobald sie abonniert wurden. Dieses abonnieren geschieht, indem sie mit der subscribe()-Methode der Instanz aufgerufen werden.

4.5.4 mainpage

Alle neu erstellen Komponente müssen natürlich in die Mainpage-Komponente eingetragen werden, damit sie auf der Seite sichtbar sind. Dies geschieht ganz einfach, indem die jeweiligen Selector-Elemente in einen eigenen div-class-Bereich der mainpage.component.html eingetragen werden.

4.6 Model-Interfaces

4.6.1 person

Um die Struktur der Vorgängergruppe beizubehalten, haben wir, um Personen hinzufügen zu können, eine „Person“-Klasse erstellt. Diese Klasse enthält alle nötigen Variablen, um eine Person hinzufügen zu können. Jede Person hat natürlich einen Namen, einen Vornamen, sowie ein Kommentar-Feld, in welches zusätzliche Infos hinterlegt werden können und natürlich kann man bei jeder Person auch Bilder hinzufügen. Diese „Person“-Klasse wurde in die „person-formular“ Komponente eingebunden.

4.6.2 event-log-entry

Das Interface „event-log-entry“ konnte auch für die neue „fr-event-log“ Komponente verwendet werden. Die ID wird hier vom Backend generiert. Das Attribut „message“ ist die Anzeigenachricht zum Log-Eintrag, der „timestamp“ ist der Zeitpunkt der Erstellung des Log-Eintrages. Das „thumbnail“ ist der erste Frame des aufgezeichneten Videos als Base64-Image und „recording“ ist der Dateiname des aufgezeichneten Videos, der als ID der Video Datei dient.

4.6.3 person.service

Bei diesem Model handelt es sich um einen Service, der für das Hinzufügen von neuen Personen wichtig ist, dies geschieht mit der Methode „addNewPerson“. Die meisten Frontend-Anwendungen kommunizieren mit den Backend-Diensten über das HTTP-Protokoll. Deshalb wird in diesem Service im Konstruktor eine Variable http vom Typ HttpClient übergeben, dies geschieht über eine Dependency Injection. In der Methode „addNewPerson“ wird das Personen Objekt als Parameter übergeben. Eine Transformation des Personen Objekts in eine für den Datentransfer zum Backend geeignete Datenstruktur, welches das JSON-Format ist, findet automatisch statt. Der im Konstruktor erzeugte http-Client schickt eine POST-Anfrage mit den Daten an das Backend.

4.6.4 Settings

Das Settings Model wurde im Zuge der Erstellung von neuen Komponenten entsprechend erweitert. Analog zu der bisher existierenden Logik der Benutzeroberfläche, wurden zwei neue Parameter definiert. In nachstehender Tabelle ist sind die erweiterten Parameter sowie eine Beschreibung zu finden.

Name	Parameter	Beschreibung
Settings	Fr_log_enabled	Gibt an, ob das Logging für die Gesichtsidentifikation aktiviert ist
	Cam_mode	Gibt an, welcher Streammodus aktiv ist. 0= Nur Kamera 1= Bewegungserkennung 2= Gesichtsidentifikation

Tabelle 1: Settingserweiterung durch zwei neue Parameter

4.6.5 Komponenten-Service-Diagramm

In der folgenden Abbildung wird der Aufbau des Frontends in Bezug auf seine Komponenten und deren Nutzung von Services dargestellt. Dieses Diagramm wurde von den Vorgängern übernommen und erweitert. Die neu hinzugefügten Komponenten und Services wurden in einer anderen Farbe hervorgehoben.

