Diplomarbeit

Entomophagie

Untertitel der Arbeit

Imst, 21. 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 verbwendet 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

z. B. Hinweise, wie das bearbeitete Thema gefunden wurde oder Dank für die Betreuung (Kooperationspartner/in, Betreuer/innen, Sponsoren) etc.

Abstract (Deutsch)

(ca. ½ bis max. 2 Seiten) Kurzbeschreibung von Aufgabenstellung und Problemlösung.

Abstract (Englisch)

(ca. ½ bis max. 2 Seiten)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltexte

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

2.4.2 Meilensteine

2.4.3 Gant-Chart

2.4.4 Abnahmekriterien

2.4.5 Pläne zur Evaluierung

2.4.6 Ergänzungen und zu klärende Punkte

3 Vorstellung des Produktes

Vorstellung des fertigen Produktes anhand von Screenshots, Bildern, Erklärungen.

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 aus 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 (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(15)

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 umsetzten.

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-Practicse Beispiele in vielen verschiedenen Sprachen. Diese findet man auf der Website. (10)

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. (9)

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

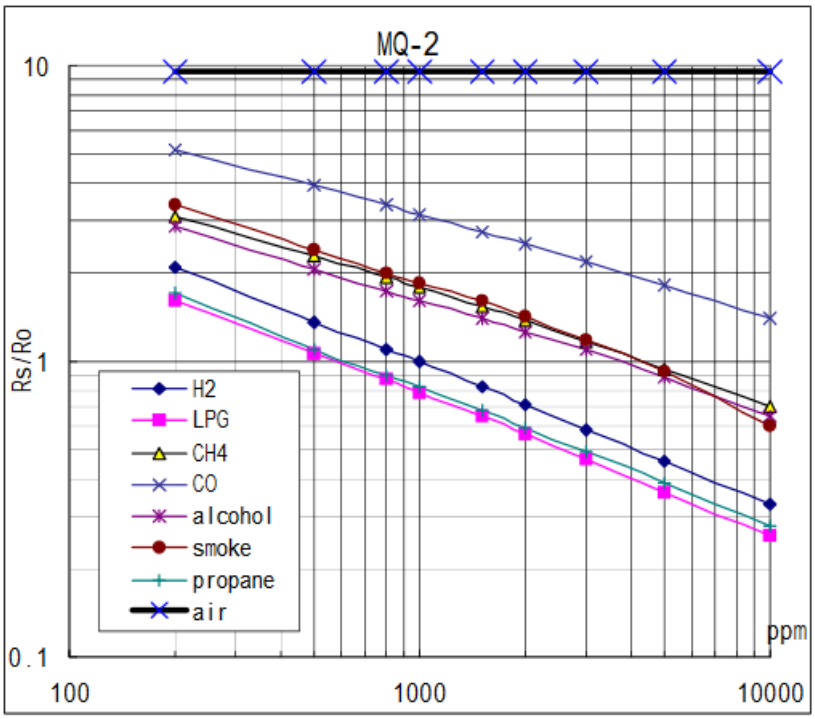
Für die Auswertung der Gaswerte stehen uns mehrere Module zur Verfügung. Zum einen stand uns die vielfältige MQ-Serie zur Verfügung oder der spezialisierter CCS811 von Adafruit.

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: (4)



(6)

(4)

Im Datasheet (6) kann man herauslesen das der Sensor MQ2 (4) 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 eCO2 in einem Bereich von 400 bis 8192 ppm (parts per million) auszulesen. (8)

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. (8)

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 CO2 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

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. (14)

Vor- und Nachteile

Vorteile	Nachteile
Gleichzeitiges Programmieren	Schlechte Übersicht
Hohe Kohäsion (12)	Konsistente Programmierung notwendig
Lose Kopplung (13)	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 viel 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 Datenbankanbindung:

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

(11)
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.(15)

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 weiterm Verlauf dann nach eigenen Wünschen verändern.

