

# Projektbericht "T15 - Fernabfrage von Kamerabild und Wetterdaten"

Hannes Fogut, Josefin Hähne, Philipp Barth, Justin Schirdewahn, Alexander  
Schoch, Agustin Calvimontes, Clemens Kujus

14. August 2020

# Inhaltsverzeichnis

Planung	1
1. Aufgabenstellung	2
1.1. Auftraggeber	3
1.2. Ausgangssituation zu Semesterbeginn	3
2. Projektorganisation	5
2.1. Team	5
2.2. Rollen	5
2.3. Kommunikation	5
2.4. Eingesetzte Tools	6
3. Eingesetzte Techniken und Praktiken	7
3.1. Rollenverteilung	7
3.2. Nutzung von GitHub zur Projektverwaltung	7
3.3. Arbeit mit Git-Branches und Commitabstände	8
3.4. Regelmäßige Treffen im Team und mit den Kunden	8
3.5. Agile und iterativ-inkrementelle Vorgehensweise	9
Durchführung	10
4. Projektphasen	11
4.1. Iterationen	11
5. Hauptaktivitäten zusammengefasst	19
5.1. Anforderungserhebung und -analyse	19
5.2. Entwurf	19
5.3. Implementierung	24
5.4. Test	25
5.5. Übergabe und Dokumentation	25
6. Wesentliche Entscheidungen	27
6.4. Django als Webframework für das Back-End	27
7. Aufgetretene Probleme	28
7.1. Aufgabenverteilung	28
7.2. Git-Branches und Commitabstände	28
7.3. Vorstellungen und Ideen der Kunden	29
7.4. Mangelnde Erfahrung in der Entwicklung mit Angular	29
Besprechungsprotokolle	31
8. Protokoll zum Treffen am 09.01.2020	32
8.3. Iterationsplan	32
8.4. Themen	32
8.5. Offenes	33
9. Protokoll zum Treffen am 08.04.2020	34
9.3. Iterationsplan	34

9.4. Themen .....	34
10. Protokoll zum Treffen am 20.05.2020 .....	35
10.3. Iterationsplan .....	35
10.4. Themen .....	35
10.5. Ergebnisse .....	35
11. Protokoll zum Treffen am 27.05.2020 .....	37
11.3. Iterationsplan .....	37
11.4. Themen .....	37
11.5. Ergebnisse .....	37
Ergebnisse .....	39
12. Projektergebnisse .....	40
12.1. Ziel am Projektanfang .....	40
12.2. Zielerreichung .....	40
13. Reflexion Philipp Barth .....	42
14. Reflexion Hannes Fogut .....	43
15. Reflexion Clemens Kujus .....	45
16. Reflexion Justin Schirdewahn .....	47
16.2. Reflexion Agustin Calvimontes .....	48
17. Reflexion Alexander Schoch .....	49
18. Reflexion Josefin Hähne .....	51

# Planung

# 1. Aufgabenstellung

Autoren: Josefin Hähne

Der Modellflugclub Rossendorf e.V. hat den Wunsch nach einer vereinseigenen Wetterstation und Webcam, wobei die entsprechenden Daten auf der Website <http://mfc-rossendorf.de/wichtige-infos-bitte-lesen/wetter.html> direkt dargestellt werden sollen. Da der Modellflugplatz keine Infrastruktur wie Strom, Telefon oder Internet hat, bleibt nur eine solarbetriebene Anlage. Ein Computermodule sammelt dabei die Daten angeschlossener Sensoren wie Bild, Temperatur, Wind (Stärke + Richtung)), speichert sie lokal und sendet sie zyklisch an einen Webserver.

Es kommt folgende Hardware zum Einsatz:

- Raspberry Pi 4
- diverse Sensoren
- Webcam (via Raspi on-board Camera Connector)
- UMTS-Modul (via USB)
- Akku (LiPo)
- Solarzelle
- Lade-Management
- Gehäuse

Auf dem Raspi läuft eine Standard-Linux-Distribution. Die auszuführenden Aktionen werden in definierten Zeitabständen per CronJob angestoßen. Das System sowie der Raspi laufen ununterbrochen.

