Smart Mirror

Baris,Tikir, Leon Dodrimong

20. Februar 2020

Inhaltsverzeichnis

1 Grundidee **3**
1.1 Spiegel mit integriertem Display 3
1.2 Konzept Benutzerfreundlichkeit 3

2 Aufbau **4**
2.1 Strukturierung 4
2.2 Materialien 4

3 Software **4**
3.1 Prerequisites 4
3.2 Frontend - SmartMirrorWeb 4
3.3 Backend - SmartMirror.WebApi 5

4 Erweiterungen **6**
4.1 API's 6
4.2 Hardware 6

5 Über uns **6**

6 Anhang **6**
6.1 Verweise 6

1 Grundidee

Wie wir auf die Idee gekommen sind... Auf die eigentliche Idee ist Barsi Tikir eines morgens gekommen als er im Bad stand. Während er sich für den Tag fertig machte und auf seinem Handy noch seine passsende Bahnverbindung raus suchte, dachte er darüber nach, wie praktisch es wäre, wenn man seine Bahnverbindungen nicht mühsam im Handy über die BVG App suchte müsste, sondern diese im irgendwie direkt präsent wäre. Als er daruf-hin in den Spiegel blickte kam im die Idee.. ein Spiegel der seine Bahnverbindung anzeigen könnte. Morgens In den Spigel gucken, beim Zähneputzen, Haare machen, SChminken, usw.... Warum nicht die Zeit auch geich nutzen für ein kleines Update, was in der Welt gerade so passiert oder wann der nächste Bus zur Arbeit fährt. Nach kurzer Recherche fand er auch eine passende Bauanleitung für einen solchen Spiegel. Jedoch gab es neben dem Zusammenbau noch das Problem etwas sinnvolles auf dem Display anzuzeigen. kleinere Projekt mit Uhrzeit und Wetter gab es bereits, als Beispiele zum nach-programmieren.
Ein paar Wochen später stieß er auf den Paulaward. Als wir uns trafen, erzählte Baris über die Idee und den Paul Award. Zusammen haben wir dann ein Konzept entwickelt, welches möglichst viele Infomationen aus verschiedenen Bereichen anzeigen kann. Da es jedoch schon sehr viele Dienste, wie zum Beispiel Google Kalender, ToDoIst, BVG App und so weiter gibt, wollte wir nicht einen weiteren doppelten Dienst entwickeln, sonder auf diese Zugreifen, sodass wir die Daten, welche wir hinter diesen Diensten bereits abgelegt hatten, nutzen können. So entstand die Gesamtidee vom Smart-Mirror.

1.1 Spiegel mit integriertem Display

Ein Spiegel mit integriertem Display ist wohl nichts neues. Die Idee ist einfach hinter einem Einwegspiegelglass ein Display zu montieren, sodass dieses durch das Glass durch scheint und man dies auf der anderen Seite des Spiegels sehen kann. Von der Anderen Seite wirkt das Galss spiegelnd, wodurch es ganz Normal als Spiegel genutzt werden kann.

1.2 Konzept Benutzerfreundlichkeit

Die eigentliche Idee dabei ist den Spiegel nicht einfach nur spiegeln zu lassen oder die Uhrzeit anzeigen zu lassen, sondern ihn ßmartßu machen, sodass man ihn praktischen und effizient Nutzen kann. Man geht morgens Zudem war uns auch wichtig, dass es **Benutzerfreundlich** ist, da nicht jeder das Know-How hat sich einen Spiegel für seine Eigenen Bedrüfnisse zusammen zu bauen bzw. zu programmieren.

