

Beat Temperli, Céline Solenthaler,  
Caroline Badoud, Patricia Roth  
08. Dezember 2014

Media Computing (meco)  
Dozent: Frederik Gundelsweiler  
Fachhochschule Nordwestschweiz

# Face Recognition

[Grundidee und Ziele](#)  
[Recherche und Technologien](#)  
[Zeitplan und Rollen](#)  
    [Rollen](#)  
    [Milestones](#)  
    [Aktivitäten](#)  
[Ermittlung der Anforderungen](#)  
    [Anforderungen](#)  
    [Abgeleitete User Stories](#)  
[UI-Design](#)  
    [Ablauf](#)  
    [Prototyp](#)  
        [Erklärungen](#)  
    [Farbpalette](#)  
[Aufbau Systemarchitektur](#)  
    [Sequenzdiagramm](#)  
    [State-Diagramm](#)  
    [Entwicklungsrichtlinien](#)  
[Matching-Funktion](#)  
[Erstellung der Software](#)  
    [Hardware](#)  
    [Software](#)  
    [Server](#)

## 1. Grundidee und Ziele

Die Idee dieses Projektes liegt darin, dass die eigene Front-Kamera vom Laptop einen Gegenstand erkennt und verfolgt. Dabei sollen folgende Ziele verfolgt werden:

- das Programm muss erkennen ob der Gegenstand ein menschliches Gesicht hat
- das Programm muss erkennen ob das erkannte Gesicht mit einem Gesicht in der "Datenbank" übereinstimmt
- das User Interface zeigt das erkannte Gesicht an
- das User Interface zeigt eine Bildübereinstimmung an

Realisiert wird das Face Recognition Projekt mit JavaScript und der API von Beyond Reality Face.

## 2. Recherche und Technologien

Folgende Technologien haben wir analysiert:

### **Faint**

Fertiges Framework, welches OpenCV und Haarclassifier (OpenCV-API) verwendet.

### **OpenCV Face-Recognizer**

OpenCV ist komplex, jedoch sehr gut dokumentiert. Da die Anleitungen jedoch grösstenteils in C++ ist und alle im Team keine Erfahrung haben, haben wir uns aus zeitlichen Gründen gegen OpenCV entschieden.

[http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec\\_api.html](http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec_api.html)

Genauer beschrieben hier:

[http://docs.opencv.org/modules/contrib/doc/facerec/facerec\\_tutorial.html](http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html)

### **Vuforia**

Vuforia ist vor allem bei der Darstellung von 3D Objekte als AR mächtig.

<https://www.qualcomm.com/products/vuforia>

### **OpenCV und Processing**

Dank OpenCV und Processing konnten wir erkennen, ob ein Gesicht im Bild enthalten ist. Die Library ist unter folgendem Link einsehbar:

<https://github.com/atduskgreg/opencv-processing>

## Beyond Reality Face

<https://www.beyond-reality-face.com>

Bietet eine “Trial SDK” an, die Gesichter über eine laufende Webkamera erkennen kann. Einen kleinen Nachteil ist, dass die SDK nicht viele Funktionen anbietet und man bei der Implementation eingeschränkt ist. Ebenfalls ist die SDK nicht wirklich dokumentiert, aber die eigentliche “Einarbeitung” benötigte nicht viel Zeit.

## 3. Zeitplan und Rollen

### 3.1. Rollen

Name	Rolle
Céline Solenthaler	Product Owner / Dev Team
Patricia Roth	Scrum Master / Dev Team
Caroline Badoud	Dev Team
Beat Temperli	Dev Team

### 3.2. Milestones

Datum	Beschreibung
24.11.2014	Erstellung Mock-Up + User Stories
08.12.2014	Erstellung GUI mit Starten der Front-Kamera
15.12.2014	Laufende Face Recognition API
20.01.2015	Erkennung, welches Gesicht es ist

### 3.3. Aktivitäten

Iterativer Prozess mit Trello: <https://trello.com/b/SWCKkEVg/meco2014b5>

Phase	Tasks	Aufwand

1	Dokumentation Grundidee + Ziele, Zeitplan + Milestones, Teamrollen zuteilen, Konzept Technik + Systemarchitektur, Screen-Konzept	2
2	Programmierung GUI gemäss Screen-Konzept, Einbindung Front-Kamera, Front-Kamera starten, testen	1
3	Face Recognition API einbauen, Gesicht wird erkannt, testen	3
4	Bilder von den Gesichtern in einem Ordner ablegen, Programmieren, dass das Gesicht zugeordnet werden kann, testen	5