Die auszuführenden Funktionen sollen in einer Skriptsprache umgesetzt werden. Die Sensordaten werden über die entsprechenden IO-Pins gesammelt. Die Webcam ist per on-board Connector angeschlossen, die Bildakquise erfolgt über die entsprechenden Kommandozeilen-Funktionen. Als Funkmodul kommt ein UMTS-Stick zum Einsatz. Systemseitig stellt sich dies als normales Netzwerk Device dar.

Die Visualisierung der Werte soll auf der Vereinswebseite erfolgen. Die Daten sollen direkt an den entsprechenden Webserver gesendet und dort in einer Datenbank gespeichert und zur Anzeige gebracht werden. Über einen separaten „Admin-Bereich“ soll es möglich sein, weitere Informationen zum Systemzustand abzufragen (Datenverbrauch, Energieverbrauch).

Es wird eine MySQL Datenbank verwendet.

Alle Messwerte sollen dynamisch sortiert, gefiltert und in grafischer Form (als Diagramm) zur Anzeige gebracht werden können. Dabei sollen alle Aktionen des Nutzers, die die Auswahl der anzuzeigenden Daten betreffen (z.B. Zeitraum) „dynamisch“ erfolgen – ein „Neu laden“ der Seite soll also nicht erforderlich sein. Die Abfrage der Datenbank und Aktualisierung der Darstellung soll via JavaScript (AJAX) im Browser erfolgen. Die Werte sollen durch ein „passendes“ Verfahren interpoliert werden, so dass eine sinnvolle Darstellung auch durch die Messungen nicht erfasster Werte möglich ist (Darstellung als „Kurvenzug“).

Die von der Kamera aufgenommenen Bilder sollen übersichtlich als Galerie entsprechend der Auswahl des Nutzers (Zeitraum, Intervall, Start- und Enddatum) dargestellt werden. Bei der Auswahl eines Bildes soll dies in voller Auflösung zur Anzeige gebracht werden. Die Interaktion des Nutzers mit dem Browser soll ebenfalls „dynamisch“, wie im vorigen Punkt beschrieben, erfolgen.

Ein Nutzer-/Rechtemanagement ist nicht erforderlich, lediglich der oben genannte „Admin-Bereich“ soll nur nach einem Login erreichbar sein.

Die webserverseitige funktionelle Umsetzung ist noch nicht spezifiziert. Letztlich ist eine Vielzahl von Sprachen geeignet, vorzugsweise sollte die Umsetzung jedoch in einer Skriptsprache erfolgen, um die spätere Wartbarkeit und Erweiterbarkeit zu vereinfachen.

Browser/Frontendseitig kommen die „Standardtechniken“ HTML, CSS und JavaScript zum Einsatz. Hinsichtlich der Verwendung spezieller JavaScript-Frameworks gibt es keine Festlegungen.

## **1.1. Auftraggeber**

Der Auftraggeber des Projekts ist der Modellflugclub Rossendorf e.V. Die Betreuung übernahmen Prof. Dr. rer. nat. habil. Heino Iwe und der Vorsitzende des Vereins Thomas Brenner, die uns auch unterstützt und Vorschläge gemacht haben.

## **1.2. Ausgangssituation zu Semesterbeginn**

Zu Semesterbeginn standen wir vor einer Herausforderung, die wir noch nicht bedacht haben, dass diese eintreten könnte. Durch die Pandemie konnten wir uns nicht mehr persönlich treffen und es fiel der tägliche Kontakt in der Hochschule weg. Somit fielen auch die Treffen an der HTW mit den Auftraggebern aller zwei Wochen weg und wir mussten uns einen neuen Kommunikationsweg suchen. Dies galt nicht nur für die internen Teammeetings, sondern auch für die Treffen mit den Auftraggebern sowie dem Coach. Dazu wählten wir anfangs Google Hangouts, welches wir auch schon von unseren Vorlesungen kannten.

Wir haben uns als Team weiterentwickelt, besser kennengelernt und in Erfahrung gebracht, welches Teammitglied bestimmte Interessen und Stärken besitzt, die wir in unserem Projekt einsetzen können. Ebenfalls bekamen wir einen neuen Tester in unser Team, Hannes Fogut, da Martin Großmann unser Team verlassen hat. Wir haben eine geregelte Kommunikation zur Aufgabenverteilung, aber auch für weitere Absprachen gefunden.

