

Projekt SBB-Reader

M242

Dokumentation

von Gökmen Erdem, Teodora Todorovic, Justin Feliz

03.04.2022

Inhalt

1	Informieren.....	2
1.1	Übersicht & Projektbeschreibung	2
1.1	Analyse Auftrag	5
1.1.1	Ausgangssituation	5
1.1.2	Aufgabenstellung.....	5
1.1.3	Umsetzung.....	5
2	Projektstruktur und Aufbau.....	5
2.1	GitHub Repository:	5
2.2	Mbed-Studios	5
2.3	Swisspass	6
2.4	Netzwerk-Anbindung.....	6
2.5	XAMPP	6
	MySQL-Datenbank.....	7
	Flask.....	7
3	Umsetzung des Projekts	8
3.1	HTML	8
3.1.1	Index-Seite.....	8
3.1.2	About-Seite.....	9
3.1.3	Profile-Seite	9
3.2	Request.py.....	10
3.2.1	POST-Methode	11
3.2.2	GET-Methode	11
3.3	Main.cpp.....	12
4	Reflexionen.....	15
4.1	Reflexion Teodora Todorovic	15
4.2	Reflexion Gökmen Kaan Erdem.....	15
4.3	Reflexion Justin Raúl Feliz.....	16

1 Informieren

1.1 Übersicht & Projektbeschreibung

Projekttitel:	Altersberechnung
Projektart:	IoT-Kit mit Clouddienst verbinden.

Projektleiter/in:	Teodora Todorovic
Projektauftraggeber/in:	Marcel Bernet
Projektkunde(n):	Offene Kundschaft
Projektdauer:	Geplanter Beginn: 17.04.2022 Geplantes Ende: 03.04.2022
Ausgangssituation / Problembeschreibung:	Wir haben den Auftrag erhalten auf einem IoT-Kit eine Cloudbasiertes Programm zu entwickeln. Gearbeitet wird auf einem IoT-Kit und programmiert wird auf MBED-Studio.
Projektgesamtziel:	Das Ziel ist eine Applikation auf dem IoT-Kit zu programmieren. Die Applikation funktioniert und ist umfangreich. Die Verbindung zwischen der Applikation, dem Notebook und dem IoT-Kit läuft einwandfrei. Die Funktion des Kartenreaders funktioniert und gibt die richtigen Daten zurück.

Projektteilziele und -ergebnisse:	Teilziele:	Ergebnisse:
	Verbindung zwischen Notebook & IoT-Kit Frontend Backend Funktion Kartenreader	
Projektorganisation:	Kernteam: <ul style="list-style-type: none"> • Justin Feliz • Gökmen Erdem • Teodora Todorovic Sonstige Projektbeteiligte: <ul style="list-style-type: none"> • Marcel Bernet (Auftraggeber) 	
Projektressourcen:	Ressourcen & Menge:	
	Personal: 3 Entwicklungsumgebung: siehe Entwicklungsumgebung	
Projektbudget:	0 CHF	
Projektrisiken und -unsicherheiten:	- Zeitmangel & Unregelmässige Produktivität	

1.1 Analyse Auftrag

1.1.1 Ausgangssituation

Wir haben den Auftrag erhalten auf einem IoT-Kit eine Cloudbasiertes Programm zu entwickeln. Gearbeitet wird auf einem IoT-Kit und programmiert wird auf MBED-Studio.

1.1.2 Aufgabenstellung

Setzen Sie ein Projekt für LB2 auf. Dieses soll das IoTKit mit einem (Cloud) Dienst, Ihrer Wahl, verbinden. -> Synchrone Kommunikation.

Nach dem Verbinden des IoT Kits mit dem Notebook und kompilieren des ersten Programmes, eignet sich als Startpunkt das HTTP/REST Kapitel.