## 4. Ermittlung der Anforderungen

### 4.1. Anforderungen

Nummer	Beschreibung	Priorität
1.	Das Programm soll erkennen ob es sich beim erkannten Gegenstand um ein Gesicht handelt.	hoch
2.	Wenn kein Gesicht erkannt werden kann, soll eine Meldung ausgegeben werden.	niedrig
3.	Wenn sich mehrere Gesichter in der Webcam befinden, dann muss das Programm nur ein Gesicht erkennen.	hoch
4.	Es wird ein Bild von einer erkannten Person in einem Ordner abgelegt.	hoch
4.1.	Es werden Bilder von den zu erkannten Personen in einem Ordner abgelegt.	niedrig
4.2.	Das Programm muss maximal vier Personen erkennen.	niedrig
5.	Die Gesichter sollen nur aus der Frontalsicht erkannt werden.	hoch
6.	Der Abgleich mit der Datenbank muss über einen Knopf gestartet werden.	niedrig
6.1.	Der Abgleich mit der Datenbank kann zu jeder Zeit abgebrochen werden.	niedrig
6.2.	Der Anwender kann die Erkennungsfunktionalität mehrmals	niedrig

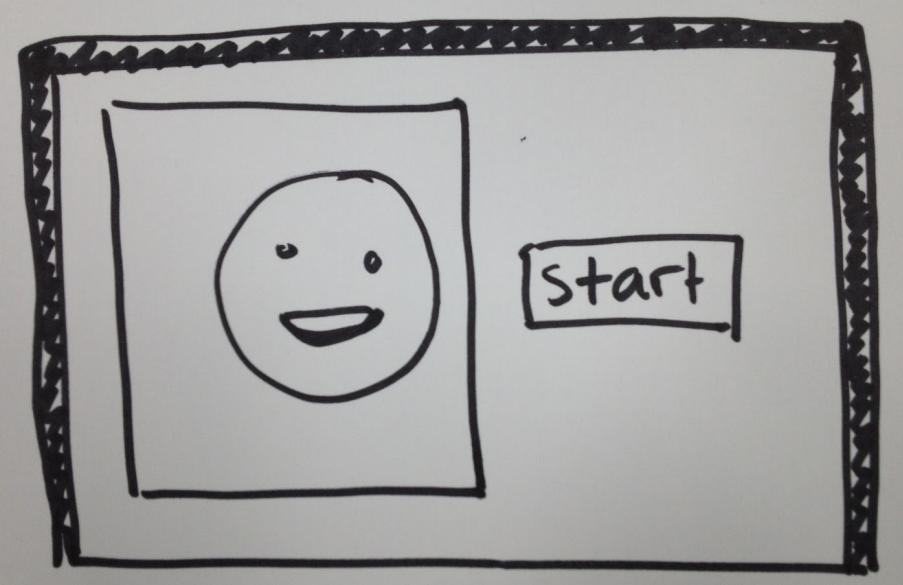
	nacheinander verwenden.	
9.	Auf dem User Interface sind alle im Ordner abgelegten Bilder ersichtlich.	hoch
9.1.	Wenn es einen "Match" gibt, dann wird das entsprechende Bild markiert (Zusatz zu Anforderung 5).	hoch
9.2.	Das Programm gibt, sobald er ein Gesicht zuordnen konnte, den Vornamen und Nachnamen aus.	hoch
9.2.1	Im Bildernamen steht jeweils der Vorname und Nachname derjenigen Person.	hoch
10.	Das Programm muss nur auf modernen Browser laufen.	hoch
11.	Die Personen dürfen auf den Bildern sowieso bei laufender Kamera nicht lachen/lächeln.	niedrig

