



## Diplomarbeit

# Entomophagie

### Untertitel der Arbeit

Imst, 22. März 2018

Eingereicht von

Leonid Hammer

Verantwortlich für IT: HTML, CSS, BWL: Kaufvertrag

Kevin Glatz

Verantwortlich für IT: SQL, C BWL: Kostenrechnung, Einsatz von Arduino

Florian Tipotsch

Verantwortlich für IT: SQL, C, Website BWL: Nutzen von Websites

Eingereicht bei

Stefan Stolz und Nina Margreiter

# Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbst verfasst und keine anderen als die angeführten Behelfe verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht. Ich bin damit einverstanden, dass meine Arbeit öffentlich zugänglich gemacht wird.

---

Ort, Datum

---

Leonid Hammer

---

Kevin Glatz

---

Florian Tipotsch

# Abnahmeerklärung

Hiermit bestätigt der Auftraggeber, dass das übergebene Produkt dieser Diplomarbeit den dokumentierten Vorgaben entspricht. Des Weiteren verzichtet der Auftraggeber auf unentgeltliche Wartung und Weiterentwicklung des Produktes durch die Projektmitglieder bzw. die Schule.

---

Ort, Datum

---

Thorsten Schwerte

# Vorwort

Wir wollen uns bei unseren Betreuern unserer Schule und bei unserem Projektpartner bedanken für die Unterstützung und die Zurverfügungstellung von Hilfsmitteln mit welcher wir diese Projekt abschließen konnten.

# Abstract (Deutsch)

Unser Projekt Entomophagie handelt von dem Züchten und Verzehr von Insekten. Dafür entwickeln wir einen Brutkasten Prototyp, welcher in Zukunft als Basis für weitere Brutkästen dient. Der Ziel unseres Produktes ist eine umweltfreundlichere Nahrungsquelle zu erschließen, die in der Westlichen Welt fast komplett ungenutzt ist. Mithilfe des Brutkastens soll es für jeden möglich sein, Essen günstig und selber 'anzubauen'. Insekten sind aufgrund ihrer niedrigen Energieverbrauch während der Züchtung viel umweltfreundlicher als, zum Beispiel, Kühe.

# Abstract (Englisch)

Our project Entomophagie is about growing and eating insects at home. To complete our task, we create a prototype of a breeding hub, which could be used for future adaptations of our model. The goal of our product is to show the western world the environmental friendly food source of insects, which is almost unused there. Furthermore we want to make it available for everyone to create their own food at home by simply breeding insects and eating them. The main reason insects are that more environmental friendly than, for example, cows is the energy consumption while breeding them.

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>11</b>
<b>Tabellenverzeichnis</b>	<b>12</b>
<b>Quelltexte</b>	<b>13</b>
<b>1 Einleitung</b>	<b>16</b>
<b>2 Projektmanagement</b>	<b>17</b>
2.1 Metainformationen . . . . .	17
2.1.1 Team . . . . .	17
2.1.2 Betreuer . . . . .	17
2.1.3 Partner . . . . .	17
2.1.4 Ansprechpartner . . . . .	17
2.2 Vorerhebungen . . . . .	17
2.2.1 Projektzieleplan . . . . .	17
2.2.2 Projektumfeld . . . . .	18
2.2.3 Risikoanalyse . . . . .	18
2.3 Pflichtenheft . . . . .	18
2.3.1 Zielbestimmung . . . . .	18
2.3.2 Produkteinsatz und Umgebung . . . . .	18
2.3.3 Funktionalitäten . . . . .	19
2.3.4 Testszenarien und Testfälle . . . . .	19
2.3.5 Liefervereinbarung . . . . .	19

2.4	Planung . . . . .	20
2.4.1	Projektstrukturplan . . . . .	20
2.4.2	Meilensteine . . . . .	22
2.4.3	Gant-Chart . . . . .	23
2.4.4	Abnahmekriterien . . . . .	23
2.4.5	Pläne zur Evaluierung . . . . .	23
<b>3</b>	<b>Brutkasten</b>	<b>24</b>
3.1	Equipment Liste . . . . .	24
3.2	Überprüfen des Equipments . . . . .	24
3.3	Equipment Kaufen . . . . .	24
3.4	Konzeptzeichnung . . . . .	24
3.5	Zeichnen . . . . .	24
3.6	Bauen . . . . .	24
<b>4</b>	<b>Eingesetzte Technologien</b>	<b>25</b>
4.1	Technologie für Webapp . . . . .	26
4.1.1	HTML - Hypertext Markup Language . . . . .	26
4.1.2	Was ist Yii . . . . .	26
4.1.3	PureMVC . . . . .	28
4.1.4	Laravel . . . . .	28
4.2	Technologien für die Datenauslesung . . . . .	29
4.2.1	Gas Sensoren . . . . .	29
4.2.2	Wärmesensor . . . . .	31
4.2.3	Luftfeuchtigkeit . . . . .	32
4.2.4	kinetischer Motor . . . . .	32
4.2.5	Hebel . . . . .	32
4.2.6	Datenübertragung . . . . .	32
<b>5</b>	<b>Problemanalyse</b>	<b>34</b>
5.1	USE-Case-Analyse . . . . .	34
5.2	Domain-Class-Modelling . . . . .	35
5.3	User-Interface-Design . . . . .	35



<b>6</b>	<b>Systementwurf</b>	<b>36</b>
6.1	Architektur . . . . .	36
6.1.1	Design der Komponenten . . . . .	36
6.1.2	MVC - Model, View, Controller . . . . .	36
6.1.3	CRUD - Create, Read, Update, Delete . . . . .	38
6.1.4	Yii2 . . . . .	39
6.1.5	Benutzerschnittstellen . . . . .	40
6.1.6	Datenhaltungskonzept . . . . .	40
6.1.7	Konzept für Ausnahmebehandlung . . . . .	42
6.1.8	Sicherheitskonzept . . . . .	43
6.1.9	Design der Testumgebung . . . . .	43
6.1.10	Desing der Ausführungsumgebung . . . . .	43
6.2	Detaillentwurf . . . . .	44
6.3	Arduino . . . . .	45
6.3.1	Verkablung . . . . .	45
<b>7</b>	<b>Implementierung</b>	<b>48</b>
7.1	Webapp . . . . .	49
7.1.1	Mockups . . . . .	49
7.1.2	Datenbankzugriffe . . . . .	53
7.1.3	Datenübermittlung . . . . .	53
7.1.4	Rest Schnittelle . . . . .	53
7.2	Verkablung . . . . .	56
7.2.1	CCS811 . . . . .	57
7.3	Arduino . . . . .	57
7.3.1	ESP8266 . . . . .	58
7.3.2	Joystick Modul . . . . .	58
7.4	Programmierung . . . . .	58
7.4.1	DHT11 & CCS811 . . . . .	59
7.4.2	Hebel . . . . .	60
7.4.3	SG90 . . . . .	60
7.4.4	Wlan . . . . .	62

<b>8 Deployment</b>	<b>65</b>
<b>9 Tests</b>	<b>66</b>
9.1 Systemtests . . . . .	66
9.2 Akzeptanztests . . . . .	66
<b>10 Projektevaluation</b>	<b>67</b>
<b>11 Benutzerhandbuch</b>	<b>68</b>
<b>12 Betriebswirtschaftlicher Kontext</b>	<b>69</b>
12.0.1 Arduino in der Wirtschaft . . . . .	69
12.0.2 Kostenrechnung . . . . .	69
12.1 Nutzen einer Website . . . . .	71
<b>13 Zusammenfassung</b>	<b>72</b>
<b>Literaturverzeichnis</b>	<b>73</b>

# Abbildungsverzeichnis

2.1	Unser Gant Chart für das Projekt . . . . .	22
6.1	ERDiagramm unserer Datenbank . . . . .	41
6.2	Arduino mit Breadboard . . . . .	45
7.1	Mockup unserer Seite wenn man eingeloggt ist . . . . .	50
7.2	Mockup der Seite Meine-Zuchtkammer . . . . .	51
7.3	Mockup unserer Seite wenn man nicht eingeloggt ist . . . . .	52
7.4	Abbildung des API Ordners . . . . .	54
7.5	Verkablung aller Module . . . . .	56

# Tabellenverzeichnis

2.1	Projektstrukturplan . . . . .	21
2.2	Meilensteine . . . . .	22

# Quelltexte

7.1	Controller Klasse für REST . . . . .	54
7.2	Module Klasse für REST . . . . .	55
7.3	Datenauslesung CCS811 . . . . .	59
7.4	Automatisierten Servobewegung bei zu niedrigen CO2 Werten .	60
7.5	Verbindungsaufbau des ESP8266 . . . . .	62
7.6	Verbindungsaufbau Response . . . . .	64
7.7	Ausführung der REST Befehle . . . . .	64

# **Einleitende Bemerkungen**

# Notationen

Beschreibung wie Code, Hinweise, Zitate etc. formatiert werden

# **1 Einleitung**



# **2 Projektmanagement**

## **2.1 Metainformationen**

### **2.1.1 Team**

### **2.1.2 Betreuer**

### **2.1.3 Partner**

### **2.1.4 Ansprechpartner**

## **2.2 Vorerhebungen**

### **2.2.1 Projektzieleplan**

Projektziele-Hierarchie - SMART

### **2.2.2 Projektumfeld**

- Identifikation der Stakeholder
- Charakterisierung der Stakeholder
- Maßnahmen
- Grafische Darstellung des Umfeldes

### **2.2.3 Risikoanalyse**

- Risikomatrix

## **2.3 Pflichtenheft**

### **2.3.1 Zielbestimmung**

- Projektbeschreibung
- IST-Zustand
- SOLL-Zustand
- NICHT-Ziele (Abgrenzungskriterien)

### **2.3.2 Produkteinsatz und Umgebung**

- Anwendungsgebiet
- Zielgruppen

- Betriebsbedingungen
- Hard-/Softwareumgebung

### **2.3.3 Funktionalitäten**

- MUSS-Anforderungen
  - Funktional
  - Nicht-funktional
- KANN-Anforderungen
  - Funktional
  - Nicht-funktional

### **2.3.4 Testszenarien und Testfälle**

- Beschreibung der Testmethodik
- Testfall 1
- Testfall 2
- ...

