HAW HAMBURG

BACHELORTHESIS
Kim Fabian Tiedemann

Umsetzung eines berührungslosen Interfaces für Quantified Self Daten auf einem Smart Mirror

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Kim Fabian Tiedemann

Umsetzung eines berührungslosen Interfaces für Quantified Self Daten auf einem Smart Mirror

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

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Thomas Lehmann

Eingereicht am: 28. September 2021

Kim Fabian Tiedemann

Thema der Arbeit

Umsetzung eines berührungslosen Interfaces für Quantified Self Daten auf einem Smart Mirror

Stichworte

Quantified Self, Android, Gesichtserkennung, Computer Vision, Sprachsteuerung

Kurzzusammenfassung

Die zunehmende Allgegenwärtigkeit von Computern begünstigt die Anwendung natürlicher und berührungsloser Interaktionsmöglichkeiten. Im Rahmen dieser Arbeit wird daher untersucht, wie berührungslose Formen der Interaktion verwendet werden können, um Geräte in einem Smart Home zu steuern. Dafür wird ein auf einem Smart Mirror basierendes System entwickelt, dass die Möglichkeit einer sprachlichen Interaktion, sowie einer Identifikation durch Gesichtserkennung bietet und abhängig von der erkannten Person, Self-Tracking Daten visualisiert.

Kim Fabian Tiedemann

Title of Thesis

Implementation of a touchless interface for quantified self data on a smart mirror

Keywords

Quantified Self, Android, Face Recognition, Computer Vision, Voice Control

Abstract

The increasing ubiquity of computers favors the use of natural and touchless interaction possibilities. In the context of this thesis it is therefore investigated how touchless forms of interaction can be used to control devices in a smart home. For this purpose, a system based on a smart mirror is developed, which offers the possibility of a voice interaction as

well as an identification by facial recognition and visualizes self-tracking data depending on the recognized person.

Inhaltsverzeichnis

Abbildungsverzeichnis **vii**

Tabellenverzeichnis **viii**

1 Einleitung **1**

1.1 Motivation 1
1.2 Ziele 2
1.3 Aufbau der Arbeit 3

2 Analyse **4**

2.1 Projektszenario 4
2.2 Smart Environments 5
2.2.1 Ubiquitous Computing 5
2.2.2 Calm Technology 6
2.2.3 Interaktionsmöglichkeiten 6
2.2.4 Smart Devices 7
2.2.5 Living Place 8
2.2.6 Ähnliche Projekte 9
2.3 Quantified Self 10
2.4 Identifikationsverfahren 11
2.5 Computer Vision 12
2.5.1 Bibliotheken 12
2.5.2 Einflüsse auf die Gesichtserkennung 13
2.5.3 Phasen der Gesichtsidentifikation 14
2.5.4 Extraktion der Gesichtsmerkmale 14
2.5.5 Local Binary Pattern Histograms 16
2.6 Spracherkennung 17
2.6.1 Einflüsse auf die Spracherkennung 18
2.6.2 Funktionsweise 19

2.6.3 Bibliotheken . . . 20
2.7 Anforderungen an die Anwendung . . . 21

3 Umsetzung **23**
3.1 Interaktion mit dem System . . . 23
3.2 Systemkomponenten . . . 25
3.3 Module . . . 26
3.3.1 Kommunikation zwischen Modulen . . . 27
3.4 Primäre Interaktionsabläufe . . . 28
3.5 Datenmodell der Datenbank . . . 30
3.6 Implementierung der Module . . . 31
3.6.1 Interne Kommunikation . . . 31
3.6.2 Datenbankverwaltung . . . 32
3.6.3 Benutzerverwaltung . . . 33
3.6.4 Gesichtserkennung . . . 34
3.6.5 Sprachsteuerung . . . 36
3.6.6 BLE-Handler . . . 38
3.6.7 Datenvisualisierung . . . 40
3.7 Evaluation . . . 42
3.7.1 Fazit der Evaluation . . . 44

4 Zusammenfassung und Ausblick **46**
4.1 Zusammenfassung . . . 46
4.2 Ausblick . . . 47

Literaturverzeichnis **49**

Selbstständigkeitserklärung **55**

Abbildungsverzeichnis

2.1 Ansicht des Badezimmers vom Fenster 9
2.2 Ansicht des Smart mirror 9
2.3 Richters bewegliches Spiegelobjekt 9
2.4 Architektur des Bianco Spiegelsystems 10
2.5 Phasen der Gesichtsidentifikation 14
2.6 Mögliche Haar Cascade Features 15
2.7 Anwendung der Features auf ein Graustufenbild 15
2.8 Merkmalsextraktion durch das LBP Verfahren 16
2.9 Konkatenation aller LBP Histogramme aus einem Bild 17
2.10 Anwendung des LPBH Verfahrens bei unterschiedlicher Belichtung 17
2.11 Aufbau eines Spracherkennungssystems 19

3.1 Darstellung der Interaktionsmöglichkeiten eines Benutzers 24
3.2 Komponenten mit denen die App interagiert 26
3.3 Beschreibung der Interaktion der einzelnen Module 28
3.4 Ablauf des Login nach Sprachbefehl 29
3.5 Ablauf des Logout nach Sprachbefehl 29
3.6 Speicherung von akzeptierten Bluetooth Daten 30
3.7 Datenmodell für Benutzer relevante Daten 31
3.8 Facade Pattern zur Kapselung des Datenbankzugriffs 33
3.9 Ablauf der Benutzererstellung 34
3.10 Ablauf des Identifikationsprozesses 36
3.11 Ablauf der Spracherkennung 37
3.12 Verbindungsaufbau mit BLE-Geräten 39
3.13 BLE Verfahren zum Datenaustausch 40
3.14 Factory Pattern zur Verwaltung der eingehenden Bluetooth Daten 41

Tabellenverzeichnis

3.1 Broadcast Topics 32
3.2 Unterstützte Sprachbefehle 38

1 Einleitung

1.1 Motivation

Viele Bereiche des alltäglichen Lebens werden immer weiter automatisiert. Dabei können viele Systeme mehr als es zunächst den Anschein hat. Sei es ein Spiegel, der einem das Wetter anzeigt, eine Uhr, mit welcher an der Kasse bezahlt werden kann oder ein Drucker, der automatisch Tintenpatronen nachbestellt. Durch eine zunehmende Vernetzung solcher Systeme, soll dem Anwender so viel Arbeit wie möglich abgenommen und der Komfort gesteigert werden. Computer sind heutzutage allgegenwärtig, sie befinden sich in Autos, Haushaltsgeräten und vielen weiteren Gegenständen. Durch die fortschreitende Entwicklung werden die Systeme immer kleiner, leistungsfähiger und sind häufig nicht mehr als Computer zu erkennen. Gerade im Kontext eines Smart Home ist das Ziel die Systeme so unauffällig wie möglich zu gestalten, damit sie sich nahtlos in die Umgebung einfügen.

Die Interaktion mit solchen Systemen hat sich im Laufe der Zeit stark gewandelt. Lange Zeit waren die Tastatur und die Maus die klassischen Eingabemethoden, um mit Computern zu kommunizieren. Durch die ständige Weiterentwicklung der Technik sind natürlichere Verfahren zur Interaktion mit technischen Systemen realisierbar. Dadurch wird ermöglicht, dass die Systeme sprichwörtlich in den Hintergrund treten können und nicht mehr als Computer wahrgenommen werden. Ein Beispiel dafür sind Sprachassistenten, die in vielen Bereichen Anwendung finden, beispielsweise auch in der Automobilindustrie. Damit der Fahrer sich während der Fahrt nicht von der Interaktion mit Radio oder Navigationsgerät ablenken lässt, bieten die meisten Hersteller bereits die Möglichkeit, per Sprache mit dem Infotainment-System des Fahrzeugs zu kommunizieren. Solche Systeme können als eine unaufdringliche Assistenz verstanden werden.

Durch die Vielzahl der technischen Systeme, die wir heutzutage in unserem Alltag verwenden, können große Datenmengen gesammelt werden. Daraus ist das sogenannte Self-

Tracking oder auch Quantified Self hervorgegangen. Dafür werden Daten über den eigenen Körper und die damit einhergehende Lebensweise protokolliert. Dabei unterstützen den Anwender viele Apps, welche die erfassten Daten auswerten und grafisch ansprechend aufbereiten. Durch die Menge der erfassten Daten wird ein digitaler Zwilling erstellt, der den Anwender abbildet. Viele Menschen sind aus medizinischen Gründen auf die Auswertung von Blutdruck, Blutsauerstoff oder ähnlichen Werten angewiesen. Andere versuchen sich durch das Self-Tracking selbst zu optimieren und somit ein gesteigertes Wohlbefinden herzustellen. Durch die Datensammelwut vieler Systeme sollte aber auch das Bewusstsein der Menschen für ihre Daten geschärft werden. Die Technik wird immer anspruchsvoller und komplexer, sodass viele Menschen die Hintergründe nicht mehr verstehen können. Gerade in Zeiten sich häufender Hackerangriffe und Meldungen von Millionen gestohlener Datensätze, sollten die Menschen vorsichtiger werden, welche Daten sie mit den Systemen teilen. Häufig ist eine Nutzung aber auch gar nicht mehr möglich, ohne die Eingabe sensibler persönlicher Daten.

1.2 Ziele

Ziel dieser Thesis ist die Entwicklung eines Systems, mit dem ein Benutzer berührungslos und auf natürliche weise über eine Sprachsteuerung interagieren kann. Das System soll den Anwender durch eine Gesichtserkennung identifizieren und ihm seine persönlichen Self-Tracking Daten präsentieren. Zunächst sollen als Datenquellen verschiedene Bluetooth-fähige Geräte verwendet werden. Ziel ist es, dass das System vom Anwender als ein unaufdringliches Assistenzsystem in einem Smart Home Kontext verstanden wird.

Als Grundlage nutzt die Anwendung einen Smart Mirror als Präsentations- und Interaktionsobjekt. Der Smart Mirror befindet sich im Badezimmer des Living Place[1] der HAW Hamburg.

Die Darstellung der Daten spielt in dieser Arbeit nur eine untergeordnete Rolle. Hier wäre durch eine systematische Auswertung, sowie anderen Visualisierungsformen ein deutlich höherer Informationsgehalt für den Nutzer möglich.

[1] Living Place, https://livingplace.haw-hamburg.de/ [12.09.2021]

1.3 Aufbau der Arbeit

In Kapitel 2 wird zunächst durch ein kleines Szenario (2.1) die Problemstellung veranschaulicht, anschließend werden die wissenschaftlichen Hintergründe zur Einordnung dieser Arbeit gegeben. Dafür wird untersucht, was ein Smart Environment (2.2) auszeichnet und wie Menschen damit interagieren können. Des Weiteren wird ein Überblick zu Quantified Self (2.3) und verschiedenen Identifikationsverfahren (2.4) gegeben. Anschließend werden technische Grundlagen der benutzten Interaktionsformen untersucht (2.5, 2.6), um die Hintergründe in der Umsetzung besser nachvollziehen zu können. Abgeleitet aus der Analyse werden die Anforderungen (2.7) an das System definiert.

Die Umsetzung wird in Kapitel 3 erläutert. Dafür werden zunächst die verwendeten Verfahren zur Interaktion erläutert (3.1). Nachfolgend werden die verwendeten Komponenten des Smart Mirrors dargestellt (3.2). Daraus werden verschiedene Module (3.3) abstrahiert, welche die identifizierten Aufgaben übernehmen. Nach der Erläuterung des Datenmodells (3.5), geht es an die konkrete Implementation der Module (3.6). Abschließend wird evaluiert (3.7), inwieweit das Design die Anforderungen erfüllt.

In Kapitel 4 wird die Arbeit zunächst noch einmal zusammengefasst (4.1) und ein Ausblick (4.2) gegeben, was auf Grundlage dieser Thesis noch möglich ist.

2 Analyse

2.1 Projektszenario

In Ahnlehnung an Mark Weiser, der Begriffe wie das Ubiquitous Computing geprägt hat, wird durch ein kleines Szenario von Sal[1], in die Problemstellung dieser Thesis eingeführt:

Sal wacht auf. Bevor sie den ersten Kaffee des Tages trinkt, geht sie wie jeden Morgen zuerst ins Badezimmer. Normalerweise liest sie Nachrichten nur in Papierform. Da sie heute aber nicht so viel Zeit hat, entschließt sie sich, die Nachrichten auf dem Smart Mirror über dem Waschbecken zu überfliegen. Dabei fällt ihr ein, dass Joe, ihr Arbeitskollege, der den gleichen Spiegel besitzt, ihr von einer App erzählt hat. Die App ermöglicht für die gängigsten Aufgaben eine berührungslose Interaktion und visualisiert persönliche Daten. Trotz Zeitdruck möchte Sal die App noch kurz ausprobieren und lädt sie auf den Smart Mirror. Damit die App Sal anhand ihres Gesichts erkennen kann, muss sie sich zunächst registrieren. Die integrierte Kamera erstellt dafür Beispielbilder. Als nächsten Schritt verbindet Sal ihre Bluetooth-fähige Waage mit dem Spiegel. Damit ist die App bereit und Sal ist nun in der Lage, sich per Sprachbefehl anzumelden. Die App scannt ihr Gesicht und verifiziert sie als Sal. Jetzt befindet sie sich in einem geschützten Bereich, in dem nur ihre persönlichen Daten angezeigt werden. Da sie sich aber gerade erst registriert hat, ist noch nicht viel zu sehen. Daraufhin stellt sie sich auf die Waage. Sobald der Wert feststeht, erscheint in der App eine kleine Benachrichtigung. Ihre Hände sind noch nass vom Händewaschen, deshalb bestätigt sie wieder per Sprachbefehl, sodass der Wert gespeichert wird. Die Anzeige aktualisiert sich und der neue Wert wird dargestellt. Per Sprachbefehl meldet Sal sich ab, schließt die App und holt sich ihren ersten Kaffee.

[1] Mark Weiser skizziert in seinem Artikel *The Computer for the 21st Century*[42] einen fiktiven Tag im Leben von Sal.

2.2 Smart Environments

Smart Environments ist ein Sammelbegriff für Smart Homes, Smart Cities oder auch Smart Factories. Als Smart Environments werden Umgebungen betrachtet, in die Smart Devices fest eingebunden sind. Dadurch kann die Umgebung Wissen über sich und ihre Bewohner erwerben und anwenden, um die Interaktionserfahrung der Bewohner zu verbessern.[28]

2.2.1 Ubiquitous Computing

In einem Smart Environment gibt es viele unterschiedliche Arten der Interaktion mit Computern. Mark Weiser hat bereits 1996 einen Wandel in der Nutzung von Computern vorhergesagt.[19] Er beschreibt dabei, dass sich die Interaktion mit Computern aus mehreren Phasen zusammensetzt.
Die erste Phase war die *Mainframe Era*, dabei hatten nur wenige Experten Zugriff auf einen Großrechner und mussten sich ihn teilen.
Die zweite Phase war die *PC Era*. Viele Menschen haben ihren eigenen Personal Computer, jeder kann zu jeder Zeit mit ihm interagieren und persönliche Dinge erledigen. Jeder Computer, zu dem Menschen eine besondere Beziehung haben oder der sie bei der Benutzung voll und ganz in Anspruch nimmt, ist ein Personal Computer. Bereits 1984 überschritt die Anzahl der Nutzer von Personal Computer die Anzahl der Nutzer von Großrechnern.
Die letzte Phase ist das *Ubiquitous Computing*(UC), was allgegenwärtiges Computing bedeutet. Dabei hat ein Mensch Zugriff auf viele Computer, die jeweils unterschiedliche Formen annehmen können und häufig nicht direkt als solche identifizierbar sind. Computer unterstützen den Menschen bei seinen Tätigkeiten und befreien ihn von Routineaufgaben. Im besten Falle verschwindet der Computer als wahrnehmbares Gerät und arbeitet als Assistenz im Hintergrund. Weiser hat vorausgesagt, dass das Ubiquitous Computing, das Personal Computing in der Zeit von 2005-2020 überholen wird.

In der Tat können wir heute feststellen, dass die Prognose zutreffend ist. Die elektronischen Geräte werden immer kleiner und lassen sich miteinander vernetzen. Viele Geräte sind aus unserem alltäglichen Leben nicht mehr wegzudenken. Ein einfaches Beispiel ist Amazons Sprachassistent Alexa. Zunächst nicht direkt als Computer identifizierbar, ist es damit möglich, auf natürliche Weise Informationen zu erhalten oder andere Geräte steuern.

2.2.2 Calm Technology

Das heutige Leben wird immer schneller und überflutet die Menschen mit Reizen und Informationen. Die Informationstechnologie trägt einen wesentlichen Teil dazu bei. Viele Geräte und Dienste benötigen eine zentrale Aufmerksamkeit des Benutzers. Calm Technology (ruhige Technologie), interagiert mit dem Benutzer peripher und nicht zentral, sie bewegt sich leicht vom Zentrum in die Peripherie und zurück.[19] Informationen werden dabei ruhig in die Aufmerksamkeit des Benutzers verlagert. Mark Weiser beschreibt Calm Technology als das, was informiert aber nicht unseren Fokus oder unsere Aufmerksamkeit erfordert.
Ein Beispiel dafür ist eine Videokonferenz, die es im Vergleich zu einer Telefonkonferenz ermöglicht, Nuancen der Körperhaltung und des Gesichtsausdrucks wahrzunehmen. Diese Informationen blieben sonst verborgen. Dadurch wird unsere Handlungsfähigkeit erhöht, ohne die Informationsflut zu vergrößern.
Ein intelligenter Spiegel fällt ebenfalls in diese Kategorie, da er Wetterinformationen oder anstehende Termine darstellt, während der Benutzer sich auf den Tag vorbereitet. Dabei wandern die Informationen nur kurz ins Zentrum der Aufmerksamkeit und entweichen danach wieder in die Peripherie. Zusammen mit Ubiquitous Computing minimiert Calm Technology die Wahrnehmbarkeit von Computern im alltäglichen Leben.