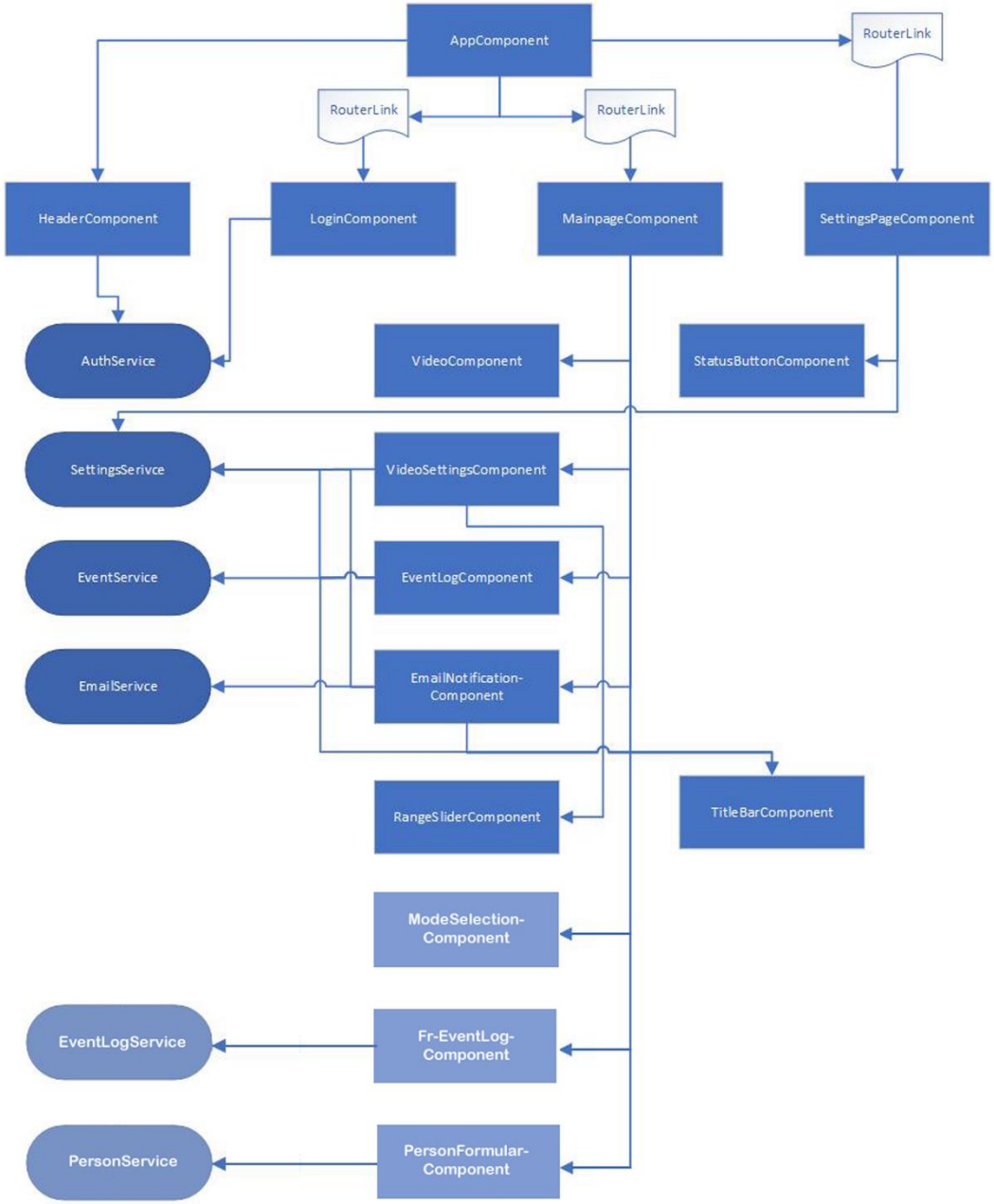


Abbildung 2: Erweitertes Komponenten-Service-Diagramm

5 Datenbank

Um eine persistente und konsistente Datenspeicherung zu gewährleisten, welche auch die Referenzbilder der angelegten Personen enthält, wurde eine relationale Datenbank verwendet. Bei der Analyse der Anforderungen ist aufgefallen, dass die Datenbank so schlank wie möglich sein soll, damit diese auch auf Systemen mit begrenzter Hardware betrieben werden kann. Mit diesem Aspekt im Hinterkopf fiel die Auswahl der Datenbank auf die Open-Source basierte Lösung, SQLite.

5.1 Aufbau

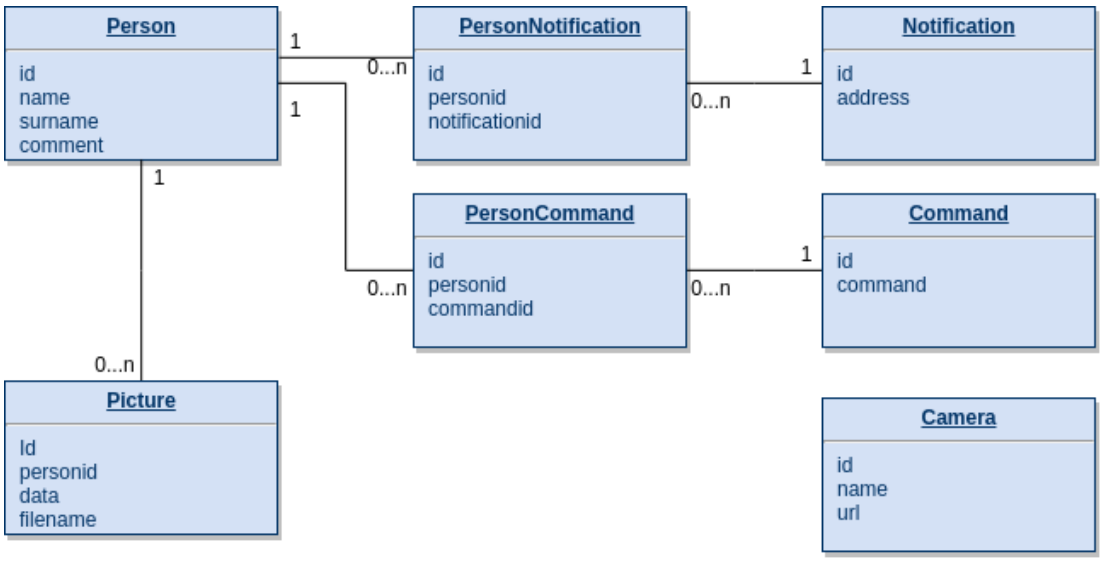


Abbildung 3: Datenbank ERM

Diese Abbildung zeigt ein Entity-Relationship-Model der erstellten Datenbank. Im Folgenden werden die einzelnen Tabellen möglichst genau spezifiziert.

5.1.1 Person

In der Tabelle Person werden Daten zu Personen hinterlegt. Jeder Eintrag entspricht dabei einer Person, welche durch eine fortlaufende und automatisch generierte **id** identifizierbar ist. Informationen, welche bezüglich einer Person, in der Datenbank gehalten werden sind zunächst der Name. Hierbei wird der verpflichtende Vorname in die Spalte **name** und der Nachname in die Spalte **surname** eingetragen. Der Datentyp der beiden Spalten ist varchar(255). In die Spalte **comment** kann ein freiwilliger Kommentar mit einer Zeichenbegrenzung von 255 eingetragen werden.

5.1.2 Picture

In die Tabelle Picture werden Daten zu Bildern abgelegt, welche eine bestimmte Person betreffen. Identifizierbar durch eine automatisch generierte und fortlaufende **id** können Datensätze dieser Tabelle eindeutig einer einzigen Person zugeordnet werden. Denn Datensätze müssen in der Spalte **personid** eine ID aus der Tabelle Person referenzieren. Zusätzlich beinhalten Datensätze das eigentliche Bild in der Spalte **data** in Form eines blobs und den Dateinamen in der Spalte **filename**, welcher auf eine Länge von 255 Zeichen begrenzt ist.

5.1.3 Notification

In der Tabelle Notification werden E-Mail-Adressen für eine Benachrichtigung per Mail abgelegt. In der Tabelle können die Datensätze eindeutig durch eine fortlaufende **id** identifiziert werden. In der Spalte **address** wird eine maximal 255 Zeichen lange E-Mail-Adresse hinterlegt, welche im Falle eines Events benachrichtigt werden soll.