### **2.3.5 Liefervereinbarung**

- Lieferumfang
- Modus
- Verteilung(Deployment)

## **2.4 Planung**

### **2.4.1 Projektstrukturplan**

Tabelle 2.1: Projektstrukturplan

P 1	Projektstart	
1	Planung	Alle
1.1	Besprechungstermine setzen	Alle
1.1.1	Vertrag erstellen	Alle
1.2	Verantwortungsmatrix	Alle
1.3	Budgetplan erstellen	Alle
2	Programmierung (Kevin + Flo)	Kevin Glatz, Florian Tiptsch
2.1	Programmiersprache	Kevin Glatz
2.1.1	Sprache Wählen Š	Kevin Glatz
2.1.2	Informationsammlung für Features	Kevin Glatz
2.1.3	Vertraut machen	Kevin Glatz
2.2	Arduino UNO kaufen (Flo)	Florian Tipotsch
2.2.1	Vertraut machen	Kevin Glatz
2.2.2	Test Programme	Kevin Glatz
2.3	Programmierung des Brutkasten	Kevin Glatz
3	Brutkasten (Leo)	Leonid Hammer
3.1	Equipment Liste erstellen	Leonid Hammer
3.1.1	Equipment überprüfen	Leonid Hammer
3.1.2	Equipment Kaufen	Alle
3.2	Konzeptzeichnung	Leonid Hammer
3.2.1	Zeichnen	Leonid Hammer
3.2.2	Bauen	Leonid Hammer
4	Datenbank	Kevin Glatz
4.1	ER-Diagramm	Florian Tipotsch + Kevin Glatz
4.2	Sprache Wählen	Florian Tipotsch
4.3	HTML Seite mit Daten	Florian Tipotsch
5	Nachforschung (Leo)	Leonid Hammer
5.1	Welches Tier	Leonid Hammer
5.2	Wie viele	Leonid Hammer
5.3	Futter	Leonid Hammer
6	Webapp	Florian Tipotsch
6.1	Prototyp	Florian Tipotsch
6.2	Login am Server	Florian Tipotsch
6.3	Registrierung Zuchkammer	Florian Tipotsch

Verantwortlich für den Inhalt: Florian Tipotsch

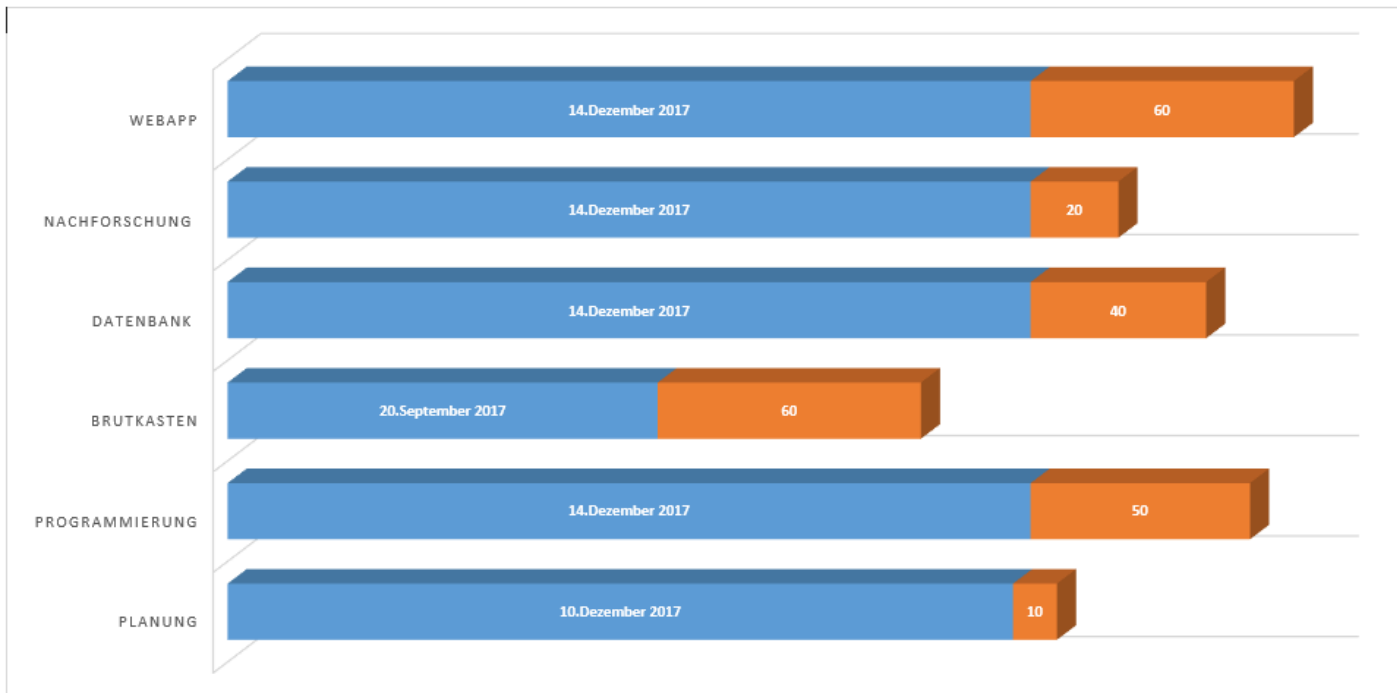


Abbildung 2.1: Unser Gant Chart für das Projekt

## 2.4.2 Meilensteine

Tabelle 2.2: Meilensteine

1	Planung	Alle
2	Programmierung	Kevin Glatz, Florian Tiptsch
3	Brutkasten	Leonid Hammer
4	Datenbank	Kevin Glatz
5	Nachforschung	Leonid Hammer
6	Webapp	Florian Tiptsch

### **2.4.3 Gant-Chart**

### **2.4.4 Abnahmekriterien**

Die Abnahmekriterien wurden von unserem Auftraggeber so gesetzt das es einen Prototypen geben soll welcher Daten auslesen kann und darauf reagiert.

### **2.4.5 Pläne zur Evaluierung**

Zuerst wollen wird die Daten auf unserem Arduino auslesen und diese dann mithilfe eines Wlan Modul es an unsere Website senden.

## **3 Brutkasten**

Hier werden wir uns dem Aufbau, der Planung und Ausführung des Brutkastens widmen.

### **3.1 Equipment Liste**

### **3.2 Überprüfen des Equipments**

### **3.3 Equipment Kaufen**

### **3.4 Konzeptzeichnung**

### **3.5 Zeichnen**

### **3.6 Bauen**



## 4 Eingesetzte Technologien

- Kurzbeschreibung aller Technologien, die verwendet wurden.
- Technologien die aus dem Unterricht bekannt sind, nur nennen und deren Einsatzzweck im Projekt beschreiben, nicht die Technologien selbst.
- Technologien die aus dem Unterricht nicht bekannt sind, im Detail beschreiben incl. deren Einsatz im Projekt
- Fokus auf eingesetzten Frameworks

## **4.1 Technologie für Webapp**

- PHP - Für Webapp
- Html - Für Webapp
- MySql - Für Datenbanken
- Yii2 - Für Webapp
- MVC - Model, View, Controller - Für Webapp
- CRUD - Create, Read, Update, Delete - Für Webapp
- REST - Für Datenbank, Webapp und Datenspeicherung

### **4.1.1 HTML - Hypertext Markup Language**

HTML sind die Grundlagen für jede Website. Sie bilden die Grundstruktur jeder Webseite, und werden von Browsern dargestellt. HTML wird von dem World Wide Web Consortium (W3C) (3) und dem Web Hypertext Application Technology Working Group (WHATWG) (2) weiterentwickelt.

Gute Ressourcen zum lernen und schreiben von HTML findet man auf [w3schools.com](http://w3schools.com) (1)

### **4.1.2 Was ist Yii**

Yii ist ein high Performance PHP Framework welches vor allem für die Entwicklung im Web2.0 eingesetzt wird. Web 2.0 fördert das User aktiv im Web mitmachen. Diese können eigenen Beiträge erstellen und diese auf der Website anzeigen lassen. Mehr dazu im Kapitel Yii2(16)

## **Alternative zu Frameworks**

Yii kann sehr weitreichend eingesetzt werden. Mit dem richtigen Wissen und Fähigkeiten kann man alles was mit einer PHP Seite möglich ist in Yii2 umsetzen.