2.2.3 Interaktionsmöglichkeiten

Heutzutage gibt es verschiedene Möglichkeiten, auf ein Computer-System einzuwirken. Dadurch, dass Sensoren immer kleiner, billiger und leistungsfähiger werden, wird die Einbettung in Alltagsgegenstände begünstigt. Dies hat viele neue Anwendungs- und Interaktionsmöglichkeiten zur Folge.[23, 12]

Die grafische Benutzeroberfläche (GUI) ist eine der gängigsten Methoden, um mit Computern zu kommunizieren. Die Interaktion erfolgt dabei mit Eingabegeräten wie einer Maus, Tastatur oder per Touch. Die Bedienung eines Computer Systems mit Maus und Tastatur war jahrzehntelang die gängigste Form der Interaktion. Die Nutzung von GUIs ist in der Regel einfach zu erlernen und bietet den Vorteil von visuellen Rückmeldungen.

Eine Erweiterung der GUIs sind berührungslose grafische Benutzeroberflächen (TGUI). Die Idee dahinter ist, dass Anwender mit ihrer Umgebung durch ihren Körper kommu-

nizieren. Dabei können verschiedene Teile des Körpers einbezogen werden. Dies ist für Menschen eine der natürlichsten Formen der Interaktion, da wir von Geburt an lernen, mit unserer Umwelt über Gesten zu kommunizieren. Um eine Gestensteuerung zu realisieren, können beispielsweise die Positionen der Hände und deren Bewegungen ausgewertet werden, um Aktionen auszulösen.

Eine weitere natürliche Möglichkeit der Interaktion bieten sprachgesteuerte Benutzerschnittstellen (VUI). Sinnvoll sind solche Systeme bei Menschen mit Beeinträchtigungen der Extremitäten oder wenn Hände und Augen während der Interaktion durch andere Aufgaben nicht vollumfänglich zur Verfügung stehen. Wenn Computer auf Sprache reagieren und Aktionen ausführen, wird eine Persönlichkeit auf die Computer projiziert. Dadurch entsteht eine andere Bindung zwischen Mensch und Maschine.[10] In Kapitel 2.2.1 wurde als Beispiel bereits Amazons Sprachassistent Alexa aufgeführt.

Für viele Anwendungen sind grafische Benutzeroberflächen die beste Wahl. Jedoch entsprechen GUIs nicht immer den Anforderungen der Nutzer. In einigen Szenarien ist eine direkte physikalische Interaktion mit den Objekten sinnvoll. Deshalb haben Ishii und Ullmer[15] die Tangible User Interfaces (TUI) entwickelt. Dadurch wird eine Brücke zwischen dem Cyberspace und dem Physischen geschaffen.[12] Das Grundprinzip sind interaktive Oberflächen, die als Schnittstelle eine Kommunikation zwischen Nutzer und System ermöglichen. Auch ganz alltägliche Oberflächen wie eine Wand oder ein Tisch können in ein Tangible User Interface verwandelt werden.

Durch Kombination der genannten Schnittstellen, die nur einen Ausschnitt darstellen, ist eine natürliche, einfache und beiläufige Bedienung möglich. Für einige der genannten Interaktionsmöglichkeiten bietet der in diesem Projekt verwendete Spiegel passende Eingabemöglichkeiten an, wie unter 2.2.5 beschrieben. Mithilfe der Kamera, kann die Position der Hände, die Gesten des Nutzers oder das Gesicht erfasst und verfolgt werden. Durch Audio und Mikrofon kann über eine sprachliche Kommunikation eine Verbindung mit dem System hergestellt werden. Die Netzwerkfähigkeit kann eine Kommunikation mit anderen Geräten oder dem Internet ermöglichen.

2.2.4 Smart Devices

Im Kontext dieser Thesis geht es primär um die Interaktion mit einem Smart Device. Dafür muss zunächst spezifiziert werden, was ein Smart Device eigentlich ist.
Smart Devices sind alltägliche Gegenstände, welche die Möglichkeit bieten, mit ihrer

Umgebung zu kommunizieren und zu interagieren. Sie benötigen dafür genügend Rechenleistung, die Möglichkeit sich mit anderen Geräten zu vernetzen und mit Sensoren und Aktoren ihre Umwelt wahrzunehmen, beziehungsweise darauf einzuwirken.[16] Sie können daher auch als Cyber-Physisches System verstanden werden. Smart Devices können als mobile und personalisierte MTOS-Geräte (Multi Task Operating System) charakterisiert werden, die als Multi-Applikations-Plattformen, Web-Portale oder Multimedia-Player verwendet werden können.[28] Smart Devices sind in den letzten Jahren immer präsenter geworden. Sie werden mittlerweile in vielen Bereichen unseres alltäglichen Lebens verwendet. Dazu gehören tragbare Accessoires wie Fitbits, Smartphones oder auch Geräte die ins Smart Home integriert sind, wie ein intelligenter Spiegel.

2.2.5 Living Place

Das Projekt wird im Badezimmer des Living Place der HAW Hamburg durchgeführt. Das Living Place wurde im Jahre 2009 gegründet, das dazugehörige Labor mit integrierter Wohnung hat eine Grundfläche von 140m^2. Es werden vorwiegend Projekte mit Bezug zu Themen wie Smart Home/Smart Living durchgeführt.

Das Testobjekt dieses Projektes ist ein Smart Mirror. Die Abmessungen des Spiegels sind 80x60x2,6cm, darin eingebettet befindet sich ein 21,5“ Tablet. Das Tablet arbeitet mit einem 1,8GHz Prozessor und 2GB RAM. Aktuell ist Android Version 8.1 installiert. Es unterstützt gängige Kommunikationstechnologien wie Wireless-LAN und Bluetooth. Zusätzlich verfügt es über eine Front-Kamera, Lautsprecher und Mikrofon. Um einem Missbrauch vorzubeugen, befindet sich im Gehäuse ein mechanischer Schalter, um die Kamera auszuschalten. Die Kamera ist außerdem losgelöst vom hinter dem Spiegel liegenden Tablet. Sie befindet sich auf Kopfhöhe, in der Mitte des Spiegels. Im Lieferumfang des Spiegels sind eine Bluetooth Waage und ein Bluetooth Hautfeuchtigkeitstester enthalten.

Wie auf den Bildern in den Abbildungen 2.1 und 2.2 zu sehen, sind die Wände in einem neutralen einheitlichen Farbton gestrichen. Außerdem wird durch mehrere Lichtquellen eine ausreichende Beleuchtung sichergestellt.

Abbildung 2.1: Badansicht vom Fenster

Abbildung 2.2: Ansicht des Smart Mirror

2.2.6 Ähnliche Projekte

Beispielhaft werden zwei Projekte vorgestellt, die ebenfalls einen Spiegel als zentrales Element beinhalten und dem Nutzer nach einer erfolgreichen Identifikation durch eine Gesichtserkennung personalisierte Daten in unterschiedlicher Form visualisieren.

Quantified Self Companion - Richter zeigt in seinem Projekt auf, wie durch Self-Tracking, Daten und Selbstwahrnehmung zusammengeführt werden können. Mithilfe einer Gesichtserkennung wird ermittelt, für welche Person die Daten angezeigt werden sollen. Zur Visualisierung der getrackten Daten nutzt er ein aus mehreren Spiegeln zusammengesetztes System, das anhand der ausgewerteten Daten eines Fitnessarmbandes die verschiedenen Spiegelelemente anordnet. Damit soll die Selbstwahrnehmung trainiert und eine emotionale Verbindung zwischen der eigenen Person und den Daten hergestellt werden.[34]

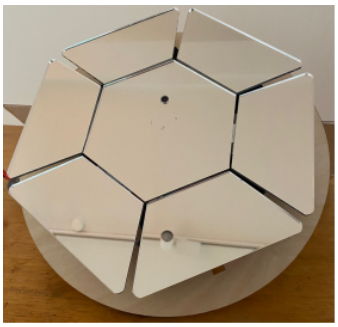

Abbildung 2.3: Richters Spiegelobjekt zur Visualisierung der Self-Tracking Daten.[34]

Magic Smart Mirror - Bianco hat ein durch Sprache steuerbares Spiegelsystem entwickelt, dass neben verschiedenen allgemeinen Informationen zum Wetter auch die emotionalen Zustände des Nutzers analysiert und darstellt. Das zentrale Element bildet ein Raspberry Pi an dem Sensoren und Eingabeelemente angeschloßen sind. Das User Interface wird auf einem Display hinter einem Zwei-Wege-Spiegel dargestellt. Zur Gesichts- und Emotionserkennung wird ein Convolutional Neural Network verwendet.[3]

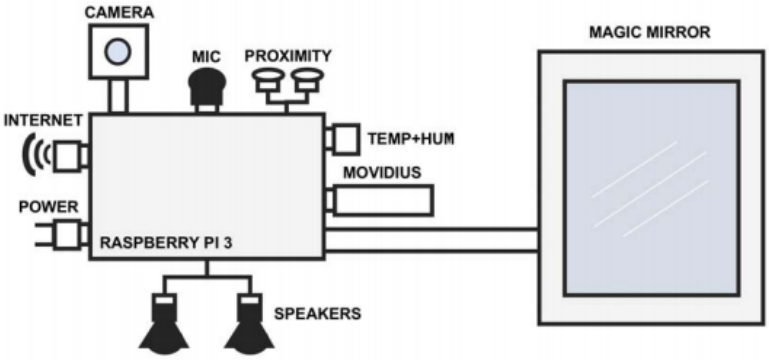

Abbildung 2.4: Architektur des Bianco Spiegelsystems.[3]

2.3 Quantified Self

Der Begriff Quantified Self (Quantifiziertes Selbst) wurde 2007 durch Gary Wolf und Kevin Kelly eingeführt.[43] Bei der dahinter stehenden Bewegung, geht es um das erfassen quantitativer und qualitativer Daten über sich selbst. Es wird versucht eine Selbsterkenntnis durch Self-Tracking zu erreichen. Es gibt unzählige Parameter die erfasst werden können, zum Beispiel die Herzfrequenz, die geschlafenen Stunden oder das Gewicht. Doch nicht alle wichtigen Dinge im Leben lassen sich messen und nicht alles, was sich messen lässt, ist wichtig. Bei QS geht es darum, den Sinn in den persönlichen Daten zu finden.[44]

Das Aufkommen neuer Praktiken zur Protokollierung des alltäglichen Lebens, wie die Standortverfolgung oder die Erfassung biometrischer Daten, ebnete den Weg für die Quantified Self Bewegung. Die beiden Gründer sahen in der Verbindung der Praktiken eine Erweiterung der gewöhnlichen Existenz, um eine rechnerische Dimension.[43]
Zunächst gab es einige Begrifflichkeiten, die parallel verwendet wurden, darunter Self-Tracking, Personal Informatics, Lifelogging, Body Monitoring oder Life Hacking. Mit der

Zeit setzte sich aber Self-Tracking und Quantified Self durch.[18] Viele Geräte, darunter Fitbits und Smartphones, erfassen digital und automatisiert viele verschiedene Daten. Aus den daraus gewonnen Erkenntnissen ist es möglich seine Lebensweise anzupassen und somit wiederum neue Erkenntnisse zu erhalten. So beginnt ein Kreislauf, der die Optimierung des eigenen Selbst zur Folge hat.

Eine Studie aus dem Jahr 2019 hat gezeigt, dass bereits die Hälfte der Deutschen eine Tracking-App nutzt oder zumindest daran interessiert ist. Hauptsächlich werden dabei Apps aus dem Bereich Sport beziehungsweise Fitness verwendet. 41% der Personen, die eine Tracking-App nutzen, erhoffen sich dadurch ein gesteigertes Wohlbefinden. Ein Viertel der Bürger nutzt zudem aktiv eine Smartwatch oder einen Fitnesstracker zur Aufzeichnung der persönlichen Daten.[37]

2.4 Identifikationsverfahren

Die Erkennung von Personen und im Speziellen deren Gesichter hat eine große Bedeutung in unserem alltäglichen Leben. Es ist nicht nur wichtig, Gesichter genau zu identifizieren und einzuordnen, sondern auch emotionale Zustände, physische Veränderungen und Auffälligkeiten schnell zu erkennen und zu interpretieren. Ohne die Fähigkeit Gesichter zu erkennen, wären wir nicht in der Lage, ein kontinuierliches soziales Leben herzustellen. Der Prozess der Gesichtserkennung im menschlichen Gehirn ist äußerst komplex und immer noch nicht vollständig ergründet. Klar ist aber das dieser Prozess extrem leistungsfähig ist, da wir Tausende Gesichter in Bruchteilen von Sekunden erkennen und klassifizieren können.[5]
Ein Gesicht besteht aus vielen Merkmalen, wobei jedes Merkmal unterschiedlichste Werte annehmen kann. Wenn wir die Anzahl der Merkmale auf 10 begrenzen (zum Beispiel Haare, Stirn, Augen, Augenbrauen, Nase, Mund, Kinn, Haut, Ohren und Form), dann ist es möglich, daraus schon $10^{10} = 10 Milliarden$ unterschiedliche Gesichter zu erstellen.[32] Der Wert ist größer als die aktuelle Gesamtbevölkerung und zeigt, wie vielfältig Gesichter sind. Studien zeigen, dass sich schon Neugeborene stärker auf Gesichter als auf andere Objekte fixieren.[33] Forscher gehen davon aus, dass diese Fähigkeit angeboren ist, um Tiere und Menschen von ihrer Umwelt zu unterscheiden.

Die Auswertung von optischen Reizen ist für uns Menschen die gängigste Methode, um Personen zu identifizieren, jedoch ist die Gesichtserkennung in der heutigen Zeit nicht die einzige Möglichkeit, eine Person zu erkennen. Daneben gibt es weitere biometrische

Identifikationsverfahren wie die Gangart oder Fingerabdrücke. Beispielsweise hat eine Firma in China eine künstliche Intelligenz entwickelt, die anhand des Ganges und der Körperform eine Person identifizieren kann. Die Fehlerquote liegt der Studie zufolge nur bei 0,7 Prozent.[9] Biometrische Verfahren werden schon seit mehreren Hundert Jahren benutzt. Bereits im 14. Jahrhundert wurden Chinesische Kaufleute durch Fingerabdrücke identifiziert.[6] Diese Verfahren sind so sicher, dass nach *EU-Verordnung 2019/1157* ab August 2021 europaweit, in allen neu ausgestellten Personalausweisen, zwei Fingerabdrücke gespeichert werden müssen.[4]

2.5 Computer Vision

Computer Vision ist ein stark wachsendes Gebiet der Informatik, dass sich mit der Analyse, Verarbeitung und semantischer Interpretation von Bildern beschäftigt. Es gibt viele Anwendungsbereiche in unterschiedlichen Branchen, wie zum Beispiel Automobil, Fotografie und Medizin.[31] Im Automobilbereich sollen Kraftfahrzeuge zukünftig autonom fahren können, das wäre ohne Computer Vision gar nicht denkbar. Bereits heute verfügbare Fahrassistenzsysteme nutzen intensiv Computer Vision Algorithmen, um dem Insassen, sowohl mehr Sicherheit, als auch Komfort zu bieten.

2.5.1 Bibliotheken

Am Markt gibt es eine Reihe von Computer Vision Bibliotheken, welche die oben genannten Aufgaben erledigen können. Nachfolgend sollen Bibliotheken vorgestellt werden, die vorwiegend im Kontext der Gesichtserkennung Lösungen anbieten. Jede dieser Bibliotheken hat dabei eigene Spezialisierungen und nutzt unterschiedliche Werkzeuge, um die Ziele zu erreichen.

OpenCV
OpenCV[2] begann im Jahr 1998 als ein Forschungsprojekt von Intel und ist seit dem Jahr 2000 unter der BSD Open-Source Lizenz verfügbar. Die Bibliothek verfügt über mehr als 2500 optimierte Algorithmen, die für Objekt- und Gesichtserkennung, Bildbearbeitung, Augmented Reality und viele weitere Anwendungsbereiche genutzt werden können. Dabei werden sowohl klassische Algorithmen als auch machine learning unterstützt. Es ist eine der am meisten eingesetzten Bibliotheken für Computer Vision Anwendungen.

[2] https://opencv.org/ [12.09.2021]

Ursprünglich wurde OpenCV in der Programmiersprache C++ entwickelt, mittlerweile gibt es aber auch Schnittstellen zu anderen Programmiersprachen wie Python oder Java. Ein weiterer Vorteil von OpenCV ist die Plattformunabhängigkeit, die Bibliothek kann auf den gängigsten Betriebssystemen eingesetzt werden, unter anderem Android. Durch die gute Dokumentation ist ein schneller Einstieg möglich.

OpenFace
OpenFace[3] ist eine Open Source Bibliothek, die Convolutional Neural Networks (CNN) zur Gesichtserkennung nutzt. Mehrere Studien zeigen, dass CNN bei der Objekt- und Gesichtserkennung deutlich präzisere Resultate liefern.[8, 24] OpenFace bietet aber keine Möglichkeit zur Bildbearbeitung wie beispielsweise OpenCV, sondern konzentriert sich nur auf die Gesichtserkennung.