Wir hatten die Inception- und Elaborationsphasen abgeschlossen, sowie die Anforderungserhebung und -analyse bis zu dem Zeitpunkt. Auch in der Architektur hatten wir Fortschritte gemacht. So haben wir erste Wireframes für die Benutzeroberfläche erstellt, die auch von den Auftraggebern abgesegnet wurden. Es gab einen ersten Entwurf, wie das System aufgebaut ist und eine Architekturbeschreibung. Des Weiteren waren der Projektplan, Use Cases mit Beschreibungen, ein Glossar, das ERM, Aktivitätsdiagramme und die Vision bis zu der Zeit ausgearbeitet.

# Alphas the things to work with

Software System	ARCHITECTURE SELECTED (1/6)
Requirements	COHERENT (3/6)
Team	PERFORMING (4/5)
Stakeholders	IN AGREEMENT (4/6)
Opportunity	VIABLE (4/6)
Way of Working	FOUNDATION ESTABLISHED (2/6)
Work	STARTED (3/6)



Alphas Overview

Abbildung 1. Projektstatus zu Semesterbeginn

## 2. Projektorganisation

### 2.1. Team

Unser Team besteht aus 7 Mitgliedern: Clemens Kujus, Alexander Schoch, Philipp Barth, Justin Schirdewahn, Agustin Calvimontes, Josefin Hähne und Hannes Fogut, welcher erst im vierten Fachsemester in unser Team kam, da Martin Großmann unser Team nach SE1 verlassen hat.

### 2.2. Rollen

Jedem von uns sind 2 Rollen zugeteilt, einmal die Hauptrolle und meist noch ein Backup. Clemens Kujus übernahm nach einem Tausch mit Martin Großmann die Rolle als Projektmanager und Analyst, Alexander Schoch ist Entwickler und Projektmanager, Philipp Barth Architekt und Entwickler, Justin Schirdewahn Entwickler und Tester, Agustin Calvimontes Deployment Engineer und Technical Writer, Josefin Hähne Analyst und Architekt und Martin Großmann beziehungsweise in SE2 Hannes Fogut Tester und Entwickler, da Martin Großmann unser Team verlassen hat.

### 2.3. Kommunikation

#### 2.3.1. Im Team

Im Team findet die Kommunikation über WhatsApp für kleine und schnelle Absprachen statt. Wir haben regelmäßige Teamtreffen, sowohl an der HTW als auch online, in denen wir größere Probleme diskutieren, Absprachen halten, Aufgaben der aktuellen Iteration bewerten, Iterationen auswerten sowie neue Aufgaben für die nächste Iteration verteilen. Es wird sich auch in kleineren Gruppen getroffen um beispielsweise nur ein Thema oder eine Aufgabe zu bearbeiten oder besprechen. Des Weiteren gab es alltäglichen Kontakt in Vorlesungen und Übungen/Praktika an der HTW, bei dem z.B. an den nächsten Termin für ein Treffen erinnert wurde oder man den aktuellen Stand austauschte (entfällt in SE2 aufgrund einer Pandemie). Für die Themenbearbeitung und Aufgabenverteilung nutzen wir die in GitHub vorgesehenen Issues.

Seit diesem Semester findet die Kommunikation nur noch online statt. Dazu wählten wir zusätzlich für Teamtreffen Google Hangouts und Google Meet.

#### 2.3.2. Zum Auftraggeber

Für die Kommunikation mit den Auftraggebern nutzen wir hauptsächlich die regelmäßigen Treffen, die anfangs aller zwei Wochen in der Bibliothek der HTW stattfanden. Seit diesem Semester wurden sie jedoch mit Videochats abgehalten und ab Mitte April auch wöchentlich. Dabei erklärten sie uns immer mehr ihre Vorstellungen, wie das System aufgebaut sein soll, was abgebildet werden soll und wie sowie was für Anforderungen sie an das System haben. Wir klären auch welche Ressourcen sie uns zur Verfügung stellen und welche Aufgaben von ihnen übernommen werden. Es werden auch unsere neuen Ergebnisse vorgeführt und geklärt, ob die Kunden damit einverstanden sind oder es noch verbessert werden soll. Des Weiteren nutzen wir Emails für die Terminvereinbarung, neue Anforderungen oder auch für mehr Informationen z.B.