## 4.2. Abgeleitete User Stories

ID	User story	Priorität
1	Als Kunde möchte ich, dass das Programm mein Gesicht erkennt, damit ich anschliessend die Gesichtserkennung laufen lassen kann.	1
2	Als Kunde möchte ich, dass das System eine Meldung ausgibt, falls in der Webcam kein Gesicht erkannt werden kann, damit der Anwender laufend informiert ist.	3
3	Als Kunde möchte ich, dass das Programm nur ein Gesicht erkennt, damit die Applikation einfach und verständlich bleibt.	1
4	Als Kunde möchte ich, dass ein Foto von mir abgelegt werden kann, damit ich persönlich erkannt werde.	1
5	Als Kunde möchte ich, dass das Programm den Vorname und Nachname der erkannten Person anzeigt, damit ich sehe ob das "Matching" erfolgreich war.	1
6	Als Kunde möchte ich, dass mein Gesicht aus der Frontalsicht erkannt wird, damit das Programm die Mindestanforderungen erfüllt.	3
7	Als Kunde möchte ich einen Knopf auf dem User Interface haben, damit ich die Erkennungsfunktionalität selbst starten kann.	3

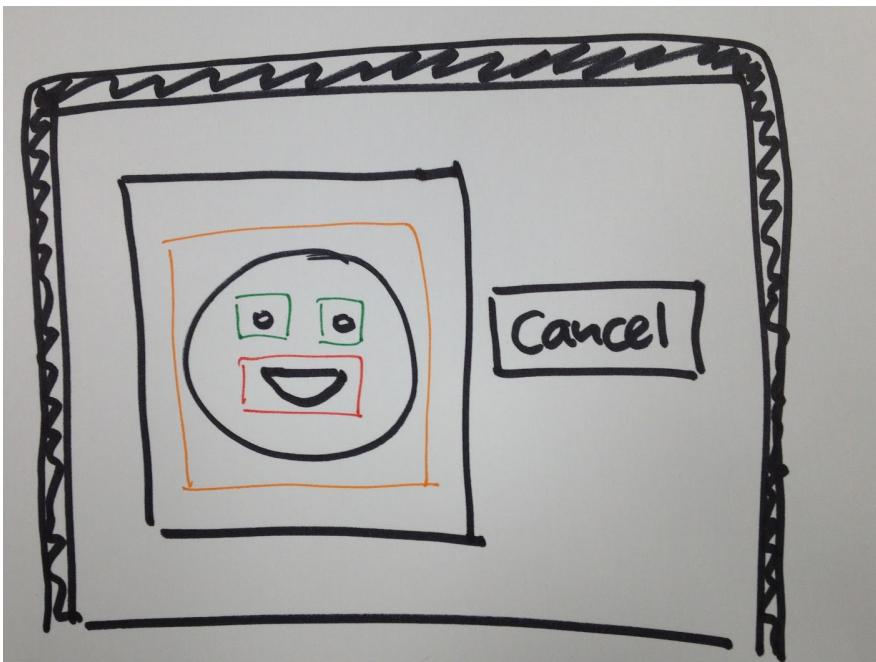
8	Als Kunde möchte ich einen Knopf auf dem User Interface haben, damit ich die Erkennungsfunktionalität zu jeder Zeit beenden kann.	3
9	Als Kunde möchte ich die Erkennungsfunktionalität mehrmals nacheinander abspielen können, damit verschiedene Gesichter nacheinander erkannt werden können.	3
10	Als Kunde möchte ich alle abgelegte Bilder in der UI sehen können, damit ich weiss, wieviele es sind.	3
11	Als Kunde möchte ich dass das Bild der erkannte Person markiert wird, damit ich weiss, dass die Erkennung funktioniert hat.	3

## 5. UI-Design



**Screen 1**

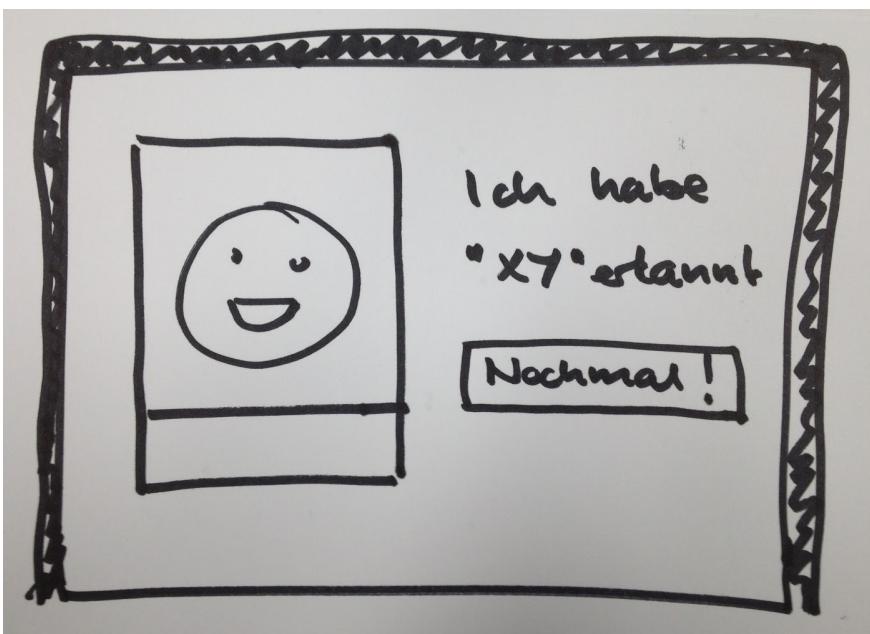
Das Kamerabild erscheint und daneben ein Startknopf, mit dem die Gesichtserkennung gestartet werden kann.