Google Vision API
Die Vision API[4] von Google Cloud bietet über die REST API und die RPC API leistungsstarke, vorab trainierte Modelle für machine learning. Damit lassen sich Objekte und Gesichter erkennen, sowie gedruckter und handgeschriebener Text lesen. Als Nachteil kann jedoch ausgelegt werden, dass eine permanente Internetverbindung erforderlich ist. Zudem ist nur ein kleines Kontingent an Erkennungen kostenlos nutzbar.

2.5.2 Einflüsse auf die Gesichtserkennung

Die Gesichtserkennung ist eine der anspruchsvollsten Aufgaben der Bildverarbeitung. Durch die gestiegene Leistungsfähigkeit der Computer ist die Erkennung von Gesichtern heutzutage nahezu in Echtzeit durchführbar. Es gibt jedoch verschiedene Aspekte, die eine erfolgreiche Erkennung beeinflussen können.[36] Gesichtserkennungsalgorithmen tolerieren die Einflüsse bis zu einem gewissen Grad. Für eine gute Erkennung gilt es die Einflüsse jedoch so gering wie möglich zu halten.

Ausrichtung des Kopfes - Für ein optimales Ergebnis sollte das Gesicht frontal zur Kamera ausgerichtet sein. Sobald der Kopf seitlich gedreht oder geneigt wird, kann es zu Problemen bei der Erkennung kommen.

Verdeckung des Gesichts - Durch Kleidungsstücke oder andere Gegenstände, die einen Teil des Gesichts bedecken, kann die Erkennungsrate deutlich reduziert werden.

[3] https://cmusatyalab.github.io/openface [13.09.2021]
[4] https://cloud.google.com/vision [12.09.2021]

Belichtung - Für gute Ergebnisse sollte eine gleichmäßige Belichtung sichergestellt sein. Eine zu dunkle, zu helle oder einseitige Belichtung kann die Erkennung stören.

Gesichtsausdrücke - Lachen oder andere Ausdrücke, die die Gesichtsmerkmale verzerren, können das Ergebnis verfälschen.

2.5.3 Phasen der Gesichtsidentifikation

Die Identifikation von Gesichtern setzt sich aus mehreren Phasen zusammen, wie unter Abbildung 2.5 beschrieben. In einem eingehenden Bild werden zunächst alle Gesichter anhand von bestimmten Merkmalen, wie zum Beispiel den Augen oder der Nase, lokalisiert. Im nächsten Schritt müssen die Gesichter vorverarbeitet werden. Dabei soll der Fokus auf die Gesichter gerichtet werden, deshalb werden sie ausgeschnitten und unwichtige Informationen ausgeblendet. Zusätzlich müssen alle Bilder einheitlich sein. Dies bedeutet, dass sie die gleiche Größe und Ausrichtung benötigen. Je nach Identifikationsalgorithmus kann es nötig sein, das Gesichtsbild in ein Graustufenbild umzuwandeln. Im letzten Schritt findet die eigentliche Identifikation statt. Das vorverarbeitete Gesicht wird durch einen Algorithmus mit Testgesichtern verglichen. Jedes Testgesicht ist dabei mit einer eindeutigen ID gekoppelt, die eine bestimmte Person referenziert. Tritt bei einem Vergleich der Bilder eine sehr hohe Ähnlichkeit auf, wird die ID extrahiert und zurückgegeben. Ab welchen Punkt zwei Gesichter als ähnlich eingestuft werden, ist in den meisten Algorithmen definierbar.

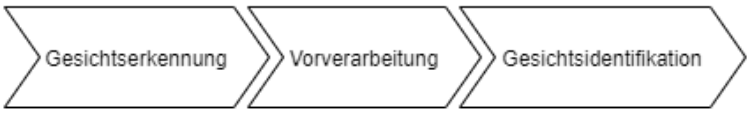

Abbildung 2.5: Phasen der Gesichtsidentifikation

2.5.4 Extraktion der Gesichtsmerkmale

Die Erkennung von Gesichtern durch Cascade Classifier (Kaskaden-Klassifikator) ist eine effektive Methode, die von Paul und Viola Jones im Jahr 2001 beschrieben wurde.[41] Der Klassifikator beruht auf Haar-features (Abbildung 2.6). Dabei wird die Summe der Grauwerte unter dem weißen Anteil des Haar-Features von der Summe der Grauwerte des

schwarzen Anteils des Features subtrahiert. Die Features werden dabei wiederholt mit unterschiedlicher Skalierung auf das Bild gelegt. Im Jahre 1998 wurde erstmals beschrieben, wie mit dieser Methode Objekte in einem Bild identifiziert werden können.[22]

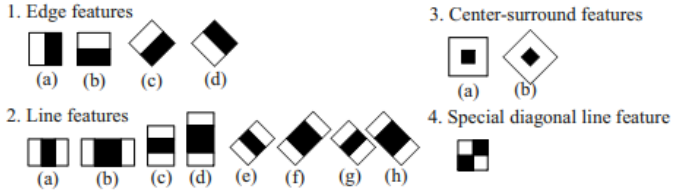

Abbildung 2.6: Mögliche Haar Cascade Features[17]

Mit machine learning Verfahren wird der Klassifikator trainiert, dazu werden viele positive Bilder (Bilder mit Gesichtern) und negative Bilder (Bilder ohne Gesicht) benötigt, auf welchen die Haar-Features angewandt werden.[13] Da dies ein sehr rechenintensiver Vorgang ist, wird das Bild in ein Integralbild umgewandelt, um die Pixelsummen schneller zu berechnen. Das Prinzip wurde erstmals durch Franklin Crow 1984 beschrieben.[7] Viele Regionen eines Bildes sind jedoch irrelevant. Für die Erkennung eines Gesichts sind die Augen- und Nasenregion häufig interessant, da dort große charakteristische Helligkeitsunterschiede vorhanden sind (Abbildung 2.7). Um nur relevante Merkmale herauszufiltern, werden Verfahren wie Adaboost verwendet.[17] Die so bestimmten Merkmale werden zu einer Kaskade zusammengeschaltet. Bibliotheken wie OpenCV bieten die Klassifikatoren bereits vorgefertigt an. Dabei gibt es viele unterschiedliche Klassifikatoren, die nicht nur für die Gesichtserkennung genutzt werden können.[13]

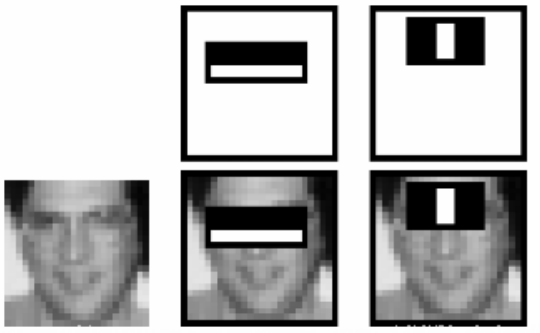

Abbildung 2.7: Anwendung der Haar Features auf ein Graustufenbild zur Extraktion der Gesichtsmerkmale[36]

In einem zu prüfenden Eingabebild wird ein rechteckiges Suchfenster schrittweise über das gesamte Bild geschoben. Der Ausschnitt des Fensters wird dem trainierten Cascade Classifier übergeben, der prüft, ob sich in dem Fenster das gesuchte Objekt befindet. Nach

jedem Durchlauf wird das Suchfenster vergrößert. Am Ende des Prozesses gibt der Klassifikator die ROIs (Region of Interest) zurück, Rechtecke, die anzeigen, dass Suchfenster durch alle Stufen des Klassifikators gelaufen sind und mit hoher Wahrscheinlichkeit das gesuchte Objekt beinhalten. Die extrahierten Rechtecke, die im besten Falle ein Gesicht beinhalten, können dann einem Identifikationsalgorithmus übergeben werden.

2.5.5 Local Binary Pattern Histograms

OpenCV bietet insgesamt drei Algorithmen für die Gesichtserkennung an: Eigenfaces[40], Fisherfaces[2] und Local Binary Pattern Histograms (LBPH). Nachfolgend wird LBPH nähergehend erläutert.

Local Binary Pattern (LBP) sind beliebt in der Computer Vision, da es eine schnelle und einfache Möglichkeit ist, Texturen in einem Bild zu erkennen. In diesem Projekt wird das Verfahren zur Identifikation von Gesichtern benutzt. Das Konzept wurde erstmals im Jahre 1994 beschrieben.[20]
Die Grundidee hinter der Entwicklung war, durch lokale räumliche Muster und Grauwertkontrast, Oberflächentexturen zu beschreiben. Das Verfahren ist in Abbildung 2.8 dargestellt. Zur Anwendung von LBP muss ein Bild in Graustufenformat vorliegen. Jedes Pixel kann somit einen Wert zwischen 0 und 255 annehmen, wobei 0 für Schwarz und 255 für Weiß steht. Der LBP-Operator weist jedem Pixel einen Wert zu, indem er den Graustufenwert, also die Helligkeit, mit den Nachbarpixeln vergleicht. Typischerweise werden dafür die angrenzenden Pixel verwendet. Es werden somit 3x3 Pixel in die Berechnung einbezogen. Überschreitet ein Nachbarpixel den Helligkeitswert, wird ihm eine 1 zugewiesen. Unterschreitet das Pixel den Wert, wird ihm eine 0 zugewiesen. Das Ergebnis ist wiederum eine Binärsumme, die 256(2^8) verschiedene Werte annehmen kann. Daraus wird die Dezimalzahl ermittelt und in einem Histogramm eingetragen. Der Vorgang wird für jedes Pixel wiederholt.[27]

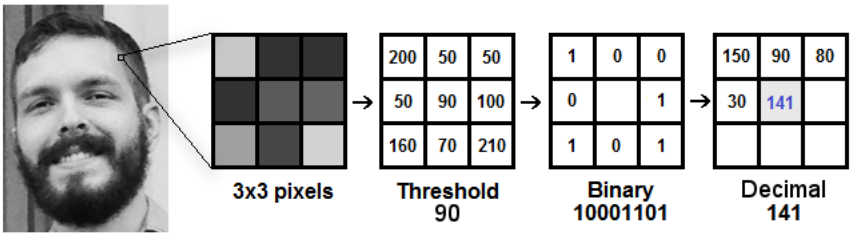

Abbildung 2.8: Merkmalsextraktion durch das LBP Verfahren[30]

Ein Problem der Local Binary Pattern ist, dass die räumlichen Information durch die Nutzung des Histogramms verloren gehen. Deswegen wird das Bild in kleine Segmente aufgeteilt, wie in Abbildung 2.9 zu sehen. Für jedes Segment wird ein Histogramm, wie oben beschrieben, berechnet. Alle Histogramme werden dann zu einem einzelnen Histogramm zusammengefügt.
Um eine Person in einem Bild zu identifizieren, werden, wie unter Kapitel 2.5.3 beschrieben, Testbilder benötigt. Für alle Testbilder müssen ebenfalls Histogramme erstellt werden. Um die Differenz der Histogramme zu ermitteln, kann der euklidische Abstand verwendet werden. Je geringer die Differenz, desto ähnlicher sind sich die Gesichter.

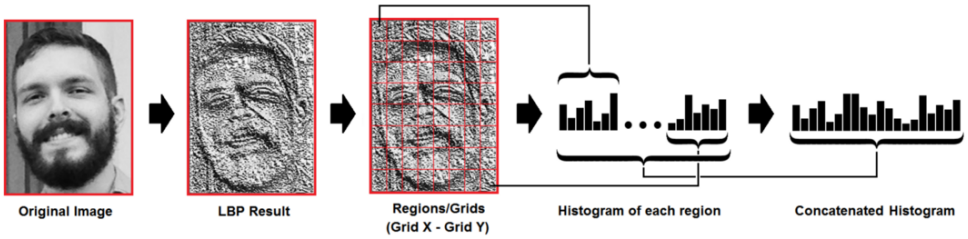

Abbildung 2.9: Konkatenation aller LBP Histogramme aus einem Bild[30]

Wie in Abbildung 2.10 zu sehen, haben unterschiedliche Belichtungen kaum einen Einfluss auf das Resultat. Jedoch nur, sofern sich die Belichtung gleichmäßig ändert.

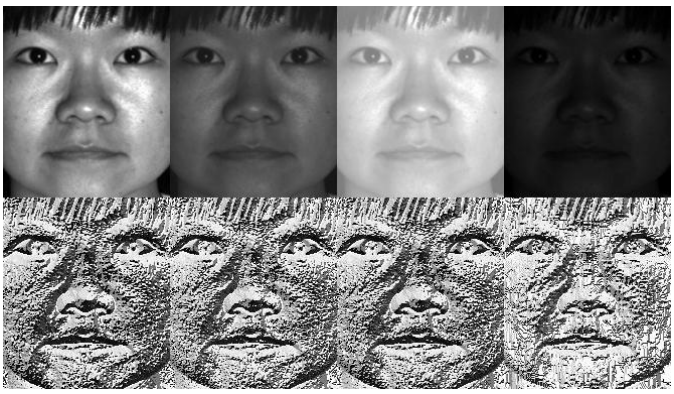

Abbildung 2.10: Anwendung des LPBH Verfahrens bei unterschiedlicher Belichtung[21]

2.6 Spracherkennung

Sprache ist die natürlichste und einfachste Möglichkeit der Kommunikation. Weltweit gibt es heutzutage etwa 6500 Sprachen, allein dieser Wert zeigt die Wichtigkeit dieser

Kommunikationsform für den Menschen. Aufgrund dessen, wird seit einigen Jahrzehnten untersucht, wie Menschen mit technischen Systemen sprachlich kommunizieren können. Viele Aufgaben, die ein Mensch händisch ausführen muss, können durch eine automatische Spracherkennung (ASR) vereinfacht werden. Jedoch ist die Erkennung und Interpretation der Sprache eine anspruchsvolle Aufgabe für Computer. Eine Sprache hat viele Facetten und die Art und Weise, wie sie gesprochen wird, variiert von Mensch zu Mensch.[39] Deshalb versucht die Informatik, zusammen mit der Computerlinguistik, Verfahren zu entwickeln, die eine zuverlässige Erkennung der gesprochenen Wörter ermöglichen. Der Begriff Spracherkennung wird für verschiedene Probleme verwendet. Zum einen geht es um die Identifikation des Sprechers als biometrisches Merkmal, zum anderen um die Übersetzung gesprochener Sprache in Text (Speech-to-Text).[26] In dieser Arbeit steht das zweitgenannte Problem im Vordergrund.

2.6.1 Einflüsse auf die Spracherkennung

Wie die Gesichtserkennung, stellt auch die Spracherkennung eine große Herausforderung für den Computer dar. Es gibt verschiedene Faktoren, die eine korrekte Erkennung beeinflussen können.[26] Die nachfolgenden Faktoren bilden nur einen Teil der möglichen Einflüsse ab.

Unklare Aussprache - Eine unklare oder zu schnelle Aussprache beeinflusst die Erkennung. Wörter, die isoliert gesprochen werden, also von Pausen umgeben sind, können einfacher erkannt werden als kontinuierliche Sprache. Wenn ein Sprecher mit einem Gerät spricht, kann es nötig sein, die Stimme anzupassen, um eine geringere Fehlerrate zu erreichen.

Unterstützte Wörter - Ein Sprachmodell, das eine große Anzahl von Wörtern unterstützt, benötigt ein Computersystem mit erhöhter Rechenleistung, da die Komplexität mit der Anzahl der Wörter steigt. Je geringer das Vokabular des Spracherkennungssystems, desto einfacher ist die Erkennung.

Nebengeräusche - Ein ruhiger Ort ohne Nebengeräusche begünstigt eine erfolgreiche Erkennung. Hinzu kommt die Qualität der Audiohardware, die den Einfluss von äußeren Störungen reduzieren kann.

Akzent - Sofern ein Sprachmodell nicht mit einem bestimmten Akzent oder Dialekt trainiert wurde, kann es zu Problemen kommen. Die Betonung einzelner Wörter kann von Region zu Region unterschiedlich sein.

2.6.2 Funktionsweise

Die Spracherkennung setzt sich aus mehreren Phasen zusammen. In den letzten Jahren haben sich Verfahren herauskristallisiert, auf deren Grundlage die meisten heutigen Spracherkennungssysteme basieren. Nachfolgend werden grob die grundlegenden Komponenten des in Abbildung 2.11 beschriebenen Systems erläutert.

