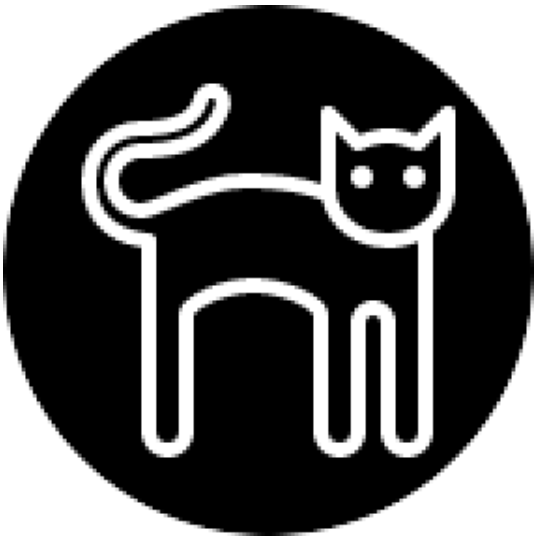

Katzenfütterungsanlage

HTBLA Kaindorf an der Sulm
Grazer Straße 202, A-8430 Kaindorf an der Sulm
Ausbildungsschwerpunkt Mechatronik und Automatisierungstechnik

Florian Greistorfer, Florian Harrer, Dominik Pichler, Julian Wolf

Abgabedatum: 05.04.2018

Betreut von:
Dipl.-Ing. Manfred Steiner
Dipl.Ing. Dr. Gerhard Pretterhofer
Otto

Inhaltsverzeichnis

KAPITEL 1

Webserver und Client

1.1 Begriffserklärungen

1.1.1 Server

Ein Computer oder Programm, der oder das Zugriff auf eine Resource oder einen Dienst in einem Netzwerk ermöglicht

1.1.2 Client

Ein Computer oder Programm, der oder das auf einen Server zugreift

1.2 Anforderungen

1.2.1 Webserver

Auf der Katzenfütterungsanlage läuft ein Webserver, der es ermöglicht, dass der Benutzer das Gerät über das Internet erreichen kann. Hauptaufgaben des Servers sind dabei, Daten bereitzustellen, zu verabeiten und zu speichern und den Webclient zur Verfügung zu stellen.

1.2.2 Client

Der Client soll dem Benutzer ermöglichen, die Katzenfütterungsanlage über einen Webbrowser zu steuern. Ein Benutzername und ein Passwort sind erforderlich, damit man das Gerät bedienen kann. Das Design soll eindeutig und übersichtlich gehalten sein. Auf der Startseite sollen die eingestellten Fütterungszeiten zu sehen sein und eine allgemeine Übersicht. Über eine Navigationsleiste sollen die weiteren Seiten erreichbar sein:

- Fütterrungszeiten
- Positionsinfo
- Geräteinfo
- Update

1.3 Voruntersuchung

1.3.1 JavaScript

JavaScript ist die Sprache des Internets. Jeder herkömmliche Browser ist in der Lage, JavaScript auszuführen. Mit JavaScript ist es möglich, das Aussehen einer Webseite während der Laufzeit zu ändern, Dinge zu entfernen, hinzufügen und animieren. Javascript ist eine objektorientierte Sprache. Es hebt sich von anderen Sprachen vorallem dadurch ab, dass die Datentypen von Variable nicht fix sind. Das bedeutet, wenn eine Variable den Datentyp *number* hat und es wird ein String angehängt, verändert sich der Datentyp automatisch auf *string*.

1.3.2 Node.js

Node.js ist eine Laufzeitumgebung, die es ermöglicht, dass Javascript direkt auf einem Rechner ausgeführt werden kann. Node.js kommt mit dem Node Package Manager (npm). Mithilfe diesem Tools ist es möglich, Module zu installieren, updaten, löschen und veröffentlichen. Diese Module werden im Ordner */node_modules* installiert und in der Datei *package.json* unter *dependencies*, oder mit der option `--save-dev` unter *dev-dependencies* eingetragen. Ein neues Projekt erstellt man mit `npm init`. Dieses Tool erstellt die Datei *package.json*, in der alle Abhängigkeiten und Informationen über das Projekt stehen. Wenn man ein Projekt kopiert, braucht man den */node_modules* Ordner nicht mit kopieren. Man muss nur im Zielordner einmal `npm install` aufrufen.