Allerdings sind Frameworks nicht Administratoren freundlich da sie sehr viel Vorwissen erfordern um diese richtig zu implementieren und zu warten. Einfacher zu implementieren sind CMS Systeme. Es gibt sehr viele Große CMS Systeme zum Beispiel:

- Joomla
- Wordpress
- Drupal
- Contao

Diese haben wir auch schon im Unterricht kennengelernt und damit Websites erstellt. Vorteile sind vor allem die einfache Implementierung und rasche Einrichtung einer Website. Auch SEO wird von den CMS Systemen vereinfacht. Nachteile sind allerdings oft eingeschränkte Möglichkeiten und Grenzen welches das CMS setzt.

## **Warum haben wir uns für Yii2 entschieden**

Der Hauptgrund warum wir uns gegen CMS Systeme entschieden haben sind die eingeschränkten Möglichkeiten die wir damit hätten. Bei Yii2 können wir die gesamte Website nach unseren Bedarf zusammenstellen und auch so bearbeiten wie wir es wollen. Es war uns auch wichtig das wir nach modernen Entwurfsmustern arbeiten. Bei Yii2 wird das MVC - Model, View, Controller Muster eingesetzt.

Wir hätten uns auch für andere Frameworks entscheiden können allerdings war uns Yii2 schon bekannt und wir haben damit schon einige Websites erstellt.

Alternativen für Yii2 sind:

- PureMVC
- Laravel

### **4.1.3 PureMVC**

PureMVC ist seit dem Release in 2008 unverändert. Das hat den Vorteil das der administrative aufwand sehr gering ist aufgrund nicht vorhandener Updates. Außerdem muss man das Framework nur einmal lernen und kann dieses dann meistern ohne irgendwelche Änderungen zu befürchten. Es gibt auch Best-Practice Beispiele in vielen verschiedenen Sprachen. Diese findet man auf der Website. (11)

### **4.1.4 Laravel**

Laravel: PHP That Doesn't Hurt. Code Happy and Enjoy The Fresh Air. Laravel will PHP einfach und übersichtlich Programmierbar machen dazu verwendet es auch das MVC Muster und auch den PHPComposer um sehr einfach neue Erweiterungen zu installieren. Auch bei Laravel gibt es sehr gute Documentationen. Diese findet man auf der Website. ( ? )

## **4.2 Technologien für die Datenauslesung**

- Gas Sensor - Für Luftqualität
- Wärmesensor - Für Raumtemperatur
- Luftfeuchtigkeit - Für Luftqualität
- Schalter - Für Futtermenge
- kinetischer Motor - für geregelte Luftzufuhr
- WLAN - Für Datenübertragung

### **4.2.1 Gas Sensoren**

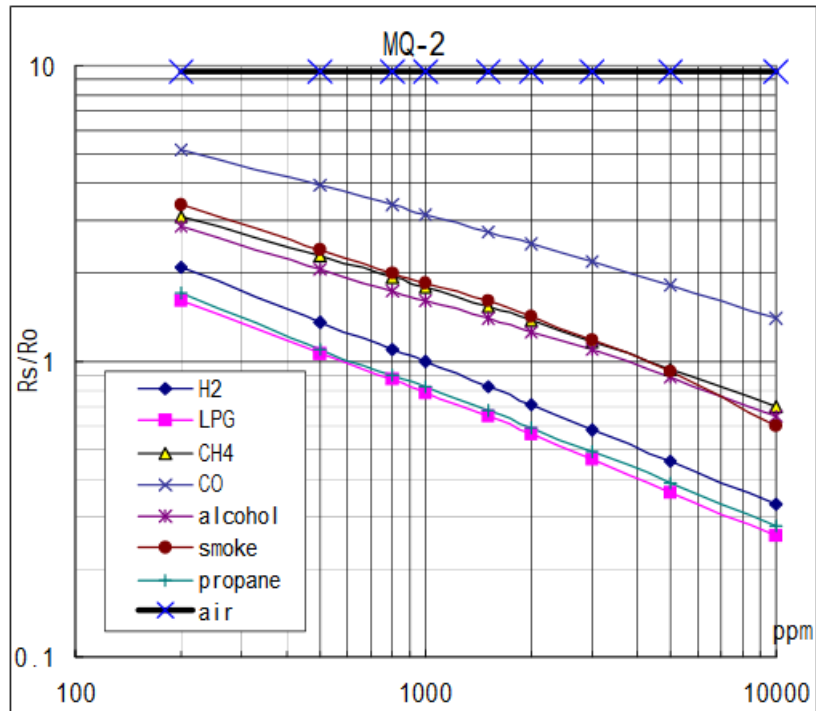
Für die Auswertung der Gaswerte stehen uns mehrere Module zur Verfügung. Zum einen stand uns die vielfältige MQ-Serie zur Verfügung oder der spezialisierte CCS811 von Adafruit.

In der Schule haben wir nicht nur den MQ2 Sensor zur Verfügung bereitgestellt bekommen, sondern auch den Adafruit CCS811. Wir bedanken uns dafür vielmals.

#### **MQ Gas Sensoren**

- MQ2 Methane, Butane, LPG, smoke
- MQ3 Alcohol, Ethanol, smoke
- MQ4 Methane, CNG Gas
- MQ5 Natural gas, LPG
- MQ6 LPG, butane gas

- MQ7 Carbon Monoxide
- MQ8 Hydrogen Gas
- MQ9 Carbon Monoxide, flammable gasses
- Mehr gibt es auf der Website: (5)



(8)

(5)

Im Datasheet (8) kann man herauslesen das der Sensor MQ2 (5) H2, LPG, CH4, CO, Alkohol, Rauch und Propan in einem Bereich von 200 bis 10000 Parts per million (Anteil pro Million) messen kann. Wie empfindlich der Sensor ist, hängt von den RS und RO werten ab.

- RS: Sensor Widerstand bei verschiedenen Konzentrationen von Gas
- RO: Sensor Widerstand bei 1000ppm von H2 bei sauberer Luft.

Da der MQ-2 auf so viele Gase außer Co2 reagiert haben wir uns entschlos-

sen eine Alternative zu suchen. Glücklicherweise hat uns die Schule auch den CCS811 von Adafruit zur Verfügung stellen können,

### **Adafruit CCS811**

In unserem Projekt haben wir uns letztendlich für den CCS811 von Adafruit entschieden. Im Gegensatz zu dem MQ-Sensoren erlaubt der CCS811 eCO<sub>2</sub> in einem Bereich von 400 bis 8192 ppm (parts per million) auszulesen. (10)

Dadurch können wir uns komplizierte mathematische Formelrechnungen sparen und direkt den gewünschten Datentypen verwenden.

### **4.2.2 Wärmesensor**

#### **DHT11**

In dem Arduino Uno Set war der DHT 11 Wärme und Temperatur Sensor von Adafruit direkt mitgeliefert. Dieses Modul erlaubt es uns digitale Daten in einem zwei Sekunden Takt auszulesen. Es liest die Wärme in einem Bereich von 0 bis 50°C, die Daten können allerdings um etwa 2°C variieren.

#### **CCS811**

Der CCS811 besitzt auch einen Thermistor, mit welchem wir direkt die Raumtemperatur auslesen können. Da der DHT11 allerdings auch die Luftfeuchtigkeit anzeigen kann, verwendeten wir diesen anstatt den CCS811. (10)

### **4.2.3 Luftfeuchtigkeit**

Da der DHT direkt bei dem Arduino Kit mit dabei war, gab es keinen Grund einen anderen Sensor zu suchen. Das Modul erlaubt uns die Luftfeuchtigkeit in einem Bereich von 20 bis 80 Prozent zu messen. Diese Daten können allerdings um beinahe 5% variieren.

### **4.2.4 kinetischer Motor**

Für eine automatisierte Luftzufuhr verwenden wir den Servo-Mikrocontroller SG90. Sowie der DHT 11 war dieser Motor schon im Arduino Uno Set beigelegt. Der 3-polige Servo wird für die Luftzufuhr verwendet. Der Propeller am drehenden Motor öffnet und schließt die Luftklappe des Kastens. Das ermöglicht eine höhere Kontrolle und Regulierung des CO<sub>2</sub> Wertes. (<https://servodatabase.com/servo/towerpro>)

### **4.2.5 Hebel**

Um zu sehen wie viel Futter für die Insekten vorhanden ist, verwendeten wir einen Joystick. Dieser Joystick war beim Arduino Uno Set mitgeliefert und kann die X- und Y-Achse ablesen sowie auf Druck zu reagieren. Anhand der Y-Achse kann man den Joystick als Hebel verwenden um den momentanen Futterstand abzulesen.

### **4.2.6 Datenübertragung**

Damit der Nutzer die Daten letztendlich auf der Website abrufen kann, verwenden wir das WLAN-Modul ESP8266. Dieser sehr beliebter und billiger Mikrocontroller fungiert als eine voll Funktionsfähige Netzwerkschnittstelle.