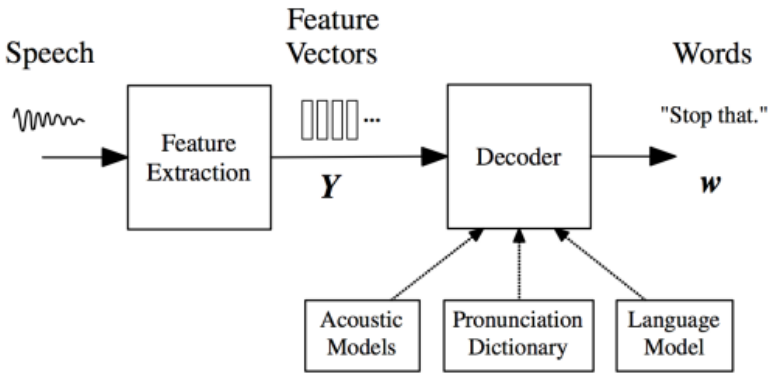

Abbildung 2.11: Aufbau eines Spracherkennungssystems[26]

Die ausgehenden Schallwellen des Sprechenden werden zunächst digitalisiert. Aus diesem Signal gilt es die Laute beziehungsweise die Phoneme[5] zu bestimmen. Dafür werden aus kurzen Signalabschnitten (typischerweise 10 Millisekunden), zwischen 20 und 50 Kennzahlen, dazu gehören die Spektralinformationen und die Signalenergie, extrahiert und in einem Merkmalsvektor zusammengefasst. Jeder Merkmalsvektor bildet charakteristische Eigenschaften des Signalabschnittes ab. Ähnliche Laute erzeugen dabei gleichartige Merkmalsvektoren, wohingegen unterschiedliche Laute zu deutlich verschiedenen Merkmalsvektoren führen.[11]

Aus den erstellten Merkmalsvektoren müssen nun das gesprochene Wort beziehungsweise die Wortfolge bestimmt werden. Dafür werden verschiedene Modelle verwendet, die zusammen mit dem Decoder die eigentliche Erkennung vornehmen. Das akustische

[5] http://www.fb10.uni-bremen.de/homepages/hackmack/phonmorph/pdf/Phonem.pdf [23.09.2021]

Modell soll die wahrscheinlichsten Phone auf Basis der zuvor erstellten Merkmalsvektoren bestimmen. Jedes Wort ist aus einer Sequenz von Phonemen zusammengesetzt, die wiederum durch unterschiedliche Phone abgebildet werden. Phone sind die kleinsten akustischen Einheiten einer Sprache. Hidden-Markov-Modelle (HMM) bilden typischerweise die Grundlage des akustischen Modells. HMM sind eine stochastische Methode, die die Wahrscheinlichkeit bestimmen, aus welcher Folge von Phonemen ein Wort zusammengesetzt ist. Für diese Aufgabe können auch tiefe neuronale Netze verwendet werden, häufig in Kombination mit HMM.[26, 35] IBM konnte im Jahre 2017 mit Deep Learning Verfahren eine Fehlerrate bei der Worterkennung von 5,5% erreichen. Damit konnte sich der menschlichen Fehlerquote von 5,1% genähert werden.[25]
Das phonetische Wörterbuch ist eine Teilkomponente des akustischen Modells. Es gibt an, welche Wörter durch das Spracherkennungssystem erkannt werden können und aus welchen Phonemen sie bestehen.[26, 35]
Das Sprachmodell repräsentiert die Grammatik einer Sprache. Das Modell ermittelt die Wahrscheinlichkeit, ob bestimmte Wörter, die durch das akustische Modell erkannt wurden, nacheinander auftreten können. Damit können Fehler durch ungenaue Aussprache ausgeglichen und nicht unterscheidbare Wörter (Meer != mehr) korrekt zugeordnet werden. Dafür können N-Gramm-Modelle oder kontextfreie Grammatiken verwendet werden.[26, 11]
Im letzten Schritt sucht der Decoder, auf Grundlage der Informationen der verschiedenen Modelle, nach der am wahrscheinlichsten gesprochenen Wortfolge. Dabei handelt es sich um ein komplexes Suchproblem, da mit der Länge des Satzes, die möglichen Wortfolgen exponentiell zunehmen. Um dieses Problem zu lösen, finden häufig der Viterbi-Algorithmus, die Strahlsuche (beam search) oder der A*-Algorithmus Verwendung, häufig auch kombiniert.[35]

2.6.3 Bibliotheken

Für die Spracherkennung können verschiedene frei verfügbare Bibliotheken verwendet werden. Einige Hochschulen bieten für Forschungszwecke Spracherkennungssysteme an, auf deren Grundlage weitere frei verfügbare Bibliotheken erstellt wurden.

Kaldi - Kaldi[6] wird seit 2009 von der John Hopkins University entwickelt. Als Grundlage für das Toolkit wird die Programmiersprache C++ verwendet. Ziel der Entwicklung war es, Kaldi einfach erweiterbar für neue Sprachen und Algorithmen zu gestalten.[29]

[6] http://kaldi-asr.org/ [25.09.2021]

CMUSphinx - CMUSphinx[7] (auch als Sphinx bekannt) wird von der Carnegie Mellon University entwickelt. Es bietet verschiedene Pakete für unterschiedliche Anwendungszwecke, unter anderem ein leichtgewichtiges Spracherkennungssystem für mobile Geräte. Entwickelt wird die Bibliothek in der Programmiersprache Java.[14]

2.7 Anforderungen an die Anwendung

Durch die Geschichte von Sal in 2.1 bereits beschrieben, werden nachfolgend die konkreten Anforderungen an die Anwendung aufgeführt. Es wird von einem 1-Personenhaushalt ausgegangen, dies hat zur Folge, dass sämtliche Interaktion mit dem Spiegel, zunächst nur für eine Person zurzeit unterstützt werden muss. Dementsprechend, können verschiedene Punkte identifiziert werden, welche die Anwendung umsetzen muss. Des weiteren sollen, wenn möglich, die vorhandenen Komponenten des Smart Mirror genutzt werden.

Interaktion mit dem Spiegel

1. Ein Nutzer kann mit dem System über eine Sprachsteuerung auf Natürliche und Berührungslose weise interagieren.
2. Das System bietet durch eine Gesichtserkennung die Möglichkeit einer berührungslosen Identifikation der Person, die vor dem Spiegel steht.

Beschaffung von Daten

1. Das System muss sich mit Bluetooth Low Energy (BLE) Geräten verbinden können.
2. Daten der BLE-Geräte können vom System ausgelesen werden.
3. Sobald sich ein verbundenes BLE-Gerät in der Nähe des Spiegels befindet, soll es möglich sein, Daten auszutauschen.

[7] https://cmusphinx.github.io/ [25.09.2021]

Verwaltung der Daten

Alle personenbezogenen Daten, dazu gehören sämtliche über Bluetooth empfangene Daten, müssen persistent in einer Datenbank vorgehalten werden. Es muss die Möglichkeit geben, Benutzer zu erstellen, zu bearbeiten und zu löschen. Wenn ein Benutzer gelöscht wird, sollen auch alle personenbezogenen Daten vom System entfernt werden.

Visualisierung der Daten

Nach erfolgreicher Identifikation eines Benutzers sollen die gespeicherten Bluetooth Daten in einfacher weise visualisiert werden. Die Anzeige der Daten soll automatisch aktualisiert werden, sobald ein neuer Wert in der Datenbank abgespeichert wurde.

3 Umsetzung

Im nachfolgenden Kapitel wird die Umsetzung der in 2.7 erläuterten Anforderungen beschrieben. Die Anwendung wird mittels einer App implementiert, da der Smart Mirror Android als Betriebssystem verwendet(siehe 2.2.5). Sämtliche Logik der Anwendung wird in der App umgesetzt. Die einzige Kommunikation mit externen Geräten wird über Bluetooth realisiert, jedoch nur um Daten abzufragen. Bis einschließlich Kapitel 3.5 werden Konzeptentscheidungen erläutert. Ab 3.6 wird die konkrete Implementierung des Konzepts beschrieben.

3.1 Interaktion mit dem System

Es bieten sich verschiedene Möglichkeiten an, berührungslos mit dem System zu interagieren, wie in 2.2.3 beschrieben. Ziel ist es, die bereits vorhandene Hardware des Smart Mirror zu nutzen (2.2.5).

Eine einfache Möglichkeit, die Person vor dem Spiegel zu identifizieren, bietet eine Gesichtserkennung. Dafür kann die vorhandene Kamera verwendet werden. Durch die zentrale, Kopf-orientierte Position der Kamera ist es mit den vorhandenen Komponenten einfach möglich, das Gesicht eines Benutzers zu erkennen, um es für die Identifikation zu nutzen. Andere Identifikationsverfahren, in 2.4 beschrieben, sind unter Umständen komplizierter in der Implementierung, benötigen externe Hardware oder sind für die örtlichen Gegebenheiten nicht geeignet. Hinzu kommt, dass eine Person, die den Spiegel verwendet, den Blick im wesentlichen Teil der Interaktion auf den Spiegel richtet und ihm somit das Gesicht zuwendet. Die Identifikation mittels Gesichtserkennung passiert in gewisser Weise beiläufig, da der Nutzer die Erkennung nur startet und der Rest automatisiert abläuft.

Für eine berührungslose Interaktion wird eine Sprachsteuerung verwendet. Der Spiegel bietet mit einem Mikrofon die passende Schnittstelle dazu an. Eine Gestensteuerung

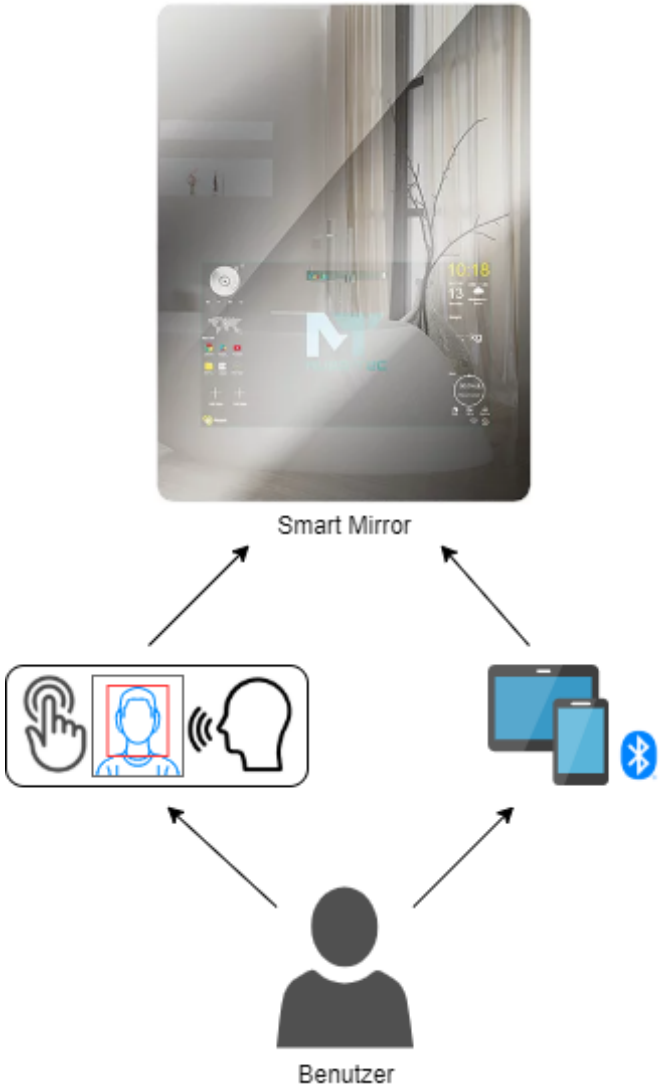

Abbildung 3.1: Darstellung der Interaktionsmöglichkeiten eines Benutzers

würde eine weitere Möglichkeit darstellen, das System zu bedienen. Da das Projekt im Badezimmer des Living Place durchgeführt wird und Benutzer eines Badezimmers häufig die Hände benutzen, wäre diese Form der Interaktion mit zusätzlichem Aufwand verbunden, da so das normale Verhalten eines Nutzers gestört wird. Vorteilhaft für eine Gestensteuerung wäre zudem eine Tiefenkamera, um auch dreidimensionale Gesten zu erkennen. Mit einer einfachen und schlecht auflösenden Kamera, wie der des Smart Mirror, ist es kaum möglich, komplexe Gesten zu erkennen. Dementsprechend wäre für eine effektive Gestensteuerung externe Hardware nötig.
Die Sprachsteuerung stellt eine einfachere und natürliche Bedienung des Systems sicher. Dafür können die bereits vorhandenen Komponenten des Spiegels genutzt werden. Ein Benutzer kann die Sprachsteuerung zudem einfacher in seinen Ablauf integrieren. Für eine schnellere Implementation sollen zunächst jedoch nur die Aktionen, die am häufigsten ausgeführt werden, wie der Login oder Logout, per Sprachbefehl gesteuert werden. Da es unter Umständen in einigen Situation nicht erwünscht ist, die App per Sprachsteuerung zu bedienen, kann ein Nutzer alle Aktionen auch standardmäßig per Touch ausführen. Abbildung 3.1 zeigt die möglichen Aktionen, die ein Benutzer ausführen kann.

Eine weitere Möglichkeit der Interaktion bieten Bluetooth Low Energy (BLE) Geräte. Damit sollen Nutzer ihre Self-Tracking Daten an das System übermitteln. Bei den BLE Geräten gibt es zunächst die Einschränkung, dass nur die bereits im Badezimmer vorhandenen Geräte (Waage, Hautfeuchtigkeitsmesser) verwendet werden können. Andere Geräte werden nicht unterstützt. Daten der Bluetooth Geräte werden nur zugelassen, sofern ein Benutzer zuvor mittels Gesichtserkennung identifiziert wurde.

3.2 Systemkomponenten

Die App kommuniziert mit verschiedenen Komponenten des Spiegels und des Betriebssystems. Abbildung 3.2 beschreibt alle verwendeten Komponenten. Android bietet für Kamera, Speicher, Lautsprecher, Mikrofon und Bluetooth Schnittstellen an, über die der Zugriff erfolgt. Die Systemnachrichten können von anderen Apps oder durch bestimmte Ereignisse (beispielsweise Ein- und Ausschalten von Bluetooth) empfangen werden. Die persönlichen Daten, sowie die von Bluetooth Geräten empfangenen Daten werden in einer Datenbank im internen Speicher persistiert. Der einzige Zugriff auf die Kamera erfolgt während einer aktiven Identifikation, sowie bei der Erstellung oder Bearbeitung eines

Benutzers. Lautsprecher und Mikrofon werden für die Sprachsteuerung benötigt. Durch Interaktion des Nutzers mit dem System werden die Komponenten angesprochen.

Abbildung 3.2: Komponenten mit denen die App interagiert

3.3 Module

Entsprechend der Anforderungen in 2.7 und den Komponenten mit denen das System interagiert, können verschiedene Module identifiziert werden, die sich jeweils um einen Teilbereich der Anwendung kümmern. Die Aufteilung der App in Module hat den Vorteil, dass die Zuständigkeiten entkoppelt sind und klare Grenzen zwischen den Aufgabenbereichen definiert werden können. Teile der Module sind gekapselt und abstrahiert, um einfacher Änderungen an der darunter liegenden Logik vornehmen zu können. Somit ist es später einfacher, eine andere Datenbank einzubinden, den Gesichtserkennungsalgorithmus auszutauschen oder eine neue Spracherkennung zu verwenden. Einzelne Module können neben der Logik auch verschiedene Android Activities[1] (sichtbare Seiten, mit der ein Benutzer interagiert) beinhalten.

[1] https://developer.android.com/reference/android/app/Activity [12.09.2021]

Nachfolgend werden die Module mit ihren Aufgaben kurz erläutert:

BLE-Handler - Kümmert sich um den Verbindungsaufbau und Datenaustausch mit BLE-Geräten. Enthält auch sichtbare Seiten, die zum einen bereits verbundene Geräte anzeigen und zum anderen die Möglichkeit bieten, nach neuen Geräten zu suchen.

Datenvisualisierung - Nachdem ein Nutzer sich mittels Gesichtserkennung identifiziert hat, visualisiert das Modul die relevanten Daten eines Benutzers.

Datenbankverwaltung - Dient als Schnittstelle zwischen der App und der Datenbank (DB), sämtliche Zugriffe auf die DB werden über das Modul geregelt. Die Datenbank liegt im internen Speicher des Smart Mirror. Das Modul wird nur nach Anfragen anderer Module aktiv, für einen Nutzer gibt es keine direkte Interaktionsmöglichkeit.

Sprachsteuerung - Bietet Funktionalitäten, um sprachlich mit der App zu interagieren. Zunächst steht jedoch nur ein eingeschränkter Zugriff auf die Startseite und die Datenvisualisierung zur Verfügung. Andere Komponenten der App können nicht durch Sprache gesteuert werden.

Gesichtserkennung - Erledigt sämtliche Aufgaben, die zur Gesichtserkennung und Identifikation gehören. Das Modul wird bei der Erstellung und Bearbeitung eines Nutzers, sowie beim Login aufgerufen.

Benutzerverwaltung - Alle Benutzer-relevanten Funktionen, wie das Erstellen, Bearbeiten oder Löschen eines Nutzers werden in diesem Modul gebündelt. Dazu gehört auch das Verwalten der Session einer authentifizierten Person.

3.3.1 Kommunikation zwischen Modulen

Nicht alle Module kommunizieren miteinander, manche arbeiten im Hintergrund und stellen nur Daten bereit, andere beinhalten auch Visualisierungen für den Nutzer. In Abbildung 3.3 sind die Interaktionen zwischen den einzelnen Modulen zu sehen. Obwohl die Startseite der App kein richtiges Modul ist, ist sie trotzdem in der Abbildung enthalten. Sie enthält keine Logik, sondern bietet die Möglichkeit andere Module anzusprechen. Deshalb sind die Verbindungen der Startseite leicht transparent dargestellt. Alle Module, die einen sichtbaren Screen beinhalten, auf dem einem Benutzer etwas präsentiert wird, sind mit einem kleinen Monitor markiert. Die Module, mit denen ein Nutzer per Touch interagieren kann, sind mit einem Finger gekennzeichnet.