1.3.3 TypeScript

TypeScript ist eine, von Microsoft entwickelte, Weiterentwicklung von JavaScript. Das bedeutet, jeder gültige JavaScript Code ist auch ein gültiger TypeScript Code. TypeScript wird vom TypeScript Compiler in sauberes JavaScript übersetzt. TypeScript ist sehr gut für größere Anwendungen geeignet. Typescript hat strenge Datentypen, Klassen und Vererbung. Die Datentypen von TypeScript sind:

- **String**: eine Unicode codierte Zeichenkette
- **Number**: eine vorzeichenbehaftete Gleitkommazahl, kann auch hexadezimal, octal oder binär sein
- **Boolean**: true oder false
- **Array**: eine Liste von Elementen des gleichen Datentyps
- **Tuple**: eine Liste von Elementen unterschiedlichen Datentyps deren Anzahl bekannt ist
- **Enum**: eine Möglichkeit, numerischen Werten Namen zu geben
- **Any**: Datentyp unbekannt, wird behandelt wie in JavaScript
- **Void**: kein Datentyp, meist Rückgabewert bei Funktionen
- **Null**: leerer Wert, kann allen anderen zugewiesen werden
- **Undefined**: kein Wert, kann allen anderen zugewiesen werden
- **Never**: wenn ein Wert niemals auftreten kann z.B. eine Funktion die immer einen Fehler produziert

1.3.4 express

Express ist ein Javascript Modul, dass auf dem Node.js Modul *http* bzw *https* aufbaut. In diesen Modulen ist bereits alles enthalten, dass benötigt wird, um einen Webserver zu programmieren. Express nimmt uns die meiste Arbeit ab und bietet viele weitere Möglichkeiten.

1.3.5 JSON

JavaScript Object Notation (JSON) ist die Textrepresentation eines JavaScript Objekts. Die möglichen Datentypen sind:

- **string**: 0 oder mehrere unicode Zeichen innerhalb Doppelhochkommas
- **boolean**: true oder false
- **number**: Eine vorzeichenbehaftete Zahl, die auch die E Notation unterstützt z.B. 0.2E4 (=2000)
- **Array**: Eine geordnete Liste von 0 oder mehreren Werten innerhalb viereckigen Klammern, Elemente sind getrennt durch Kommas.
- **Object**: Eine ungeordnete Sammlung von Name-Wert-Paaren, wo die Namen, die auch Keys genannt werden, Strings sind; Jeder Key sollte eindeutig sein; Innerhalb geschwungener Klammern; Paare sind durch Komma getrennt
- **null**: Ein leerer Wert

```
{
  "Object": {
    "string": "name",
    "number": 10E5,
    "boolean": true,
    "Array": [
      {
        "string": "wert",
        "number": 1
      },
      {
        "string": "wert",
        "number": 2
      }
    ]
  }
}
```

Listing 1.1: JSON Beispiel

1.3.6 JSON Web Token

Ein JSON Web Token (JWT)

1.3.7 MongoDB

MongoDB ist eine schemenlose Datenbank. Schemenlos bedeutet, dass die Datenbank, im vergleich zu schemenbehafteten Datenbanken, keine klare Strukturierung benötigt. Einer schemenlosen Datenbank kann man einfach Daten geben und wieder abfragen. Eine schemenbehaftete Datenbank ist in Zeilen und Spalten unterteilt. Diese müssen vorher feststehen. Da Raspian, das Betriebssystem vom Raspberry Pi, nur 32 Bit ist und MongoDB ab Version 3 nur mehr in 64 Bit erhältlich ist, mussten wir auf eine ältere Version wechseln. Die Verbindung der MongoDB Datenbank erfolgt über einen Driver. Der Driver muss mit der Datenbankversion übereinstimmen. MongoDB ist ein Database management System (DBS), das bedeutet, ein Server läuft auf dem Port 27017, über den alle Datenbanken im System erreichbar sind z.B. die Datenbank *fuettr* ist über `localhost :27017/ fuettr` erreichbar. Eine Gruppe von Daten nennt man Collection. Zugriff auf die Datenbank erfolgt serverseitig wie folgt:

```
1 const dbServer = await mongodb.MongoClient.connect(url);
```