2 Aufbau

2.1 Strukturierung

Notizen: Dagramme und erklärungen der Funktionsweise

2.2 Materialien

Der Spiegel besteht aus einem Einwegspiegelglass, welches von einer Seite spiegelt und von der anderen Seite reflektiert. Außerdem haben wir einen Holzrahmen, welcher aus Fassung für das Spiegelglass dient, genutzt. Für die Technik haben wir ein Display für die Anzeigen , ein RasperryPi für die Steuerung und die nötigen Verbindungskabel, wie Spannungsversorgung und Videokabel (HDMI) zur Übertragung der Videosignals zum Display, eingesetzt.
Für die optinalen Erweiterungen würde man je nachdem welches Feature gewünscht ist, noch eine Picamera (für Facerecognition), RasperryPi Bewegungssensor (Bewegungserkennung)[1], Gesture Sensor (Gestiksteuerung)[2] oder ein kleines Mikrophone zur Sprachsteuerung [3]

Notizen: Später eintragen welches Raspi Modell, Display, Glass verwendet wurde

3 Software

3.1 Prerequisites

- Node.js (Javascript runtime)
- Angular
- IDE (Visual Studio Code)
- Datenbank ()

3.2 Frontend - SmartMirrorWeb

Wir haben unser Frontend mithilfe von Angular 9. Für uns war dies eine Neue Erfahrung, da wir noch nie zuvor damit gearbeitet haben. Das Fron-

[1] *Raspberry Pi Infrarot Bewegungsmelder:* https://www.reichelt.de/raspberry-pi-infrarot-bewegungsmelder-hc-sr501-rpi-hc-sr501-p224216.html?&nbc=1

[2] *3D Gesture Tracking Shield for Raspberry Pi:* http://wiki.seeedstudio.com/3D-Gesture-Tracking-Shield-for-Raspberry-Pi-MGC3130/

[3] *Ansteckmikrofon über Klinke:* https://www.amazon.de/dp/B073GJQKL1/ref=psdc_1384055031_t1_B07WQFNVVQ

tend soll sich lediglich um die Anzeige der Daten kümmern und folgende Funktionen übernehmen:

- Daten vom Backend (SmartMirror.WebApi) anfragen
  - alle möglichen Dienste (Widgets)
  - vom User angemeldeten Dienste
- Daten senden
  - Dienst aktivieren/ deaktivieren
  - Benutzereingaben zur Erstellung eines neuen Benutzers
- User zur Anmeldung von weiteren externen Diensten, zur jeweiligen Website weiterleiten[4]

3.3 Backend - SmartMirror.WebApi

Das Backend besteht aus einer eigens Entwickelten Web API, welche zum einen die Anfragen und Daten vom Frontend (SmartMirror.Web) entgegennimmt und bearbeitet, und zum anderen den Datenaustausch mit der eigenen Datenbank und Kommunikation mit den externen API kommuniziert.
Das Backend besteht zum einen Client-Server. Zum einen stellt er als Server eine eigene API dar, welche vom Frontend genutzt wird. Zum anderen fungiert dieser auch als Client und nutzt die externen API Schnittstellen von Drittanbietern. Besitzt dieser eine extra Komponente, welche auf die interne Datenbank zugreift

Client Funktion Da der Smart Mirror möglichst viele externe Features verbinden soll, muss die Schnittstelle zu diesen gut strukturiert werden. Daher haben wir uns dazu entschieden, dass jede externe Kommunikation ihre eigene Komponente bekommet, welche bestimmte Standards (Interfaces) bedient. Somit können auch später leicht neue externe API's eingebunden werden, indem die Kommunikation in einer neuen Komponente implementiert wird. Dort müssen dann nur die vorgegebenen Interfaces definiert sein und dann anschließend bekannt gemacht werden, indem diese in eine interne Liste von allen möglichen externen Diensten eingetragen wird.

Server Funktion

[4] zum Beispiel: wenn der User den neuen Dienst Google Kalender für sich registrieren möchte, muss er sich auf der Website von Google Kalender anmelden um sich zu zertifizieren. Diese sendet dann die zur Authentifizierung notwendigen Credentials, welche vom Backend gespeichert werden müssen

Kommunikation zur Datenbank

4 Erweiterungen

4.1 API's

- Google Kalender
- Todoist
- BVG
- Wetter

4.2 Hardware

- Face Recognition
- Voice Control
- Bewegungssensor

5 Über uns

Wir sind zwei Studenten, Baris Tikir und Leon Dodrimong, von der HTW - Hochschule für Technik und Wirtschaft Berlin aus dem Fachbereich der Ingenieurwissenschaften und Technik.

6 Anhang

6.1 Verweise