## *Entomophagie*

Es erlaubt uns auch den Brutkasten kabellos und benutzerfreundlicher zu Gestalten

# 5 Problemanalyse

## 5.1 USE-Case-Analyse

- UseCases auf Basis von Benutzerzielen identifizieren:
  - Benutzer eines Systems identifizieren
  - Benutzerziele identifizieren (Interviews)
  - Use-Case-Liste pro Benutzer definieren
- UseCases auf Basis von Ereignissen identifizieren:
  - Externes Event triggert einen Prozess
  - zeitliches Event triggert einen Prozess (Zeitpunkt wird erreicht)
  - State-Event (Zustandsänderung im System triggert einen Prozess)
- Werkzeuge:
  - USE-Case-Beschreibungen (textuell, tabellarisch)
  - USE-Case-Diagramm
  - Aktivitätsdiagramm für den Use-Case (Interaktion zwischen Akteur und System abbilden)
  - System-Sequenzdiagramm (Spezialfall eines Sequenzdiagramms: Nur 1 Akteur und 1 Objekt, das Objekt ist das komplette System, es geht um die Input/Output Requirements, die abzubilden sind)

## **5.2 Domain-Class-Modelling**

- "Dinge" (Rollen, Einheiten, Geräte, Events etc.) identifizieren, um die es im Projekt geht
- ER-Modellierung oder Klassendiagramme
- Zustandsdiagramme (zur Darstellung des Lebenszyklus von Domain-Klassen darstellen)

## **5.3 User-Interface-Design**

- Mockups
- Wireframes

# 6 Systementwurf

## 6.1 Architektur

### 6.1.1 Design der Komponenten

Unsere Webapp ist in Yii2 programmiert und ist deshalb nach dem MVC Muster aufgebaut.

### 6.1.2 MVC - Model, View, Controller

#### Was ist MVC?

MVC, auch Model, View, Controller ist ein modernes Entwurfsmuster das meist für Anwendungen die ein User Interface beinhalten, eingesetzt wird. Zum Beispiel für PHP, JAVA, C# und Ruby Anwendungen. (15)

## **Vor- und Nachteile**

### Vorteile

Gleichzeitiges Programmieren

Hohe Kohäsion (13)

Lose Kopplung (14)

### Nachteile

Schlechte Übersicht

Konsistente Programmierung notwendig

Steile Lernkurve

Das MVC Entwurfsmuster ist sehr weit verbreitet und hoch angesehen, es wird deshalb in den meisten Anwendungen mit User Interface angewandt. Der Grund für die Verwendung dieses Musters sind die vielen Vorteile die es bietet. Außerdem können viele Nachteile über Frameworks behoben werden.

## **Aufbau von MVC**

Im Grunde arbeitet MVC mit 3 Klassen:

- Modell
- View
- Controller

Die unterschiedlichen Klassen haben unterschiedliche Aufgaben.

## **View**

Die View Klasse ist nur zum Anzeigen der einzelnen Teile des User Interfaces verantwortlich. Bei Webseiten zum Beispiel die Index Seite oder das Login Formular. Außerdem nimmt es alle Benutzer/innen Eingaben entgegen.

## **Controller**

Die Controller Klasse Steuert die ganze Anwendung, sie nimmt die User eingaben von der View Klasse entgegen und verarbeitet diese. Außerdem ändert sie auch die View Klasse um andere Seiten anzuzeigen. Sie nimmt auch die Daten von der Modell Klasse entgegen, verarbeitet diese und gibt sie an die View Klasse weiter.

## **Modell**

Die Modell Klasse enthält die darzustellenden Daten, sie ist unabhängig von der View und der Controller Klasse

### **6.1.3 CRUD - Create, Read, Update, Delete**

#### **Was ist CRUD?**

CRUD sind die 4 grundlegenden Aufgaben einer Datenbankbindung:

- Create - Erstellen neuer Datensätze
- Read - Auslesen der Datensätze
- Update - Aktualisieren von vorhandenen Datensätze
- Delete - Löschen von Datensätzen

(12)

CRUD ist für alle Datenbankzugriffe verantwortlich die gemacht werden.

## **6.1.4 Yii2**

### **Was ist Yii**

Yii ist ein high Performance PHP Framework welches vor allem für die Entwicklung im Web2.0 eingesetzt wird. Web 2.0 fördert das User aktiv im Web mitmachen. Diese können eigene Beiträge erstellen und diese auf der Website anzeigen lassen.(16)

### **Gii**

Gii ist der Yii eigene Model, Crud, Controller, Form, Module und Extension Generator. Mit ihm kann man sehr einfach eine Model Klasse mit einer Unterliegenden Datenbanktabelle erstellen. Aus dieser Model klasse kann man dann wiederum CRUD befehle erzeugen. Mit Gii kann man die gesamte Grund MVC Struktur für Yii2 erzeugen und im weitem Verlauf dann nach eigenen Wünschen verändern.

### **Vor- und Nachteile**

Yii hat sehr viele Vorteile, allerdings auch einige Nachteile: Vorteile sind:

- CRUD-Creator (Gii)
- Model Generator (Gii)
- Einfache Implementierung von HTML Formulare
- Einfach Datenbankzugriffe

Nachteile ist der Hohe Setup aufwand und die benötigte Kenntnisse in PHP und SQL, welche allerdings bei fast allen Frameworks von Nöten sind.

### **6.1.5 Benutzerschnittstellen**

Die Benutzerschnittstelle unseres Brutkasten besteht hauptsächlich über die Webapp. Dort kann man sich anmelden und die Daten für den eigenen Brutkasten auslesen. Um sicherzugehen das man nur die eigenen Daten sieht, muss man sich auf der Website mit der Seriennummer die man beim Kauf eines Kasten erhält registrieren. Dadurch kann man dann auf die Daten zugreifen welche für die jeweilige Seriennummer gespeichert sind.

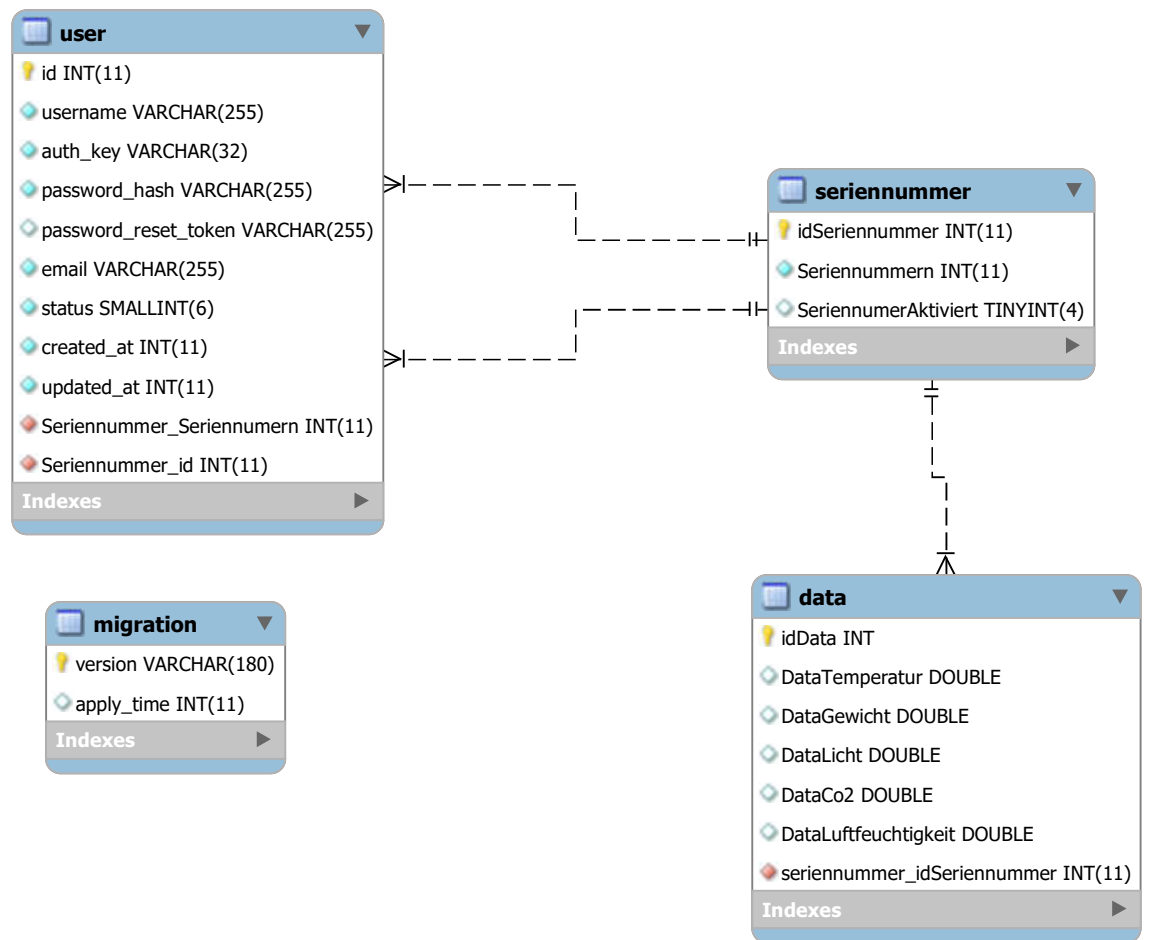
Wenn man noch nicht registriert wird kann man auf der Website einen Link zu unserem Webshop sehen, den wir allerdings nicht in diesem Projekt erstellen werden da es den Rahmen dieses Projektes Sprengen würde.

### **6.1.6 Datenhaltungskonzept**

Hier ist unser ER-Diagramm

Die Datenbank werden wir bei unserem Web Hoster anlegen und dort alle Daten bis zu 7 Tage Speichern. Daten sollen alle 5-10 Minuten von unserem Brutkasten übermittelt werden. Zur Übermittlung werden wir ein WLAN Modul an unserem Brutkasten anbringen. Mehr dazu im Kapitel Datenübermittlung. Für Prototyp Zwecken werden wir die Daten vorerst lokal über PHPMyAdmin speichern.