1.1.3 Umsetzung

Die Umsetzung erfolgt im Team. Jeder hat seine Aufgaben, welche abgeschlossen werden. Die Umsetzung erfolgt auf MBED-Studio. Wir haben





2 Projektstruktur und Aufbau

Damit das Projekt einwandfrei funktioniert, werden einige Tools etc. gebraucht, die zuerst lokal ausgeführt und vorhanden sein müssen.

Das Projekt liegt grundsätzlich auf einem Github Repository und ist öffentlich. Mit dem IoTKitV3 kann eine Karte, z.B bei uns der Swisspass eingelesen werden und wird im Backend mittels C++ in Mbed-studios verarbeitet. Die Daten, bei uns die CardID, werden via HTTP-Protokoll weitergesendet an ein Endpoint (POST-Methode). Mit Flask, einem Python Webframework, werden diese Daten aus dem erzeugten JSON-Objekt genommen und an die eine MySQL Datenbank weitergesendet. Mittels einer GET-Methode in Flask werden die CardID's aus der Datenbank ausgelesen und auf der entsprechenden Webseite angezeigt.

2.1 GitHub Repository:

https://github.com/goekmenerd/m242_webwedo

 http	Project M242	1 hour ago
 request	Project M242	1 hour ago
 web	Project M242	1 hour ago
 README.md	Initial commit	1 hour ago

Hier befinden sich alle benötigten Files.

Mit dem «git clone https://github.com/goekmenerd/m242_webwedo.git» Befehl via Git CMD, Bash etc. kann jeder User den aktuellen Stand des Repository bei sich lokal ausführen und speichern.

2.2 Mbed-Studios

<https://os.mbed.com/studio/>

Mit der Entwicklungsumgebung «Mbed-Studios» können wir unser IoTKitV3 ganz einfach handhaben und verschiedene Einrichtungen vornehmen. Daher wird für die Ausführung des IoTKit-Backends dieses Tool benötigt.

Nachdem das Repository im Mbed-Studios geöffnet wurde, muss geschaut werden, dass das IoTKitV3 wirklich angeschlossen ist an das jeweilige Gerät und das korrekte «Target» bzw. Gerät, oben links im Mbed-Studios ausgewählt wurde.

WICHTIG: Ebenfalls werden verschiedene Librarys benötigt, die allenfalls manuell noch heruntergeladen werden müssen. Unter «View» im Mbed-Studios Menü kann man ganz einfach das Feld «Librarys» anwählen und sieht dann unten auf der Seite, was genau noch heruntergeladen oder geupdatet werden muss.

2.3 Swissspass

Wir haben uns überlegt, dass wir den Swissspass einlesen und Daten, wie Vorname, Gültigkeitsdatum etc. dann anzeigen auf der Webseite. Da diese Daten jedoch sensibel sind und explizit geschützt werden vom Bund, ist uns die Ausgabe der Medien-ID nur möglich.

Mit dem «RFID-Reader», welches mit dem IoTKit dabei ist, kann dann der Swissspass nun eingelesen werden. (siehe Bild rechts)



2.4 Netzwerk-Anbindung

Im Repository gibt es im Ordner «http» ein File mit dem Namen «mbed_app.json».

Die Zeilen 5 und 9 müssen, wenn sich der User nicht im gleichen Netzwerk befindet, angepasst werden.

