Diplomarbeit

Entomophagie

Untertitel der Arbeit

Imst, 8. März 2018

Eingereicht von

Leonid Hammer	Verantwortlich für IT: HTML, CSS, BWL: Kaufvertrag
Kevin Glatz	Verantwortlich für IT: SQL, C BWL: Kaufvertrag
Florian Tipotsch	Verantwortlich für IT: SQL, C BWL: Kaufvertrag

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

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(12)

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 (7)

4.1.4 Laravel

Laravel

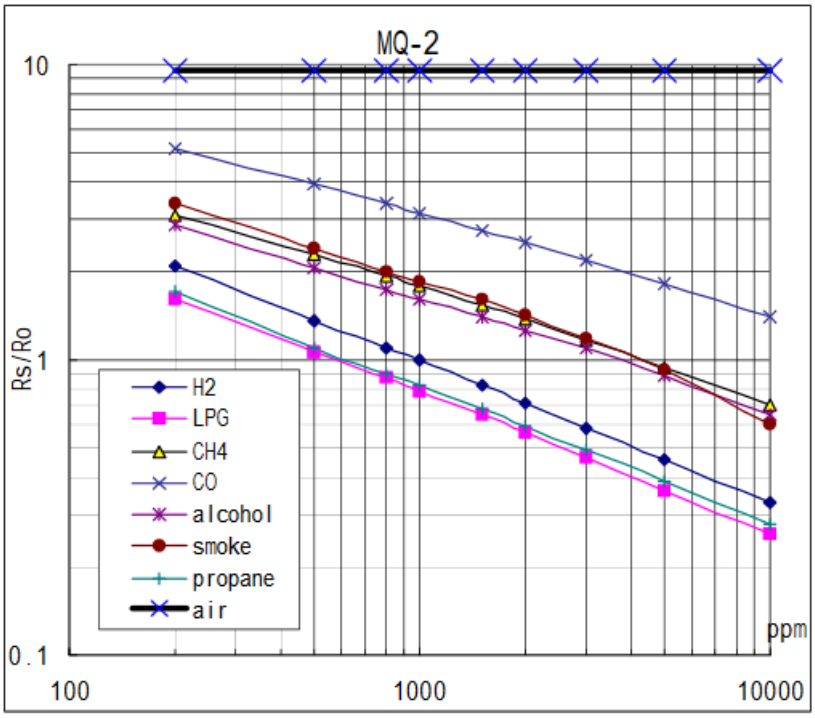
4.2 Gas-Sensoren

4.2.1 MQ Gas Sensoren

Es gibt mehrere MQ Gas Sensoren zum Beispiel:

- 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)

In der Schule haben wir den MQ2 zur Verfügung stehend werden wir auch von der Schule den Adafruit CCS811 bereitgestellt bekommen. Wir bedanken uns dafür vielmals.



(5)

(4)

Im Datasheet (5) 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.

4.2.2 Adafruit CCS811

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. Zum Beispiel für PHP, JAVA, C# und Ruby Anwendungen. (11)

Vor- und Nachteile

Vorteile	Nachteile
Gleichzeitiges Programmieren	Schlechte Übersicht
Hohe Kohäsion (9)	Konsistente Programmierung notwendig
Lose Kopplung (10)	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

(8) Sie sind verantwortlich für alle Datenbankzugriffe 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.(12)