### **6.1.7 Konzept für Ausnahmebehandlung**

- Systemweite Festlegung, wie mit Exceptions umgegangen wird
- Exceptions sind primär aus den Bereichen UI, Persistenz, Workflow-Management

### **6.1.8 Sicherheitskonzept**

Auf unsere Website kann man sich mithilfe von Username und Password anmelden. Im Weiteren verlauf des Projektes besteht noch die Möglichkeit für unsere Rest Schnittelle eine Authentifizierung einzubauen. Da unser Prototyp allerdings nur lokal vorhanden ist ist dies zum jetzigen Zeitpunkt noch nicht notwendig.

### **6.1.9 Design der Testumgebung**

Unseren Protoyp testen wird indem wir ihn einen Tag autonom laufen lassen und am Ende dieser 24h Periode alle Daten kontrollieren ob diese Sinn machen und der Realität entsprechen. Um dies zu gewährleisten bräuchten wir allerdings einen Klima Regulierten Raum der Co2 messen kann.

### **6.1.10 Desing der Ausführungsumgebung**

Das Endprodukt sollte im Freien autonom funktionieren können. Das Setup für den Endbenutzer/in besteht aus dem anstecken an eine Stromversorgung und dem Verbinden mit dem Internet, sowie das einsetzen der Insekten. Im weiteren Verlauf der Brutzeit muss der Benutzer/ die Benutzerin die Insekten füttern. Unser Produkt übernimmt das Regulieren der Temperatur sowie Co2 Werten damit die Insekten Überleben können.

## 6.2 Detailentwurf

USE-Cases Klassendiagramme vom Domain-Klassendiagramm ableiten (incl. detaillierter Darstellung und Verwendung von Vererbungshierarchien, abstrakten Klassen, Interfaces)

- Sequenzdiagramme vom System-Sequenz-Diagramm ableiten
- Aktivitätsdiagramme
- Detaillierte Zustandsdiagramme für wichtige Klassen

Verwendung von CRC-Cards (Class, Responsibilities, Collaboration) für die Klassen

- um Verantwortlichkeiten und Zusammenarbeit zwischen Klassen zu definieren und
- um auf den Entwurf der Geschäftslogik zu fokussieren

Design-Klassen für jeden einzelnen USE-Case können z.B. sein:

- UI-Klassen
- Data-Access-Klassen
- Entity-Klassen (Domain-Klassen)
- Controller-Klassen
- Business-Logik-Klassen
- View-Klassen

Optimierung des Entwurfs (Modularisierung, Erweiterbarkeit, Lesbarkeit):

- Kopplung optimieren
- Kohäsion optimieren
- SOLID
- Entwurfsmuster einsetzen

## 6.3 Arduino

Die erfolgreiche Verwendung des Arduino besteht aus zwei Punkten:

- Verkablung der einzelnen Module
- Programmierung

Wir können vor allem dem Programmcode um einiges vereinfachen indem wir diesen in drei Unterbereiche aufteilen, wobei jedes Abteil eine fest zugeteilte Aufgabe bekommt. Dasselbe zählt für die Verkablung der einzelnen Module, da diese mit einer strukturierten Farbkodierung übersichtlicher werden.

### 6.3.1 Verkablung

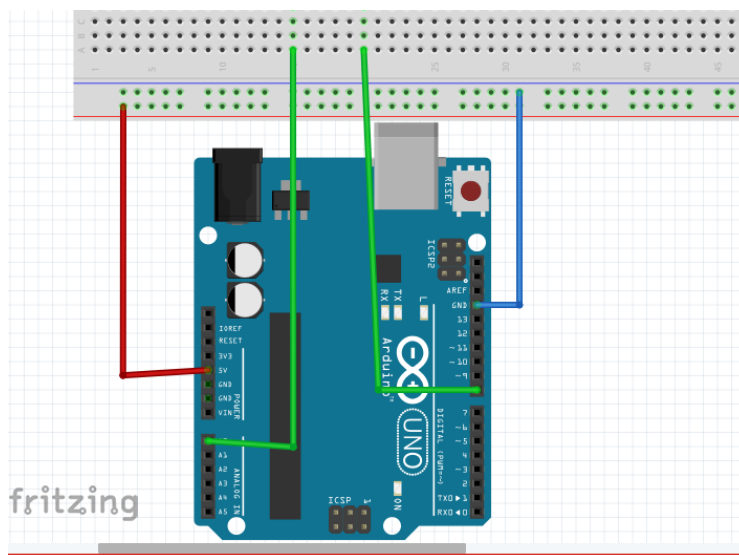


Abbildung 6.2: Arduino mit Breadboard

In Abbildung 6.2 können wir nun den Arduino und das Breadboard sehen. Für eine einfachere Unterscheidung der Kabel verwenden wir ein simples Farbschema. Rot steht für die Stromzufuhr, grün für die Datenauslesung und blau für die Erdung.

Die untersten und obersten zwei Zeilen, lassen den Strom bzw. die Erdung horizontal laufen. Die Steckplätze darüber leiten hingegen jegliche Information vertikal. Ein wichtiges Detail ist das man Stromzufuhr auf der horizontalen Plus reihe platziert (in der Grafik anhand der roten Linie zu sehen), da ansonsten keine Energie ankommt. Bei den vertikalen Steckplätzen gibt es allerdings keine negativ oder positiv gepolten Spalten.

Worauf man bei der Datenauslesung achten muss, ist das Arduino zwei Möglichkeiten dafür verwendet. Zum einen gibt es digitale Daten, welche einen Wert von 1 oder 0 besitzen können. Die Alternative dafür sind analoge Daten, die einen Wert von 0 bis 1023 Wahrnehmen können.

Vereinfacht erklärt werden bei digitalen Pins entweder 0 oder 5 Volt gelesen und bei analogen Pins 5 Volt durch 1024 Werte geteilt. Je nachdem wie viel Stromzufuhr das Modul hat, kann es somit einen Datenwert um einiges genauer zurückgeben (4)

### **Bibliotheken importieren und globale Variablen definieren**

In diesem Schritt importieren wir die nötigen Bibliotheken welche aus GitHub oder direkt von Arduino zur Verfügung gestellt bekommen. Auch werden alle Variablen die sowohl in den Methoden Setup und Loop verwendet werden erstellt.

### **Setup**

Die Funktion Setup wird verwendet um einmalige Konfigurationen zu tätigen. Dazu zählen unter anderem Aufbau einer Verbindung, Kalibrierung sensibler Controller oder ähnlichen. Die hier getätigten Aktivitäten werden nur ausgeführt wenn der Arduino startet bzw. zurückgesetzt wird.

## **Loop**

Diese Methode verarbeitet und gibt all unsere Werte aus. Besonders an der Klasse Loop ist, dass diese dauerhaft wiederholt wird. Es gibt keine maximale Durchlaufmenge und die Geschwindigkeit eines Durchlaufs wird von der Methode `delay` in Millisekunden definiert. Mit Hilfe dieser Eigenschaften, können wir zeitnahe alle Daten auslesen. Falls nun ein Wert eine Mindestgrenze kann sofort darauf reagiert werden.

Eine genauere Beschreibung der einzelnen Module und deren Programmierung erfolgt in den kommenden Seiten.

# 7 Implementierung

Detaillierte Beschreibung der Implementierung aller Teilkomponenten der Software entlang der zentralsten Use-Cases:

- GUI-Implementierung
- Controllerlogik
- Geschäftslogik
- Datenbankzugriffe

Detaillierte Beschreibung der Teststrategie (Testdriven Development):

- UNIT-Tests (Funktional)
- Integrationstests

Zu Codesequenzen:

- kurze Codesequenzen direkt im Text (mit Zeilennummern auf die man in der Beschreibung verweisen kann)
- lange Codesequenzen in den Anhang (mit Zeilennummer) und darauf verweisen (wie z.B. hier)



## **7.1 Webapp**

Für unser Projekt erstellen wir eine Webapp mit der man die Daten seiner eigenen Zuchtkammer anzeigen lassen kann. Wir haben geplant das man sich mit der Seriennummer der Box Registrieren kann und dann am Handy über eine Webapp alle Daten anzeigen lassen kann. Folgende Daten sollte man auslesen können:

- Sauerstoff
- Luftfeuchtigkeit
- Gewicht
- Temperatur
- Futtermenge
- ungefähre Zeit bis zu Reife

Als Grundlage für die Website haben wir das Framework Yii2 verwendet. Mehr dazu im Kapitel Yii2.

### **7.1.1 Mockups**

Hier sieht man die Ansicht wenn man auf unserer Website Angemeldet ist. Man kann auf seine eigene Zuchtkammer zugreifen indem man auf den Button 'Meine Zuchtkammer' klickt und dort die Daten auslesen.