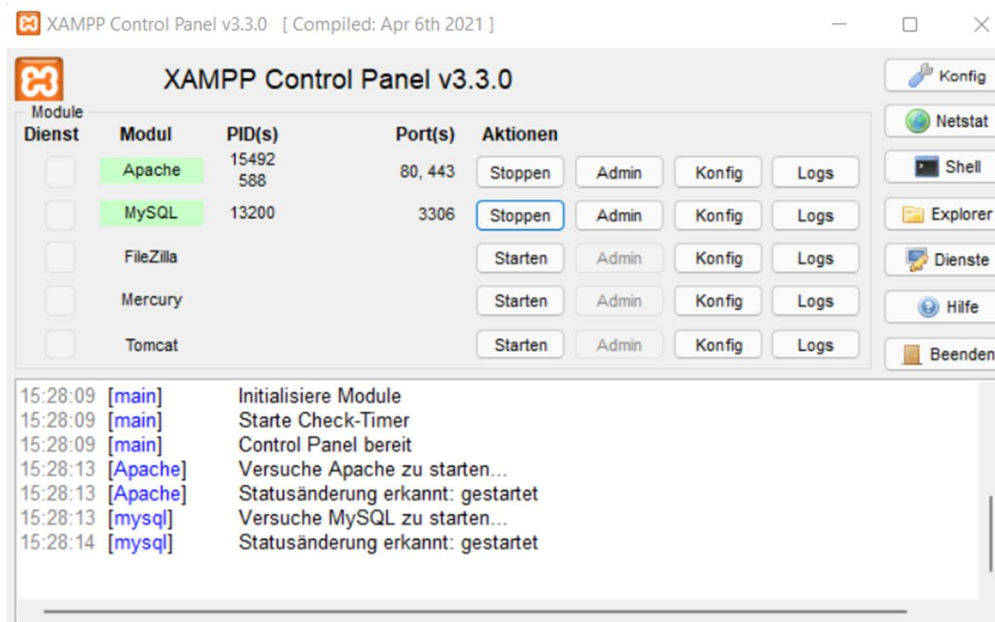
Bei der Zeile 5 wird der Namen des Netzwerks benötigt.

Bei der Zeile 9 wird das Passwort für das entsprechende Netzwerk benötigt.

```
1  {
2      "config": {
3          "wifi-ssid": {
4              "help": "WiFi SSID",
5              "value": "\"LERNKUBE\""
6          },
7          "wifi-password": {
8              "help": "WiFi Password",
9              "value": "\"l3rnk4b3\""
10         }
11     }
```

2.5 XAMPP

Apache & MySQL Port Verwaltung mit XAMPP. Damit der Port für den Localhost aktiviert ist.



MySQL-Datenbank

Mit phpMyAdmin wird eine MySQL Datenbank benötigt. Im Repository unter «web» und «model» kann der MySQL-Code für die Datenbank-Tabellen-Erstellung benutzt werden.

Zuvor wird jedoch eine Datenbank benötigt, die manuell auf phpMyAdmin mit dem Namen «sbb_profile_data» und «utf8_general_ci», erstellt werden kann.

Flask

Unsere API und Webseiten werden mit Flask verwaltet und angesprochen. Alle Endpoints werden dort zugewiesen. Die Daten, die in einem JSON-Objekt per http vom IoT-Kit zugeschickt werden, werden hier mit demselben End-Point und der POST-Methode wieder aufgenommen und verarbeitet. Mit PyCharm (Entwicklungsumgebung) wird der Code ausgeführt.

Link PyCharm: <https://www.jetbrains.com/de-de/pycharm/>

3 Umsetzung des Projekts

3.1 HTML

Für das Frontend unserer Applikation haben wir mit HTML ein schönes Frontend mit einer «Home», «About» und «Profile» Page erstellt. In der «About» Page wird unser Team vorgestellt, welches an dem Projekt gearbeitet hat. In «Home» wird die Funktion der Website und des Projekts erklärt und bei der «Profile» Page erfolgt das Lesen eines Swisspass mit dem Ausspucken von Informationen Anhand des Swisspass.



3.1.1 Index-Seite

Die Index-Seite, bzw. die Start-Seite bietet einen kurzen Überblick auf das Projekt, die Funktion der Website mit einer Beschreibung. Diese und die zwei anderen Seiten wurden mit HTML entwickelt und mit CSS designed.

