



## Diplomarbeit

# Entomophagie

### Untertitel der Arbeit

Imst, 25. Januar 2018

Eingereicht von

Leonid Hammer

Kevin Glatz

Tobias Haslwanter

Florian Tipotsch

Verantwortlich für IT: HTML, CSS, BWL: Kaufvertrag

Verantwortlich für IT: SQL, C# BWL: Kaufvertrag

Verantwortlich für IT: HTML, CSS, BWL: Kaufvertrag

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

---

Tobias Haslwanter

---

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.  $\frac{1}{2}$  bis max. 2 Seiten) Kurzbeschreibung von Aufgabenstellung und Problemlösung.

# **Abstract (Englisch)**

(ca. ½ bis max. 2 Seiten)

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>10</b>
<b>Tabellenverzeichnis</b>	<b>11</b>
<b>Quelltexte</b>	<b>12</b>
<b>1 Einleitung</b>	<b>15</b>
<b>2 Projektmanagement</b>	<b>16</b>
2.1 Metainformationen . . . . .	16
2.1.1 Team . . . . .	16
2.1.2 Betreuer . . . . .	16
2.1.3 Partner . . . . .	16
2.1.4 Ansprechpartner . . . . .	16
2.2 Vorerhebungen . . . . .	16
2.2.1 Projektzieleplan . . . . .	16
2.2.2 Projektumfeld . . . . .	17
2.2.3 Risikoanalyse . . . . .	17
2.3 Pflichtenheft . . . . .	17
2.3.1 Zielbestimmung . . . . .	17
2.3.2 Produkteinsatz und Umgebung . . . . .	17
2.3.3 Funktionalitäten . . . . .	18
2.3.4 Testszenarien und Testfälle . . . . .	18
2.3.5 Liefervereinbarung . . . . .	18

2.4	Planung . . . . .	19
2.4.1	Projektstrukturplan . . . . .	19
2.4.2	Meilensteine . . . . .	19
2.4.3	Gant-Chart . . . . .	19
2.4.4	Abnahmekriterien . . . . .	19
2.4.5	Pläne zur Evaluierung . . . . .	19
2.4.6	Ergänzungen und zu klärende Punkte . . . . .	19
<b>3</b>	<b>Vorstellung des Produktes</b>	<b>20</b>
<b>4</b>	<b>Eingesetzte Technologien</b>	<b>21</b>
4.1	Technologie für Webapp . . . . .	22
4.1.1	HTML - Hypertext Markup Language . . . . .	22
4.1.2	Alternative zu Frameworks . . . . .	23
4.1.3	Warum haben wir uns für Yii2 entschieden . . . . .	23
4.1.4	PureMVC . . . . .	24
4.1.5	Laravel . . . . .	24
4.2	Gas-Sensoren . . . . .	25
4.2.1	MQ Gas Sensoren . . . . .	25
4.2.2	Adafruit CCS811 . . . . .	26
<b>5</b>	<b>Problemanalyse</b>	<b>27</b>
5.1	USE-Case-Analyse . . . . .	27
5.2	Domain-Class-Modelling . . . . .	28
5.3	User-Interface-Design . . . . .	28
<b>6</b>	<b>Systementwurf</b>	<b>29</b>
6.1	Architektur . . . . .	29
6.1.1	Design der Komponenten . . . . .	29
6.1.2	MVC - Model, View, Controller . . . . .	29
6.1.3	CRUD - Create, Read, Update, Delete . . . . .	31
6.1.4	Yii2 . . . . .	32
6.1.5	Benutzerschnittstellen . . . . .	33
6.1.6	Datenhaltungskonzept . . . . .	33