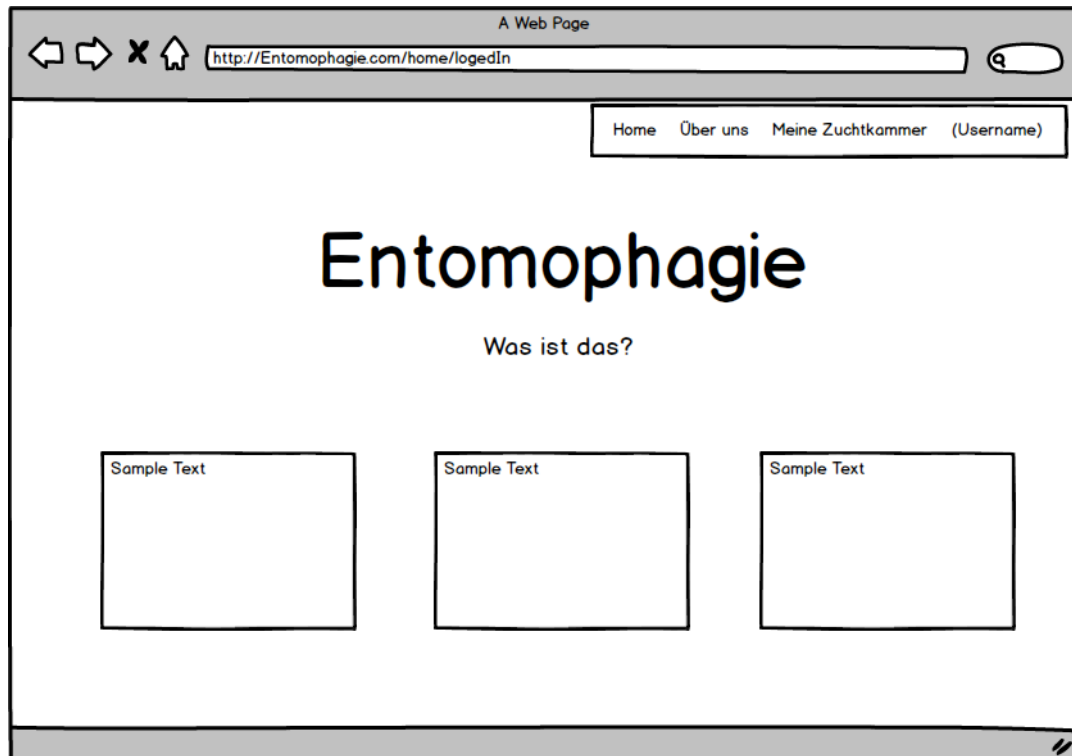


Abbildung 7.1: Mockup unserer Seite wenn man eingeloggt ist

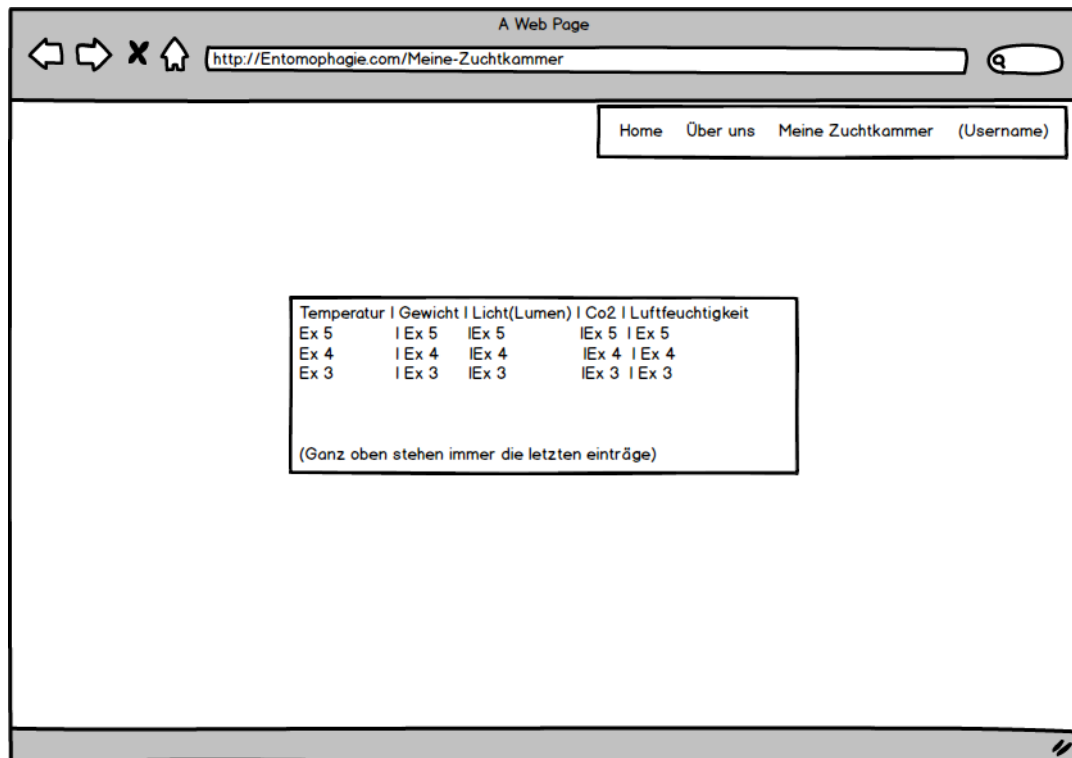


Abbildung 7.2: Mockup der Seite Meine-Zuchtkammer

Auf der Seite 'Meine Zuchtkammer' sieht die letzten Daten die die Zuchtkammer wiedergegeben hat. Diese sind in Absteigender Reihenfolge geordnet, dass heißt, dass der letzte Eintrag ganz oben steht.

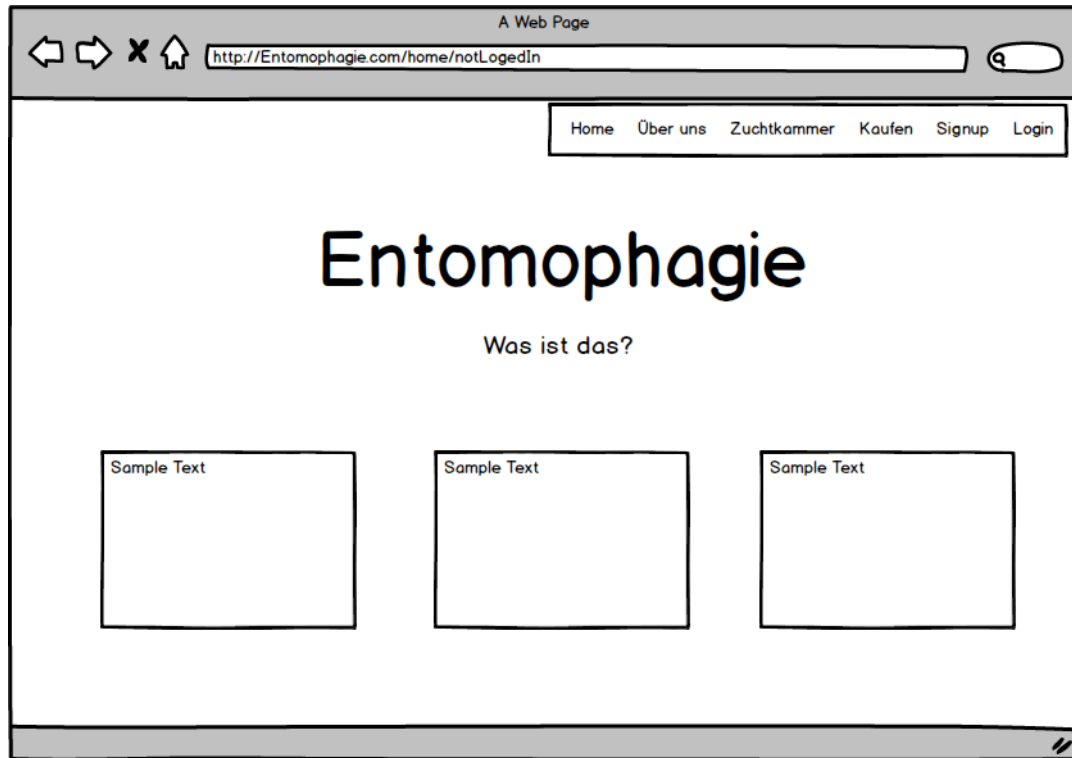


Abbildung 7.3: Mockup unserer Seite wenn man nicht eingeloggt ist

Hier Sieht man die Ansicht wenn man auf unserer Website nicht Angemeldet ist. Man kann über den Button 'Zuchtkammer' mehr über unsere Zuchtkammern herausfinden, weiter kann man über den Button 'Kaufen' seine eigene Zuchtkammer kaufen. Bei Signup kann man sich als neuer Benutzer registrieren, um sich allerdings zu registrieren braucht man zuerst eine Seriennummer die man beim kauf einer Zuchtkammer erhält.

### **7.1.2 Datenbankzugriffe**

Unserer Datenbank zugriffe werden von unserem Framework verarbeite, dabei verwendet das Framework CRUD befehle und arbeitet nach dem MVC Muster. Das heißt es gibt ein unterliegendes Modell welches Daten an unseren Controller mitgibt welche wiederum die View erzeugt.

Der gesamte Datenbankzugriff kann mittels Yii2 sehr einfach erstellt werden. Mehr dazu siehe Gii

### **7.1.3 Datenübermittlung**

Um die Daten von unsrem Arduino an unsere Datenbank zu senden werden wir in den Brutkasten ein WLAN Modul einbauen und die Daten alle 5-10 Minuten als JSON-Format an unsere REST-Schnittstelle unserer Webapp senden. Hierfür haben wir das WLAN Modul ESP8266 verwendet. Das hat den Vorteil das unsere Daten direkt von unserem Brutkasten an die Datenbank gegeben werden kann.

Für diese Lösung brauchen wir eine REST Schnittelle in unserer Webapp. Dies kann über YII2 sehr einfach realisiert werden. Yii2 hat eine vor implementierte REST Schnittelle welche man nur noch aktivieren muss. Die Daten werden aus der Modell Klasse ausgelesen und über den Controller an die REST-View weitergeleitet. Die ganze Implementierung sind 5-10 Zeilen Code. Die Code Sequenz sehen sie im Kapitel Rest Schnittelle.