5.1.4 PersonNotification

Diese Tabelle dient der Vorbeugung einer entstehenden n:m-Beziehung. In der Tabelle kann jeder Eintrag durch eine **id** eindeutig identifiziert werden. In der Spalte **personid** werden lediglich vorhandene IDs aus der Person Tabelle referenziert. Für die Spalte **notificationid** gilt dies analog. Die Referenzen sind durch eine Fremdschlüsselbeziehung geschützt.

5.1.5 Command

In dieser Tabelle werden Kommandos für eine eventuell zukünftig folgende Steuerung von Smart-Home-Hubs oder einzelnen Smart-Home-Endgeräten hinterlegt. Wie auch bei den vorherigen Tabellen, kann jeder Eintrag eindeutig durch eine **id** identifiziert werden. In der Spalte **command** werden die einzelnen Kommandos hinterlegt. Momentan ist der Datentyp dieser Spalte noch varchar(255), jedoch sollte bei einer später folgenden Umsetzung der Datentyp bzw. Auch die Tabelle entsprechend angepasst werden.

5.1.6 PersonCommand

Diese Tabelle dient der Vorbeugung einer entstehenden n:m-Beziehung. In der Tabelle kann jeder Eintrag durch eine **id** eindeutig identifiziert werden. In der Spalte **personid** werden lediglich vorhandene IDs aus der Person Tabelle referenziert. Für die Spalte **commandid** gilt dies analog. Die Referenzen sind durch eine Fremdschlüsselbeziehung geschützt.

5.1.7 Camera

In dieser Tabelle werden Daten zu vorhandenen Kamerastreams abgelegt. Zusätzlich zu der fortlaufenden **id** werden die Kamerastreams mit einer **url** als varchar(255) und mit einem **name**, also einem Namen des Streams in Form eines Zeichenkette mit maximal 255 Zeichen abgelegt.

5.2 Schnittstelle im Backend

Um die Datenbank aus dem Backend heraus zu nutzen, wurde eine Schnittstelle erstellt. Diese beinhaltet alle relevanten Anweisungen an die Datenbank.

interfacedb
-__db_location: string
-__tmp_directory: string
+ initialize(database_location, temporary_saving_path): boolean
- __check_for_initialization(): boolean
+ database_connect(): sqlite3.Connection, sqlite3.Cursor
+ database_dump(saving_path): string
+ database_import(file_path): boolean
+ get_by_person(person_name, person_surname=''): DetailedPerson
- __get_pictures_by_personid(person_id): string[]
+ get_all_persons(): Person[]
- __get_personid_by_name(person_name, person_surname=''): int
+ get_all_pictures(): string[]
+ delete_person_by_name(person_name, person_surname=''): int
+ delete_images_by_personid(person_id): int
- __delete_images_by_personid(person_id, cur): int
+ insert_person(person_name, person_surname = '', comment=''): int
+ insert_picture(personId, file): int
- __insert_picture(personId, file, cur): int
+ insert_picture_as_bytes(personId, byteObject): int
+ update_person(person_id, person_name, person_surname='', person_comment=''): boolean

Abbildung 4: Übersicht der Datenbankschnittstelle

In obenstehender Abbildung ist das Modul interfacedb mit Methoden und Variablen zu sehen. Bei der Benamung der Methoden wurde versucht möglichst selbstsprechende Namen auszuwählen. In der im Anhang als A deklarierte Tabelle werden die Funktionalitäten der Methoden in Kurzform erklärt. Nähere Dokumentation und weitere Hinweise sind dann im Quellcode zu finden.

Generell wurden die meisten Funktionen um Datenbank-Einträge zu ändern, anzulegen und zu löschen mit Transaktionen versehen. Bei Transaktionen handelt es sich um ein Konstrukt, welches dem Quellcodeverwaltungssystem Git nahekommt. Sollen Daten in der Datenbank beeinflusst werden, so werden die Änderungen zunächst mithilfe des sqlite3.Cursor-Objekts ausgeführt und dadurch zwischengespeichert. Auf der Datenbank ist bis zu diesem Zeitpunkt

keine Änderung des Datensatzes zu sehen. Erst durch ein Commit, welches auf dem sqlite3.Connection-Objekt ausgeführt wird, werden alle auf dem Cursor zwischengespeicherten Vorgänge auch auf die Datenbank übertragen. Praktisch hierbei ist, dass nicht erst bei dem Commit auf Fehler hingewiesen wird, sondern der Cursor findet bereits bei der Ausführung der Anweisungen etwaige Syntax-Fehler in den SQL-Anweisungen und macht darauf aufmerksam.

Eine Standard SQL-Funktion ist in diesem Projekt somit nach dem folgenden Schema aufgebaut.

```
(1)     try:
(2)       __check_for_initialization()
(3)       con, cur = database_connect()
(4)       sql = "Insert into…"
(5)       cur.execute(sql, [parameter1, parameter2])
(6)       con.commit()
(7)       return cur.lastrowid
(8)     except Exception as e:
(9)       con.rollback()
(10)      raise e
```

In Zeile zwei wird überprüft, ob die Datenbank bereits initialisiert wurde. Daraufhin wird eine Verbindung zur Datenbank aufgebaut und es werden ein Connection und ein Cursor Objekt definiert. In Zeile 4 wird das SQL-Statement vordefiniert und in Zeile 5 durch den Cursor ausgeführt. Sind im Statement Parameter vorhanden, so werden diese in Zeile 5 mitgegeben. In Zeile 6 werden die Statements, in diesem Beispiel aber nur das eine Statement, committet und in Zeile 7 wird die ID des erstellten Eintrages als Rückgabewert der aufrufenden Methode übergeben. Sollte eine Exception auftreten, werden die ausgeführten Statements zurückgesetzt, siehe Zeile 9, und die Exception an die aufrufende Methode weitergeleitet, siehe Zeile 10.