6.1.7	Konzept für Ausnahmebehandlung . . . . .	34
6.1.8	Sicherheitskonzept . . . . .	34
6.1.9	Design der Testumgebung . . . . .	34
6.1.10	Desing der Ausführungsumgebung . . . . .	34
6.2	Detailentwurf . . . . .	35
<b>7</b>	<b>Implementierung</b>	<b>37</b>
7.1	Webapp . . . . .	38
7.1.1	Mockups . . . . .	39
7.1.2	Datenbankzugriffe . . . . .	42
<b>8</b>	<b>Deployment</b>	<b>43</b>
<b>9</b>	<b>Tests</b>	<b>44</b>
9.1	Systemtests . . . . .	44
9.2	Akzeptanztests . . . . .	44
<b>10</b>	<b>Projektevaluation</b>	<b>45</b>
<b>11</b>	<b>Benutzerhandbuch</b>	<b>46</b>
<b>12</b>	<b>Betriebswirtschaftlicher Kontext</b>	<b>47</b>
<b>13</b>	<b>Zusammenfassung</b>	<b>48</b>
	<b>Literaturverzeichnis</b>	<b>49</b>

# **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 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)

#### **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 die User aktiv im Web mitzumachen. Diese können eigenen Beiträge erstellen und diese auf der Website anzeigen lassen.(11)

### **4.1.2 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 ganz einfach in Yii2 umsetzen.

Allerdings sind Frameworks nicht Administratoren freundlich da sie sehr viel Vorwissen erfordern um diese richtig zu implementieren. 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 besprochen 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.

### **4.1.3 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 unserem Bedarf zusammenstellen und auch so bearbeiten wie wir es wollen. Es war uns auch wichtig das wir nach modernen Entwurfsmustern arbeiten. Siehe MVC - Model, View, Controller.

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

#### **4.1.4 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 (6)

#### **4.1.5 Laravel**

Laravel



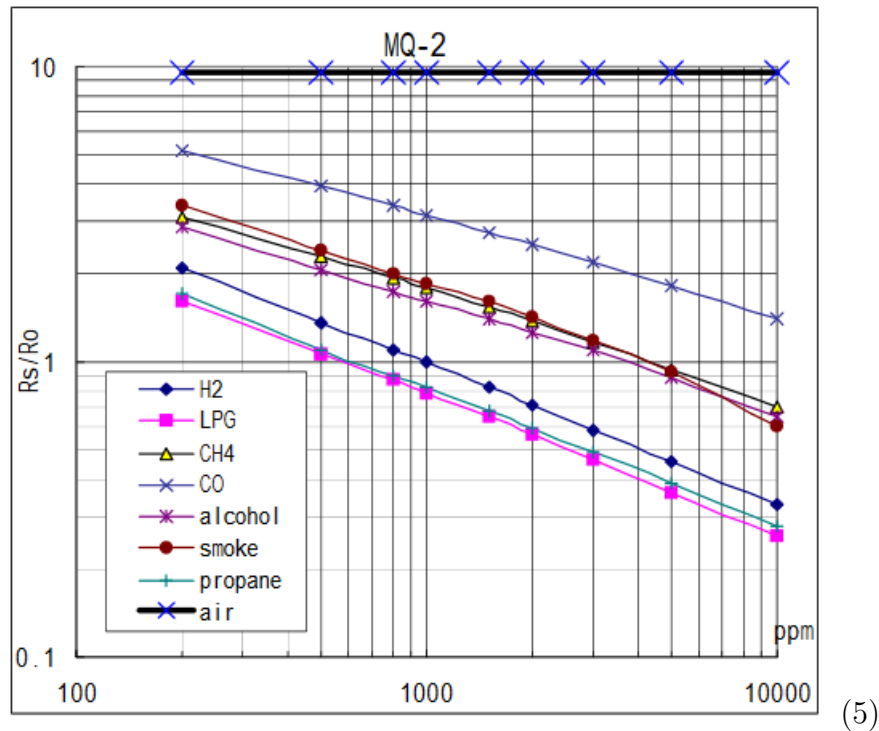
## **4.2 Gas-Sensoren**

### **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 oftmals.



(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. (10)

## **Vor- und Nachteile**

### Vorteile

Gleichzeitiges Programmieren

Hohe Kohäsion (8)

Lose Kopplung (9)