### Screen 2

Nachdem die Gesichtserkennung gestartet wurde, "tastet" das Programm das Kamerabild ab, um herauszufinden, ob darauf ein Gesicht zu erkennen ist und, wenn ja, welches.

Neben dem Kamerabild erscheint ein Cancel-Button. Wird dieser gedrückt, wird die Gesichtserkennung abgebrochen und es erscheint wieder Screen 1.



### Screen 3

Wurde ein Gesicht erkannt, erscheint links das Foto aus der Datenbank und rechts davon erscheint der Text "Ich habe "XY" erkannt.

Drückt man den Button darunter, erscheint wieder Screen 1.

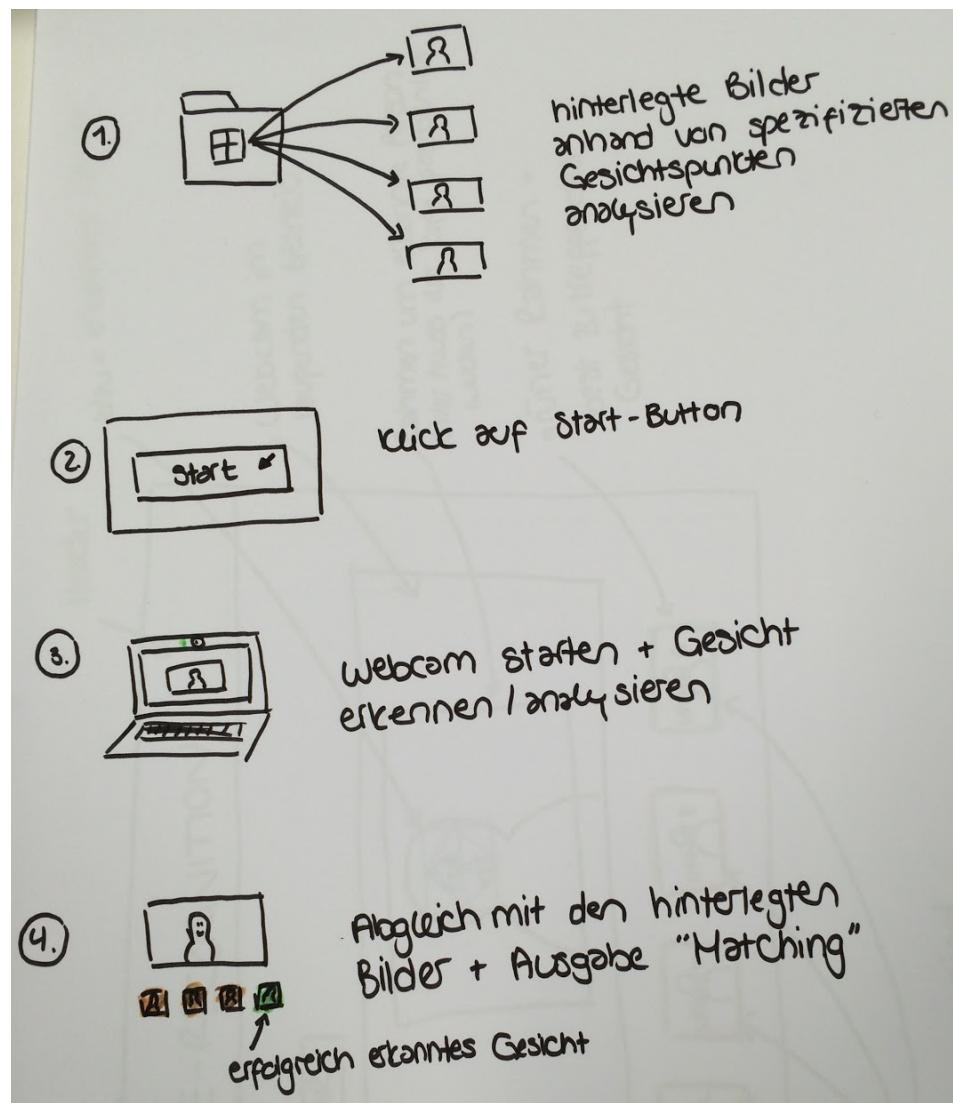


#### Screen 4

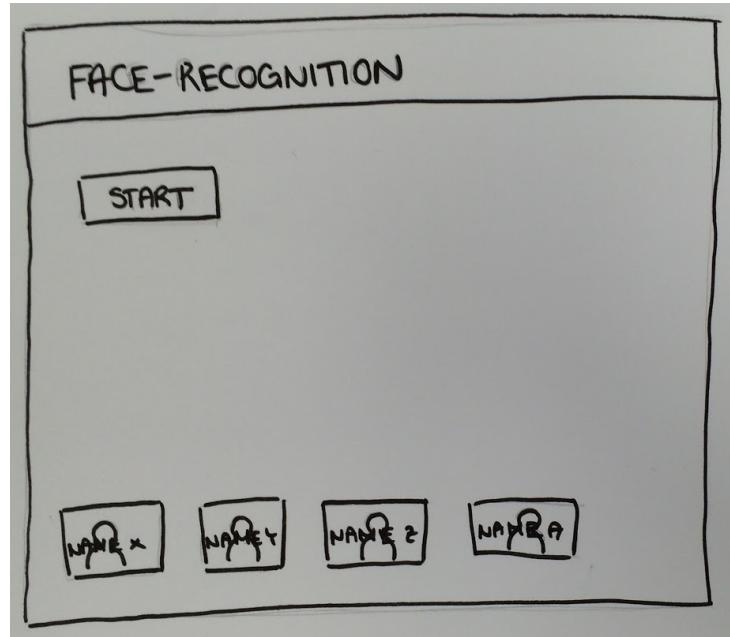
Wenn das Programm kein Gesicht erkennen konnte, erscheint die Meldung "Ich habe leider niemanden erkannt". Drückt man den Button darunter, erscheint wieder Screen 1.

#### 5.1. Ablauf

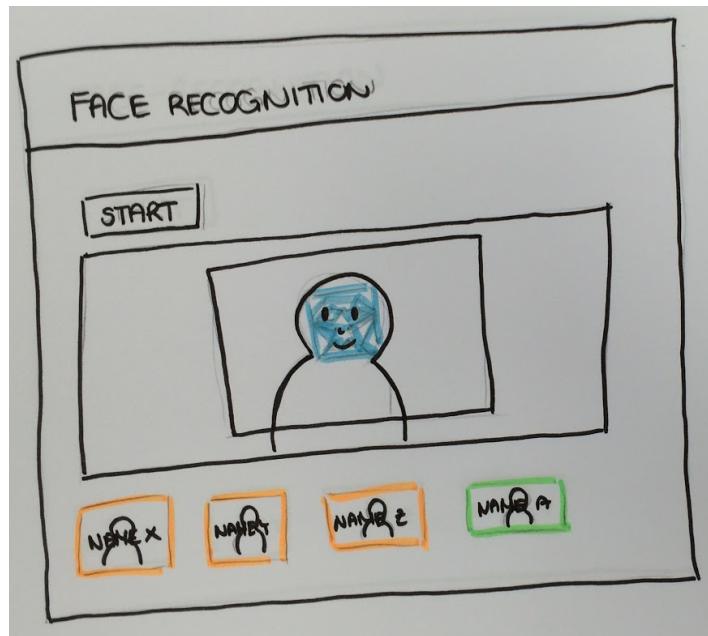
Das folgende Bild stellt den Ablauf dar, welcher vom Programm durchlaufen wird, sobald dieses gestartet wird.