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="css/style.css">
  <title>LB02 SBB Profile reader</title>
</head>
<body>
<header>
  <a class="ribbon" tabindex="-1" title="zurück zur Startseite!" href="index.html">
    <h1 id="logo">SBB Profil reader</h1>
    <p>WebWeDo LB02 - TBZ</p>
  </a>
  
</header>
<main>
  <nav id="navigation">
    <ul>
      <li><a aria-current="page" class="nava" href="index.html">Home</a></li>
      <li><a aria-current="page" class="nava" href="view/profile/profile.html">Mein Profil</a></li>
      <li><a aria-current="page" class="nava" href="view/about/about.html">Über uns</a></li>
    </ul>
  </nav>
  <section id="intro">
    <h2>Willkommen <span>bei unserem SBB Profile reader von WebWeDo</span></h2>
  </section>
  <section class="spalte">
    <h2>Was du bei uns findest</h2>
```


3.1.2 About-Seite

Die About-Seite beschreibt unser Team, liefert zusätzlich Informationen über das Projekt mit einer kleinen Beschreibung wie es umgesetzt wurde.

```
<header>
  <a class="ribbon" tabindex="-1" title="zurück zur Startseite!" href="../../index.html">
    <h1 id="logo">SBB Profil reader</h1>
    <p>WebWeDo LB02 - TBZ</p>
  </a>
  
</header>

<main>

  <nav id="navigation">
    <ul>
      <li><a aria-current="page" class="nava" href="../../index.html">Home</a></li>
      <li><a aria-current="page" class="nava" href="../../profile/profile.html">Mein Profil</a></li>
      <li><a aria-current="page" class="nava" href="about.html">Über uns</a></li>
    </ul>
  </nav>

  <section id="intro">
    <h2>Über uns <span>erfährst du hier alles Mögliche.</span></h2>
    <div class="box_about">
      <p align="justify">Wir sind drei Informatiker Applikationsentwicklung EFZ Lernende in der Schweiz in Zürich.
        Unsere Firma "WebWeDo" hat ebenfalls seinen Sitz in Zürich und wird aktuell von <b>Teodora Todorovic</b> geleitet.
        Wichtiger Bestandteil des Teams sind ebenfalls <b>Gökmen Kaan Erdem und Justin Raul Feliz</b>, die das Team schlussendlich
        <p align="justify">Der SBB Profile reader von WebWeDo wird am <b>01.04.2022</b> offiziell produktiv gestellt.
        Verschiedene User, also ihr, könnt dann die Seite nutzen und von den vielen guten und nützlichen Features profitieren.
      <p align="justify">Nutzung: Mit dem IotKit kannst du deinen Swisspass einlesen. Deine Daten werden dann genommen und bei uns si
        Diese Daten werden anschliessend auf der übersichtlichen <a href="../../profile/profile.html">Mein Profil</a> Seite angezeigt
        Du siehst nebst all deinen möglichen Swisspass-Daten zusätzlich die, aus den Daten berechnete, Gültigkeitsdauer.
        Auf der Seite kannst du auch noch einige andere Elemente nutzen, wie die Ausgabe verschiedene nützlicher Abos von der SBB
      <p align="justify"><b>Am besten erkundigst du dich selbst und stöberst durch die Seite! Los gehts!</b></p>
    </div>
  </section>
</main>
```

3.1.3 Profile-Seite

Die Profile-Seite beinhaltet die Funktion, welche programmiert wurde. Hier werden die Daten ausgegeben, welche durch den Kartenreader empfangen werden.

```
<header>
  <a class="ribbon" tabindex="-1" title="zurück zur Startseite!" href="../../index.html">
    <h1 id="logo">SBB Profil reader</h1>
    <p>WebWeDo LB02 - TBZ</p>
  </a>
  
</header>

<main>

  <nav id="navigation">
    <ul>
      <li><a aria-current="page" class="nava" href="../../index.html">Home</a></li>
      <li><a aria-current="page" class="nava" href="../../profile/profile.html">Mein Profil</a></li>
      <li><a aria-current="page" class="nava" href="../../about/about.html">Über uns</a></li>
    </ul>
  </nav>