### 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. Bei Webseiten zum Beispiel die Index Seite oder das Login Formular. Außerdem nimmt es Benutzer 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

(7)

## **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 die User aktiv im Web mitzumachen. Diese können eigenen Beiträge erstellen und diese auf der Website anzeigen lassen.(11)

### **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 weiter 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

Darstellung und Beschreibung der Systemarchitektur;



- statische Zerlegung des Systems in seine physischen Bestandteile (Komponenten, Komponentendiagramm)
- (textuelle) Beschreibung des dynamischen Zusammenwirkens aller Komponenten
- (textuelle) Beschreibung der Strategie für die Architektur, d. h. wie die Architektur in Statik und Dynamik funktionieren soll.
- Verwendung von Referenzarchitekturen bzw. Architekturmustern (als Schablonen, z.B. MVC, Plugin, Pipes and Filters)
  - MVC
  - Schichten
  - Pipes
  - Request Broker
  - Service-Oriented

### **6.1.5 Benutzerschnittstellen**

- Design des UIs
- Dialoge, Dialogsteuerung, Ergonomie, Gestaltung, Eingabeüberprüfungen

### **6.1.6 Datenhaltungskonzept**

- Design der Datenbank (ER-Modell)
- Design des Zugriffs auf diese Daten (Datenhaltungskonzept)
- Caching, Transaktionen

### **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

- Design der Security-Elemente
- Design von Safety-Elementen (Fehlertoleranz, Verfügbarkeit etc.)

### **6.1.9 Design der Testumgebung**

- wie wird getestet (Unit-Testing, Integrationstesting, Systemtests, Akzeptanztests)
- Testumgebung, Testprozess, Teststrategie, Testmethoden, Testfälle

### **6.1.10 Desing der Ausführungsumgebung**

- Deployment (DevOps)
- Betrieb (besonders Hoch- und Hertzenerfahren der Anwendung)