## 5.2. Prototyp

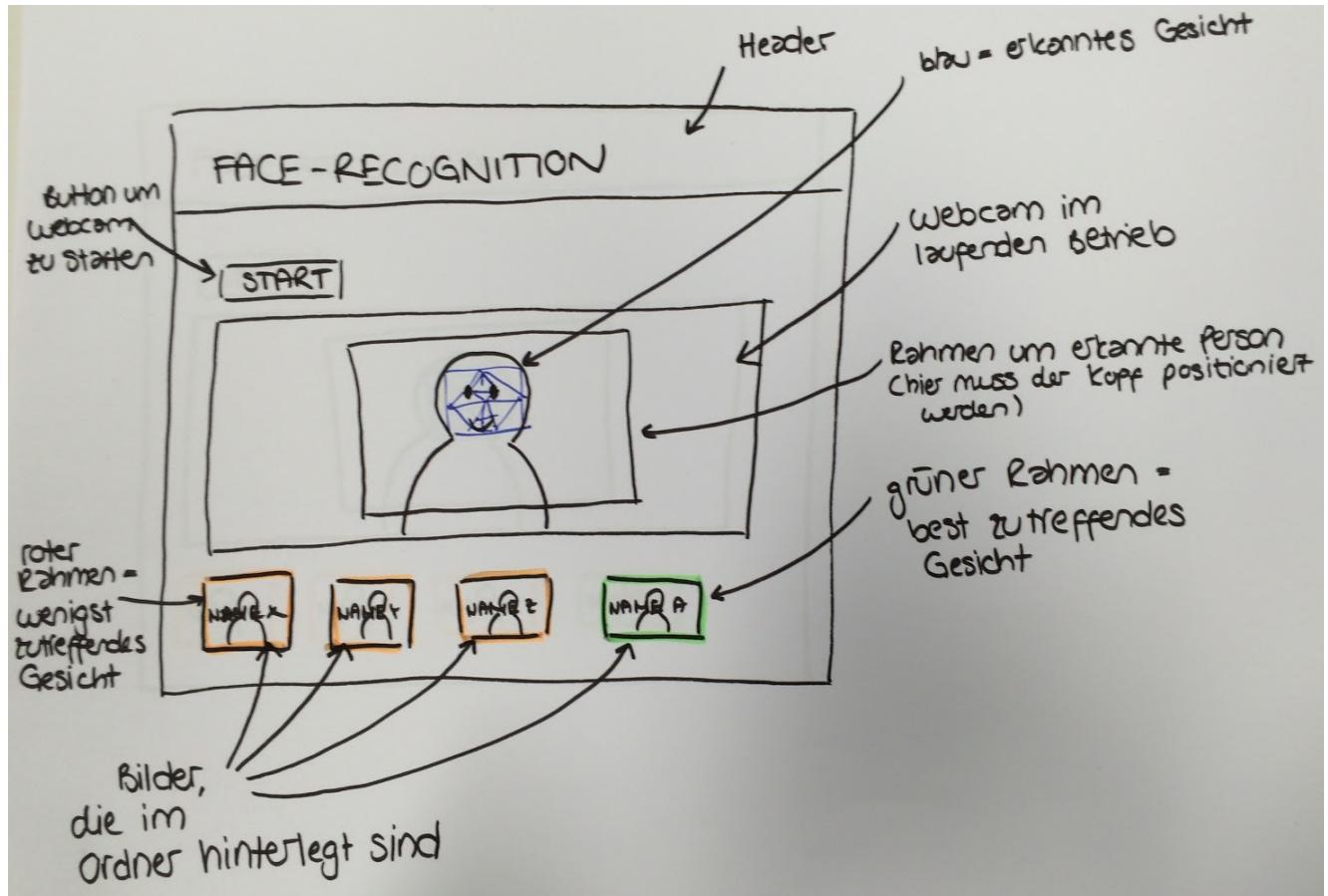


erster Screen wenn Webapplikation gestartet wird



folgender Screen wenn auf Start geklickt wird und Gesicht analysiert wird

### 5.2.1. Erklärungen

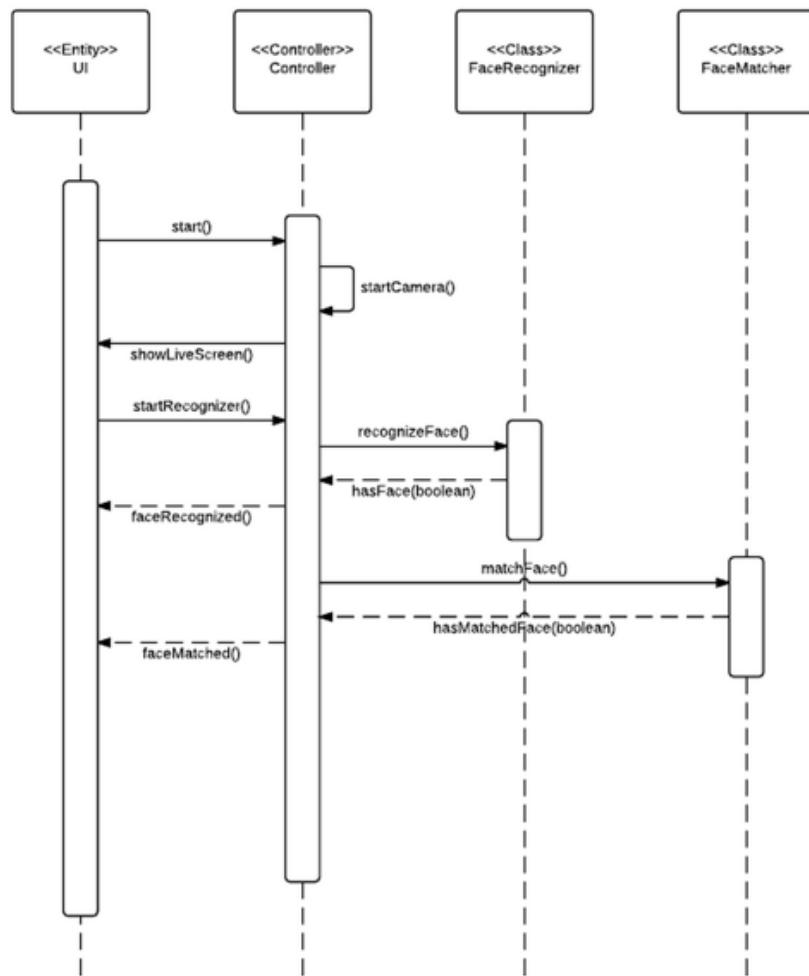


### 5.3. Farbpalette

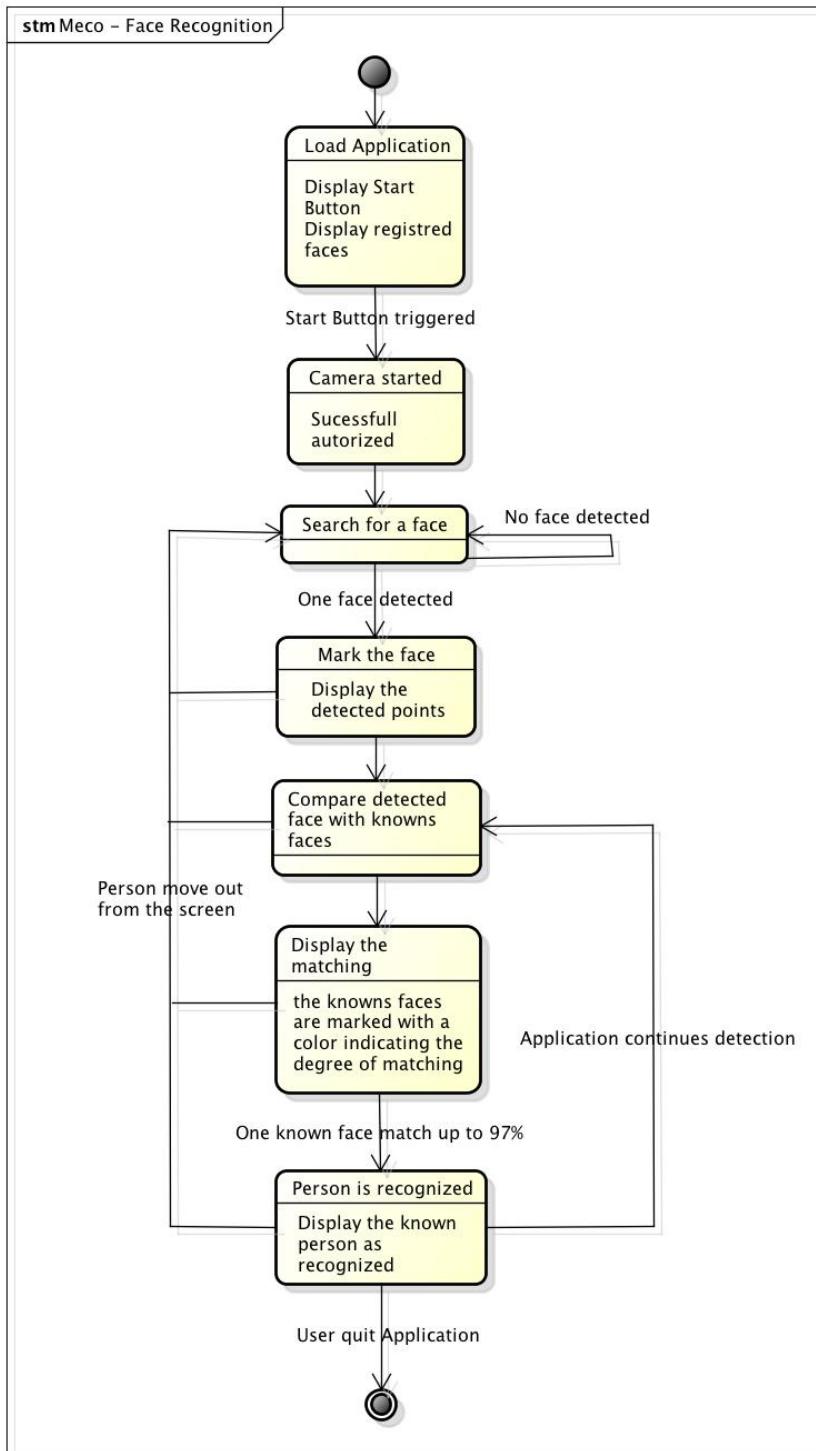


## 6. Aufbau Systemarchitektur

### 6.1. Sequenzdiagramm



## 6.2. State-Diagramm



### **6.3. Entwicklungsrichtlinien**

Für das Projekt wird an sich keine Datenbank verwendet. Es wird ein Ordner mit verschiedenen Bildern der zu erkennenden Personen hinterlegt. Dabei soll im Bildernamen immer den Vor- und Nachname hinterlegt werden.

Nach folgenden Regeln sollen die Bilder im Ordner hinterlegt werden:  
person\_vorname\_nachname.jpg

## **7. Matching-Funktion**

Als erster Schnitt versucht das Programm das Gesicht im Webcam-Stream zu erkennen. Das Programm zeigt im Webcam-Stream daraufhin ein Rechteck an, das das gefundene Gesicht markiert.