Vor- und Nachteile

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

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

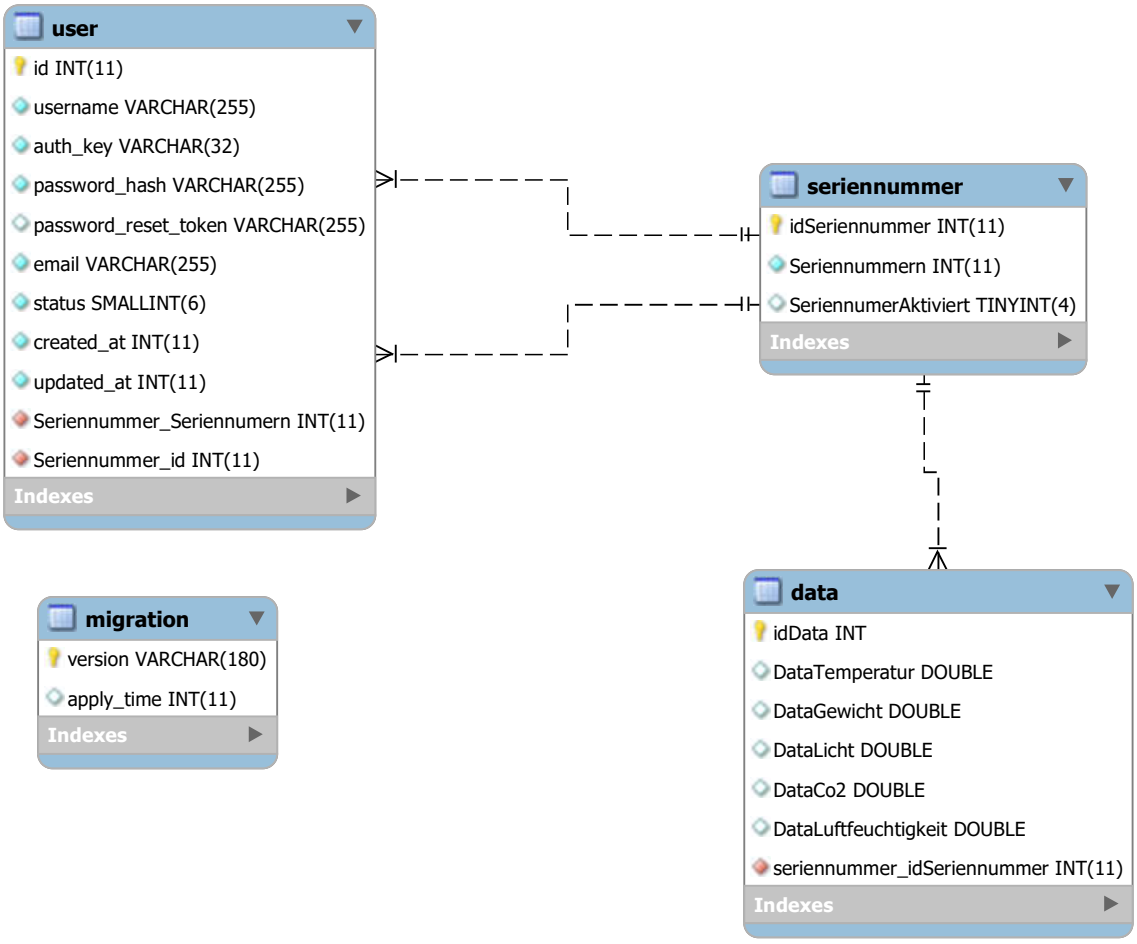
Nachteile ist der Hohe Setup aufwand und die benötigte Kenntnisse in PHP und SQL, welche allerdings bei fast allen Frameworks von Nöten sind.