5.3 Installation der Datenbank

Unter Linux Systemen kann mithilfe des Aptitude Paketmanagers die benötigte Datenbanksoftware heruntergeladen werden.

```
sudo apt-get install sqlite3
```

Mit dem Befehl sqlite3 kann nun unter Angabe einer .db Datei eine Datenbank geöffnet werden. Kann unter dem angegebenen Namen keine Datei gefunden werden, so wird eine neue .db-Datei im aktuellen Arbeitsverzeichnis angelegt.

```
sqlite3 pipco.db
```

Nun befindet man sich im Kommandomodus von SQLite. In diesem kann man nun mit untenstehendem Befehl die in den Installationsdateien beigelegte .sql-Datei einlesen. SQLite führt dabei die Befehle aus, welche in der .sql-Datei angegeben sind.

```
.read pipco_db.sql
```

Innerhalb der Datenbank werden mit Fremdschlüsselbeziehungen gearbeitet. SQLite beachtet diese Constraints aber standardmäßig nicht, weshalb diese mit dem Einlesen der .sql-Datei aktiviert werden.

Das Vorhandensein der benötigten Tabellen kann mithilfe des folgenden Kommandos überprüft werden. Es werden dann alle in der Datenbank erstellten Tabellen angezeigt.

```
.tables
```

Die Aktivierung der Überprüfung der Fremdschlüsselbeziehungen kann mithilfe des folgenden Codes abgefragt werden.

```
PRAGMA foreign_keys;
```

Sollte das System eine 1 als Antwort liefern, so sind die Constraints aktiviert. Ist jedoch eine 0 vorhanden, so müssen diese entweder durch nochmaliges einlesen der .sql-Datei oder durch manuelles Eingeben des folgenden Befehls aktiviert werden.

```
PRAGMA foreign_keys = 1;
```

6 Tests

6.1 Unit-Tests

6.1.1 Frontend

Da Unit-Tests des Frontends nur schwer umzusetzen sind, aufgrund von Abhängigkeiten bzw. Schachtelungen von Komponenten usw., haben wir uns dafür entschieden, die Komponenten mit dem Test-Framework Karma, welches Angular standardmäßig liefert, zu testen. Die Ergebnisse sind im Anhang unter F zu finden.

6.1.2 Datenbank-Interface

Im Modul interface_unittest.py wird das Datenbank Interface getestet. Es wird als Interface importiert, weshalb in der Tabelle, welche unter Anhang B zu finden ist, alle Aufrufe der interfacedb durch *interface.xy()* aufgerufen werden.

Sollen die Unit-Tests der Datenbank durchgeführt werden, so empfiehlt es sich die Testumgebung im Ordner /scripts/dbunittest/ im Quellcode zu nutzen. Die dort vorfindbare Struktur solle vor Ausführung der Unit-Tests des Datenbank-Interfaces angegeben werden, da die Datenbank speziell für die Tests angelegte Einträge enthält und die Unit-Tests auch Daten dieser Einträge als Ergebnis erwarten. Im Falle von Anpassungen, sollte immer der Unit-Test in Kombination mit den relevanten Datenbankeinträgen bearbeitet werden. Sollte die Datenbank einmal nicht funktionieren oder nicht nutzbar sein, so kann mit der in der Struktur vorhanden dumpoutput.sql-Datei eine neue Datenbank erstellt werden. Hierzu kann die Installationsanleitung, siehe Kapitel 5.3, verwendet werden. Lediglich beim einlesen der sql-Datei sollte die dumpoutput.sql Datei angegeben werden.

6.2 Manuelle Tests

Da sich das Frontend sowie das Backend nur schwer mithilfe von Unittests testen lassen, wurden die Funktionalitäten in einem Systemtest untersucht und getestet. Die Testfälle, sowie die Ergebnisse sind im Anhang unter E zu finden.

7 Fazit

Im Großen und Ganzen ist es unserer Gruppe gut gelungen unsere Aufgaben umzusetzen. Die weiterentwickelte Applikation erfüllt die meisten Anforderungen aus der Anforderungsspezifikation, es konnten jedoch nicht alle Funktionen implementiert werden. Die Alexa Kommandos mussten wir bedauerlicherweise aus zeitlichen Gründen auslassen. Die Einarbeitung in den ganzen Anwendungen (Open CV, Python, Angular usw.) hat viel Zeit in Anspruch genommen, und die Implementierung der Gesichtsidentifikation war aufwendiger als geplant.

Anfangs existierte ein Problem bei der Kommunikation zwischen AIN und ITP aufgrund verschiedener Kompetenzen. Erst nachdem diese Barriere durchbrochen wurde konnte das Team produktiv weiterarbeiten. Streitigkeiten zwischen den Projektmitgliedern gab es in unserer Gruppe nicht, wir sind immer friedlich und respektvoll miteinander umgegangen. Die Rollenverteilungen wurden gut ausgewählt, jeder hat sich mit den Aufgaben auseinandergesetzt, welche den Kompetenzen der Verantwortlichen entspricht. Als es Probleme bei der Erstellung des Frontends gab wurde sofort nachgeholfen, und somit die verlorene Zeit ausgeglichen.

Das Semesterprojekt war herausfordernd, was jedoch als positive Sache angesehen wurde, da man lernt schwierige Aufgaben zu meistern. Wir haben viel Erfahrung bezüglich Planung und Organisation dazu gewonnen. Es konnte sich Wissen über den technischen Hintergrund eines Überwachungssystems mit Gesichtserkennung angeeignet werden und Funktionalitäten sowie deren Aufbau verstanden werden. Abschließend können wir sagen, dass das Projekt Spaß gemacht hat und ein Erfolg war. Die erlernten Fähigkeiten können sicherlich für das Praxissemester oder für das zukünftige Berufsleben weiterverwendet werden.

8 Ausblick

Das Projekt bietet in vielerlei Hinsicht Ansätze für andere Projektgruppen. Zunächst muss gesagt werden, dass wir unsere gestellten Anforderungen derart zurückschrauben mussten, so dass viele Funktionen der Datenbank ungenutzt bleiben.

Zum Beispiel wurde ein Formular um Personen in der Datenbank hinzuzufügen erstellt. Generell kann man eine Verwaltung der Personen und Bilder in der Datenbank über das Frontend vornehmen. Somit würden dann neue Anforderungen entstehen, wie das aktive Einsetzen von Rollen beziehungsweise Nutzerberechtigungen.