Listing 1.2: Verbinden mit dem DBS

```
1 const dbFuettr = await dbServer.db('fuettr');
```

Listing 1.3: Auswählen der Datenbank

```
1 const collTimes = await dbFuettr.collection('data_times');
```

Listing 1.4: Auswählen der Collection

```
1 const Times = await this._times.find({ identifier: 'Times' }).toArray();
```

Listing 1.5: Auslesen aller Datensätze mit einem Identifier

```
1 this._times.updateOne({ identifier: 'Times' }, { $set: times });
```

Listing 1.6: Überschreiben eines Datensatzes mit einem Identifier

1.3.8 Angular 2/4

Angular ist ein TypeScript Framework, das aus dem Javascript-Framework AngularJS weiterentwickelt wurde. Es wird von Google entwickelt. Angular ist gegliedert in Module. Die grobe Struktur wird in der Abbildung 1.1 dargestellt.

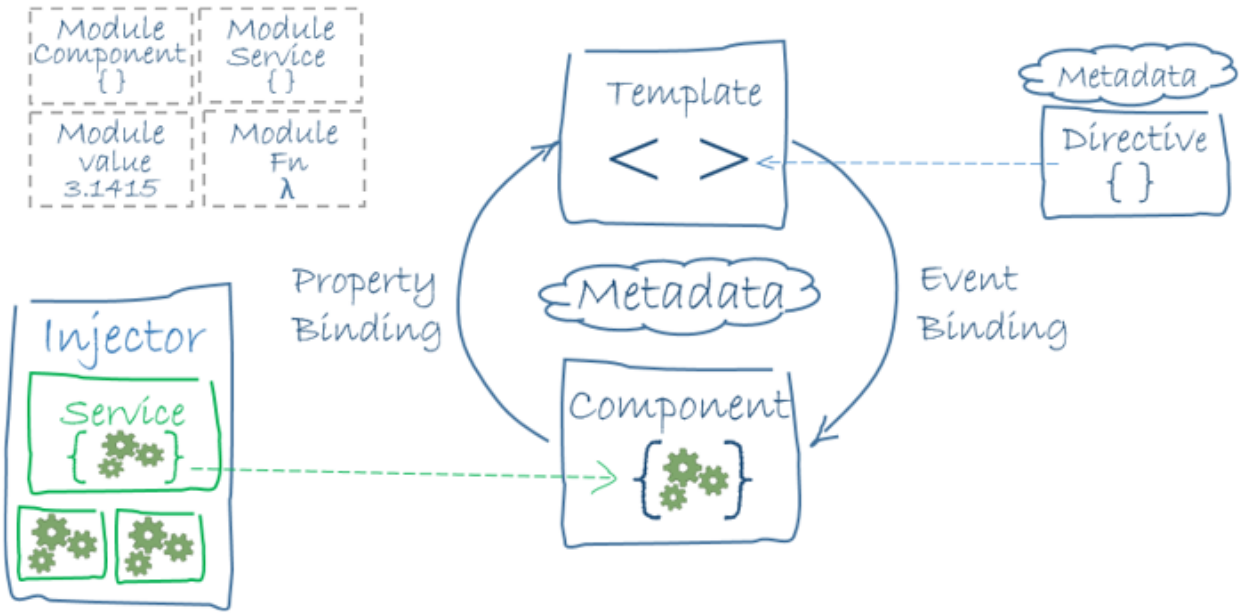

Abbildung 1.1: Angular Struktur

1.3.8.1 Modules

Jede Angular App ist in *Module*s gegliedert. *Module*s fassen meist ähnliche Funktionen zusammen. Jede App muss mindestens ein *Module* enthalten. Dies heißt standartmäßig *AppModule*. Ein *Module* ist die größte Einheit einer Angular App. Ein *Module* kann folgende Komponenten beinhalten:

- Components
- Services
-

1.3.8.2 Libraries

Eine Angular Library ist ein Modul, das *decorator* und *Modules* exportiert. Diese können von *Components* und *Modules* importiert werden. Der Name jeder Angular library beginnt mit *@angular*. Angular libraries können mit dem npm installiert werden.