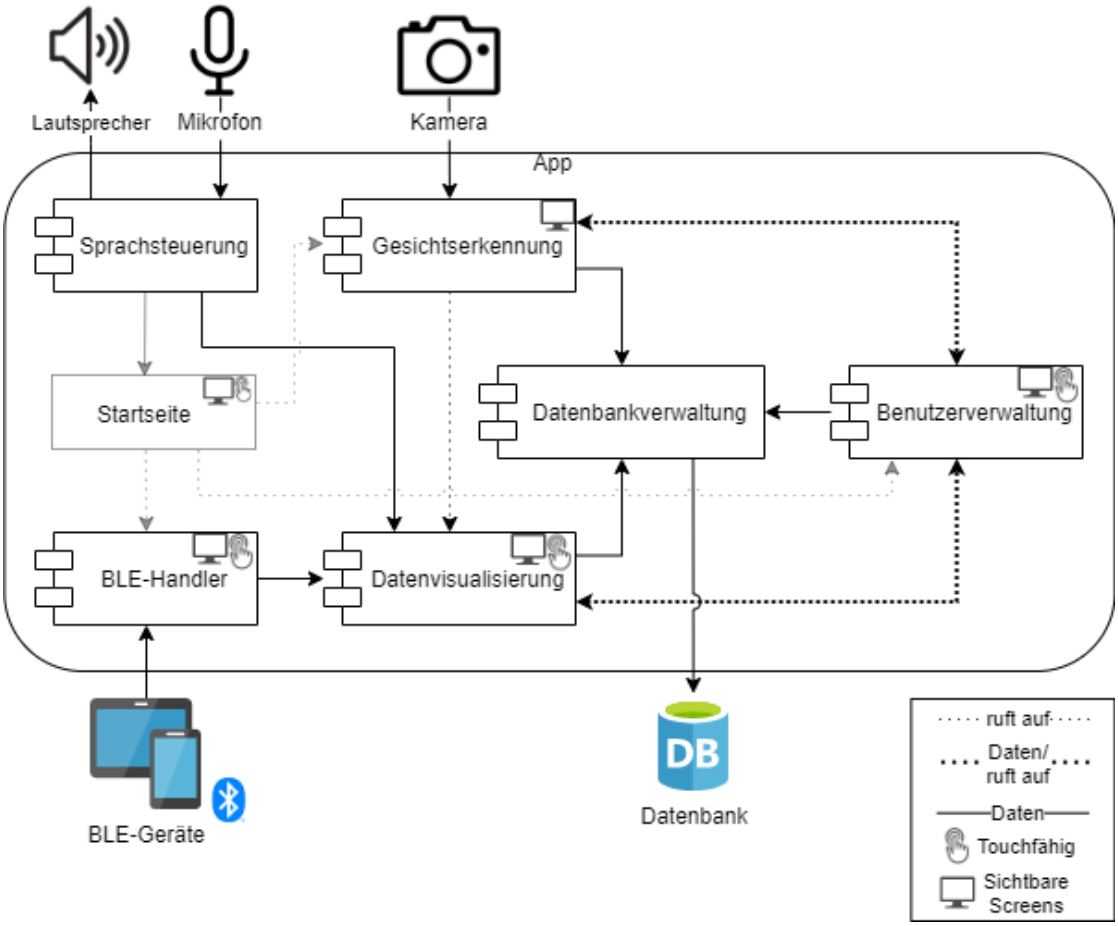

Abbildung 3.3: Grobe Beschreibung der Interaktion der einzelnen Module

3.4 Primäre Interaktionsabläufe

In 3.1 wurde aufgeführt welche Interaktionsarten das System unterstützt. In diesem Kapitel werden grob die Abläufe zwischen den Modulen (siehe 3.3.1) und während der verschiedenen Interaktionen erläutert. Dafür können drei primäre Abläufe identifiziert werden, die ein Nutzer wahrscheinlich am häufigsten durchführt. Dies sind der Login, Logout und die Datenabfrage der Bluetooth Geräte. Alles andere sind sekundäre Aktionen, die in einer geringeren Häufigkeit ausgeführt werden. Dazu gehören die Erstellung und Bearbeitung eines Benutzers und die Verwaltung der BLE-Geräte.

Damit der Spiegel sprachliche Befehle verarbeiten kann, muss er permanent die gesprochenen Wörter interpretieren. Dadurch kann es passieren, dass, obwohl die gesprochenen

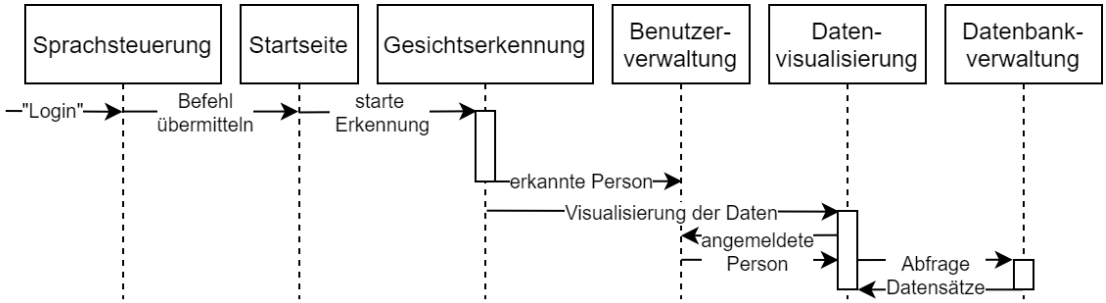

Abbildung 3.4: Beschreibung des Login nach Sprachbefehl

Wörter nicht an den Spiegel gerichtet waren, er Befehle erkennt und Aktionen ausführt. Deshalb wird ein sogenanntes Hotword verwendet, um den Spiegel mitzuteilen, dass nachfolgend Befehle zu erwarten sind. Beispiele für Hotwords sind Amazons „*Alexa*“ oder Googles „*Hey Google*“. Inspiriert davon wird der Spiegel in diesem Projekt durch „*Hey Glass*“ in einen Zustand versetzt, in welchem er Befehle verarbeiten kann. Wie in Abbildung 3.3 zu sehen, werden sprachliche Befehle zunächst nur von der Startseite und der Datenvisualisierung unterstützt.

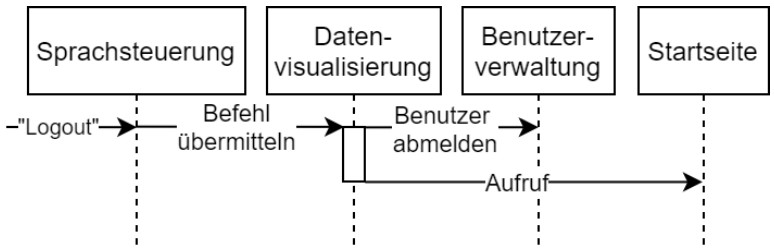

Abbildung 3.5: Beschreibung des Logout nach Sprachbefehl

In den Abbildungen 3.4, 3.5 und 3.6 sind die auszuführenden Aktionen für die primären Abläufe zwischen den einzelnen Modulen ersichtlich. Die Sequenzdiagramme stellen dabei nur einen groben Ablaufplan der nötigen Aktionen dar. Aus Gründen der Übersichtlichkeit ist die Erkennung des Hotwords nicht visualisiert. Die Sprachsteuerung an sich ist optional, da alle Aktionen auch per Touch ausgeführt werden können. Abbildung 3.6 bezieht sich auf die Abläufe zur Speicherung von Bluetooth Daten, wofür der Benutzer den Wert in einem Dialogfenster bestätigen muss. Sollte ein Nutzer den Wert nicht speichern wollen, entfällt der Aufruf der Datenbankverwaltung.

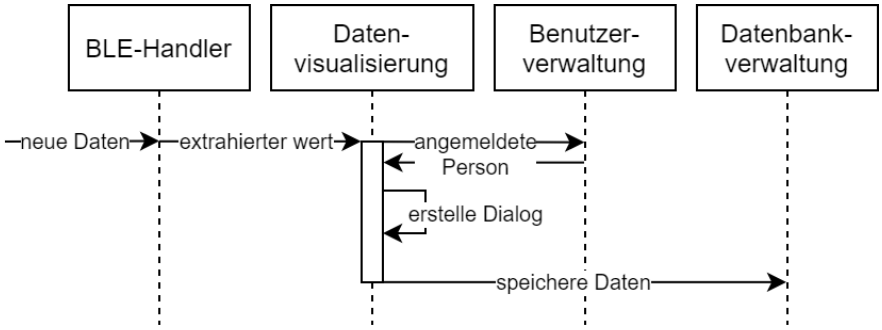

Abbildung 3.6: Module die an der Speicherung von akzeptierten Bluetooth Daten beteiligt sind

3.5 Datenmodell der Datenbank

Sämtliche Daten, die dauerhaft verfügbar sein sollen, werden in einer Datenbank vorgehalten. Da keine Kommunikation mit externen Servern stattfindet, wird die Datenbank im internen Speicher des Gerätes abgelegt. Dadurch kann eine geringe Latenz sichergestellt werden, was vor allem beim Laden der Beispielbilder für die Gesichtserkennung relevant ist. Abgeleitet aus den Anforderungen ergibt sich das unter Abbildung 3.7 skizzierte Datenmodell.

Nachfolgend werden die Tabellen mit ihrer jeweiligen Funktion erläutert:

Users - Zentrale Tabelle, die allgemeine Informationen wie Name, Alter und Größe eines Benutzers enthält. Über die *userid* als Fremdschlüssel werden alle weiteren Tabellen referenziert.

Images - Enthält alle Beispielbilder eines Benutzers für die Gesichtserkennung.

Scale - Speichert das Gewicht eines Benutzers, den Zeitpunkt der Messung und den Body-Mass-Index (BMI). Der BMI wird aus dem Alter, der Größe und dem Gewicht des Nutzers berechnet.

Humidity - Stellt die gemessenen Hautfeuchtigkeitswerte eines Benutzers mit dem jeweiligen Zeitstempel der Messung bereit.

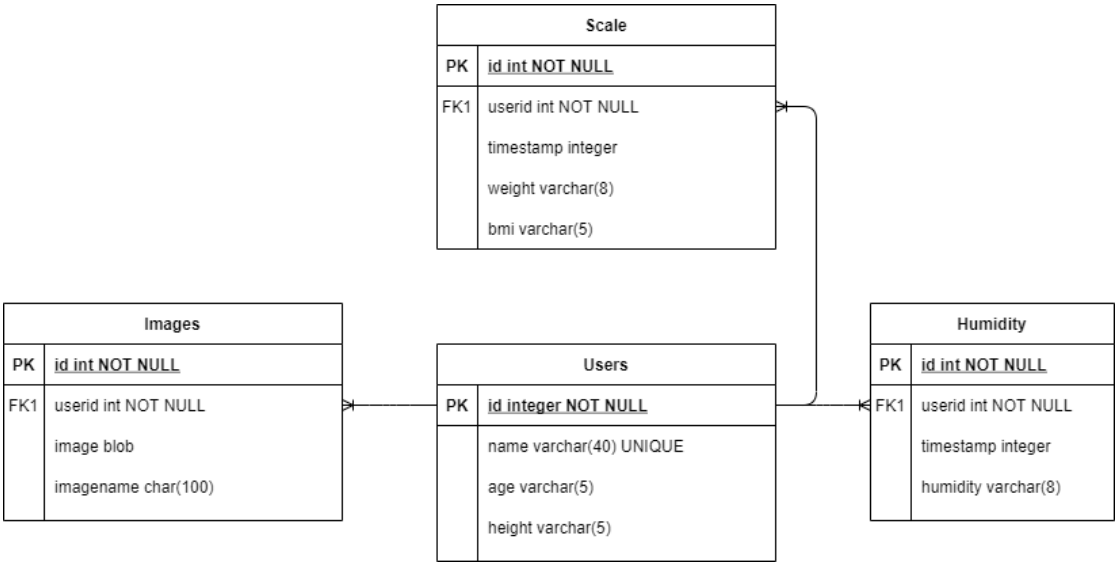

Abbildung 3.7: Beschreibung des verwendeten Datenmodells zur Speicherung aller Benutzer relevanten Daten

3.6 Implementierung der Module

In den nachfolgenden Kapiteln wird die konkrete Implementierung der bereits beschriebenen Module erläutert. Das gesamte Projekt wird in der Programmiersprache Java erstellt. Da der Smart Mirror mit Android Version 8 betrieben wird, muss darauf geachtet werden, dass alle Funktionen der App bis zu dieser Version abwärtskompatibel sind. Als Entwicklungsumgebung wird Android Studio 4.1.2 verwendet.

3.6.1 Interne Kommunikation

Für die Kommunikation zwischen den Modulen, stehen in Android sogenannte Intents[2] zur Verfügung. In diesem Projekt werden sie auf zwei Arten genutzt. Mit einem Intent kann eine Activity (eine sichtbare Seite) gestartet werden und dabei Daten, die in Key-Value Paaren geordnet sind, erhalten.
Broadcasts[3] stellen die zweite Möglichkeit dar, um mithilfe von Intents Daten zwischen Modulen auszutauschen. Broadcasts arbeiten nach dem publish-subscribe Pattern und

[2] https://developer.android.com/reference/android/content/Intent [12.09.2021]
[3] https://developer.android.com/guide/components/broadcasts [12.09.2021]

BLE-Handler	
Intern	Extern
DEVICE_DISCOVERED	*DEVICE_SET_SESSION*
DEVICE_CONNECTION_REQ	*MEASUREMENT_WEIGHT_DATA*
	MEASUREMENT_HUMIDITY_DATA
Datenvisualisierung	
Intern	Extern
ALERT_REQUESTED	
Sprachsteuerung	
Intern	Extern
	SPEECH_MIC_PANEL_OFF
	SPEECH_MIC_PANEL_ON
	SPEECH_COMMAND_LOGIN
	SPEECH_COMMAND_LOGOUT
	SPEECH_COMMAND_REGISTER
	SPEECH_COMMAND_YES
	SPEECH_COMMAND_NO
	SPEECH_COMMAND_QUIT

Tabelle 3.1: Die verwendeten Broadcast Topics der einzelnen Module

werden systemweit verschickt. Es ist damit möglich, Benachrichtigungen vom System, anderen Apps oder einem Modul der eigenen App zu erhalten. Hintergrundprozesse, wie der Datenaustausch per Bluetooth oder die Sprachsteuerung, können mit diesem Verfahren Daten an Module schicken, die gerade im sichtbaren Fokus des Anwenders liegen. Tabelle 3.1 zeigt alle Module, die Daten per Broadcasts verschicken. Da ein Modul aus mehreren Activities bestehen kann und diese auch untereinander Nachrichten austauschen, ist die Auflistung in interne und externe Broadcast Topics aufgeteilt.

3.6.2 Datenbankverwaltung

Aufgrund der einfachen Einbindung in Android Studio wird SQLite[4] als Grundlage für die Datenbank in diesem Projekt verwendet. SQLite ist ein relationales Datenbank Management System, das keine Client-Server Architektur benötigt, sondern in die Anwendung eingebettet ist. Somit wird ein schneller Zugriff auf die Daten garantiert. Zur Erstellung und Bearbeitung von SQLite Datenbanken gibt es verschiedene Möglichkeiten. In diesem Projekt wird dafür SQLiteStudio[5] verwendet.

[4] https://www.sqlite.org/index.html [12.09.2021]
[5] https://sqlitestudio.pl [12.09.2021]

In Abbildung 3.8 ist zu sehen, dass die Klasse *DBOperations* als Fassade für den Zugriff auf die Datenbank fungiert. Dies hat die Vorteile, dass es für andere Module nur einen zentralen Ansprechpartner für den Zugriff auf die Datenbank gibt und Änderungen an der darunter liegenden Logik einfach durchgeführt werden können.

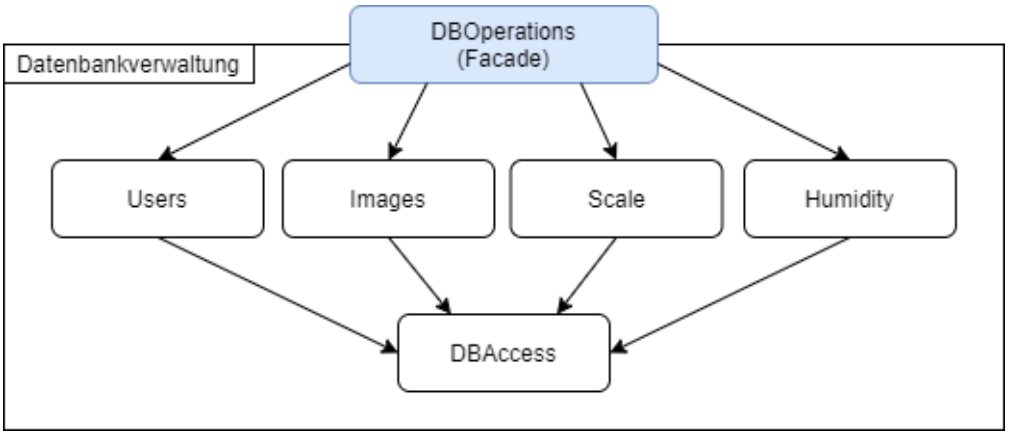

Abbildung 3.8: Mit dem Facade Pattern wird der Zugriff auf die Datenbank gekapselt

3.6.3 Benutzerverwaltung

Die Benutzerverwaltung erfüllt mehrere Aufgaben. Sie bietet die Möglichkeit, einen Benutzer zu erstellen, der Prozess kann über die Startseite angestoßen werden. Ein angemeldeter Benutzer, der sich bereits per Gesichtsidentifikation authentifiziert hat, kann seine existierenden Beispielbilder einsehen und Neue erzeugen, um bei der nächsten Anmeldung besser vom System erkannt zu werden. In der gleichen sichtbaren Activity, die über die Datenvisualisierung erreicht werden kann, ist es zudem möglich, einzelne Bilder oder den gesamten Benutzer und alle dazugehörigen Daten zu löschen. Zudem stellt die Benutzerverwaltung eine Schnittstelle für andere Module bereit, um den angemeldeten Benutzer zu verwalten und auf seine Daten zuzugreifen.