Ein weiterer Punkt wäre das Verwalten von E-Mail-Adressen bei erkannten Incidents, bzw. Vorfällen. Bisher werden E-Mails bei nicht identifizierbaren Personen verschickt. Die Datenbank ist jedoch so ausgelegt worden, dass zu jeder erkennbaren Person auch eine E-Mail-Adresse hinterlegt werden kann, welche benachrichtigt werden soll. Man könnte hierbei auch einen „Dummy"-Eintrag für nicht erkennbare Personen erstellen und somit eine Liste mit zu benachrichtigenden E-Mail-Adressen bei unbekannten Personen erstellen.

Zudem stand während unseres Bearbeitungszeitraums die Anforderung im Raum, das Gesichtsidentifikationssystem mit dem Smart-Home-Labor zu verknüpfen. Es könnten dann zum Beispiel zu jeder erkennbaren Person, bzw. angelegten Person in der Datenbank, ein oder mehrere Kommandos für ein Smart-Home-Assistant, wie Alexa, Google Home, o.Ä. hinterlegt werden. Somit wäre es möglich sobald eine Person erkannt wird, diese per Alexa-Kommando mit Namen zu begrüßen, die favorisierte Musikplaylist zu spielen und vieles mehr. Dies könnte durch vordefinierte Routinen erfolgen, welche in der Datenbank hinterlegt werden.

Bisher läuft das System auf einer Kamera im Smart-Home-Labor. Uns wurde bereits in Ausblick gestellt, dass das System auch auf mehrere Kameras des Labors übertragbar sein könnte, sofern die Zeit reichen würde. Genau aus diesem Gesichtspunkt wurde die Tabelle camera erstellt. In dieser könnten dann IP-Adressen verschiedenster Kameras hinterlegt und mit Namen versehen

werden, so dass im Frontend dann „on the fly“ zwischen verschiedenen Kameras gewechselt werden könnte.

Im Hinblick auf die Technik, genauer gesagt auf die Gesichtserkennung, gibt es auch Potenzial. Die Performance könnte durch das Anpassen der verarbeitenden Matrizen dahingehend verbessert werden, dass einerseits die Erkennung flüssiger verläuft, andererseits aber auch nicht frontal in die Kamera schauende Personen identifiziert werden können.

Generell kann gesagt werden, dass eine Verwaltung von Tabelleneinträgen über das Frontend hilfreich wäre. Dadurch könnten Personen, Bilder, Kamerastream-URLs, hinterlegte E-Mail-Adressen und hinterlegte Kommandos konfiguriert, angepasst und verändert werden.

Zusätzlich könnten bisher nicht identifizierbare, also unbekannte, Personen schneller angelegt werden. Es könnte möglich gemacht werden, unbekannte Personen, welche von der Kamera erfasst wurden, nachträglich und händisch mit Namen und Kommentar zu versehen. Die würde das kurzfristige Anlegen von Personen in der Datenbank enorm vereinfachen, da ein Bild der Person durch die Kameraaufnahme bereits vorhanden ist.

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Furtwangen, 16.07.2019 _____

Furtwangen, 16.07.2019 _____

Furtwangen, 16.07.2019 _____

Furtwangen, 16.07.2019 _____

Furtwangen, 16.07.2019 _____

Anhang

A. Methodenübersicht des Datenbankinterfaces

Methoden- / Variablensignatur	Kurzerklärung	Rückgabewert
-__db_location: string	Globale Variable mit Pfad zur Datenbank	
-__tmp_directory: string	Globale Variable mit Pfad zu einem temporären Speicherverzeichnis	
+initialize(database_location, temporary_saving_path): boolean	Setzt die globalen Variablen.	True, falls erfolgreich
-__check_for_initialization(): boolean	Prüft ob globale Variablen initialisiert wurden	True, falls initialisiert
+database_connect(): sqlite3.Connection, sqlite3.Cursor	Verbindet zur Datenbank unter __db_location	Connection und Cursor-Objekt zur Datenban
+database_dump(saving_path): string	Exportiert die DB in eine .sql Datei	Name der .sql-Datei
+database_import(file_path): boolean	Importiert eine .sql-Datei	True, falls erfolgreich
+get_by_person(person_name, person_surname=''): DetailedPerson	Sucht anhand der Parameter nach einer Person in DB	DetailedPerson Objekt mit Daten zu der gesuchten Person
-__get_pictures_by_personid(person_id): string[]	Sucht anhand des Parameters nach Bildern und legt sie im __tmp_directory ab	String-Array mit Dateinamen
+get_all_persons(): Person[]	Gibt alle in der DB vorhandenen Personen mit allen Infos zurück	Person-Array mit allen in der DB vorhandenen Personen
-__get_personid_by_name(person_name, person_surname=''): int	Sucht in der DB nach einem Eintrag mit Übereinstimmung mit den Parametern	PersonenID der gefundenen Person als int
+get_all_pictures(): string[]	Speichert alle in der DB gespeicherten Bilder im __tmp_directory	Liste aller Dateinamen
+delete_person_by_name(person_name, person_surname=''): int	Löscht eine Person anhand der Parameter	PersonenID des gelöschten Eintrags
+delete_images_by_personid(person_id): int	Löscht ein oder mehrere Bild/er anhand des Parameters	Anzahl der betroffenen DB-Einträge

+__delete_images_by_personid(person_id, cur): int	Löscht ein oder mehrere Bild/er anhand des Parameters, ohne eine Transaktion auszuführen	Anzahl der betroffenen DB-Einträge
+insert_person(person_name, person_surname = '', comment=''): int	Legt eine neue Person mit den Parametern an	ID des angelegten Person Eintrags
+insert_picture(personId, file): int	Verknüpft eine vorhandene PersonenId mit einem zu sichernden Bild	ID des angelegten Picture Eintrags
+__insert_picture(personId, file, cur): int	Verknüpft eine vorhandene PersonenId mit einem zu sichernden Bild, ohne Transaktion	ID des angelegten Picture Eintrags
+insert_picture_as_bytes(personId, byteObject): int	Verknüpft eine vorhandene PersonenId mit einem zu sichernden Bild, welches als Bytestrom vorhanden sein sollte	ID des angelegten Picture Eintrags
+update_person(person_id, person_name, person_surname='', person_comment=''): boolean	Aktualisiert einen Personen Eintrag anhand der ID mit den als Parametern übergebenen Daten	True, falls erfolgreich