  <section id="intro">
    <h2>Hier deine ID <span>von deiner Karte</span></h2>
    {{ get_id }}
  </section>
</main>

<footer class="grid">
  <ul id="footer-nav">
    <li><a href="https://data.sbb.ch/api/v1/console" title="SBB API">SBB API</a></li>
    <li><a href="../../controller/contacts/contacts.html" title="">Kontakt</a></li>
  </ul>
</footer>
```

Mit {{ get_id }} werden die gesammelten Daten vom Kartenreader ausgegeben.

3.2 Request.py

Das File „request.py“ hat zwei Funktionen, welche beide mit der Datenübergabe oder Datenempfang des Kartenreaders zu tun hat.

```
from flask import Flask, request
import requests
import json

from flask_mysql import MySQL

app = Flask(__name__)

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'sbb_profile_data'

mysql = MySQL(app)

@app.route('/data', methods=["POST"])
def post():
    data = request.get_json()
    cur = mysql.connection.cursor()
    try:
        cur.execute(
            "INSERT INTO tbl_sbb_data (cardid) VALUES (" + str(data["CardId"]) + ")"
        )
    except:
        print("Error")
    mysql.connection.commit()
    cur.close()

    return str(cur.lastrowid)

@app.route('/data', methods=["GET"])
def get():
    cur = mysql.connection.cursor()
    cur.execute("SELECT cardid FROM tbl_sbb_data")
    rv = cur.fetchall()
    response = []
```

3.2.1 POST-Methode

Die Daten, welche über den Kartenreader empfangen werden, müssen an einen http-Endpoint und danach in eine Datenbank gespeichert werden. Dafür haben wir eine Methode implementiert, welche die Daten, welche vom Kartenreader geschickt werden in die Datenbank einfügt.

```
@app.route('/data', methods=["POST"])
def post():
    data = request.get_json()
    cur = mysql.connection.cursor()
    try:
        cur.execute(
            "INSERT INTO tbl_sbb_data (cardid) VALUES (" + str(data["CardId"]) + ")"
        )
    except:
        print("Error")
    mysql.connection.commit()
    cur.close()

    return str(cur.lastrowid)
```

3.2.2 GET-Methode

Wenn Daten gebraucht werden, müssen von der Datenbank die richtigen Daten gezogen werden, deshalb haben wir eine GET-Methode entwickelt, welche von der Datenbank die Daten holt und weitergibt.

```
@app.route('/data', methods=["GET"])
def get():
    cur = mysql.connection.cursor()
    cur.execute("SELECT cardid FROM tbl_sbb_data")
    rv = cur.fetchall()
    response = []
    for row in rv:
        response.append({"CardId": row[0]})
    return app.response_class(
        response=json.dumps(response),
        status=200,
        mimetype='application/json'
    )
```

3.3 Main.cpp

Im „Main.cpp“ werden die Funktionen implementiert, welche auf das IoT-Kit bezogen sind.

Diese Funktion ruft das UI des IoT-Kits auf, also das kleine Display welches auf dem Kit zu sehen ist.

```
// UI
OLEDDisplay oled( MBED_CONF_IOTKIT_OLED_RST, MBED_CONF_IOTKIT_OLED_SDA, MBED_CONF_IOTKIT_OLED_SCL );
```

Diese Funktion ruft den RFID-Reader auf, der in unserer Applikation für das Lesen des Swisspass gebraucht wird.

```
// NFC/RFID Reader (SPI)
MFRC522 rfidReader( MBED_CONF_IOTKIT_RFID_MOSI, MBED_CONF_IOTKIT_RFID_MISO, MBED_CONF_IOTKIT_RFID_SCLK, MBED_CONF_IOTKIT_RFID_SS, MBED_CONF_IOTKIT_RFID_RST );
```

Hier wird das Display resettet und „RFID“ ausgegeben.