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

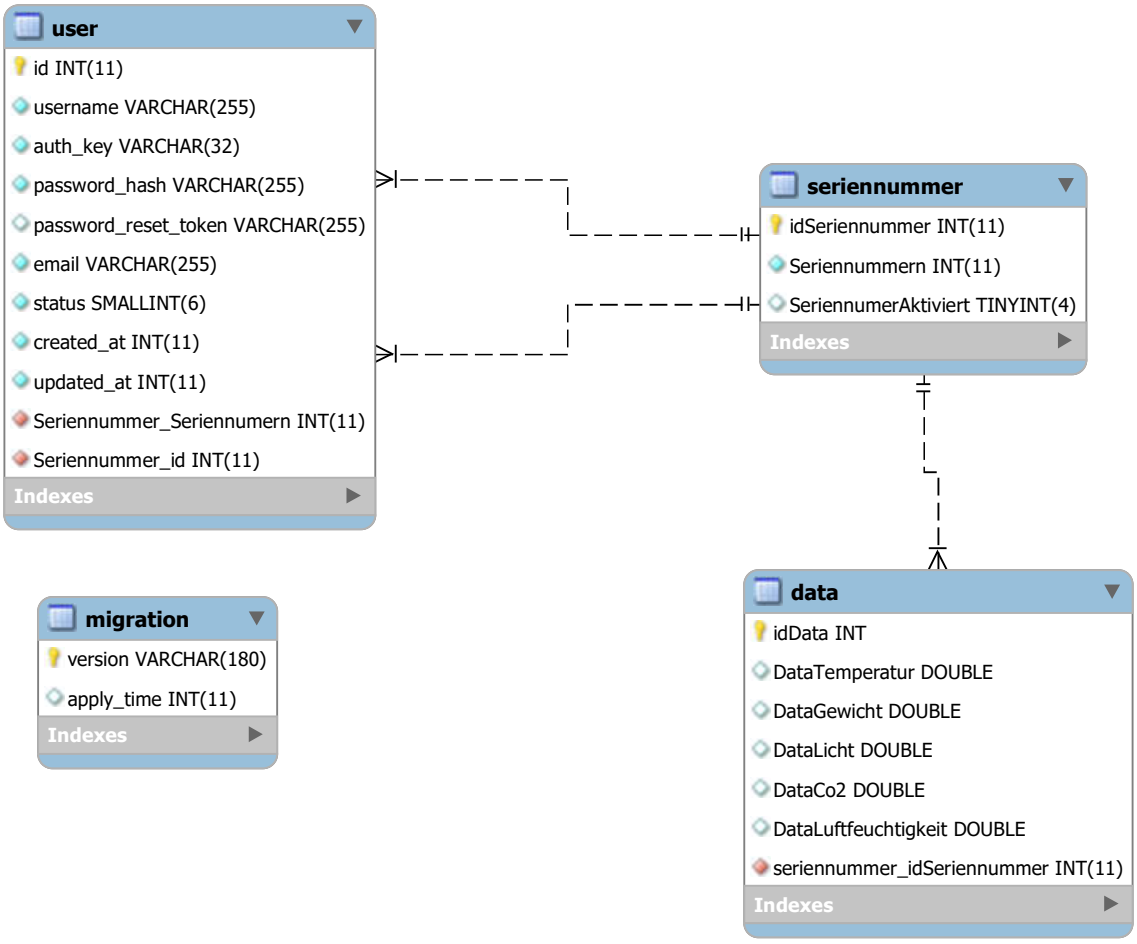
Nachteile ist der Hohe Setup aufwand und die benötigte Kenntnisse in PHP und SQL.

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 auf dem Kasten Registrieren, und dadurch auf die Daten zugreifen welche für die jeweilige Seriennummer gespeichert sind zugreifen.
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

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 auf einen Raspberry Pi 3 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

Beschreibung aller sicherheitsrelevanten Designentscheidungen

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)

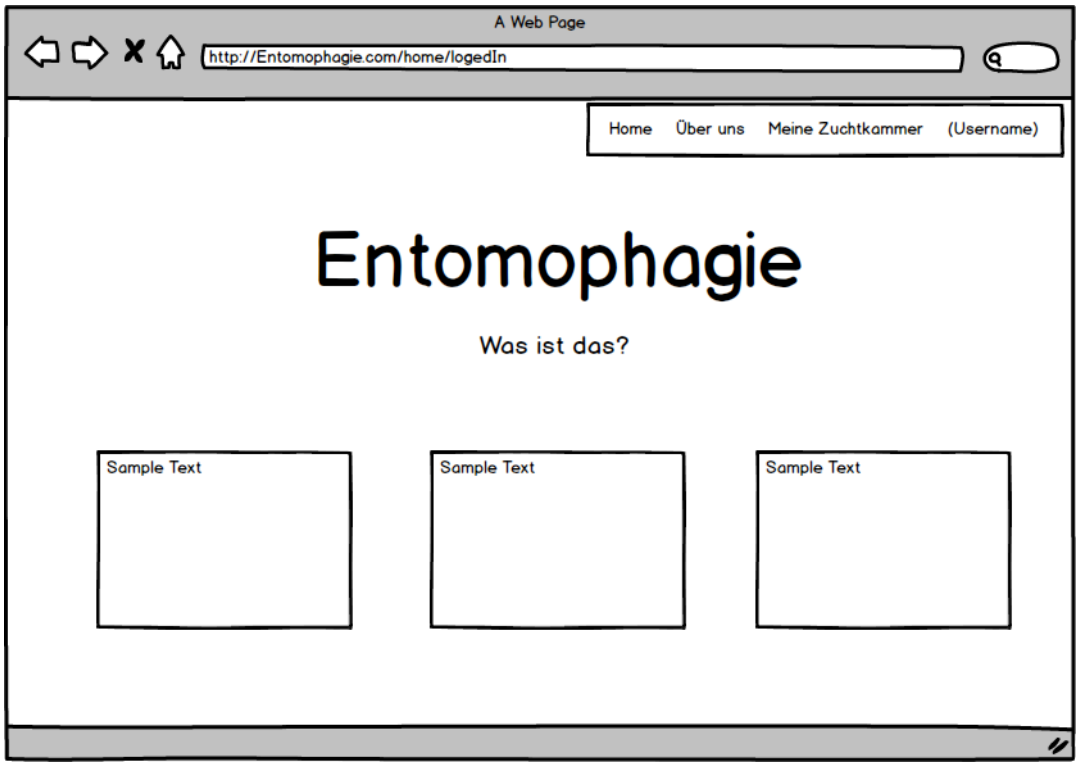
7.1 Webapp

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:

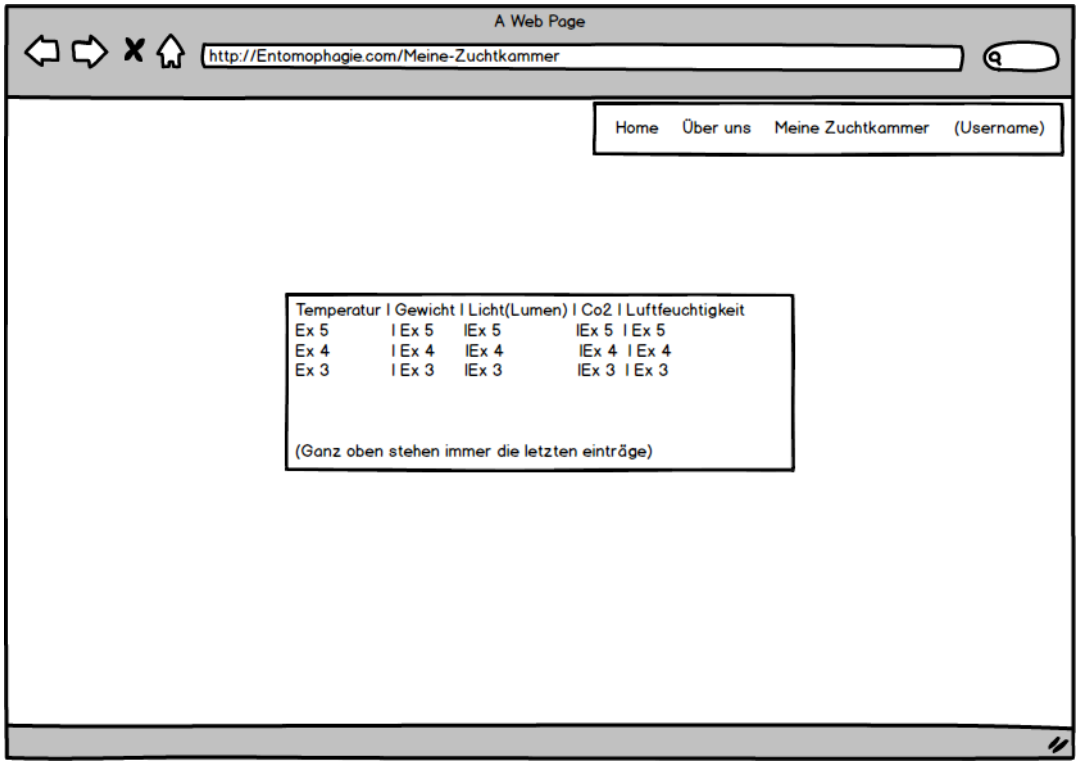
- Sauerstoff
- Luftfeuchtigkeit
- Gewicht
- Temperatur
- Futtermenge
- ungefähre Zeit bis zu Reife