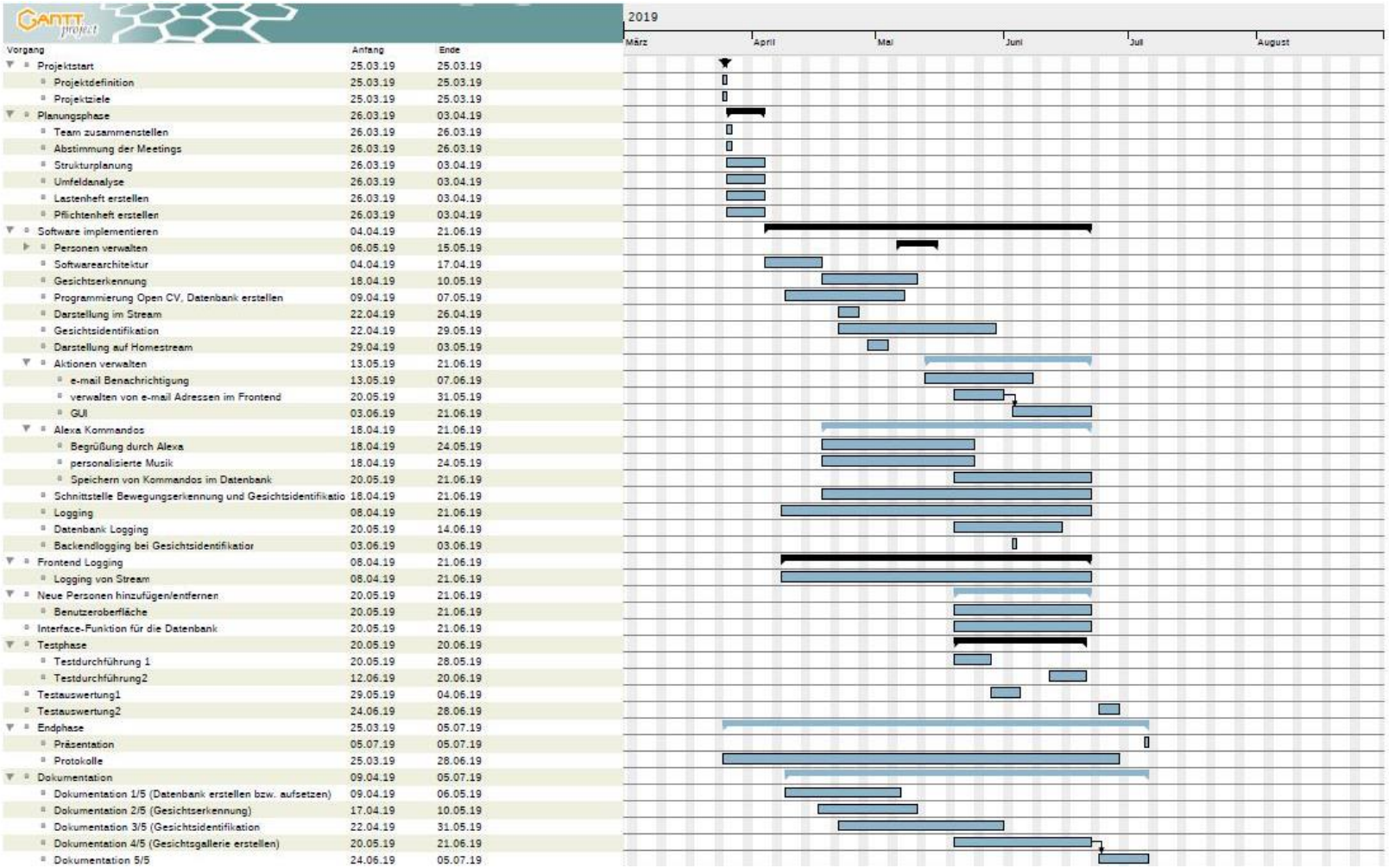
B. Unittests für das Datenbankinterface

Testname	(zusätzlich) getestete Funktionen	Beschreibung
test_Initialization	Interface.initialize() Interface.check_for_initialization()	Testet die Initialisierungsfunktion auf Korrektheit
test_ExceptionInitialization	Exception bei: Interface.check_for_initialization()	Testet ob die die Verbindung zur Datenbank ohne vorige Initialisierung funktioniert
test_DumpDatabase	Interface.database_dump()	Überprüft, ob die Datenbank Dump Funktion korrekt funktioniert und löscht die angelegte Datei.
test_ImportDatabase	Interface.database_import()	Zurzeit Auskommentiert. Überprüft ob ein dump file eingelesen werden kann. Soll dieser Test durchgeführt werden, so muss der Zeitstempel entsprechend angepasst werden.
test_ExceptionPersonInsert	Exception bei: interface.insert_person()	Prüft ob eine Person angelegt werden kann, welche den gleichen Namen hat, wie eine bereits in der DB existierende Person
test_PersonInsertAndDelete	interface.insert_person() interface.get_by_person() interface.delete_person_by_name()	Prüft ob eine Person angelegt werden kann, Daten über diese abgerufen werden kann und diese dann anhand des Namens gelöscht werden kann
test_PersonInsert	-	Prüft ob eine Person angelegt werden kann und löscht diese wieder
test_PictureInsertFile	Interface.insert_picture() Interface.delete_images_by_personid()	Fügt ein Bild als Datei ein und löscht dieses wieder.
test_ExceptionPictureNotFound	Exception bei: Interface.insert_picture()	Versucht ein Bild in die DB einzufügen, welches nicht existiert
test_PictureInsertBytes	interface.insert_picture_as_bytes()	Fügt ein Bild als Bytestrom ein und löscht es danach wieder
test_PersonUpdate	interface.update_person() interface.get_by_person()	Ändert den Namen einer Person, sucht nach diesem Namen in der Datenbank und gleicht ab, ob die ID des Eintrags noch identisch ist. Danach wird die Änderung rückgängig gemacht.
test_ExceptionPersonUpdate	Exception bei: Interface.update_person()	Testet, ob es möglich ist den Namen einer Person zu einem bereits existierenden Namen zu ändern.
test_PersonDelete	interface.delete_person_by_name()	Fügt einen Testeintrag hinzu und versucht diesen wieder zu löschen.
test_PictureDelete	interface.delete_images_by_personid()	Testet, ob ein Bild hinzugefügt werden kann und wieder gelöscht werden kann
test_PictureDeleteNone	-	Testet, ob es möglich ist, Bildern von Personen, die keine Bilder haben, zu löschen
test_GetPerson	-	Testet, ob nach einer Person anhand des Namens gesucht werden kann
test_GetNonExistingPerson	-	Testet die ordnungsgemäße Funktion, wenn nach einer Person gesucht wird, die nicht vorhanden ist
test_GetAllPersons	interface.get_all_persons()	Testet, ob es möglich ist alle Personen auszugeben
test_GetAllPictures	interface.get_all_pictures()	Testet, ob alle Bilder ausgegeben werden können.