```
int main()
{
    // OLED Display
    oled.clear();
    oled.printf( "RFID\n" );
```

Da unsere Applikation eine API verwendet, musste eine WiFi-Verbindung hergestellt werden. Es werden die Zugangsdaten aus dem File „mbed_app.json“ aufgerufen, kann keine Verbindung hergestellt werden, kommt eine Fehlermeldung.

```
// Connect to the network with the default networking interface
// if you use WiFi: see mbed_app.json for the credentials
WiFiInterface* network = WiFiInterface::get_default_instance();
if (!network) {
    printf("ERROR: No WiFiInterface found.\n");
    return -1;
}
printf("\nConnecting to %s...\n", MBED_CONF_APP_WIFI_SSID);
int ret = network->connect(MBED_CONF_APP_WIFI_SSID, MBED_CONF_APP_WIFI_PASSWORD, NSAPI_SECURITY_WPA_WPA2);
if (ret != 0) {
    printf("\nConnection error: %d\n", ret);
    return -1;
}
```

```
"config": {
    "wifi-ssid": {
        "help": "WiFi SSID",
        "value": "\"LERNKUBE\""
    },
    "wifi-password": {
        "help": "WiFi Password",
        "value": "\"l3rnk4b3\""
    }
}
```

Wenn die Verbindung erfolgreich hergestellt wurde, wird mit dieser Funktion eine MAC- und IP-Adresse vergeben.

```
printf("Success\n\n");
printf("MAC: %s\n", network->get_mac_address());
SocketAddress a;
network->get_ip_address(&a);
printf("IP: %s\n", a.get_ip_address());
int count = 0;
```

Hier verwenden wir die Vorlage von Herrn Bernet, um den RFID-Chip der Karte zu lesen.

```
while ( 1 )
{
    // RFID Reader
    if ( rfidReader.PICC_IsNewCardPresent()
        if ( rfidReader.PICC_ReadCardSerial()
            {
                count = count + 1;
                char body[1024];
                oled.cursor( 1, 0 );
                // Print Card UID (2-stellig mit Vornullen, Hexadecimal)
                string card_id = "";

                oled.printf("UID: ");
                for ( int i = 0; i < rfidReader.uid.size; i++ ){
                    oled.printf("%02X:", rfidReader.uid.uidByte[i]);
                    card_id = card_id + std::to_string(rfidReader.uid.uidByte[i]) + ":";
                }
                oled.printf("\r\n");

                HttpRequest* post_req = new HttpRequest( network, HTTP_POST, "http://192.168.1.114:5000/data");
                post_req->set_header("Content-Type", "application/json");

                sprintf( body, "{ \"CardId\": \"%s\" }", card_id.c_str());
                HttpResponse* post_res = post_req->send(body, strlen(body));

                // Print Card type
                int piccType = rfidReader.PICC_GetType(rfidReader.uid.sak);
                oled.printf("PICC Type: %s \r\n", rfidReader.PICC_GetTypeName(piccType) );
                printf("Card Count: %d \r\n", count);
            }
        thread_sleep_for( 200 );
}
```

In diesem Abschnitt des Codes musste die URL abgeändert werden für den HTTP-Request. Hier werden die Daten der Karte, die eingelesen wird via HTTP-Post Request in einem JSON Objekt abgespeichert und an die entsprechende URL gesendet.

```
HttpRequest* post_req = new HttpRequest( network, HTTP_POST, "http://192.168.1.114:5000/data");
post_req->set_header("Content-Type", "application/json");
```

Da wird der JSON mit der CardID erzeugt und als http-Request weitergesendet.