1.3.8.3 Components

Ein *Component* kontrolliert einen Teil des Bildschirms, den sogenannten *view*. Die Logik des *Component*s wird in einer Klasse definiert. Die Klasse interagiert mit dem *view* durch eine Application Programming Interface (API) von Eigenschaften und Methoden.

1.3.8.4 Templates

Das Aussehen des *view*s wird in einem *Template* definiert. Ein *Template* ist eine HyperText Markup Language (HTML) Datei, mit Angular's Template Syntax. Das bedeutet, dass einige Zusatzbefehle vorkommen können. Beispiele hierfür sind:

***ngFor**

Stellt ein HTML Element so oft dar, wie Elemente in einem Array vorhanden sind.

Listing 1.7: *ngFor Beispiel

```
1 <a *ngFor="let lang of languages" href="{{lang.href}}">{{lang.text}}</a>
```

***ngIf**

{{variable}}

(click)

[variable]

<app-route>

1.3.8.5 Data binding

1.3.8.6 Services

1.3.9 Bootstrap

Bootstrap

1.4 Umsetzung

1.4.1 Projektstruktur

```
Webserver
├── server
│   ├── dist
│   ├── keys
│   ├── node_modules
│   ├── public
│   ├── src
│   │   └── views
├── ngx
│   ├── dist
│   ├── e2e
│   ├── node_modules
│   ├── src
│   │   └── app
│   │       ├── components
│   │       └── services
```

```
assets
environments
i18n
```

1.4.2 Client

1.4.2.1 Design

Das Design sollte übersichtlich und einfach gestaltet werden. Der Benutzer soll auf den ersten Blick die wichtigsten Funktionen und Informationen erkennen können.

Auf der Starseite sind alle wichtigen Informationen übersichtlich dargestellt. Auf der linken Seite werden die Uhrzeit der letzten erfolgreichen Fütterung, die Zeit der nächsten Fütterung und die Zeit bis zur nächsten Fütterung dargestellt. Darunter werden Fehler und Warnungen, fals welche auftreten sollten, angezeigt. Da unbekannt ist, wie viele Fehler und Warnungen auftreten, werden diese in einem *ngFor aufgelistet. Auf der rechten Seite sind die aktiven Fütterungszeiten aufgelistet.

Abbildung 1.2: Startseite

Auf der Fütterungszeiten-Seite kann der Benutzer die Katzenfütterungsanlage ein und ausschalten. Dies wurde mit einer Checkbox realisiert, die durch Styles wie ein Schalter gestaltet wurde. Darunter können die Fütterungszeiten geändert und deaktiviert werden. Siehe Abbildung 1.3. Der Button 'Speichern' wird deaktiviert, sobald eine Zeit ungültig eingegeben wurde, oder wenn die Zeiten nicht in aufsteigender Reihenfolge sortiert sind.

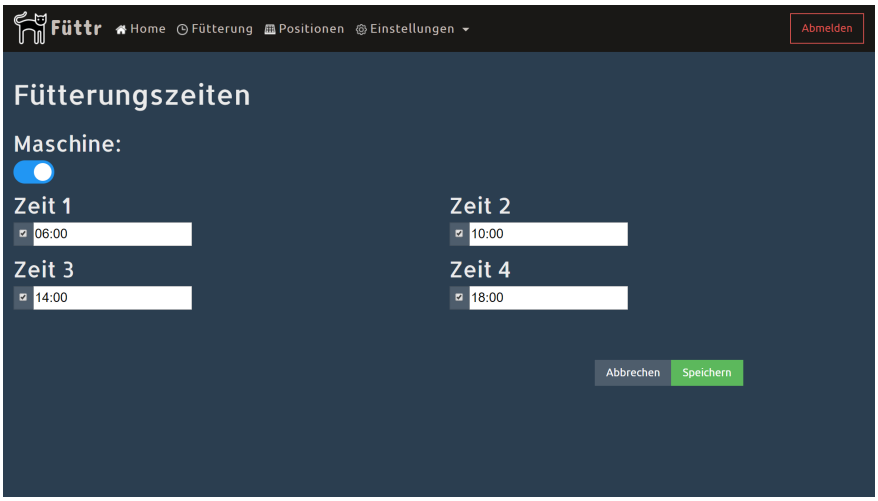

Abbildung 1.3: Fütterungszeiten

Auf der Geräteinformations-Seite werden die wichtigsten Daten über das Gerät angezeigt. diese Daten sind:

- Seriennummer
- Interner Rechner
- WLAN Status
- IP Adresse
- Softwareversion

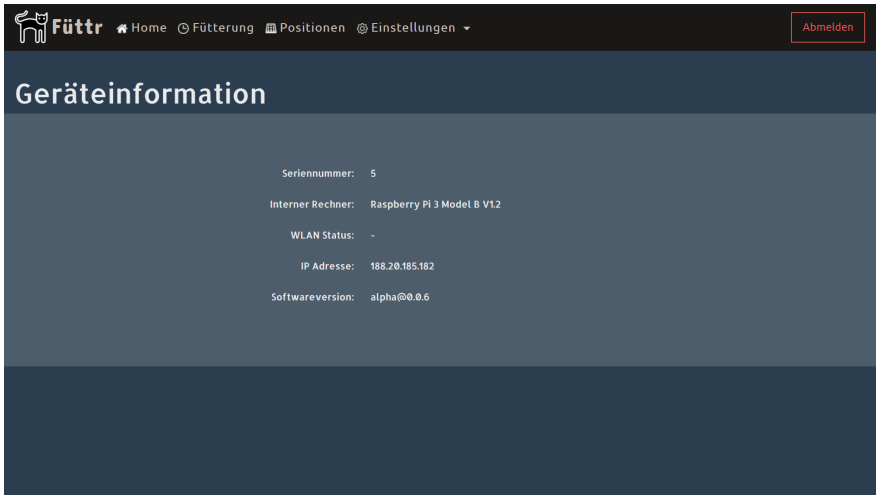

Abbildung 1.4: Geräteinformationen

Die Seriennummer ist eindeutig und wird von einem Server zugeteilt. Die IP-Adresse ist die aktuelle **externe** IP-Adresse.

Auf der Update-Seite kann der Benutzer nach Updates suchen, Updates starten oder die Maschine herunterfahren. Wenn der Benutzer auf den Herunterfahren-Button klickt, der sich auf der linken Seite ganz unten befindet, wird er gewarnt, dass die Maschine nur mehr über das Aus- und wieder Einstecken des Netzteils startbar ist.

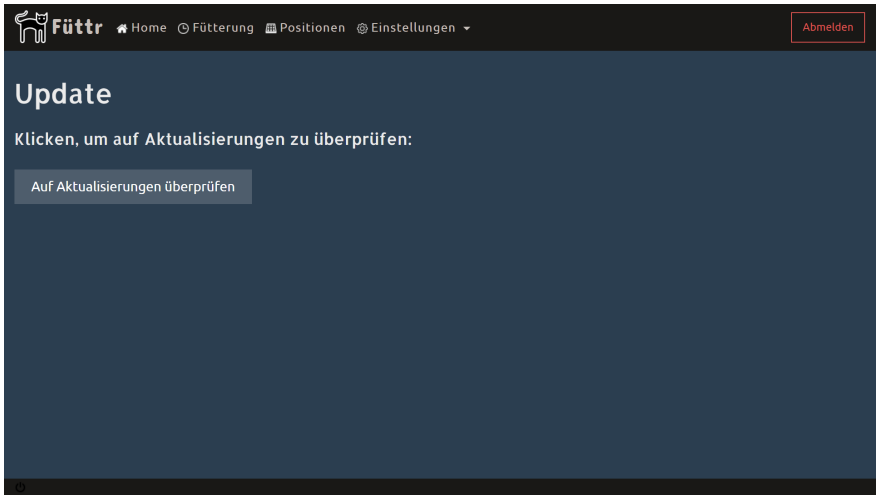

Abbildung 1.5: Update

1.4.2.2 Funktion

1.4.3 Server

1.4.3.1 Funktion

1.4.3.2 Mongodb

1.4.3.3 Kommunikation mit dem Java Programm

1.5 Zusammenfassung und Verbesserungsmöglichkeiten

Anhang

ANHANG A

Abbildungsverzeichnis

ANHANG B

Tabellenverzeichnis

ANHANG C

Listings

ANHANG D

Abkürzungsverzeichnis