## **6.2 Detailentwurf**

Design jedes einzelnen USE-Cases

- Design-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

## *Entomophagie*

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

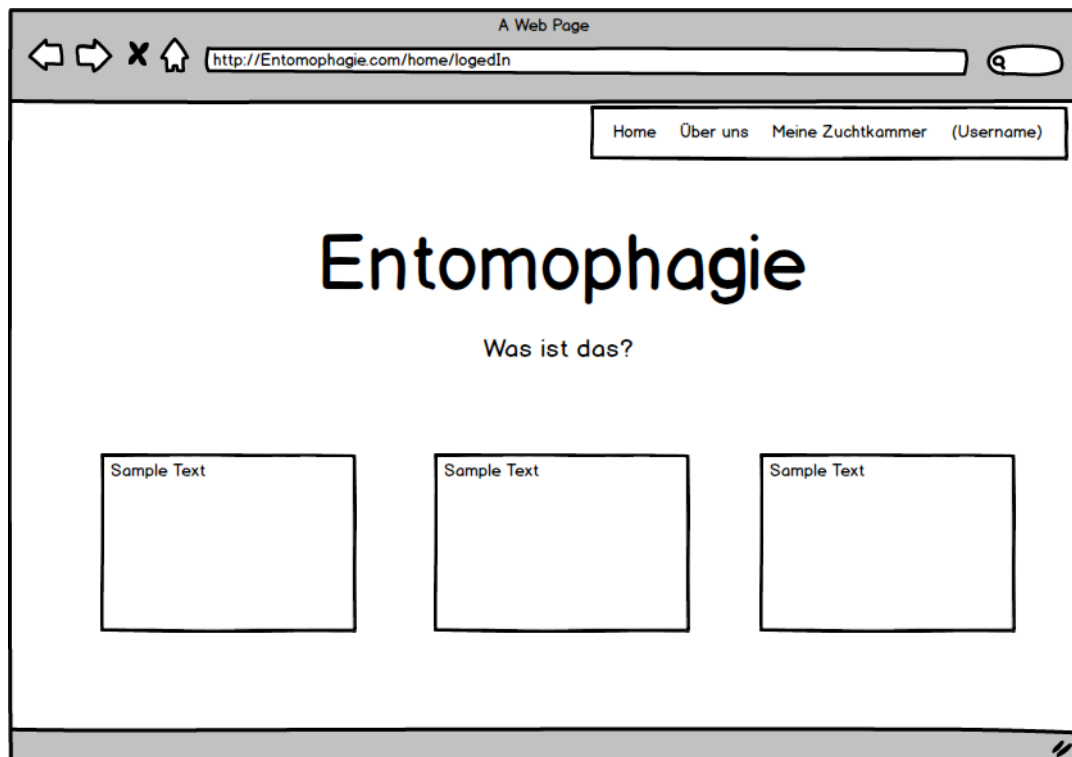
## **7.1 Webapp**

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:

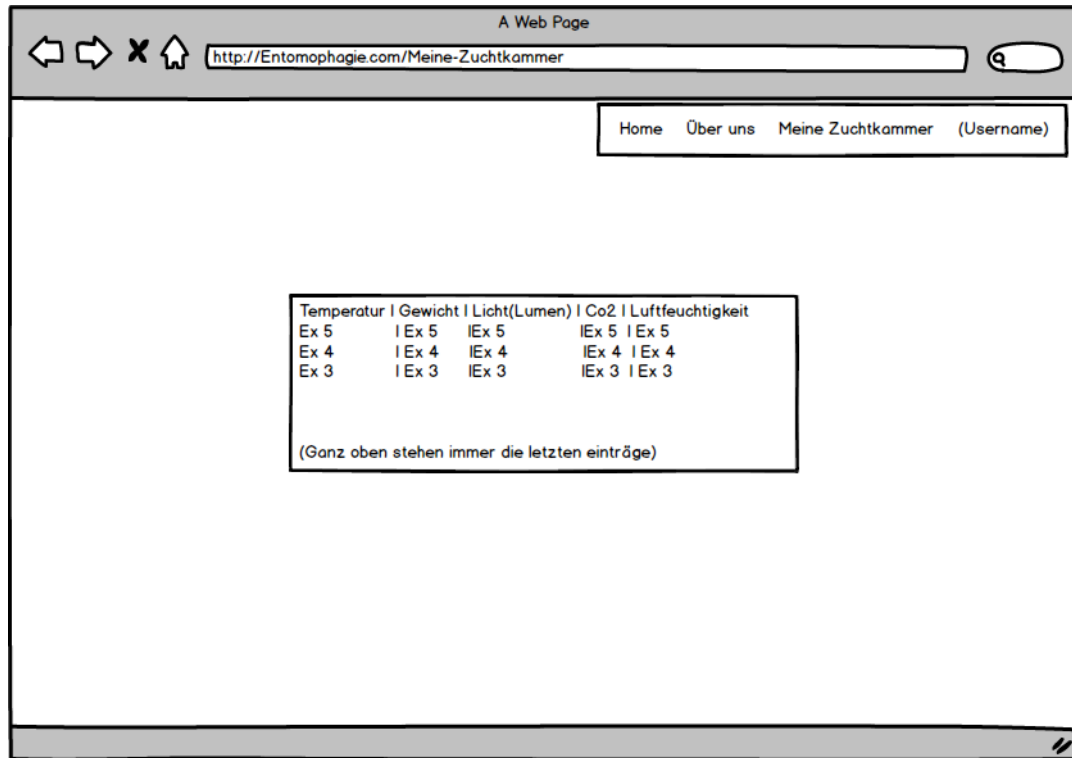
- Sauerstoff
- Luftfeuchtigkeit
- Gewicht
- Temperatur
- Futtermenge
- ungefähre Zeit bis zu Reife