über die gesendeten Daten (Datenformat, Zeitstempel, ...).

## 2.4. Eingesetzte Tools

Hier folgt eine Auflistung der im Projektverlauf genutzten Tools nach Zweck:

**Dokumentation:** Visual Paradigm, Visual Studio Code, Pydoc, Compodoc

**Programmierung/Tests:** PyCharm, Visual Studio Code, Postman, MySQL

**Browser:** Google Chrome, Microsoft Edge, Mozilla Firefox

**Kommunikation:** Google Hangout, Google Meet, Discord, E-Mail, WhatsApp

**Versionsverwaltung:** Git, GitHub

Alle Tools wurden in aktuellen Versionen benutzt.

# 3. Eingesetzte Techniken und Praktiken

Autoren: Clemens Kujus

## 3.1. Rollenverteilung

Die Rollenverteilung hat sich zwei mal grob verändert. Zum einen am Anfang des Projekts, als unser ehemaliger Projektmanager Martin Großmann seine Rolle abgegeben hat. Er hat sich in seiner Rolle nicht wohlfühlt und ist seinen Aufgaben in den ersten beiden Projektwochen auch nicht nachgekommen. Auf anraten des Teams und schließlich eigener Entscheidung trat er seine Position ab. Getauscht wurde mit Clemens Kujus, dem damaligen Tester. Somit wurde Martin primär Tester und Backup-Projektmanager. Diese Konstellation hat sich bis durch das WiSe 20 gehalten, für das SoSe 20 entschied sich Martin allerdings das Projekt zu verlassen. Neu hinzugekommen ist Hannes Fogut, der die Rolle des Testers von Martin übernommen hat, und sich aufgrund der großen Implementierungsaufgaben auch als Entwickler gemeldet hat. Damit wurde Alexander Schoch Backup-Projektmanager. Durch den Wegfall von Martin Großmann und das Hinzukommen von Hannes Fogut hat das Projekt an Qualität und Produktivität gewonnen.

Neben der Rollenverteilung nach dem OpenUP haben wir uns vor allem mit der Phase der Implementierung dafür entschieden, dass mehrere Teammitglieder auch abseits ihrer UP-Rolle teaminterne Rollen haben, die vor allem der Unterstützung der Entwickler dienen. So wurde z.B. der Projektmanager nebenläufig zum Front-End Entwickler. Aufgrund der Komplexität der Software war dies auch richtig und nötig.

## 3.2. Nutzung von GitHub zur Projektverwaltung

In unserem Projekt dient Git nicht nur der Versions-, sondern auch der Projektverwaltung. Das bedeutet, dass wir GitHub-Issues und im Verlauf des Projekts auch GitHub-Projekte eingeführt haben. Die Entscheidung für GitHub-Issues liegt darin begründet, dass diese eine übersichtliche und intuitive Möglichkeit bilden, Aufgaben bzw. Work Items anzubieten und die Bearbeitung kenntlich zu machen, damit zum einen Aufgaben klar definiert sind und zum anderen keine doppelte Aufgabenlösung entsteht.

<input type="checkbox"/>	Treff mit Coach am 08.07.2020 17 Uhr <span>coach</span>		2
	<small>#85 by ClemensKHTW was closed 27 days ago</small>		
<input type="checkbox"/>	Adminpanel Diagramm für Leistungsaufnahme <span>development</span>		
	<small>#82 by ClemensKHTW was closed on 30 Jun</small>		
<input type="checkbox"/>	Adminpanel graphisch anpassen <span>development</span> <span>enhancement</span>		
	<small>#74 by ClemensKHTW was closed on 13 Jun</small>		
<input type="checkbox"/>	Adminpanel mit Back-End zusammenführen		
	<small>#71 by ClemensKHTW was closed on 29 Jun</small>		
<input type="checkbox"/>	Einsatz von Mat-Cards im Adminpanel prüfen		1
	<small>#70 by ClemensKHTW was closed on 1 Jul</small>		
<input type="checkbox"/>	Treff am 20.05.2020		
	<small>#59 by ClemensKHTW was closed on 20 May</small>		
<input type="checkbox"/>	Skript erstellen was Daten auf dem Raspi einsammelt und postet <span>development</span>		
	<small>#58 by philippBa13 was closed on 2 Jul</small>		