### **7.1.4 Rest Schnittelle**

Hier sieht man die Ordnerstruktur. Wir erstellen im Obersten Verzeichnis einen neuen Ordner namens 'api' um im Web über folgende

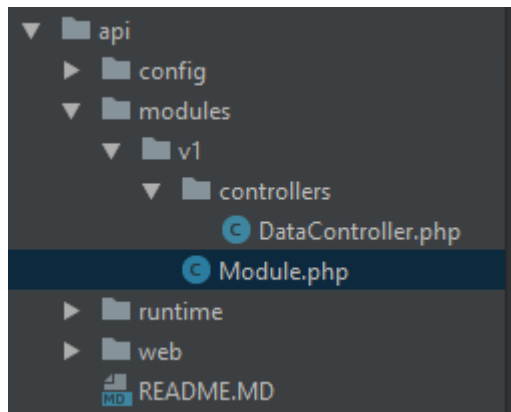


Abbildung 7.4: Abbildung des API Ordners

URL: 'www.entomophagie/api/web/v1/datas' auf unsere Restschnittelle Zugreifen können. Dabei müssen wir darauf achten das wir Groß- und Kleinschreibung beachten. Da es sonst passieren kann das wir keine Anzeige bekommen. Um dieses Problem zu lösen ist es am einfachsten alle Ordner in unserem Verzeichnis immer klein zuschreiben.

Weiters Kopieren wir 'web', 'config' und den 'runtime' Ordner aus unserem Front- bzw. Backend Ordner.

Quelltext 7.1: Controller Klasse für REST

```
1 <?php
2
3 namespace api\modules\v1\controllers;
4 use yii\rest\ActiveController;
5
6 class DataController extends Active Controller
7 {
8     public $modelClass = 'common\models\Data';
9 }
```

Hier sieht man die Controller Klasse. In der Controller Klasse müssen wir den Yii eigenen ActiveController einbinden und die Modell Klasse festlegen welche die Daten ausliest.

## Quelltext 7.2: Module Klasse für REST

```
1 <?php
2
3 namespace api\modules\v1;
4
5 class Module extends \yii\base\Module
6 {
7     public $controllerNamespace =
8         'api\modules\v1\controllers';
9
10    public function init()
11    {
12        parent::init();
13    }
14 }
```

Hier sieht man die Module Klasse. In der Module Klasse wird unser neuer Rest Controller initialisiert, damit unsere Anwendung diese Verwenden kann. Dort müssen wir unseren Controller den wir zuvor erstellt haben festlegen. Wir haben den Rest Controller mit folgendem Online Tutorial erstellt (9).

## 7.2 Verkablung

Die tatsächliche Verkablung der einzelnen Module war am Anfang recht komplex. Trotz des richtigen Setups unterliefen Fehler die verhinderten das Daten sinnvoll ausgelesen werden konnten. Verwendung von Widerständen führte zu falschen oder unrealistischen Daten, da der Strom der darüber lief zu gering war bzw. der Widerstand zu hoch.

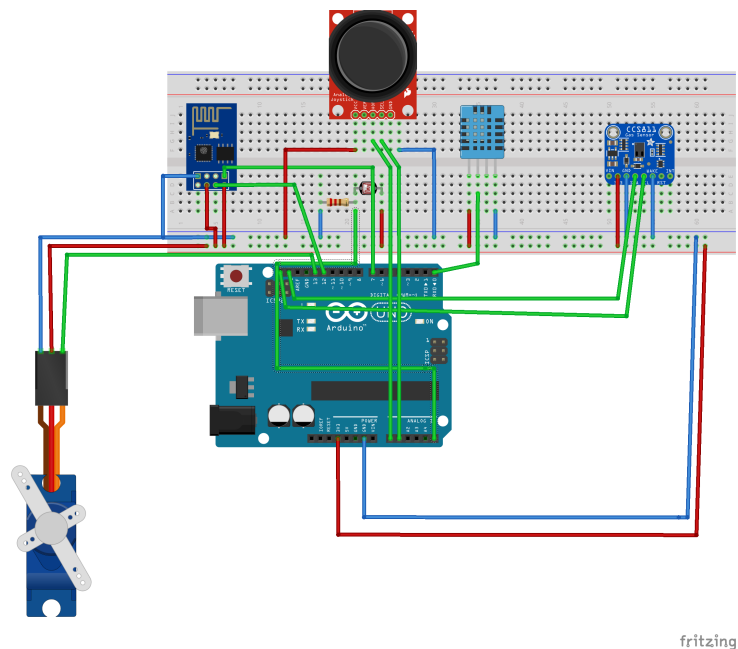


Abbildung 7.5: Verkablung aller Module

In dieser Grafik kann man die Verkablung aller Module sehen. Abgebildet von links anfangend:

- Servomotor SG0
- WLAN Modul ESP8266



- Joystick Modul (oben)
- Lichtempfindlichkeit (unten)
- Temperatur und Luftfeuchtigkeitssensor DHT11
- CO2 Sensor CCS811

Die größte Herausforderung bot der CCS811 sowie der ESP8266, da beide keine typische Verkabelung verwenden. Wie man an der Grafik erkennen kann werden mehrere Strom oder Datenquellen benötigt.

### **7.2.1 CCS811**

=====

## **7.3 Arduino**

Die Verkablung der einzelnen Module ist aus verschiedenen Gründen wichtig. Zum einen braucht jedes Modul genügend Strom um seine Aufgabe zu erfüllen, gleichzeitig würde es aber ohne die Erdung zu einem Kurzschluss kommen. Des weiteren ist es wichtig das die Module Daten auslesen und weitersenden können mit stromfluss einfügen) »»»»> 3e06afbafba374741d0d74fa5cbde2bc3cc60033

Der CCS811 benötigt weder digitale noch analoge Werte da er die Pins von der I2C Uhr und Datenlinien verwendet. Des weiteren ist es wichtig anzumerken das dieser Sensor nur mit Mikrocontroller funktioniert die intern I2C "clock stretching" unterstützen. In unserem Projekt mussten wir nur die Steckplätze SCL und SDA für den CO2 Sensor verwenden und es funktionierte.

### **7.3.1 ESP8266**

Da der ESP als eingestehende Netzwerkschnittstelle zählt braucht diese besonders viel Energie. Daher muss man beim Verkabeln vor allem darauf achten dass beide Stromanschlüsse richtig plaziert wurden. Ähnlich wie das CO2 Messmodul verwendet der WLAN Adapter 2 Datenpins. Diese sind allerdings beide digital und können an beliebigen stellen zwischen 0 und 13 gesetzt werden

### **7.3.2 Joystick Modul**

Der Joystick verwendet ursprünglich zwei analoge Pins und einen Analogen. Damit benötigt dieser Mikrocontroller die meisten Pins. Damit eine variable X und Y Achse ausgelesen werden kann benötigen wir die zwei analogen Pins. Der digitale Wert fungiert als Knopfdruck, dieser kann in unserem Projekt allerdings ignoriert werden. Auf die X Ache können wir hingegen nicht verzichten, da beide Werte benötigt werden um die Position auslesen zu können.

«««< HEAD

## **7.4 Programmierung**

=====

### **Bibliotheken importieren und globale Variablen definieren**

itHub oder direkt von Arduino zur verfügung gestellt bekommen. Auch werden alle Variablen die sowohl in den Methoden Setup und Loop verwendet werden

erstellt. äten werden nur ausgeführt wenn der Arduino startet bzw. zurückgesetzt wird. en definiert. Mit Hilfe dieser Eigenschaften, können wir zeitnahe alle Daten auslesen. Falls nun ein Wert eine Mindestgrenze kann sofort darauf reagiert werden. kt kann der Quellcode in drei Bereiche eingeteilt werden. »»»> 3e06afbabfa374741d0d74fa5cbde2bc3cc60033

### 7.4.1 DHT11 & CCS811

Für den DHT 11 sowie dem CCS811 gibt es eine von Adafruit frei verwendbare Bibliothek. Mithilfe von dieser kann man ein Objekt erstellen und es mithilfe diesem Objektes die Momentane Temperatur/Luftfeuchtigkeit auslesen.

Der Parameter `#include` fügt externe Klassen in unser Projekt ein und erlaubt es uns in das Objekt dht sowie ccs zu erstellen. Des weitere können wir mit dem Parameter `#define` relevante Daten festlegen. Ein solches Beispiel ist von welchem Pin Daten empfangen werden.

Quelltext 7.3: Datenauslesung CCS811

```
1
2 delay(2000);
3 if(ccs.available())
4 {
5     if(!ccs.readData())
6     {
7         Serial.print("CO2: ");
8         Serial.println(ccs.geteCO2());
9     }
10    else
11    {
12        Serial.println("ERROR!");
13        //while(1);
14    }
```

```
}
```

(10)

Die Loop Funktion, wiederholt sich im vorgegebenen Rhythmus immer wieder und gibt keine Variablen zurück. Dort werden alle Daten ausgelesen und an das WLAN Modul weitergeschickt bzw. in diesem Fall wird es auf den Serial Monitor ausgegeben.

Die WENN abfrage prüft als erstes ob der CO2 Sensor kalibriert wurde oder nicht. Falls es gelungen sein sollte gibt es die Werte mit dem Befehl `Serial.println(ccs.getCO2());` im Serial Monitor aus.