In den Prozess zur Erstellung eines neuen Benutzers sind mehrere Module eingebunden. Abbildung 3.9 zeigt den generellen Ablauf. Der Prozess kann über die Startseite durch einen Sprachbefehl oder durch das Klicken auf den „Register“ Button gestartet werden. Nach Eingabe der Benutzerinformationen (Name, Alter, Größe) werden standardmäßig für 10 Sekunden Beispielbilder des Benutzers aufgenommen und in der Datenbank abgespeichert. Dabei wird das erkannte Gesicht auf dem Bildschirm mit einem grünen Rechteck gekennzeichnet.

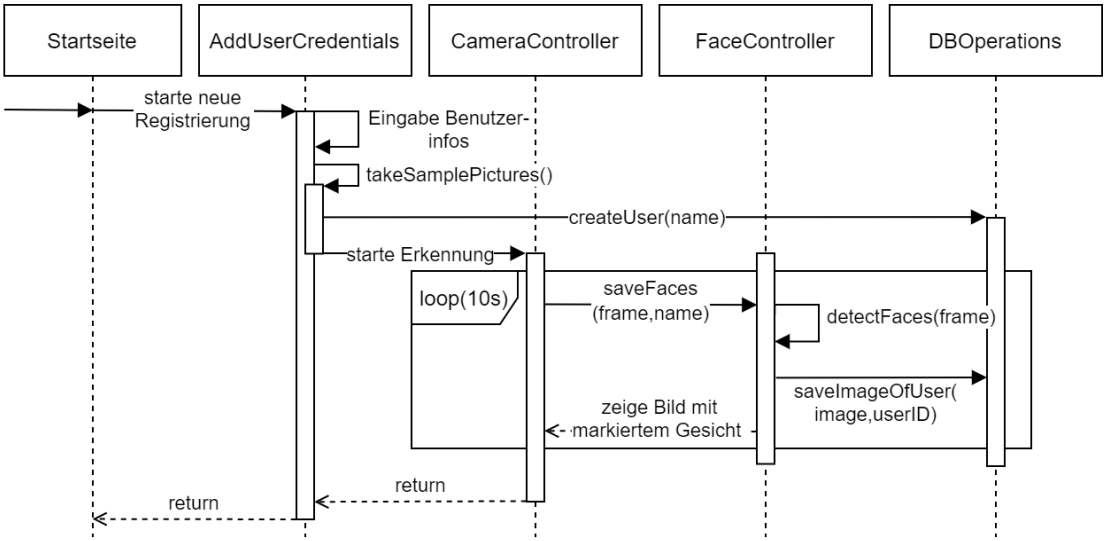

Abbildung 3.9: Beschreibung der eingebundenen Komponenten zur Benutzererstellung

3.6.4 Gesichtserkennung

Wie in Kapitel 2.5.1 beschrieben, bieten sich verschiedene Bibliotheken an, um die Anforderungen an die Gesichtserkennung umzusetzen. Aufgrund der einfachen Einbindung, der Vielzahl an Möglichkeiten und der guten Dokumentation wird in diesem Projekt OpenCV als Computer Vision Bibliothek verwendet. Hinzu kommt, dass am Anfang des Projektes angedacht war, den Fokus nicht nur auf die Gesichtserkennung und die Identifikation zu legen, sondern auch Bilder zu bearbeiten.
Als Algorithmus für die Gesichtsidentifikation wird LBPH (siehe 2.5.5) verwendet. Untersuchungen[1, 38] kommen zu dem Schluss, dass LBPH eine zuverlässigere und schnellere Erkennung als die anderen OpenCV Algorithmen sicherstellt. Dabei kann allerdings die programmtechnische Konfiguration des Algorithmus eine große Rolle spielen. Zudem ist LBPH relativ robust, da es auch gute Ergebnisse bei unterschiedlichen Belichtungen liefert. Da der Rechenaufwand aufgrund des einfachen Prinzips gering ist, eignet sich LBPH am ehesten auf Systemen wie dem Smart Mirror, der nur über begrenzte Rechenkapazitäten verfügt.

Die Gesichtsidentifikationsalgorithmen sind nicht im normalen OpenCV Umfang enthalten. Für experimentelle oder kaum benutzte Funktionalitäten wird das OpenCV-Contrib Modul bereitgestellt. Um Funktionalitäten von OpenCV und dem Contrib Modul zu nutzen, müssen beide zusammengeführt werden. Da die manuelle Einbindung von OpenCV

und dem Contrib Modul in Android Studio einen nicht unerheblichen Konfigurationsaufwand erfordert, wird ein bereits vorgefertigtes und einfach einzubindendes Package[6] verwendet.

Trainieren des Algorithmus

Wie in Kapitel 2.5.5 bereits angedeutet, muss der Gesichtsidentifikationsalgorithmus trainiert werden, bevor er Personen in einem Bild identifizieren kann. Das bedeutet nichts anderes, als das Histogramme aller Testbilder erstellt und vorgehalten werden müssen, damit ein zu testendes Bild verglichen werden kann. Das Erzeugen der Histogramme wird an drei Stellen durchgeführt, zum einem beim Start der Anwendung, nach Erstellung eines neuen Benutzers und beim Hinzufügen von neuen Bildern für einen bestehenden Nutzer.

Damit die Testbilder einer Person zugeordnet werden können, stehen Identifikatoren im Dateinamen des Bildes: <Label>_<Name>_<Index>. Das Label stellt für den Identifikationsprozess die wichtigste Größe dar. Jeder Nutzer hat ein eindeutiges Label, das der Identifikationsalgorithmus zurückgibt, sofern er eine Person erkannt hat. Über das Label ist es dann möglich, den Namen der Person aus der Datenbank zu erhalten. Der Index wird bei jedem Bild erhöht und dient lediglich der Unterscheidung von verschiedenen Bildern einer Person.

Ablauf der Identifikation

Der Ablauf der Identifikation für ein Bild ist in Abbildung 3.10 beschrieben. OpenCV stellt über ein zu implementierendes Interface jeden Frame der Kamera zur Verfügung. Der Frame wird dann dem *FaceController* zur dargestellten Verarbeitung übergeben. Die Gesichter werden in dem eingehenden Frame nach dem Verfahren in Kapitel 2.5.4 bestimmt. Der Cascade Classifier ist im internen Speicher hinterlegt. Daraus wird ein Task erstellt, der dem *RecognizerThread* übergeben wird, sofern er nicht gerade mit einer laufenden Identifikation beschäftigt ist. Die Frequenz, mit der die Frames von der Kamera eingehen, ist höher als die Verarbeitungsgeschwindigkeit des Identifikationsalgorithmus, besonders dann, wenn viele Beispielbilder vorhanden sind, dadurch kann nicht

[6] https://github.com/quickbirdstudios/opencv-android [12.09.2021]

jeder Frame verarbeitet werden. Somit kann es sein, dass über einen Zeitraum von mehreren Frames ein altes Identifikationsergebnis angezeigt wird. Dies hat aber den Vorteil, dass das Kamerabild flüssig wiedergegeben wird und nicht auf eine Identifikation gewartet werden muss. Damit ein Benutzer angemeldet werden kann, muss er in mindestens 80% der vom Identifikationsalgorithmus verarbeiteten Frames erkannt werden.

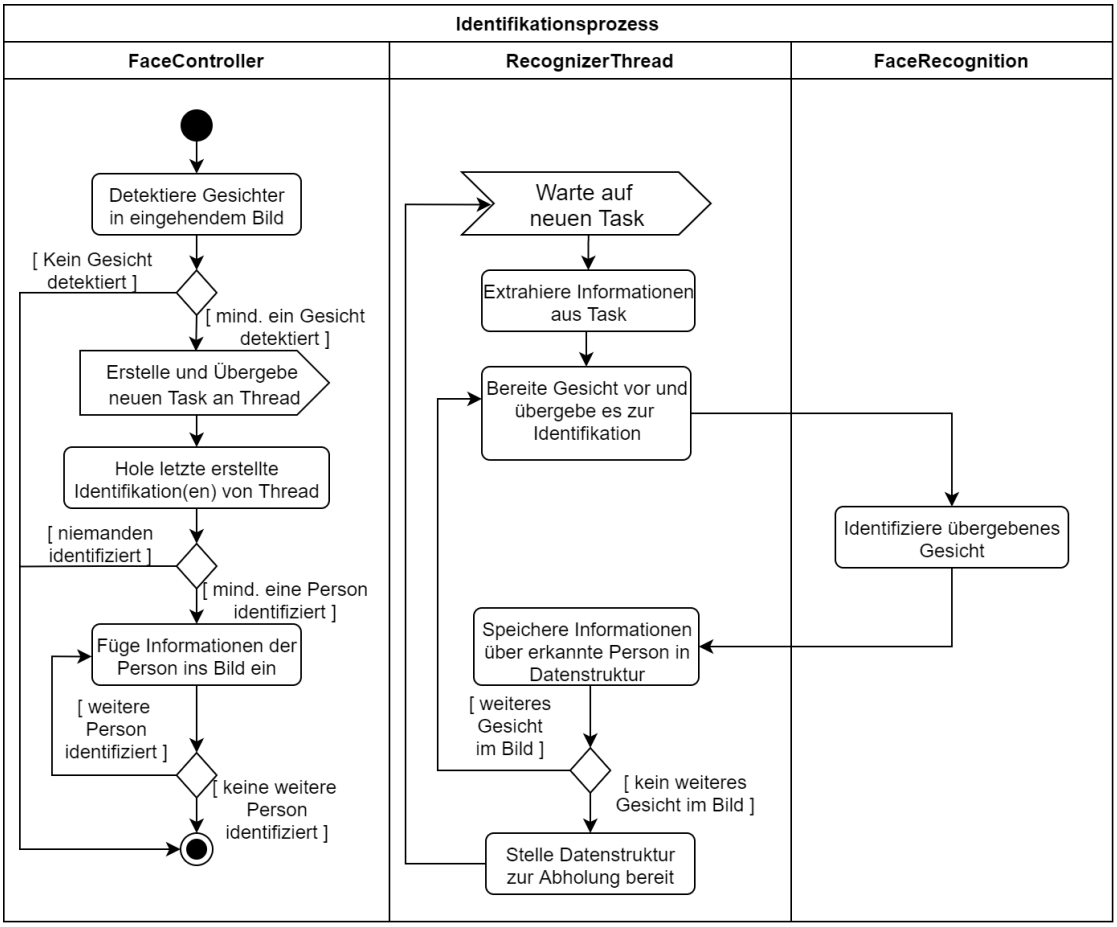

Abbildung 3.10: Beschreibung des Prozesses zur Identifikation von Personen aus Kamerabild

3.6.5 Sprachsteuerung

Für die Sprachsteuerung wird der frei verfügbare Dienst Vosk[7] von Alphacephei verwendet. Als Grundlage für Vosk dient die unter 2.6.3 vorgestellte Bibliothek CMUSphinx.

[7] https://alphacephei.com/vosk/ [12.09.2021]

Vosk unterstützt eine Vielzahl von Sprachen und ist dabei im Gegensatz zu der von Android bereitgestellten *SpeechRecognizer*[8] Klasse, nicht auf einen Onlinedienst angewiesen, sondern erledigt die Erkennung offline. Weitere Vorteile gegenüber der Standard Android API sind zum einen der geringere Implementierungsaufwand und die Möglichkeit zur Verwendung eines Hotwords. Als Sprache wird zunächst Englisch verwendet, auf der Website stehen jedoch weitere Sprachen zum Download bereit.

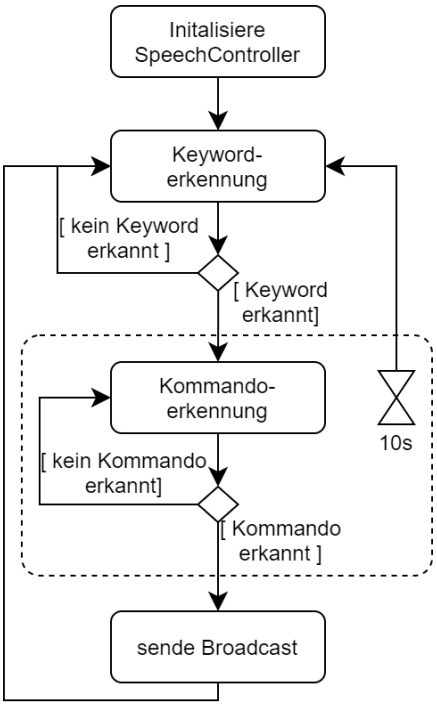

Abbildung 3.11: Beschreibung des Spracherkennungssystems

Konkret wurde für diese Arbeit das Vosk Beispielprojekt für Android[9] in die App eingebunden und entsprechend angepasst. Wie bereits in 3.4 erwähnt, wird als Hotword „*Hey Glass*“ verwendet. Sobald das Hotword erkannt wird, startet ein 10-Sekunden-Timer. Durch ein akustisches Signal, sowie durch Einblendung eines kleinen Mikrofons am unteren Bildschirmrand, wird dem Benutzer signalisiert, dass der Spiegel aufnahmebereit für die unterstützten Befehle (siehe Tabelle 3.2) ist. Abbildung 3.11 beschreibt den internen Ablauf im Sprachsteuerungsmodul. Sobald ein Befehl detektiert wird, wird wie in 3.6.1 beschrieben ein Broadcast versendet, sodass andere Module auf den Befehl reagieren können.

[8] https://developer.android.com/reference/android/speech/SpeechRecognizer [12.09.2021]
[9] https://github.com/alphacep/vosk-android-demo [12.09.2021]

Intern heißt die Klasse, welche die Sprachsteuerung abbildet, *SpeechController*. Damit nicht mehrere Instanzen nebeneinander existieren und sich überlagern, wird mit dem Singleton-Pattern sichergestellt, dass nur eine Instanz zurzeit, auf das Sprachmodell zugreift.

Startseite		**Datenvisualisierung**	
Befehl	Zweck	Befehl	Zweck
Login	Starten der Gesichtsidentifikation.	*Logout*	Abmelden des Benutzers und Rückkehr zur Startseite.
Register	Anlegen eines neuen Benutzers.	*Yes*	Bestätigung des Popup, sodass übermittelter Bluetooth Wert in Datenbank gespeichert wird.
Quit	Schließen der App.	*No*	Ablehnung des Popup, sodass übermittelter Bluetooth Wert nicht in Datenbank gespeichert wird.
		Quit	Schließen der App.

Tabelle 3.2: Auflistung der zurzeit unterstützten Sprachbefehle

3.6.6 BLE-Handler

Neben dem klassischen Bluetooth, bei welchem einmal eine Verbindung zwischen zwei Geräten hergestellt wird und solange diese existiert, dauerhaft Daten ausgetauscht werden, gibt es noch Bluetooth Low Energy (BLE). BLE wird in Systemen eingesetzt, die periodisch oder auf Anfrage Daten senden. Dadurch werden die Batterien der Geräte geschont, wodurch eine lange Akkulaufzeit ermöglicht wird. Deshalb werden BLE-fähige Geräte häufig in Anwendungen eingesetzt, die dem Internet der Dinge zuzurechnen sind. Dazu gehören die in diesem Projekt verwendete Waage und der Hautfeuchtigkeitsmesser.

Android bietet zwar eine API an, mit der eine Verbindung zu BLE-Geräten hergestellt werden kann, jedoch bezieht sich die Dokumentation häufig auf das klassische Bluetooth. Hinzu kommt das die API auf einer sehr niedrigen Abstraktionsebene operiert, sodass ein Entwickler sich um Dinge wie Verbindungsmanagement oder eine korrekte Reihenfolge bei der Abarbeitung von Befehlen selber kümmern muss. Deshalb wird zur Anbindung und Verwaltung von BLE-Geräten, die frei verfügbare *Blessed*[10] Bibliothek verwendet, die einiges an Arbeit abnimmt und eine einfache Erstellung einer Bluetooth Low Energy Schnittstelle ermöglicht.

[10] https://github.com/weliem/blessed-android [12.09.2021]

Der BLE-Handler hat die Aufgabe, Verbindungen mit den Geräten herzustellen und Daten auszutauschen. Dabei obliegt ihm aber nicht die Interpretation der gesendeten Daten. Das Modul schickt die Daten per Broadcast an andere Module, die damit dann weiter arbeiten. Durch zwei Activities, welche über die Startseite erreicht werden können, kann der Benutzer verbundene Geräte einsehen und neue Geräte verbinden. Zurzeit können neben der Waage und dem Hautfeuchtigkeitsmesser keine weiteren Geräte angebunden werden. Alle Geräte, mit denen eine Verbindung hergestellt werden soll, müssen in der Software abgebildet werden. Aufgrund des Aufwandes wurde hierbei kein generischer Ansatz implementiert. Wenn der Benutzer eine Suche nach BLE-Geräten durchführt, werden nur die Waage und der Feuchtigkeitsmesser angezeigt, allerdings nur, sofern noch keine Verbindung besteht.