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 Datenhaltunskonzept

Hier ist unser ER-Diagramm


user
id INT(11)
username VARCHAR(255)
auth_key VARCHAR(32)
password_hash VARCHAR(255)
password_reset_token VARCHAR(255)
email VARCHAR(255)
status SMALLINT(6)
created_at INT(11)
updated_at INT(11)
Seriennummer_Seriennumern INT(11)
Seriennummer_id INT(11)
Indexes
seriennummer
idSeriennummer INT(11)
Seriennummern INT(11)
SeriennumerAktiviert TINYINT(4)
Indexes
migration
version VARCHAR(180)
apply_time INT(11)
Indexes
data
idData INT
DataTemperatur DOUBLE
DataGewicht DOUBLE
DataLicht DOUBLE
DataCo2 DOUBLE
DataLuftfeuchtigkeit DOUBLE
seriennummer_idSeriennummer INT(11)
Indexes

ten 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 Zeitpunk 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

Design jedes einzelnen USE-Cases

- Design-Klassendiagramme vom Domain-Klassendiagramm ableiten (incl. detaillierter Darstellung und Verwendung von Vererbungshierarchichen, 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

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

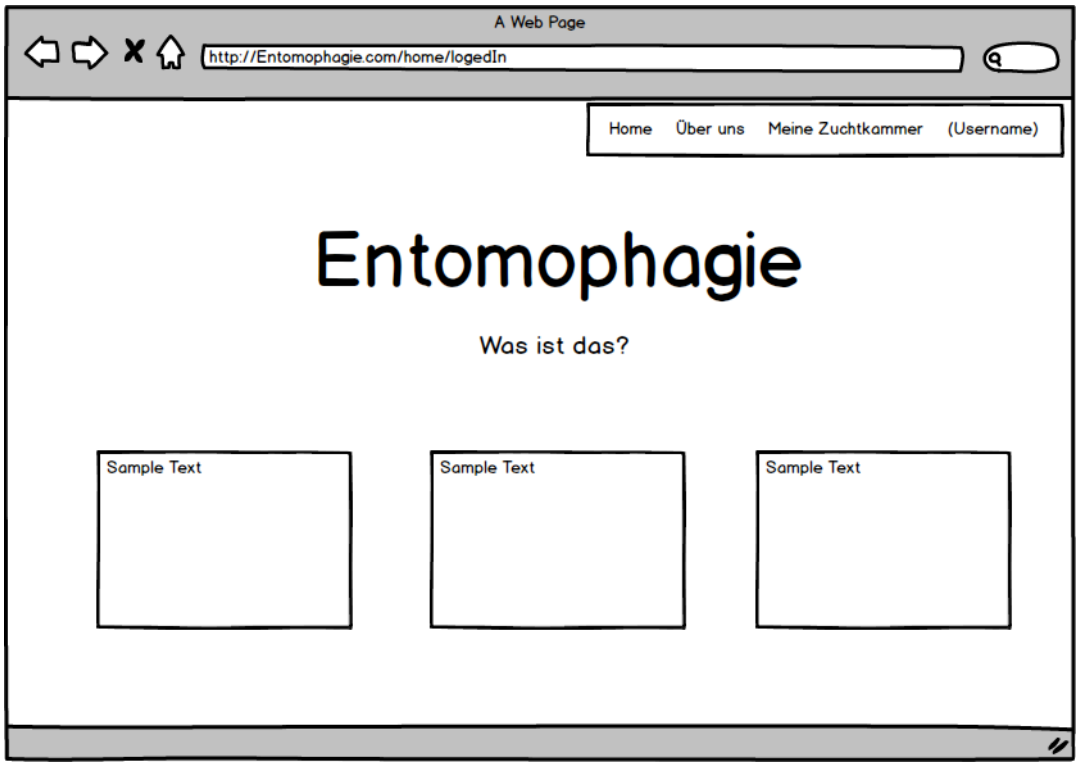
7.1 Webapp

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:

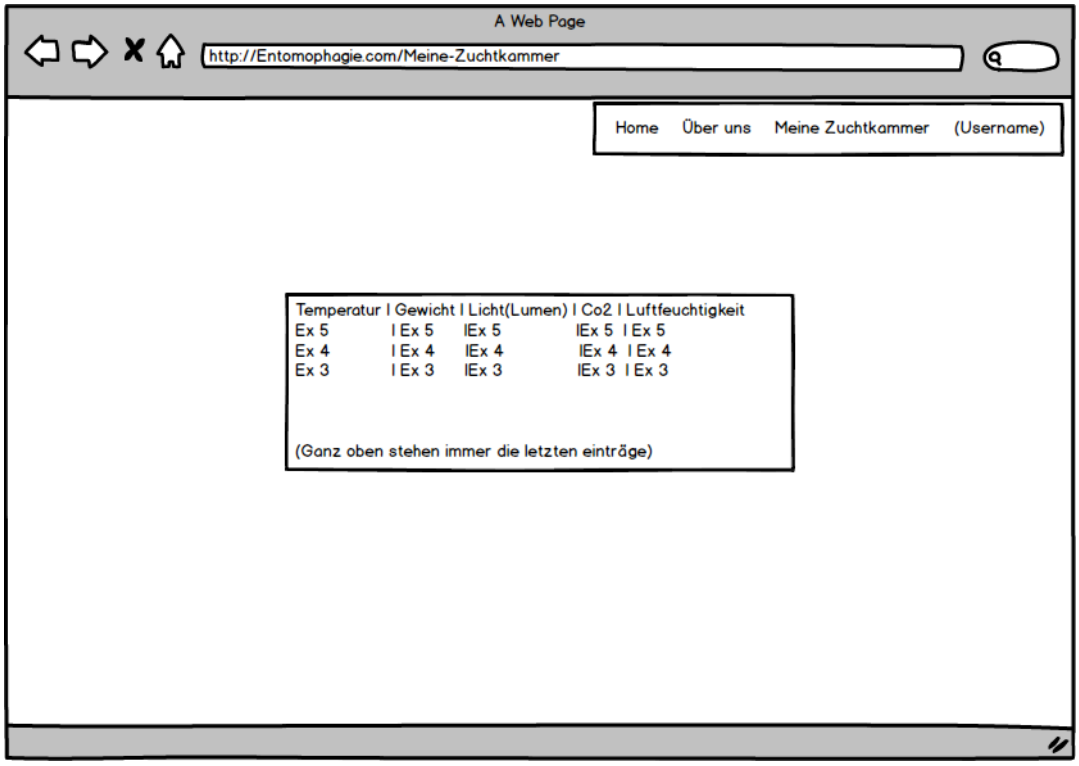
- Sauerstoff
- Luftfeuchtigkeit
- Gewicht
- Temperatur
- Futtermenge
- ungefähre Zeit bis zu Reife