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

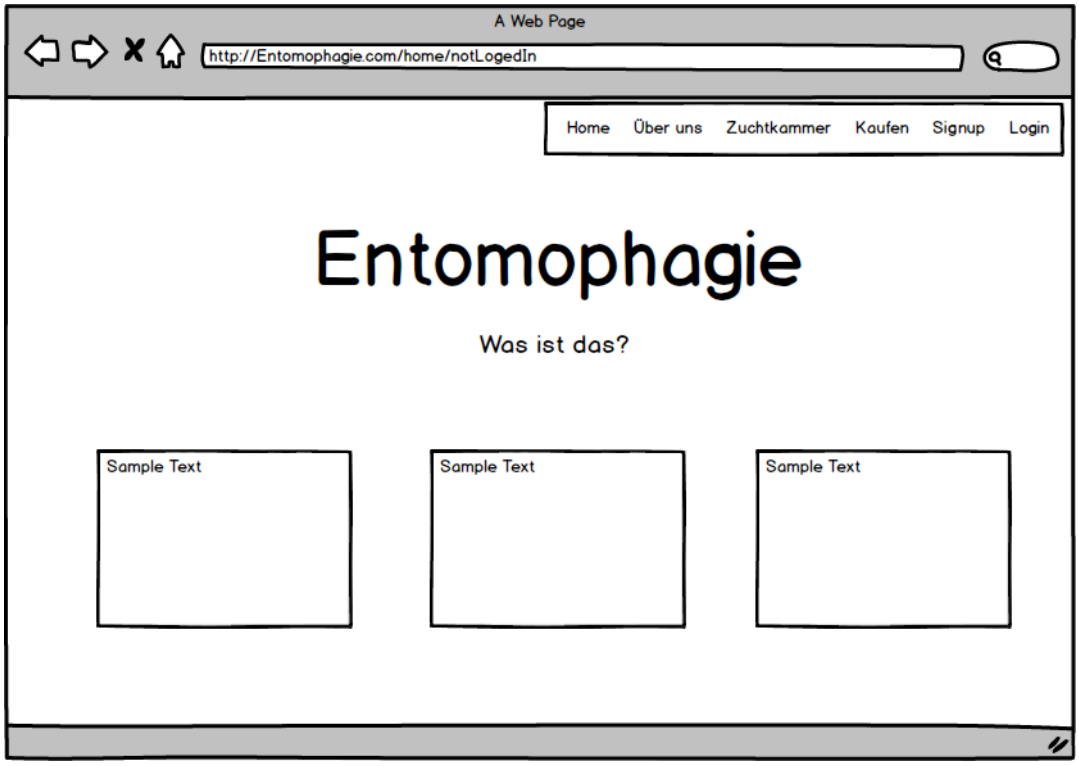
7.1.1 Mockups



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 Zuchtkammerßieht die letzten Daten die meine 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 SZuchtkammer"mehr über unsere Zuchtkammern herausfinden, weiter kann man über den Button "Kaufenßeine 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

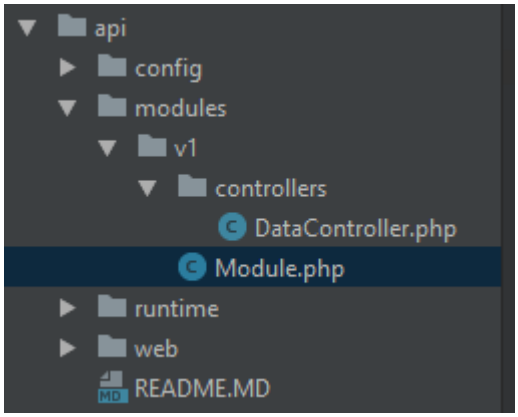
Unserer Datenbank zugriffe werden von unserem Framework verarbeite, dabei verwendet das Framework CRUD befehle und nach dem MVC Muster. Das heißt es gibt ein unterliegendes Modell welches Daten unserem Controller mitgibt welche wiederum die View erzeugt.

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

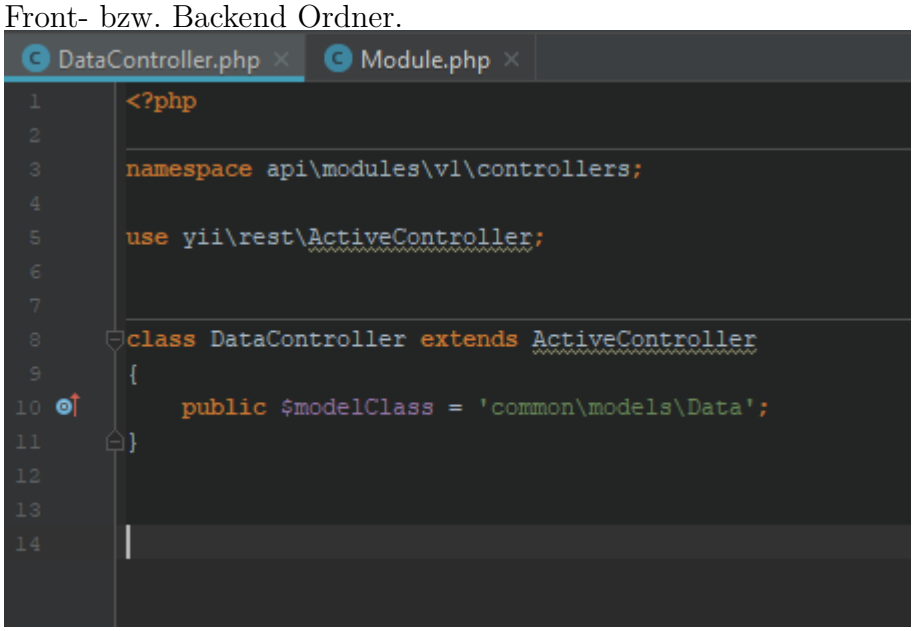
7.1.3 Datenübermittlung

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. 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.



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.

DataController.php | Module.php

```php
<?php
namespace api\modules\v1;

class Module extends \yii\base\Module
{
    public $controllerNamespace = 'api\modules\v1\controllers';

    public function init()
    {
        parent::init();
    }
}
```

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 (6).

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

BW-Teil

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] 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.

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

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

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

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

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

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

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