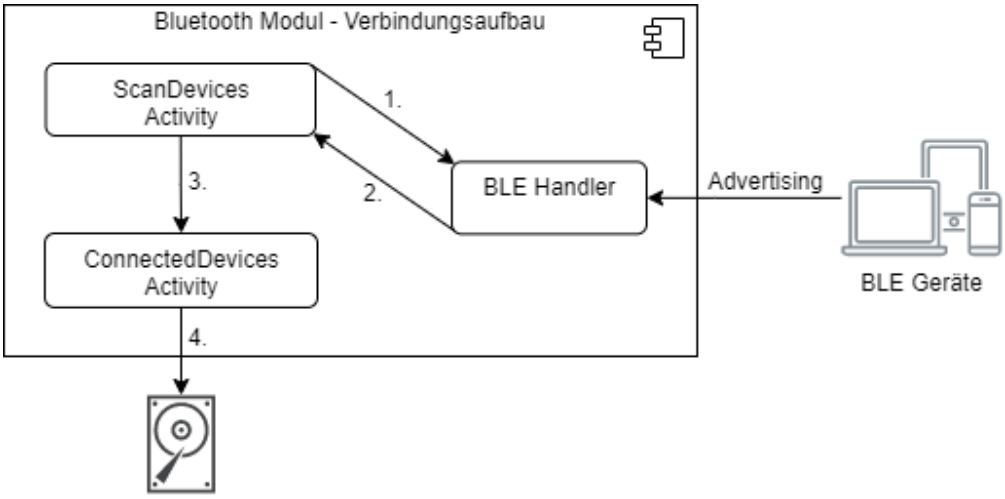

Abbildung 3.12: Komponenten die am Verbindungsaufbau beteiligt sind

Abbildung 3.12 beschreibt den Verbindungsprozess mit den eingebundenen Komponenten. Eingeschaltete BLE-Geräte senden periodisch Advertising Pakete, über die ein Empfänger die nötigen Informationen für einen Verbindungsaufbau extrahieren kann. Startet der Benutzer über die *ScanDevices* Activity einen neuen Suchlauf, wird dem BLE-Handler mitgeteilt, dass er auf die Advertising Pakete reagieren soll (Punkt 1 in Abb. 3.12). Sobald der Handler ein Gerät identifizieren konnte, sendet er die Daten des Gerätes per Broadcast zurück an die Activity (Punkt 2 in Abb. 3.12). Daraufhin wird das Gerät visualisiert und ein Nutzer kann eine Verbindung initiieren. Sobald die Verbindung aufgebaut wurde, wird die Activity geschlossen und zur *ConnectedDevices* Activity gewechselt (Punkt 3 in Abb. 3.12). Dort wird das neue Gerät in eine Datenstruktur im Speicher des Smartpho-

nes eingetragen, sodass auch beim nächsten Öffnen der App Daten ausgetauscht werden können, ohne erneut den Verbindungsprozess durchlaufen zu müssen (Punkt 4 in Abb. 3.12).

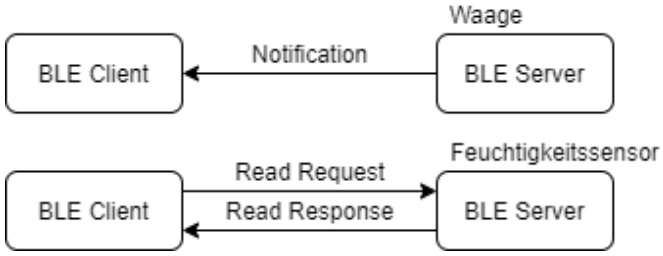

Abbildung 3.13: Die benutzten Methoden zum Datenaustausch zwischen Client und Server

Wie bereits erläutert, versendet das Modul die empfangenen Daten der BLE-Geräte (Server) per Broadcast an das Datenvisualisierungsmodul. Um Daten von Geräten zu erhalten, gibt es zurzeit zwei Möglichkeiten, die in Abbildung 3.13 dargestellt sind. Beim Verbindungsaufbau wurde der Waage mitgeteilt, dass sie, sobald ein neuer Wert vorhanden ist, eine Notification mit dem Wert versenden soll. Es findet auf der Seite der App (Client) kein Polling statt. Anders verhält es sich beim Hautfeuchtigkeitsmesser, dieser unterstützt keine Notifications, sodass der BLE-Handler iterativ per Read Request die Daten anfordert. Das Intervall zur Abfrage der Daten liegt zurzeit bei 500 Millisekunden. Da beispielsweise eine Waage nicht nur den Gewichtswert versenden kann, sondern auch weitere Werte (siehe Body Composition Service[11]), die in einem Byte-Array enthalten sind, muss in einem letzten Schritt der jeweilige Wert extrahiert werden, bevor er per Broadcast versendet werden kann.

3.6.7 Datenvisualisierung

Nachdem ein Benutzer sich durch die Gesichtserkennung authentifiziert hat, werden die persönlichen Self-Tracking Daten aus der Datenbank visualisiert. Die Anzeige der Daten ist rudimentär und soll nur die Möglichkeiten aufzeigen. Für die Anzeige der Gewichtsdaten wird ein Diagramm erzeugt, welches auf dem frei verfügbaren Android Plugin GraphView[12] beruht. Zusätzlich wird der Body-Mass-Index (BMI) des letzten bekannten Gewichtswertes dargestellt. Eine kleine Grafik stellt die verschiedenen Bereiche des BMI

[11] https://www.bluetooth.com/specifications/specs/ [12.09.2021]
[12] https://github.com/jjoe64/GraphView [12.09.2021]

dar und ermöglicht eine Einordnung des aktuellen Wertes. Für den Hautfeuchtigkeitswert wird nur der letzte Wert eines Benutzers aus der Datenbank geladen und visualisiert.

Wie bereits in 3.3.1 erläutert, bietet die Visualisierung die Möglichkeit der bedingten sprachlichen Interaktion. Es ist möglich sich per Sprachbefehl abzumelden, einen eingehenden Wert eines Bluetooth Gerätes zu bestätigen oder abzulehnen oder die App komplett zu schließen. Bei Letzterem wird ein Benutzer ebenfalls abgemeldet, sodass beim erneuten Öffnen der App eine neue Authentifizierung stattfinden muss.

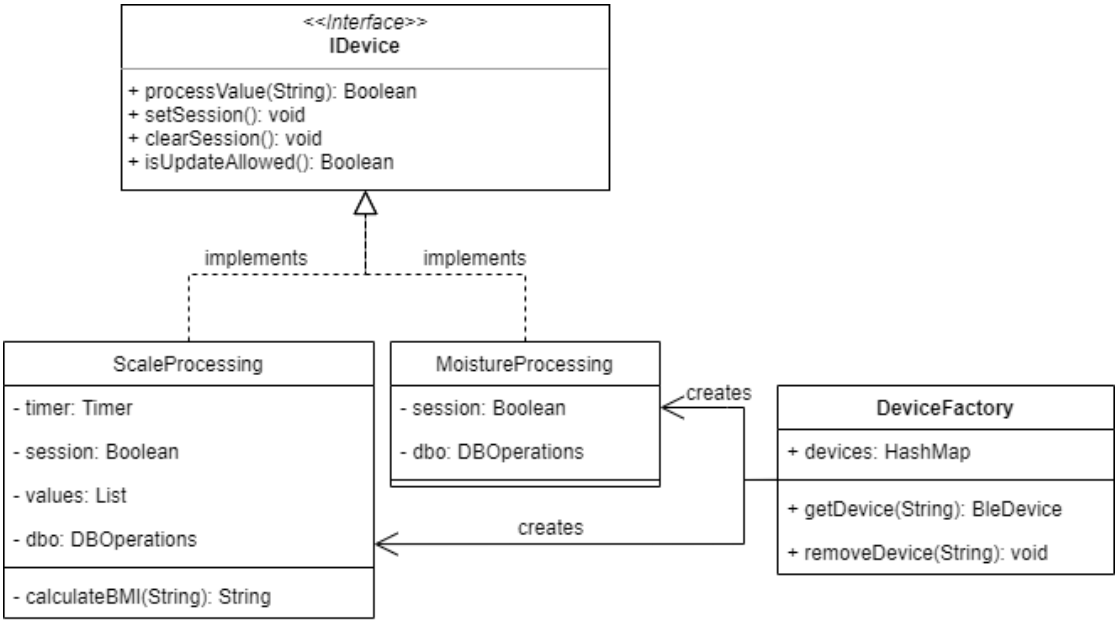

Abbildung 3.14: Factory Pattern zur einfacheren Verwaltung der angebundenen Bluetooth Geräte und deren Daten

Dem Modul fällt nicht nur die Aufgabe der Visualisierung zu, sondern auch die Interpretation der Bluetooth Daten. Wie in 3.6.6 beschrieben, senden die eingeschalteten Geräte, bei bestehender Verbindung, periodisch die gemessenen Daten. Sobald eine Verbindung mit einem Gerät aufgebaut wurde, wird ein Session-Flag gesetzt. Solange die Session gesetzt ist, können Daten verarbeitet werden. Pro Session darf nur ein Wert in der Datenbank gespeichert werden. Erst wenn über einen definierbaren Zeitraum kein anderer Wert mehr eintrifft, kann davon ausgegangen werden, dass es der letzte Wert ist und die Messung abgeschlossen wurde. Daraufhin wird dem Benutzer ein Dialog angezeigt, in dem er den gemessenen Wert bestätigen oder ablehnen kann.

Um eine einheitliche Schnittstelle für die Verarbeitung der Bluetooth Daten zu gewährleisten und um einfacher neue Geräte hinzufügen zu können, werden die Bluetooth Geräte im Visualisierungsmodul mit dem Factory-Pattern verwaltet. Abbildung 3.14 zeigt die aktuelle Struktur.

3.7 Evaluation

Abschließend wird überprüft, ob die Umsetzung in Kapitel 3 den funktionalen Anforderungen aus 2.7 genügt. Dafür werden die genannten Punkte aus den Anforderungen nacheinander bewertet.

Interaktion mit dem Spiegel

Für eine natürliche und berührungslose Interaktion mit dem Spiegel wird eine Sprachsteuerung verwendet. Die Sprachsteuerung wird zunächst jedoch nur mit wenigen Befehlen auf der Startseite und der Datenvisualisierung unterstützt. Der Fokus lag dabei auf den gängigsten Interaktionen eines Nutzers mit dem Spiegel: Login, Logout und der Datenbeschaffung von BLE-Geräten. Die Sprachsteuerung arbeitet recht zuverlässig, es kann jedoch durch verschiedene Faktoren zu Fehlinterpretationen kommen, wie in 2.6.1 beschrieben. Durch eine kleine Meldung am unteren Bildschirmrand ist die letzte Interpretation der Sprachsteuerung ersichtlich, sodass man den nächsten Befehl korrigieren kann. Durch die Verwendung eines Hotwords, welches die Sprachsteuerung aktiviert, wird vermieden, dass durch Umgebungsgeräusche oder Konversationen, Aktionen ausgeführt werden.

Für eine berührungslose Identifikation einer Person wird eine Gesichtserkennung eingesetzt. Die Gesichtserkennung funktioniert relativ verlässlich, es kann jedoch durch verschiedene Faktoren (siehe 2.5.2), wie zum Beispiel die Belichtung passieren, das Gesichter nicht erkannt oder identifiziert werden. Es kann also sein, dass Personen, die Beispielbilder an einem sonnigen Tag und ohne eingeschaltetes Licht aufgenommen haben, an einem wolkigen Tag mit eingeschaltetem Licht nicht mehr erkannt werden. Es besteht auch die Möglichkeit, dass nicht die Person, welche vor dem Spiegel steht erkannt wird, sondern ein anderer registrierter Benutzer.

Der Smart Mirror besitzt jedoch nur sehr begrenzte Rechenkapazitäten, zudem hat die Kamera keine hohe Auflösung. Aufgrund der Coronasituation wurde die Entwicklung

hauptsächlich mit privaten Smartphones und Tablets vorangetrieben. Bei Tests im Living Place hat sich herausgestellt, dass die Performance des Smart Mirror gegenüber den Entwicklungssystemen deutlich abfällt und es sogar häufiger zu Systemabstürzen kommt. Sowohl die Spracherkennung, als auch die Gesichtserkennung sind rechenintensive Aufgaben, die den Smart Mirror offensichtlich an seine Grenzen bringen.

Beschaffung von Daten

Das System bietet die Möglichkeit, sich mit Bluetooth Low Energy Geräten zu verbinden. Jedoch nur mit der bereits vorhandenen Waage und dem Hautfeuchtigkeitsmesser. Eine allgemeine Schnittstelle für alle BLE-Geräte wäre in der gegebenen Zeit schwer zu realisieren gewesen. Bei einem Suchlauf nach verfügbaren Geräten werden auch nur die beiden genannten Geräte angezeigt, sofern nicht schon eine aktive Verbindung aufgebaut wurde. Neue Geräte in der Software abzubilden, sollte aber relativ einfach möglich sein. Beim Hautfeuchtigkeitssensor kommt zudem hinzu, dass er keine standardisierten Bluetooth Services unterstützt. Dadurch ist nicht ganz klar, welcher der gesendeten Werte den eigentlichen Feuchtigkeitswert darstellt. Generell ist es aber möglich, Daten mit den BLE-Geräten auszutauschen. Sind die Geräte einmal verbunden, werden Informationen dazu in einer internen Datenstruktur hinterlegt, sodass immer Daten ausgetauscht werden können, sobald sie sich in Reichweite des Spiegels befinden. Um Bluetooth Daten speichern zu können, muss ein Benutzer sich vorher authentifizieren, andernfalls werden die gesendeten Daten verworfen.

Verwaltung der Daten

Die personenbezogenen Daten werden in einer SQLite Datenbank gespeichert. Die Datenbank liegt im internen Speicher des Systems und bietet somit eine schnelle und zuverlässige Verbindung. Verschiedene Module des Systems greifen dabei auf die Datenbank zu. Beim Aufbau der Datenbankverwaltung war das Ziel, den Zugriff zu abstrahieren, sodass später einfach die Datenbank ausgetauscht oder auf Daten aus dem Netzwerk zugegriffen werden kann, ohne das in anderen Teilen der Anwendung Änderungen durchgeführt werden müssen. Daneben ist es möglich, einen neuen Benutzer zu erstellen und einen bestehenden Benutzer zu löschen, dies muss ein Nutzer aber selber durchführen, nachdem er sich angemeldet hat. Nachdem ein Nutzer gelöscht wurde, sind keine seiner Daten mehr in der Datenbank enthalten. Ein Benutzer kann insofern bearbeitet werden,

dass er neue Beispielbilder für die Identifikation erstellen kann. Dies ist jedoch nur möglich, nachdem er sich angemeldet hat. Weitere Daten, die bei der Anmeldung erhoben wurden, wie das Alter, können zunächst nicht angepasst werden.

Visualisierung der Daten

Nachdem ein Benutzer durch die Gesichtserkennung identifiziert wurde, werden die Gewichtswerte des Nutzers in einem einfachen Diagramm dargestellt. Zudem werden der aktuellste Body-Mass-Index und der letzte Feuchtigkeitswert der Haut angezeigt. Dies sind die einfachsten Formen der Visualisierung, sie beinhalten zudem keine Interpretationen der Daten. Es wäre absolut denkbar, durch Data Mining Verfahren, Muster in den Daten zu erkennen, um so dem Nutzer Vorschläge zu machen, wie er seine Gesundheit verbessern kann. Zudem wäre es möglich, die Daten in einem deutlich umfangreicheren Maße, beispielsweise durch Augmented Reality, darzustellen. Nach dem Erhalt neuer Bluetooth Daten wird dem Benutzer ein Dialog angezeigt, der eine Interaktion erfordert. Durch den Dialog können die eingegangenen Bluetooth Daten, entweder bestätigt und somit in der Datenbank gespeichert oder abgelehnt werden. Empfindet der Benutzer die Daten als plausibel und bestätigt sie, wird die Anzeige aktualisiert.

3.7.1 Fazit der Evaluation

Die Anforderungen konnten durch das implementierte System direkt oder in leicht abgewandelter Form umgesetzt werden. Damit kann auch gezeigt werden, dass das in 2.1 skizzierte Szenario von Sal umsetzbar ist. Als einziges, größeres Problem stellte sich die Rechenleistung des verwendeten Smart Mirror heraus. Durch Verwendung der Sprach- und Gesichtserkennung kommt es hier teilweise zu Performance Problemen. Interessant wäre es herauszufinden, ob durch andere Bibliotheken zur Sprach- und Gesichtserkennung, die gleichen Probleme auftreten oder gänzlich andere Möglichkeiten der Interaktion besser geeignet wären. Trotz des genannten Problems bietet das System eine gute Grundlage, um weitere Untersuchungen im Bereich der berührungslosen Interaktion mit einem Smart Home durchzuführen. Auch wenn die Anforderungen zunächst nur für einen 1-Personenhaushalt definiert wurden, ist die Anwendung bereits darauf ausgelegt, mehrere Personen zu unterstützen. Die Gesichtserkennung, sowie die Sprachsteuerung kann zunächst nur von einer Person zurzeit bedient werden. Zusammen mit der Benutzerver-

waltung ist aber eine Grundlage vorhanden, um eine Interaktion mit mehreren Person zu ermöglichen.