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

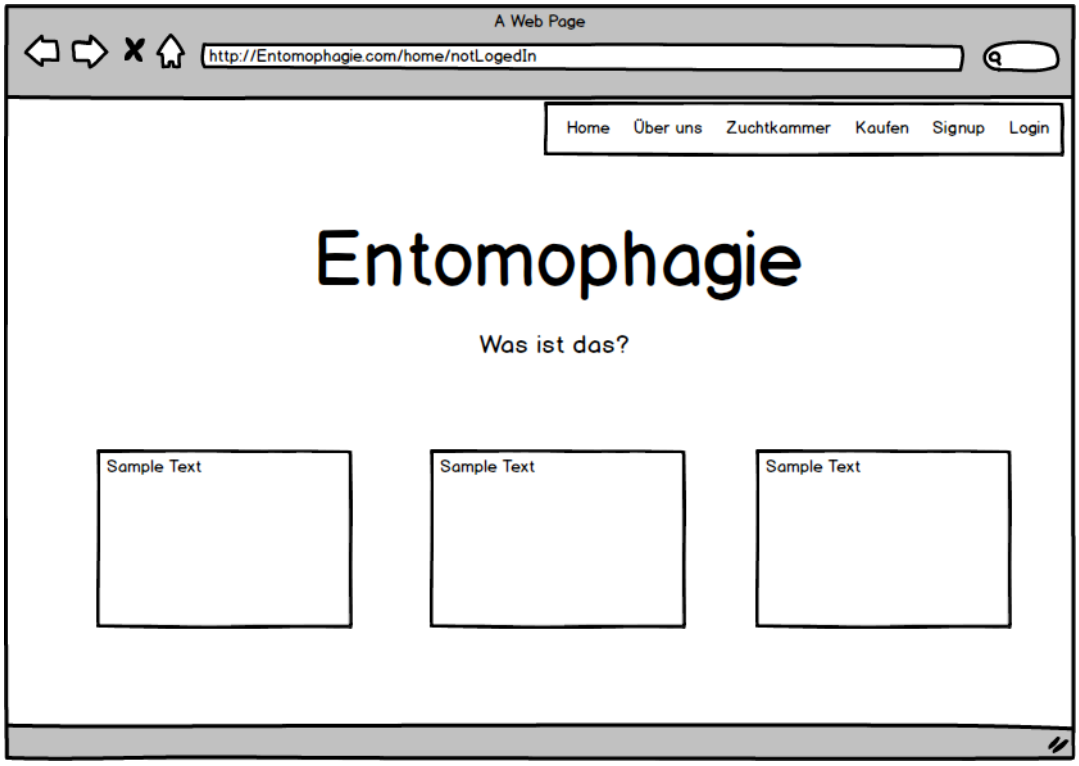
7.1.1 Mockups



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.



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.



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 brauch man zuerst eine Seriennummer die man beim kauf einer Zuchtkammer erhält.

7.1.2 Datenbankzugriffe

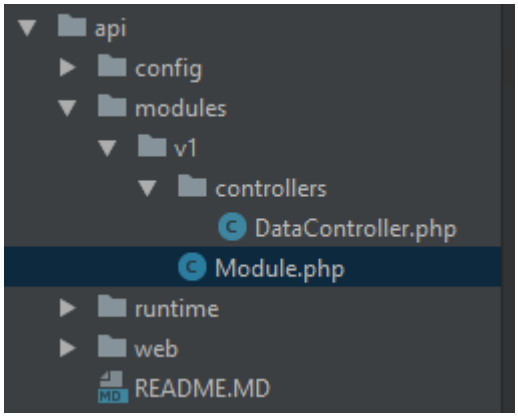
Unserer Datenbank zugriffe werden von unserem Framework verarbeite, dabei verwendet das Framework CRUD befehle und arbeitet nach dem MVC Muster. Das heißt es gibt ein unterliegendes Modell welches Daten an unseren Controller mitgibt welche wiederum die View erzeugt.