### 7.4.2 Hebel

Das Joystick bzw. der Hebel konnte direkt abgelesen werden. Wichtig hierbei ist es das man nicht nur eine Achse im Setup verwendet sondern beide, ansonsten kommt es bei dem Loop zu einem Fehler und der Wert der benötigten Achse verändert sich nicht.

### 7.4.3 SG90

Die vom verwendeten Bibliotheken waren bei der Installierung der Arduino eigenen IDE direkt dabei. Die Servo kontrolliert die Luftzufuhr und muss daher Werte vom CCS811 wiederverwenden

Quelltext 7.4: Automatisierten Servobewegung bei zu niedrigen CO2 Werten

1

```
2 if(ccs.getCO2() <= co2Min)
```

```
3 {
4     //move the micro servo from 0 degrees to 180
        degrees
5     for(; servoAngle < 180; servoAngle++)
6     {
7         servo.write(servoAngle);
8         delay(10);
9     }
10 }
11 if (ccs.getCO2() > co2Min && servoAngle != 0)
12 {
13     servo.write(45);
14     servoAngle = 0;
15     Serial.println("RETURN");
16 }
```

(6)

Die If Abfrage prüft ob CO2 einen Mindestwert (co2Min) unterschreitet. Falls das passiert wird eine Schleife ausgeführt die den Servo 180° dreht. Diese 180° würde die Lüftungsklappe aufhalten. Falls der CO2 Wert wieder in einem akzeptablen Bereich liegt und die Position des Motors nicht null beträgt, wird die zweite Schleife aktiviert die den Motor zur Position 0 zurückbringt

## 7.4.4 Wlan

Für die Implementierung der Wlan anbiendung haben wir ein ESP8266 verwendet. Das Modul ist sehr einfach einzurichten das ganze haben wir mit den Beispielen von den ESP8266 in der Arduino IDE gemacht.

### ESP8266

Das Wlan Modul ESP866 wird verwendet um die Verbindung mit dem Internet aufzubauen und die Daten mithilfe der Rest Schnittelle an unsere Website senden. Dafür haben wir die ESP8266 Library von Arduino verwendet.(? )

Quelltext 7.5: Verbindungsaufbau des ESP8266

```
1 #include <ESP8266.h>
2
3 const char* ssid = "SSID des Benutzers";
4 const char* password = "password des Benutzers";
5
6 const char* host = "Unsere Website";
7
8 void setup()
9 {
10     Serial.begin(115200);
11     dela(10);
12
13     Serial.println();
14     Serial.println();
15     Serial.println("Connecting to ");
16     Serial.println(ssid);
17
18
19     WiFi.mode(WIFI_STA);
```

```
20     WiFi.begin(ssid,password);  
21 }
```

Hier sieht man den Verbindungsaufbau des ESP8266. In der Zeile WIFI.begin wird dann die Verbindung gestartet vorher übergeben wir die SSID und das Password des WLAN womit das Modul sich verbinden soll.

Quelltext 7.6: Verbindungsaufbau Response

```
1 while(WiFi.status() != WL_CONNECTED)
2 {
3     delay(500);
4     Serial.print(".");
5 }
6
7 Serial.println("");
8 Serial.println("WiFi connected");
9 Serial.println("IP address: ");
10 Serial.println(WiFi.localIP());
```

Bei erfolgreicher Verbindung wird die IP Adresse unseres Moduls ausgegeben.

Quelltext 7.7: Ausführung der REST Befehle

```
1 client.print(String("GET ") + url + " HTTP/1.1\r\n") +
2 "Host: " + host + "\r\n" +
3 "Connection: close\r\n\r\n");
4 unsigned long timeout = millis();
5 while(millis() - timeout > 5000)
6 {
7     Serial.println(">>> Client Timeout !");
8     client.stop();
9     return;
10 }
```

Anschließend können wir in unserer Loop REST Befehle wie Get, Post, Put, etc ausführen.



# 8 Deployment

- Umsetzung der Ausführungsumgebung
- Deployment
- DevOps-Thema

# 9 Tests

## 9.1 Systemtests

Systemtests aller implementierten Funktionalitäten lt. Pflichtenheft

- Beschreibung der Teststrategie
- Testfall 1
- Testfall 2
- Testfall 3
- ...

## 9.2 Akzeptanztests

# **10 Projektevaluation**

siehe Projektmanagement-Unterricht

# **11 Benutzerhandbuch**

falls im Projekt gefordert

# 12 Betriebswirtschaftlicher Kontext

## 12.0.1 Arduino in der Wirtschaft

Mikrocontroller gewinnen vor allem beim Endverbraucher immer mehr an Qualität, allerdings wird auch im Landwirtschaftlichen Bereich von diesen Technologien Gebrauch gemacht. In dem Artikel *Ärduino Projekte in der Landwirtschaft und Fischzucht* wird beschrieben wie Privatpersonen ihre eigene Lösung zu Problemen wie Fischzucht, Lenksysteme für Erntemaschinen und weitere Fälle gestalten. (7)

Für einen Insekten Zuchtkasten existiert in der westlichen Welt keine große Nachfrage, im Asiatischen Raum könnte ein solches Produkt hingegen große Popularität gewinnen. In der Kostenrechnung werden die einzelnen Teile nochmals genauer besprochen, allerdings ist der Anschaffungswert alle benötigten Teile unter € 100.

## 12.0.2 Kostenrechnung

Die Kostenrechnung ist ein mächtiges Werkzeug die mehrere wichtige Aufgabenbereiche deckt. Dazu zählen unter anderem Grundlagen der Preisbildung,

## *Entomophagie*

Verwendung als Entscheidungsinstrument oder auch als Planungsinstrument. Grundlage für die gängigen Kostenrechnungsarten besteht aus der Istkostenrechnung, diese listet alle tatsächlich angefallenen Kosten eines Produktes oder einer Periode auf. ( ? )

## **12.1 Nutzen einer Website**

Es ist immer wichtiger für Unternehmen eine Website zu besitzen den für viele Personen ist das Internet der erste Ansprechpartner um Informationen zu erhalten. Dabei ist es wichtig das man die gewünschte Website schnell und einfach findet. Dafür ist eine gute Search engine optimization(SEO) wichtig, damit die Website in Google auf der Ersten Seite angezeigt wird. Das ist deshalb wichtig da die zweite Ergebnis Seite von Google oft ignoriert wird.

Auch in unserem Projekt haben wir eine Website erstellt, diese ist allerdings eine Reine Funktionswebsite. Deshalb kann man auf unserer Website erst nur Daten seines Brutkasten auslesen. Man könnte die Website noch mit einem Webshop erweitern, da dies allerdings den Rahmen des Projektes sprengen würde haben wir dies nicht gemacht.

Auf Websites ist es auch wichtig dass das Unternehmen widergespiegelt wird, deshalb sollte man dort möglichst ein Corporate Design anwenden falls eines vorhanden ist. Den in der heutigen Modernen Zeit ist es für viele der Erste Eindruck über ein Unternehmen.

# 13 Zusammenfassung

- Etwas längere Form des Abstracts
- Detaillierte Beschreibung des Outputs der Arbeit



# Literaturverzeichnis

- [1] W3schools. URL: <https://www.w3schools.com/>.
- [2] Web hypertext application technology working group. URL: <https://whatwg.org/>.
- [3] World wide web consortium. URL: <https://www.w3.org/>.
- [4] Arduino. analogread(). URL: <https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/>.
- [5] Arduino. Mqgassensor. URL: <https://playground.arduino.cc/Main/MQGasSensors>.
- [6] Calin Dragos. Tutorial: How to control the tower pro sg90 servo with arduino uno, October 2. URL: <https://www.intorobotics.com/tutorial-how-to-control-the-tower-pro-sg90-servo-with-arduino-uno/>.
- [7] Thomas Fischer. Arduino projekte in der landwirtschaft und fischzucht, June 2017. URL: <https://www.expertenderit.de/blog/arduino-projekte-in-der-landwirtschaft-und-fischzucht>.
- [8] LTD HANWEI ELETRONICS CO. Technical data mq-2 gas sensor. URL: <https://www.mouser.com/ds/2/321/605-00008-MQ-2-Datasheet-370464.pdf>.

- [9] Budi Irwan. Setup restful api in yii2, July 2014. URL: <http://budiirawan.com/setup-restful-api-yii2/>.
- [10] Dean Miller. Adafruit ccs811 air quality sensor, January 2018. URL: <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ccs811-air-quality-sensor.pdf>.
- [11] PureMVC. Implementation idioms and best practices. URL: <http://puremvc.org/>.
- [12] Wikipedia. Crud. URL: <https://de.wikipedia.org/wiki/CRUD>.
- [13] Wikipedia. Kohäsion. URL: [https://de.wikipedia.org/wiki/Koh%C3%A4sion\\_\(Informatik\)](https://de.wikipedia.org/wiki/Koh%C3%A4sion_(Informatik)).
- [14] Wikipedia. Lose kopplung. URL: [https://de.wikipedia.org/wiki/Lose\\_Kopplung](https://de.wikipedia.org/wiki/Lose_Kopplung).
- [15] Wikipedia. Model view controllers. URL: [https://de.wikipedia.org/wiki/Model\\_View\\_Controller](https://de.wikipedia.org/wiki/Model_View_Controller).
- [16] Wikipedia. Web 2.0. URL: [https://en.wikipedia.org/wiki/Web\\_2.0](https://en.wikipedia.org/wiki/Web_2.0).