```
sprintf( body, "{ \"CardId\": \"%s\" }", card_id.c_str());  
HttpResponse* post_res = post_req->send(body, strlen(body));
```

3.4 Known issues

In diesem Kapitel werden bekannte Probleme und Fehler genauer beschrieben und analysiert.

Unsere Idee am Anfang war es, dass Daten aus der SBB App auf unserer Webseite von den einzelnen User angezeigt werden. Da wir zu spät gemerkt haben, dass die SBB ihre Daten schützt und wir diese daher nicht auslesen können. Eine andere Möglichkeit wäre gewesen, dass wir die ID der User auslesen und diese auf der Webseite anzeigen. Wir haben uns Hilfe geholt und viel nach gefragt bei anderen Gruppen, doch keiner konnte sich erklären, wieso es nicht funktioniert. Das Auslesen in die Datenbank und Webseite der Daten funktioniert daher in unserem Projekt nicht.

4 Reflexionen

4.1 Reflexion Teodora Todorovic

Ich wusste am Anfang vom Modul nicht was mich erwarten wird. Ich habe bisher noch nie etwas mit IoT zu tun gehabt, genau so wenig mit IoTKit. Ich habe vieles gelernt in diesem Projekt und kann einiges davon sicher im nächsten umsetzen.

Ich hatte kein Vorwissen in diesem Projekt (bis auf wie man eine Webseite erstellt und diese mit einer Datenbank verbindet), daher war der Einstieg schwer und wir hatten nicht so viel Zeit. Ich habe jetzt ein grösseres Wissen darüber, was IoT ist und wie damit umgegangen wird, sowie wie ein IoTKit funktioniert und wie gebrauch davon gemacht werden kann. All das wusste ich von einem Monat nicht. Ich bin ohne Vorwissen in dieses Modul gestartet.

Mein persönlicher Wissensstand zum IoT ist gewachsen, wie bereits erwähnt. Ich weiss jetzt, dass IoT dort eingesetzt wird, wo ein physisches Gerät mit dem Internet verbunden wird. Ich kenne mich ebenfalls besser mit den einzelnen Komponenten ums IoTKit aus wie zum Beispiel den Sensoren (diese haben viele Gruppen für die Temperaturmessungen gebraucht), Aktoren, Service, etc. Das Gerät wurde mir bekannt und ich habe gelernt, dass viel Verschiedenes damit getätigt werden kann.

4.2 Reflexion Gökmen Kaan Erdem

Erste Eindrücke zum Thema

In diesem Projekt war die Voraussetzung und das Hauptziel, dass wir uns mit dem Oberbegriff „IoT“ oder auf Deutsch auch „Internet der Dinge“ auseinandersetzen und dies gleichzeitig unsere Grundlage für das LB02 Projekt ist. Der Begriff „IoT“ kam mir zwar bekannt vor, da sich viele Unternehmen und Bereiche heutzutage mit diesem Thema auch auseinandersetzen und verschiedene Produkte nach diesem Prinzip und Verfahren herstellen. Jedoch hatte ich selbst nie die Erfahrung in diesem Bereich gemacht. Sprich, die Umsetzung, der Prozess und die Elemente, die gebraucht und eingesetzt werden, waren mir alle unbekannt.

Start in das Modul

Das Modul startete sehr gut. Herr Bernet hat uns von Anfang an aufgezeigt, was die Grundlagen für die späteren Ausführungen sind und welche Tools etc. benötigt werden. Ich konnte mich daher dem entsprechend orientieren und meine Umgebung frühzeitig einrichten. Dass wir mit einem physischen Gerät, dem IoTKitV3, auch wirklich arbeiten durften, war ein kleines Highlight für mich. Dadurch recherchierte ich auch noch zuhause, was es sonst noch für Alternativen gibt und wo diese eingesetzt werden. Herr Bernet hat uns neben seinen Inputs in den Lektionen auch immer wieder viele nützliche Beispiele und Code-Elemente zur Verfügung gestellt, die wir schlussendlich auch in unserem Projekt wiederverwendet haben. Diese Beispiele halfen mir bei der Umsetzung des IoTKit-Backends auch sehr, da ich an vielen Stellen nur noch logisch überleben musste und wenig Anpassung nötig war. Ich konnte mich daher auf das Wesentliche fokussieren im Projekt und verlor nicht unnötig Zeit.