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 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.

```
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.

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

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 (7).

7.2 Arduino

7.3 Verkablung

Die Verkablung der einzelnen Module ist aus verschiedenen Gründen wichtig. Zum einen brauch 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

7.3.1 Aufbau

(bild einfügen)

In Abbildung x können wir nun den Arduino und das Breadboard sehen. Dort kann man direkt die drei Farben erkennen die jeweils für Stromzufuhr (blau), Datenauslesung (grün) und Erdung (braun) stehen.

(bild mit stromfluss einfügen)

Hier sieht man wie der Strom im Breadboard fließt. Das heißt wenn man in der Untersten Reihe das Board mit 5 Volt versorgt sind alle weiteren Steckplätze verwendbar. Wichtig hierbei ist, dass das Kabel für die Stromzufuhr auch auf der richtigen Reihe platziert wurde. Das erkennt man bei dieser Leiterplatte an dem Plus und Minus Zeichen Plus steht hierbei für Strom und Minus für die Erdung.

Falls man das Falsch verkabelt funktioniert das Modul nicht, es werden allerdings keine Schäden angerichtet

(Bild mit analog und digitalwerten einfügen)

Der Arduino hat zwei Möglichkeiten Daten auszulesen. Einmal spricht man hier von digitaler und analogen Datenauslesung. (internet suche) Digitale Werte bedeutet hier allerdings nur das der ausgelesene Wert als 0 oder 1 abgespeichert werden kann, analoge Daten sind bei diesem Punkt anders, da diese einen Wert zwischen 0 und x haben können (zitat)

7.4 Programmierung

Um die Daten der einzelnen Sensoren verwenden zu können, müssen diese ausgelesen und sinnvoll wiedergegeben werden. Nur sobald die Sensoren richtig Programmiert und verkabelt sind, kriegt man lesbare Information. Diese können anschließend weiter interpretiert und verwendet werden. In diesem Projekt kann der Quellcode in drei Bereiche eingeteilt werden.

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 ̈delayïn 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.

7.4.1 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.

```
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    }
15 }
```

(8)

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.geteCO2()); 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

```
1
2  if(ccs.geteCO2() <= 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.geteCO2() > co2Min && servoAngle != 0)
12 {
13    servo.write(45);
14    servoAngle = 0;
15    Serial.println("RETURN");
16 }
```

(5)

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.

```
#include <ESP8266.h>

const char* ssid = "SSID des Benutzers";
const char* password = "password des Benutzers";

const char* host = "Unsere Website";

void setup()
{
   Serial.begin(115200);
   dela(10);

   Serial.println();
   Serial.println();
   Serial.println("Connecting to ");
   Serial.println(ssid);


   WiFi.mode(WIFI_STA);
   WiFi.begin(ssid,password);
}
```

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

```
while(WiFi.status() != WL_CONNECTED)
{
   delay(500);
   Serial.print(".");
}

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP adress: ");
Serial.println(WiFi.localIP());
```

Bei erfolgreicher Verbindung wird die IP Adresse unseres Moduls ausgegeben.

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

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
- Tesfall 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 Fischzuchtẅird beschrieben wie Privatpersonen ihre eigene Lösung zu Problemen wie Fischzucht, Lenksysteme für Erntemaschinen und weitere Fälle gestalten. (**?**)

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

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. Mqgassensor. URL: https://playground.arduino.cc/Main/MQGasSensors.

[5] Calin Dragos. Tutorial: How to control the tower pro sg90 servo with arduino uno, October 2016. URL: https://www.intorobotics.com/tutorial-how-to-control-the-tower-pro-sg90-servo-with-arduino-uno/.

[6] 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.

[7] Budi Irwan. Setup restful api in yii2, July 2014. URL: http://budiirawan.com/setup-restful-api-yii2/.
«««< HEAD

[8] Dean Miller. Adafruit ccs811 air quality sensor, January

2018. URL: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ccs811-air-quality-sensor.pdf. =======

[9] Laravel. Documentation. URL: https://laravel.com/docs/5.6. »»»> 31a14c04eb76a54432c9493d6eddb6c490aab7a9

[10] PureMVC. Implementation idioms and best practices. URL: http://puremvc.org/.

[11] Wikipedia. Crud. URL: https://de.wikipedia.org/wiki/CRUD.

[12] Wikipedia. Kohäsion. URL: https://de.wikipedia.org/wiki/Koh%C3%A4sion_(Informatik).

[13] Wikipedia. Lose kopplung. URL: https://de.wikipedia.org/wiki/Lose_Kopplung.

[14] Wikipedia. Model view controllers. URL: https://de.wikipedia.org/wiki/Model_View_Controller.

[15] Wikipedia. Web 2.0. URL: https://en.wikipedia.org/wiki/Web_2.0.