Im zweiten Schritt analysiert das Programm das erkannte Gesicht und verwendet dazu eine Form mit individuellen Gesichtszugpunkten, die vordefiniert sind. Die wichtigsten Gesichtspunkten sind dabei die Augen, die Nase sowie das Maul. Ebenfalls wird die Position, Rotation und Skala des erkannten Gesichts abgeschätzt. So kann auch ein sich bewegendes Gesicht erkannt werden.

In einem weiteren Schritt rechnet das Programm die Distanz zwischen den festgelegten Gesichtspunkten des erkannten Gesichts aus. Vor allem die Distanzen zwischen den einzelnen Gesichtspunkten zu den Augen werden berücksichtigt. Daraufhin werden die Distanzen der Gesichtspunkten der hinterlegten Datenbank-Bilder analysiert und mit den Daten aus dem Webcam-Stream verglichen.

Im letzten Schritt werden im User Interface die Datenbank-Bilder farblich gekennzeichnet. Wenn ein Bild grün umrandet ist, dann gibt es eine hohe Übereinstimmung mit dem Gesicht aus dem Webcam-Stream. Rot bedeutet gar keine Übereinstimmung. Sobald das Matching bis und mit 3 Prozent abweicht wird der Name der erkannten Person ausgegeben.

## **8. Erstellung der Software**

Das Projekt ist unter folgendem Github-Link abgelegt:  
[https://github.com/btemperli/fhnw\\_meco\\_face-recognition](https://github.com/btemperli/fhnw_meco_face-recognition)

### **8.1. Hardware**

Die Anforderungen an die Hardware ist ein moderner Laptop oder Computer mit einer Webcam, die mit einem aktuellen Treiber läuft.

### **8.2. Software**

Die Anforderung an die Software ist, dass sie auf einem modernen Laptop läuft. Veraltete Browser werden nicht unterstützt.

Aufgrund der Recherche von dem zweiten Kapitel, haben wir uns dazu entschieden, dass wir als Basis die SDK von Beyond Reality Face benutzen.

### **8.3. Server**

Als Server wird Apache2 verwendet.