Erweiterung meines Wissensstands

Wenn ich nun zurückschaue, kann ich von mir selbst sagen, dass ich sehr viel Nützliches, Neues und Wichtiges mitnehmen konnte aus diesem Projekt/Modul. Ich verstehe nun nicht nur den Nutzen einer solchen Umgebung, sondern kann vieles im Backend auch umsetzen und nachvollziehen. Die Cloud-Anbindung mittels HTTP, das Ein- und Auslesen verschiedener Daten mit dem IoTKit etc. kann ich nun selbst umsetzen und angehen. Ich habe auch erfahren, dass durch diese Beziehungen und

Abhängigkeiten von Cloud, Backend etc. viele verschiedene Sprachen, Services, Elemente etc. zum Einsatz kommen. Daher fand ich auch die Aufteilung der Arbeiten innerhalb unserer Gruppe ideal.

Verbesserungsbedarf:

Im Grossen und Ganzen finde ich meine Weiterentwicklung in den letzten paar Wochen sehr gut. Einige Stellen und Anforderungen konnten wir in diesem Projekt nicht vollständig bearbeiten. Zum Beispiel gelang uns das Auslesen der Swisspass-Daten nicht so, wie wir uns das vorgestellt hatten. Ich denke, dass wir bei offenen Fragen lieber früher oder überhaupt zu Herr Bernet hätten gehen sollen, da er uns diese Möglichkeiten auch immer wieder anbot. Ich habe sehr oft versucht, selbst z.B. im Backend zu verstehen, wie etwas funktionieren muss, auch wenn es mir manchmal nicht so gut gelang. Das nächste Mal werde ich daher diese gegebene Chance von Herr Bernet nutzen, um Fehler und Probleme beim Endprodukt zu meiden.

4.3 Reflexion Justin Raúl Feliz

Modulstart und Erfahrung:

Das Modul 242 hat mir gezeigt, wie es auf der Welt immer wichtiger wird, physische Geräte und Maschinen vom Internet oder einem Netzwerk abhängig zu machen. Der Oberbegriff „IoT“ stand in diesem Modul im Vordergrund. In unserem Projekt mussten wir uns mit einigen neuen Services, Elementen, Umgebungen und Programmiersprachen auseinandersetzen. Vor diesem Modul hatte ich keine Projekte oder Arbeiten, in denen ich Komponenten wie z.B. das IoTKitV3 benutzen musste.

Individuelle Arbeit:

Bei dieser Arbeit beschäftigte ich mich unter anderem mit den API's (Flask), der Datenbank, die schlussendlich nur aus einer Tabelle besteht, und dem Aufbau und Inhalt der Dokumentation. Ich empfand die Gruppenarbeit als sehr angenehm und entspannt, da wir uns in der Konstellation bereits in vorherigen Modulen und Projekten wiedergefunden hatten, und daher auch wussten, wie wir zusammenarbeiten können.

Prozess Entwicklung:

In den Lektionen von Herr Bernet konnte ich viel Neues mitnehmen, wie z.B. die Datenübertragung via HTTP etc. In diesem Modul habe ich auch gelernt, wie wichtig es ist, dass eine Struktur-Planung im Vorhinein vorhanden ist, damit das Gerüst auch funktioniert und das Projekt auch einfach erweiterbar bleibt. Ich denke, dass ich mich sehr gut in diesem Bereich weiterentwickeln konnte durch diese LB02 Arbeit, da ich, wie bereits erwähnt, bisher den Begriff „IoT“ nur aus den Medien und der Aussenwelt kannte und diesmal selbst die Architektur und den Hintergrund bearbeiten musste.

Verbesserungspotenzial:

Schlussendlich finde ich, dass wir als Gruppe sehr gut gearbeitet haben, die Kommunikation stets vorhanden war, und sich mein Wissensstand positiv entwickelt hat. Ich hätte jedoch von Anfang an 100% geben sollen, damit ich am Ende nicht zu diesem Stress komme.