

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

1	Webserver und Client	1
1.1	Begriffserklärungen	1
1.1.1	Server	1
1.1.2	Client	1
1.2	Anforderungen	1
1.2.1	Webserver	1
1.2.2	Client	1
1.3	Voruntersuchung	2
1.3.1	Typescript	2
1.3.2	Node.js	2
1.3.3	Angular 2/4	2
1.3.3.1	Modules	2
1.3.3.2	Libraries	3
1.3.3.3	Components	3
1.3.3.4	Templates	3
1.3.3.5	Data binding	4
1.3.3.6	Services	4
1.3.4	Bootstrap	4
1.3.5	express	4
1.3.6	JSON	4
1.3.7	JSON Web Token	5
1.3.8	MongoDB	5
1.4	Umsetzung	5
1.4.1	Client	5
1.4.1.1	Design	5
1.4.1.2	Funktion	7
1.4.2	Server	7
1.4.2.1	Funktion	7
1.4.2.2	Mongodb	7
1.4.2.3	Kommunikation mit dem Java Programm	7
1.5	Zusammenfassung und Verbesserungsmöglichkeiten	7
A	Abbildungsverzeichnis	11
B	Tabellenverzeichnis	13

C Listings	15
D Abkürzungsverzeichnis	17

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 verarbeiten 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:

- Fütterungszeiten
- Positionsinfo
- Geräteinfo
- Update

1.3 Voruntersuchung

1.3.1 Typescript

Typescript ist eine Weiterentwicklung der Sprache Javascript, die strenge Datentypen hat. Typescript muss von einem Transpiler (=Übersetzer) in Javascript übersetzt werden. Javascript kann direkt von jedem herkömmlichen Browser ausgeführt werden.

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 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.

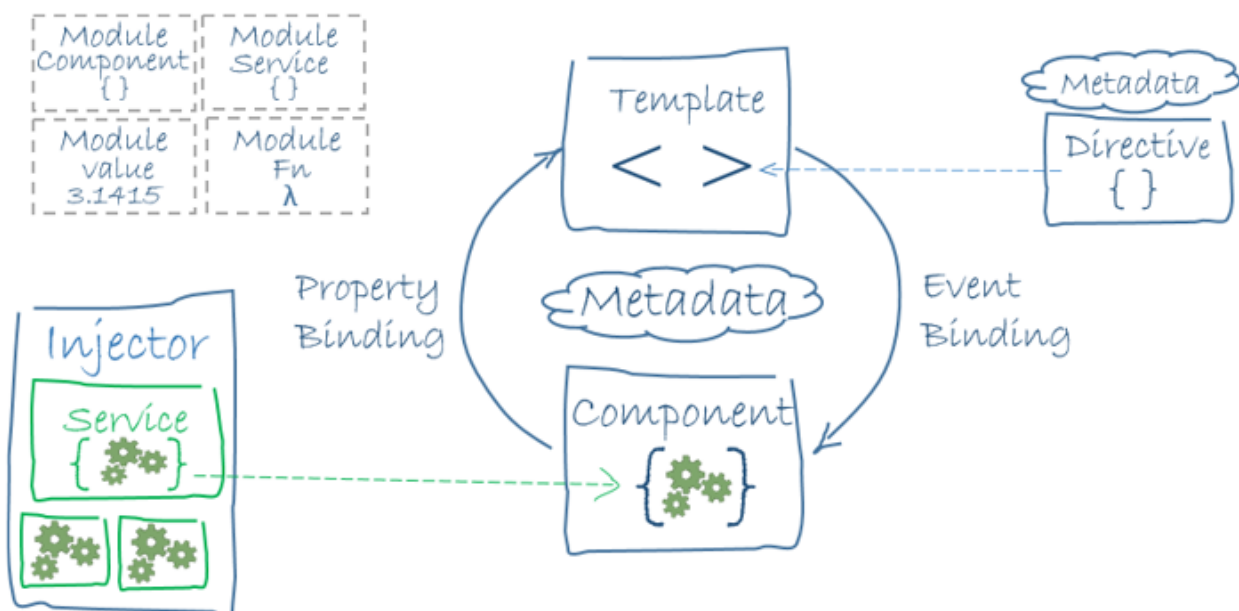


Abbildung 1.1: Angular Struktur

1.3.3.1 Modules

Jede Angular App ist in *Modules* gegliedert. *Modules* 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
-

1.3.3.2 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.

1.3.3.3 Components

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

1.3.3.4 Templates

Das Aussehen des *views* 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 1.1: *ngFor Beispiel

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

***ngIf**

{{variable}}

(click)

[variable]

<app-route>

1.3.3.5 Data binding

1.3.3.6 Services

1.3.4 Bootstrap

Bootstrap

1.3.5 express

Express ist ein Javascript Modul, dass auf dem Node.js Modul *http* bzw *https* aufbaut. Express ist ein Webserver. Das *http* Modul kann bereits alles, das für einen Webserver nötig ist. Express macht es allerdings leichter, diesen zu programmieren.

1.3.6 JSON

JavaScript Object Notation (JSON) ist die Textrepräsentation 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

```
1 {
2   "Object": {
3     "string": "name",
4     "number": 10E5,
5     "boolean": true,
6     "Array":
7       [
8         {
9           "string": "wert",
10          "number": 1
11        },
12        {
13          "string": "wert",
14          "number": 2
15        }
16      ]
17   ]
18 }
```

Listing 1.2: JSON Beispiel

1.3.7 JSON Web Token

1.3.8 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 1.3: Verbinden mit dem DBS

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

Listing 1.4: Auswählen der Datenbank

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

Listing 1.5: Auswählen der Collection

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

Listing 1.6: Auslesen aller Datensätze mit einem Identifier

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

Listing 1.7: Überschreiben eines Datensatzes mit einem Identifier

1.4 Umsetzung

1.4.1 Client

1.4.1.1 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 Startseite 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, falls 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.

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

1.4.1.2 Funktion

1.4.2 Server

1.4.2.1 Funktion

1.4.2.2 MongoDB

1.4.2.3 Kommunikation mit dem Java Programm

1.5 Zusammenfassung und Verbesserungsmöglichkeiten

Anhang

ANHANG A

Abbildungsverzeichnis

1.1	Angular Struktur	2
1.2	Startseite	5
1.3	Fütterungszeiten	6
1.4	Geräteinformationen	6
1.5	Update	7

ANHANG **B**

Tabellenverzeichnis

ANHANG C

Listings

1.1	*ngFor Beispiel	3
1.2	JSON Beispiel	4
1.3	Verbinden mit dem DBS	5
1.4	Auswählen der Datenbank	5
1.5	Auswählen der Collection	5
1.6	Auslesen aller Datensätze mit einem Identifier	5
1.7	Überschreiben eines Datensatzes mit einem Identifier	5

ANHANG D

Abkürzungsverzeichnis

npm Node Package Manager	2
API Application Programming Interface	3
HTML HyperText Markup Language	3
JSON JavaScript Object Notation	4
DBS Database management System	5