C. Rollenverteilung

Rolle	Beschreibung	Verantwortlichkeiten	Personen
Projektplanungs- und Steuerungs-Verantwortlicher	Planung und Steuerung des Projekts	Erfüllung der gesetzten Ziele innerhalb des Projektzeitraums	Jack, Cherubin
Frontend-Verantwortlicher	Planung und Implementierung der grafischen Oberfläche	Funktionsfähige grafische Oberfläche	Janine, Michael
OpenCV-Verantwortlicher	Planung und Implementierung von OpenCV-Funktionalitäten	Funktionsfähige OpenCV-Funktionen	Valentin, Janine
Projektkoordinations-Verantwortlicher	Koordination von Meetings und anderen Terminen	Termineinhaltung der Projektmitgliedern	Cherubin, Jack
Dokumentations-Verantwortlicher	Dokumentationen verwalten	Vollständigkeit der Projektdokumentation	Valentin, Jack
Datenbank- und Systemadministrator	Datenbank und andere Systeme und Dienste zur Verfügung stellen	Funktionsfähige Systeme, Dienste und eingerichtete Datenbank	Michael, Janine
Test-Verantwortlicher	Prüfung der Funktionalitäten durch die Analyse der Software mithilfe von Testfällen	Korrekte Tests	Jack, Cherubin
Schnittstellen-Verantwortlicher	Stellt die Kommunikation zwischen einzelner Softwaremodulen sicher	Kommunikation zwischen Backend, Frontend und Kamera	Michael, Janine
Software-Architekt	Erstellung der Software-Architektur	Einhaltung der Architektur	Janine, Valentin
Protokoll-Verantwortlicher	Protokolle der Meetings schreiben und den Teammitgliedern zur Verfügung stellen	Vollständige und genaue Protokolle der Meetings	Michael, Cherubin