Abbildung 2. GitHub-Issues

Für die Projekte in GitHub haben wir uns entschieden, damit wir die Iterationen besser auswerten können. Eingeführt wurden sie Ende Januar 2020 auf anraten des Coaches, da die Auswertung der Iterationen bis dahin in Textform der thematisierten Iteration angeheftet wurde. Dadurch wurde zwar auch ersichtlich, welche Aufgaben wie gelöst wurden, allerdings ist ein Blocktext nicht schnell zu erfassen und Issues und Auswertung liefen parallel. In den Projekten haben wir uns aufgrund der Intuition für das "Basic kanban"-Template entschieden. Die Auswertung in Textform lief auf Entscheidung des Projektmanagers (besserer Abschluss der Iterationspläne) bis zur 13. Iteration parallel zu den Projekten. Ab der 14. Iteration haben wir uns dann allerdings dafür entschieden, wie vom Coach angeraten nur noch die Projekte zur Auswertung zu nutzen.

### Bewertung

Das Front-End wurde mit Grundfunktionalitäten ausgestattet, lediglich die Bildergalerie fehlt. Das Back-End und das Front-End sind soweit zusammengeführt, dass eine Kommunikation dazwischen möglich ist. Das Datenformat der Wetterdaten wurde uns vom Kunden noch nicht vorgelegt. Die Rest-API wurde ausgebaut, allerdings ist die Implementierung einer Authentifizierung in der Priorität nach hinten gerückt. Die Überarbeitung der Dokumentation wurde angefangen. Ein Testdokument wurde erstellt, bei welchem die Grundsätze des Testens für die Wetterstation aufgeschrieben wurden. Die Projektabgabe wurde nicht weiter spezifiziert.

Abbildung 3. Alte Auswertung in Textform

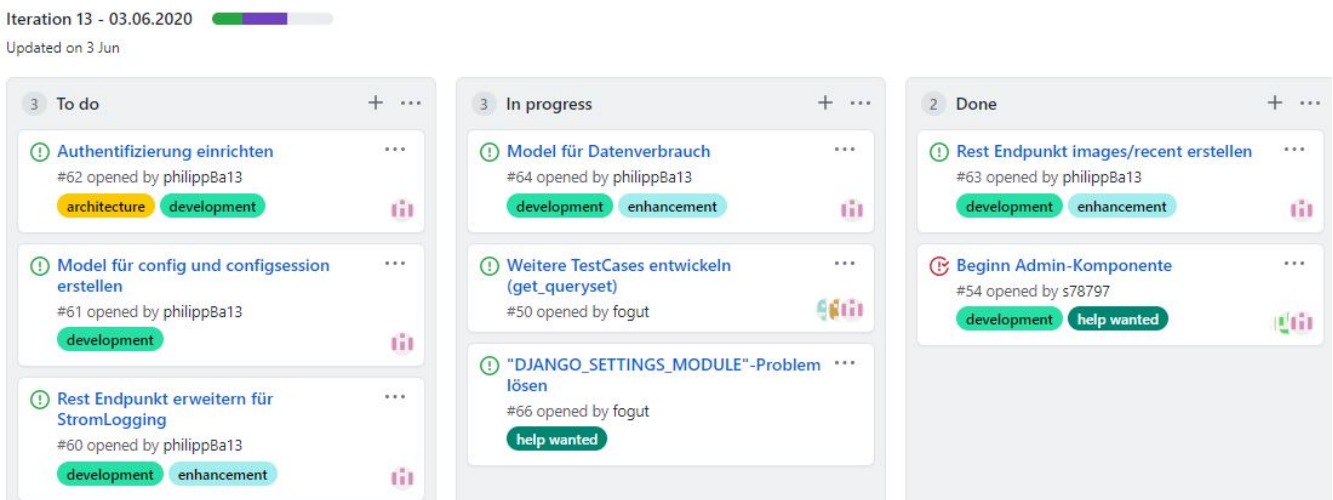


Abbildung 4. Neue Auswertung in GitHub-Projekten

Wir haben uns nicht aktiv gegen die Nutzung anderer Tools entschieden, sondern haben uns auf die Notwendigkeit dieser konzentriert. Für unseren Arbeitsablauf waren die Möglichkeiten in Git völlig ausreichend, weshalb wir ausschließlich die GitHub Standardtools gewählt haben.