4 Zusammenfassung und Ausblick

4.1 Zusammenfassung

Im Rahmen dieser Arbeit sollte ein System entwickelt werden, mit dem ein Benutzer berührungslos interagieren kann, das Daten von Bluetooth Geräten (Quantifief Self-Daten) erfassen und in einfacher Form visualisieren kann. Das erstellte System, soll ermöglichen, in einer unaufdringlichen, beiläufigen weise Daten zu erheben. Grundlage dafür bildete ein Smart Mirror der im Badezimmer des Living Place angebracht ist und auf dem das mobile Betriebssystem Android verwendet wird. Bei der Erstellung des Systems wurde von einem 1-Personenhaushalt ausgegangen.

Dafür wurde zunächst in Kapitel 2 das Projektszenario vorgestellt, um einen Einstieg in diese Arbeit zu erhalten. Danach wurde erläutert, was ein Smart Environment ist, auf welche Weise man mit Geräten interagieren kann und wie der verwendete Smart Mirror einzuordnen ist. Es wurde ein kurzer Überblick zu Quantified Self gegeben und verschiedene Identifikationsverfahren beleuchtet. Des Weiteren wurde ein Einblick in einen Teilbereich der Computer Vision gegeben, um zu verstehen, wie die verwendete Gesichtserkennung aufgebaut ist. Nachfolgend wurde eine Einführung zur Spracherkennung gegeben und beschrieben, aus welchen Komponenten ein Spracherkennungssystem zusammengesetzt ist. Abschließend wurden, aufbauend auf den vorherigen Kapiteln, die Anforderungen an die zu implementierende Anwendung definiert.

In Kapitel 3 wurde zunächst die Architektur des Systems entworfen. Dafür wurde erläutert, welche Interaktionsverfahren in der Anwendung eingesetzt werden und welche Komponenten für das System zur Verfügung stehen. Dafür wurden dann Module definiert, welche die einzelnen Aufgaben übernehmen. Es wurde gezeigt, auf welche Weise die Module miteinander kommunizieren und wie Interaktionen eines Benutzers mit dem System ablaufen. Danach wurde die konkrete Implementation der Module auf Basis der

Architektur dokumentiert. Zum Ende wurde eine funktionale Analyse durchgeführt, dafür wurde die Umsetzung mit den in 2.7 genannten Anforderungen verglichen.

Die Evaluation hat gezeigt, dass die Anforderungen größtenteils direkt umgesetzt werden konnten. Es stellte sich jedoch heraus, dass die verwendeten Verfahren zur berührungslosen Interaktion den Smart Mirror an seine Grenzen bringen und es zu Performance Problemen kommen kann.

4.2 Ausblick

Aufbauend auf dieser Arbeit sind einige weitere Funktionalitäten denkbar, die nicht nur auf den verwendeten Smart Mirror beschränkt sind. Teile der Arbeit sind auch für andere Systeme im Smart Home Kontext anwendbar.

Um eine noch natürlichere Interaktion mit dem Smart Mirror zu gewährleisten, könnte eine Gestensteuerung verwendet werden. Da die Kamera des Smart Mirror aber keine hohe Auflösung besitzt, würde man hier wahrscheinlich externe Hardware in Form einer Tiefenkamera oder eines Lidar benötigen. Das Erkennungssystem könnte autark arbeiten, die Gesten erkennen und sie über das Netzwerk an den Smart Mirror senden, der daraufhin Aktionen ausführt. Dies hätte auch einen positiven Einfluss auf die Performance des Smart Mirror, da er die Erkennung nicht selbst durchführen müsste. Zusätzlich wäre es interessant, durch Verwendung von weiteren Datenquellen, sowie einer Auswertung der Daten, eine noch intensivere Nutzung des Smart Mirror zu ermöglichen. Self-Tracking zeichnet aus, dass es in vielen Lebensbereichen angewandt werden kann und je mehr Daten zur Verfügung stehen, desto umfangreicher kann ein Abbild des eigenen Selbst erstellt werden.

Zurzeit werden die Datenquellen von allen Nutzern geteilt, dies sind aktuell die Waage und der Hautfeuchtigkeitssensor. Da es aber auch einige Geräte gibt, die dauerhaft nur von einer Person verwendet werden, beispielsweise eine Smart Watch, kann die Anwendung dahingehend angepasst werden, dass jeder Benutzer seine eigenen Datenquellen verwaltet. Dementsprechend müsste dann auch die Visualisierung individualisierbar sein, damit verschiedene Geräte mit ihren Daten präsentiert werden können und jeder Nutzer sein eigenes Dashboard erhält.

Losgelöst vom Smart Mirror kann das Self-Tracking, die Auswertung der Daten oder die berührungslose Interaktion auch in einem größeren Maße auf das Smart Home oder das

Smart Environment übertragen werden. Das Living Place verfügt über einige Kameras, die in der Wohnung verteilt sind. Damit wäre es zum Einen möglich, eine Person zu tracken und zum Anderen eine Identifikation durchzuführen. Abhängig davon könnten sich dann alle Komponenten der Wohnung komplett auf den Nutzer einstellen, um ihn eine optimale Wohnungserfahrung zu garantieren. Zusätzlich dazu kann eine Sprachsteurung verwendet werden, um auf natürliche Weise mit verschiedenen Komponenten der Wohnung zu interagieren. Durch weitere smarte Geräte, die Self-Tracking Daten bereitstellen, beispielsweise ein smartes Bett, das aufzeichnet, wie gut und erholsam der Schlaf war, könnten mit Data Mining Verfahren die Daten analysiert werden. Die daraus resultierenden Erkenntnisse könnten verwendet werden, um verschiedene Geräte zu steuern, die mit dem Bewohner abhängig von seinem allgemeinen Zustand interagieren. Hat der Bewohner schlecht geschlafen, könnte die Espressomaschine automatisch, nach dem Aufwachen, einen starken Espresso zubereiten.

Für ein Automobil wäre es denkbar, ein Fahrassistenzsystem zu entwickeln, dass über verschiedene Sensoren, unter anderem im Sitz oder im Lenkrad, permanent den Zustand des Fahrers analysiert. Ergänzt um eine Gesichtserkennung, wäre es damit möglich, das Auto individuell an den Fahrer anzupassen und Fehlverhalten im Straßenverkehr durch beispielsweise Müdigkeit vorzubeugen. Das Auto könnte dem Fahrer daraufhin vorschlagen, zur nächsten Raststätte zu fahren, um eine Pause einzulegen.

Literaturverzeichnis

[1] Ahsan, Md M. ; Li, Yueqing ; Zhang, Jing ; Ahad, Md T. ; Gupta, Kishor D.: Evaluating the Performance of Eigenface, Fisherface, and Local Binary Pattern Histogram-Based Facial Recognition Methods under Various Weather Conditions. In: *Technologies* 9 (2021), 04

[2] Anggo, Mustamin ; Arapu, La: Face Recognition Using Fisherface Method. In: *Journal of Physics: Conference Series* 1028 (2018), jun, S. 012119. – URL https://doi.org/10.1088/1742-6596/1028/1/012119

[3] Bianco, Simone ; Celona, Luigi ; Napoletano, Paolo: Visual-based sentiment logging in magic smart mirrors. In: *2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, September 2018, S. 1–4. – ISSN: 2166-6822. – ISSN 2166-6822

[4] BMIBund: *Der Personalausweis mit Online-Ausweisfunktion*. 2021. – URL https://www.bmi.bund.de/DE/themen/moderne-verwaltung/ausweise-und-paesse/personalausweis/personalausweis-node.html

[5] Carbon, Claus-Christian: Face processing: Early processing in the recognition of faces. (2003). – URL https://refubium.fu-berlin.de/handle/fub188/2491. – Zugriffsdatum: 2021-03-27

[6] CH, Kalyani: Various Biometric Authentication Techniques: A Review. In: *Journal of Biometrics & Biostatistics* 08 (2017), 01

[7] Crow, Franklin C.: Summed-Area Tables for Texture Mapping. In: *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA : Association for Computing Machinery, 1984 (SIGGRAPH '84), S. 207–212. – URL https://doi.org/10.1145/800031.808600. – ISBN 0897911385

[8] Dirin, Amir ; Kauttonen, Janne: Comparisons of Facial Recognition Algorithms Through a Case Study Application. In: *International Journal of Interactive Mobile Technologies (iJIM)* 14 (2020), 08, S. 121

[9] Fan, C. ; Peng, Y. ; Cao, C. ; Liu, X. ; Hou, S. ; Chi, J. ; Huang, Y. ; Li, Q. ; He, Z.: GaitPart: Temporal Part-Based Model for Gait Recognition. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Juni 2020, S. 14213–14221. – ISSN: 2575-7075. – ISSN 2575-7075

[10] Flounders, Christine: Speech Recognition as a Mirror. In: *CHI '01 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA : Association for Computing Machinery, 2001 (CHI EA '01), S. 459–460. – URL https://doi.org/10.1145/634067.634332. – ISBN 1581133405

[11] Gallwitz, Florian ; Niemann, Heinrich ; Noeth, Elmar: Spracherkennung — Stand der Technik, Einsatzmöglichkeiten und Perspektiven. In: *Wirtschaftsinformatik* 41 (1999), 12, S. 538–547

[12] Gentile, Antonio ; Santangelo, Antonella ; Sorce, Salvatore ; Vitabile, Salvatore: Novel Human-to-Human Interactions from the Evolution of HCI. In: *2011 International Conference on Complex, Intelligent, and Software Intensive Systems*, Juni 2011, S. 600–605

[13] Huamán, Ana: *Cascade Classifier*. https://docs.opencv.org/4.5.1/db/d28/tutorial_cascade_classifier.html. Dezember 2020. – URL https://docs.opencv.org/4.5.1/db/d28/tutorial_cascade_classifier.html

[14] Huggins Daines, David ; Kumar, M. ; Chan, A. ; Black, A.W. ; Ravishankar, M. ; Rudnicky, Alexander: Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices, 06 2006, S. I – I

[15] Ishii, Hiroshi ; Ullmer, Brygg: Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA : Association for Computing Machinery, 1997 (CHI '97), S. 234–241. – URL https://doi.org/10.1145/258549.258715. – ISBN 0897918029

[16] Kortuem, Gerd ; Kawsar, Fahim ; Sundramoorthy, Vasughi ; Fitton, Daniel: Smart objects as building blocks for the Internet of things. In: *IEEE Internet Computing* 14 (2010), Januar, Nr. 1, S. 44–51. – ISSN 1941-0131

[17] Lienhart, Rainer ; Kuranov, Alexander ; Pisarevsky, Vadim: Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. In: Michaelis, Bernd (Hrsg.) ; Krell, Gerald (Hrsg.): *Pattern Recognition*. Berlin, Heidelberg : Springer, 2003 (Lecture Notes in Computer Science), S. 297–304. – ISBN 9783540452430

[18] Lupton, Deborah: *The Quantified Self*. 1st. Polity Press, 2016. – URL https://dl.acm.org/doi/book/10.5555/3052450. – ISBN 1509500596

[19] Mark Weiser, John B.: *The Coming Age of Calm Technology*. 1996. – URL https://calmtech.com/papers/coming-age-calm-technology.html

[20] Ojala, T. ; Pietikainen, M. ; Harwood, D.: Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. In: *Proceedings of 12th International Conference on Pattern Recognition* Bd. 1, Oktober 1994, S. 582–585 vol.1

[21] OpenCV: *Face Recognition with OpenCV*. Dezember 2019. – URL https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms

[22] Papageorgiou, C.P. ; Oren, M. ; Poggio, T.: A general framework for object detection. In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Januar 1998, S. 555–562

[23] Paravati, Gianluca ; Gatteschi, Valentina: Human-Computer Interaction in Smart Environments. In: *Sensors* 15 (2015), 08, S. 19487–19494

[24] Paul, Sanmoy ; Acharya, Sameer K.: *A Comparative Study on Facial Recognition Algorithms*. Dezember 2020. – URL https://papers.ssrn.com/abstract=3753064. – Zugriffsdatum: 2021-09-13

[25] Penchikala, Srini: *Using Deep Learning Technologies IBM Reaches a New Milestone in Speech Recognition*. 2017. – URL https://www.infoq.com/news/2017/03/ibm-speech-recognition/

[26] Peter Flock, Helge S.: *Spracherkennung*. 2013. – URL https://www.h-brs.de/files/20171215_fbinf_mclab_ss15_spracherkennung_flock_spieker_sa_mk.pdf

[27] Pietikäinen, Matti: Local Binary Patterns. In: *Scholarpedia* 5 (2010), März, Nr. 3, S. 9775. – URL http://www.scholarpedia.org/article/Local_Binary_Patterns. – Zugriffsdatum: 2021-07-02. – ISSN 1941-6016

[28] Poslad, Stefan: *Ubiquitous Computing: SmartDevices, Environments and Interactions Wiley*. 2009. – URL http://www.eecs.qmul.ac.uk/~stefan/ubicom/chapter-summaries-and-keywords.html

[29] Povey, Daniel ; Ghoshal, Arnab ; Boulianne, Gilles ; Burget, Lukáš ; Glembek, Ondrej ; Goel, Nagendra ; Hannemann, Mirko ; Motlíček, Petr ; Qian, Yanmin ; Schwarz, Petr ; Silovský, Jan ; Stemmer, Georg ; Vesel, Karel: The Kaldi speech recognition toolkit. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding* (2011), 01

[30] Prado, Kelvin Salton d.: *Face Recognition: Understanding LBPH Algorithm*. Februar 2018. – URL https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b. – Zugriffsdatum: 2021-03-27

[31] Pulli, Kari ; Baksheev, Anatoly ; Kornyakov, Kirill ; Eruhimov, Victor: Realtime Computer Vision with OpenCV: Mobile Computer-Vision Technology Will Soon Become as Ubiquitous as Touch Interfaces. In: *Queue* 10 (2012), April, Nr. 4, S. 40–56. – URL https://doi.org/10.1145/2181796.2206309. – ISSN 1542-7730

[32] Rakover, Sam S. ; Cahlon, Baruch: A face recognition by similarity (FRBS) conjecture. In: *Perception & Psychophysics* 70 (2008), August, Nr. 6, S. 969–982. – URL https://doi.org/10.3758/PP.70.6.982. – Zugriffsdatum: 2021-04-11. – ISSN 1532-5962

[33] Reid, Vincent M. ; Dunn, Kirsty ; Young, Robert J. ; Amu, Johnson ; Donovan, Tim ; Reissland, Nadja: The Human Fetus Preferentially Engages with Face-like Visual Stimuli. In: *Current Biology* 27 (2017), Juni, Nr. 12, S. 1825–1828.e3. – URL https://www.cell.com/current-biology/abstract/S0960-9822(17)30580-8. – Zugriffsdatum: 2021-04-11. – ISSN 0960-9822

[34] Richter, Paul: *Ein Embedded Systembasiertes Designobjekt als Quantified Self Companion*. 2021

[35] S, Karpagavalli ; Chandra, Evania: A Review on Automatic Speech Recognition Architecture and Approaches. In: *International Journal of Signal Processing, Image Processing and Pattern Recognition* 9 (2016), 04, S. 393–404

[36] Sharifara, Ali ; Rahim, Mohd Shafry M. ; Anisi, Yasaman: A general review of human face detection including a study of neural networks and Haar feature-based cascade classifier in face detection. Kuala Lumpur, Malaysia : IEEE, 2014, S. 73–78. – ISBN 978-1-4799-6443-7

[37] SplendidResearchGmbH: *Studie: Einstellung der Deutschen zum ThemaSelf-Tracking*. https://www.splendid-research.com/de/statistiken/item/studie-einstellung-deutsche-tracking.html. 2019. – URL https://www.splendid-research.com/de/statistiken/item/studie-einstellung-deutsche-tracking.html. – Zugriffsdatum: 13.01.2021

[38] SudhaNarang, MeghaSaxena A.: *Comparison of Face Recognition AlgorithmsUsing Opencv for Attendance System*. 2018. – URL http://www.ijsrp.org/research-paper-0218/ijsrp-p7433.pdf

[39] Thorat, Roopa A. ; Jadhav, Ruchira. A.: Speech Recognition System. In: *Proceedings of the International Conference on Advances in Computing, Communication and Control*. New York, NY, USA : Association for Computing Machinery, 2009 (ICAC3 '09), S. 607–609. – URL https://doi.org/10.1145/1523103.1523226. – ISBN 9781605583518

[40] Turk, Matthew ; Pentland, Alex: Eigenfaces for Recognition. In: *J. Cognitive Neuroscience* 3 (1991), Januar, Nr. 1, S. 71–86. – URL https://doi.org/10.1162/jocn.1991.3.1.71. – ISSN 0898-929X

[41] Viola, P. ; Jones, M.: Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* Bd. 1, Dec 2001, S. I–I. – ISSN 1063-6919

[42] Weiser, Mark: The Computer for the 21st Century. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 3 (1999), Juli, Nr. 3, S. 3–11. – URL https://doi.org/10.1145/329124.329126. – ISSN 1559-1662

[43] Wolf, Gary: *What is The Quantified Self?* https://quantifiedself.com/blog/what-is-the-quantified-self/. 2011. – URL https://quantifiedself.com/blog/what-is-the-quantified-self/. – Zugriffsdatum: 30.12.2020

[44] Wolf, Gary: *What is quantified self?* Oktober 2016. – URL http://qsinstitute.com/about/what-is-quantified-self/. – Zugriffsdatum: 2021-07-26

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

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

Ort	Datum	Unterschrift im Original