D. Gantt-Diagramm

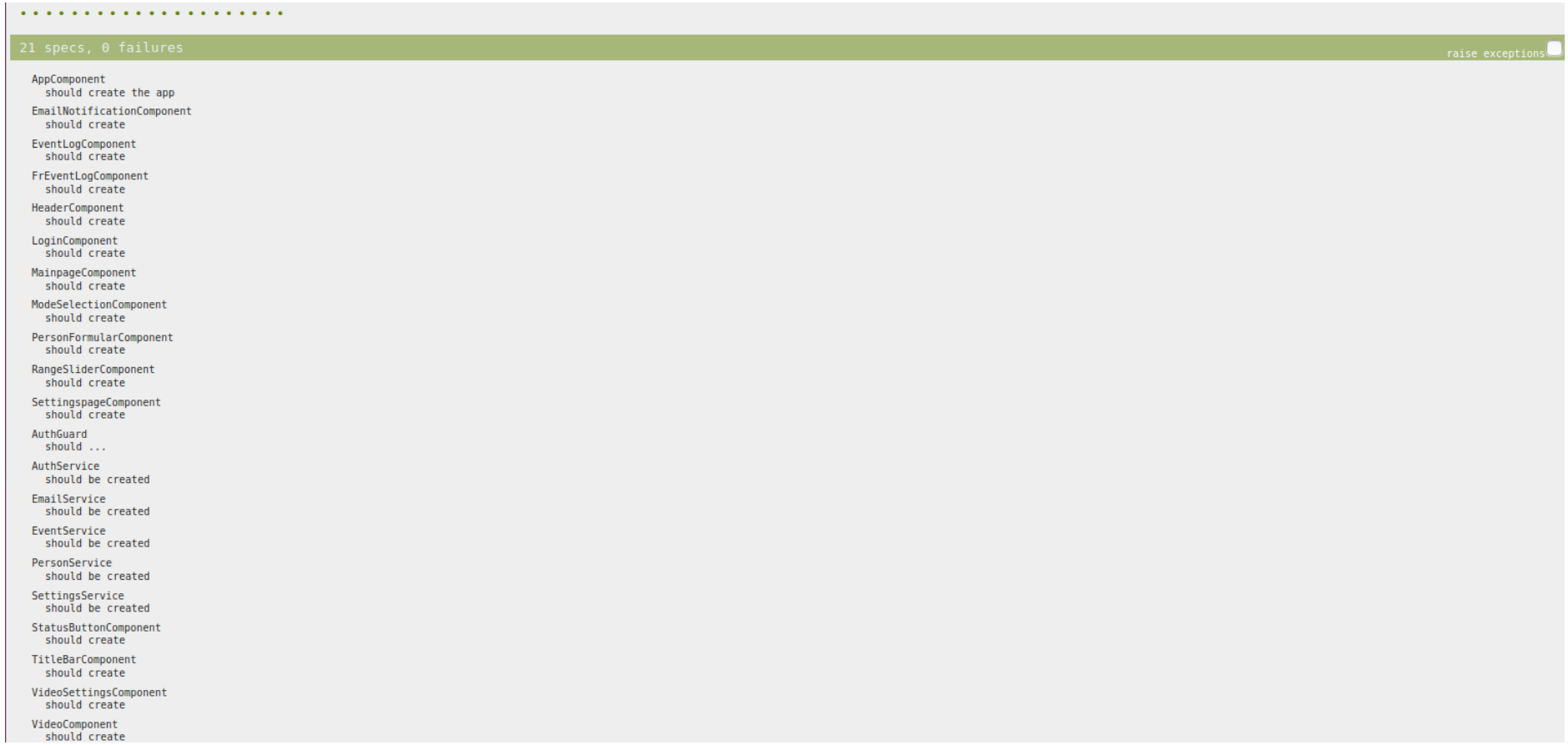


Vorgang	Anfang	Ende
Projektstart	25.03.19	25.03.19
Projektdefinition	25.03.19	25.03.19
Projektziele	25.03.19	25.03.19
Planungsphase	26.03.19	03.04.19
Team zusammenstellen	26.03.19	26.03.19
Abstimmung der Meetings	26.03.19	26.03.19
Strukturplanung	26.03.19	03.04.19
Umfeldanalyse	26.03.19	03.04.19
Lastenheft erstellen	26.03.19	03.04.19
Pflichtenheft erstellen	26.03.19	03.04.19
Software implementieren	04.04.19	21.06.19
Personen verwalten	06.05.19	15.05.19
Softwarearchitektur	04.04.19	17.04.19
Gesichtserkennung	18.04.19	10.05.19
Programmierung Open CV, Datenbank erstellen	09.04.19	07.05.19
Darstellung im Stream	22.04.19	26.04.19
Gesichtsidentifikation	22.04.19	29.05.19
Darstellung auf Homestream	29.04.19	03.05.19
Aktionen verwalten	13.05.19	21.06.19
e-mail Benachrichtigung	13.05.19	07.06.19
verwalten von e-mail Adressen im Frontend	20.05.19	31.05.19
GUI	03.06.19	21.06.19
Alexa Kommandos	18.04.19	21.06.19
Begrüßung durch Alexa	18.04.19	24.05.19
personalisierte Musik	18.04.19	24.05.19
Speichern von Kommandos im Datenbank	20.05.19	21.06.19
Schnittstelle Bewegungserkennung und Gesichtsidentifikatio	18.04.19	21.06.19
Logging	08.04.19	21.06.19
Datenbank Logging	20.05.19	14.06.19
Backendlogging bei Gesichtsidentifikatior	03.06.19	03.06.19
Frontend Logging	08.04.19	21.06.19
Logging von Stream	08.04.19	21.06.19
Neue Personen hinzufügen/entfernen	20.05.19	21.06.19
Benutzeroberfläche	20.05.19	21.06.19
Interface-Funktion für die Datenbank	20.05.19	21.06.19
Testphase	20.05.19	20.06.19
Testdurchführung 1	20.05.19	28.05.19
Testdurchführung2	12.06.19	20.06.19
Testauswertung1	29.05.19	04.06.19
Testauswertung2	24.06.19	28.06.19
Endphase	25.03.19	05.07.19
Präsentation	05.07.19	05.07.19
Protokolle	25.03.19	28.06.19
Dokumentation	09.04.19	05.07.19
Dokumentation 1/5 (Datenbank erstellen bzw. aufsetzen)	09.04.19	06.05.19
Dokumentation 2/5 (Gesichtserkennung)	17.04.19	10.05.19
Dokumentation 3/5 (Gesichtsidentifikation	22.04.19	31.05.19
Dokumentation 4/5 (Gesichtsgallerie erstellen)	20.05.19	21.06.19
Dokumentation 5/5	24.06.19	05.07.19

E. Systemtestfälle und Ergebnisse

Test	Testaktion	Erwartetes Ergebnis:	Tatsächliches Ergebnis:
1	Button gedrückt auf Cam-Only	Nur Kamera wird aktiviert	Kamera wird aktiviert
2	Button auf Face Recognition gedrückt	Kamera + Gesichtserkennung wird aktiviert	Gesichtserkennung ist aktiviert. Personen werden erkannt oder mit fortlaufender Zahl zugeordnet.
3	Button bei Save unknown person incidents gedrückt	Es wird ein Videoclip der nicht identifizierbaren Person erstellt.	Speicherung von Videoclips. Horizontale Darstellung der Informationen: Thumbnail, Timestamp, Message, Delete.
4	Button Delete bei Aufnahmen gedrückt	Die Aufnahme wird gelöscht.	Die Aufnahme wird gelöscht
5	Upload Button des Personen-Formulars wird gedrückt, aber es wird keine Datei ausgewählt.	Keine Reaktion, ein Anlegen einer Person ohne Bild soll nicht möglich sein	Person wird nicht angelegt.
6	Upload Button des Personen Formulars wird gedrückt. Datei ist ausgewählt	Person wird in die Datenbank gespeichert.	Person wird in die Datenbank gespeichert. Alle Eingaben, bis auf die ausgewählte Bilddatei, werden gelöscht bzw. Eingabefelder geleert

F. Testergebnis Frontend mit Karma



```
• • • • • • • • • • • • • • • • • • • • •

21 specs, 0 failures                                                    raise exceptions

  AppComponent
    should create the app
  EmailNotificationComponent
    should create
  EventLogComponent
    should create
  FrEventLogComponent
    should create
  HeaderComponent
    should create
  LoginComponent
    should create
  MainpageComponent
    should create
  ModeSelectionComponent
    should create
  PersonFormularComponent
    should create
  RangeSliderComponent
    should create
  SettingspageComponent
    should create
  AuthGuard
    should ...
  AuthService
    should be created
  EmailService
    should be created
  EventService
    should be created
  PersonService
    should be created
  SettingsService
    should be created
  StatusButtonComponent
    should create
  TitleBarComponent
    should create
  VideoSettingsComponent
    should create
  VideoComponent
    should create
```