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

### 7.1.1 Mockups

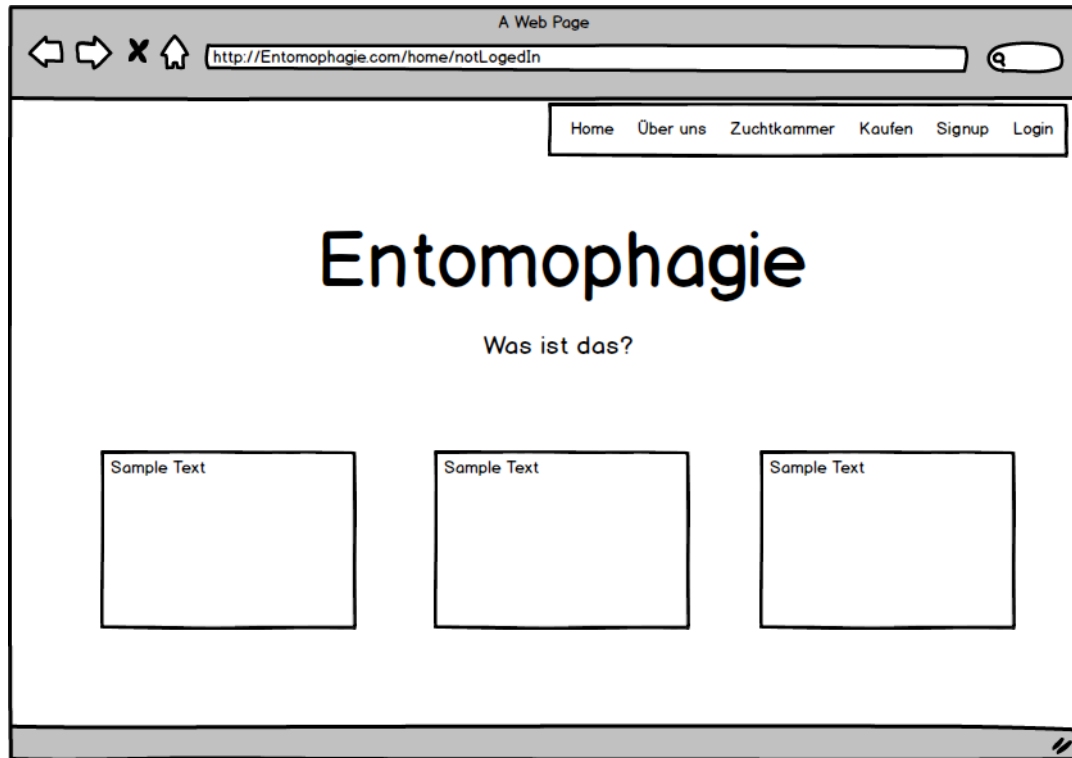


Hier sieht man die Ansicht wenn man auf unserer Website Angemeldet ist. Man kann auf seine eigene Zuchtkammer zugreifen indem man auf den Button "Meine Zuchtkammer" klickt und dort die Daten auslesen.



Auf der Seite "Meine Zuchtkammer" sieht die letzten 10 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 "Zuchtkammer" mehr über unsere Zuchtkammern herausfinden, weiter kann man über den Button "Kaufen" eine 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 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

# 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**

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] PureMVC. Implementation idioms and best practices. URL: <http://puremvc.org/>.
- [7] Wikipedia. Crud. URL: <https://de.wikipedia.org/wiki/CRUD>.
- [8] Wikipedia. Kohäsion. URL: [https://de.wikipedia.org/wiki/Koh%C3%A4sion\\_\(Informatik\)](https://de.wikipedia.org/wiki/Koh%C3%A4sion_(Informatik)).
- [9] Wikipedia. Lose kopplung. URL: [https://de.wikipedia.org/wiki/Lose\\_Kopplung](https://de.wikipedia.org/wiki/Lose_Kopplung).

- [10] Wikipedia. Model view controllers. URL: [https://de.wikipedia.org/wiki/Model\\_View\\_Controller](https://de.wikipedia.org/wiki/Model_View_Controller).
- [11] Wikipedia. Web 2.0. URL: [https://en.wikipedia.org/wiki/Web\\_2.0](https://en.wikipedia.org/wiki/Web_2.0).