## 3.3. Arbeit mit Git-Banches und Commitabstände

Für Git-Banches haben wir uns entschieden, da in der Implementierung Versionskonflikte abzusehen waren. Aufgrund des unten beschriebenen Problems der Git-Banches und Commitabstände siehe Abschnitt "Aufgetretene Probleme" setzten wir detaillierte Banches mit möglichst kleinen Commits ein.

## 3.4. Regelmäßige Treffen im Team und mit den Kunden

Im WiSe 19/20 haben wir uns jeweils zum Iterationsabschluss getroffen, oft auch mit den Kunden. Mit Anfang der Implementierung haben wir dann festgestellt, dass diese Vorgehensweise zum

einen zur teaminternen Kommunikation einfach nicht mehr ausreicht, zum anderen aber auch durch die oft geänderten Anforderungen der Kunden nicht mehr reicht. Demzufolge haben wir uns ab Mitte April wöchentlich online getroffen, später teilweise auch unter den Meetings persönlich, und dabei immer die Kunden eingeladen. Letztlich war diese Entscheidung maßgeblich für das Projekt und aus unserer Sicht und der der Kunden absolut richtig.

### **3.5. Agile und iterativ-inkrementelle Vorgehensweise**

Wir haben uns für den Ansatz des selbstorganisierenden Teams entschieden, da wir zum einen die Interessen jedes Mitglieds beachten wollten und zum anderen eine Schätzung von Aufwand und Zeit zum gegebenen Umfang schwieriger erschien als den Rahmen (Iterationen) festzulegen und die Aufgaben nach Priorität abzuarbeiten.

Für die iterativ-inkrementelle Vorgehensweise nach dem Vorbild des OpenUP haben wir uns aufgrund des Fokus im Modul Software Engineerig 1 und des Status als state of the art entschieden. Anfangs haben wir sehr strikt darauf geachtet, dass die Rollen auch möglichst nur ihre zugehörigen Aufgaben wahrnehmen. Da uns allerdings auffiel, dass dadurch ein starkes Ungleichgewicht vor allem bei der Implementierung entstehen würde, haben wir uns letztendlich für eine teainterne Rollenverteilung entschieden, bei der jeder nach seinen Stärken und Interessen auch andere unterstützende Aufgaben wahrgenommen hat.

# Durchführung

# 4. Projektphasen

Autoren: Agustin Calvimontes, Clemens Kujus

Das Projekt wurde in Iterationen mit einer Iterationslänge von 2 Wochen durchgeführt. Dabei kann das Projekt grob in 2 Phasen untergliedert werden. Die erste Phase ist die der Analyse und Projektplanung im WiSe 19/20, die zweite Phase ist die der Implementierung im SoSe 20, wobei im Sinne der Iterativen Entwicklung keine strikte Trennung erfolgte. Die Iterationen wurden mit Ablauf der vorhergehenden angefangen, 2 Wochen lang bearbeitet und mit einem Meeting abgeschlossen, bei dem möglichst das gesamte Team anwesend sein sollte. Während des Meetings wurde dann die neue Iteration geplant. Im folgenden sind ausgewählte Ziele, Aktivitäten, Ergebnisse und Probleme der Iterationen aufgelistet und verantwortliche Rollen spezifiziert. Dabei sind nicht alle Aktivitäten und Ergebnisse dargestellt, sondern nur die wichtigsten und/oder Beispiele (d.h. es werden hier nicht alle ausgearbeiteten Wireframes, Use Cases, ERMs, ... dargestellt).

## 4.1. Iterationen

### 4.1.1. Iteration 1 - 28.11.2019-11.12.2019

#### Ziele / Aktivitäten

- Team zusammensetzen → Alle
- Arbeitsweise und Tools zur Teamarbeit herausuchen → Alle

#### Ergebnisse

- Team zusammengesetzt, Kommunikation per WhatsApp etabliert
- Git für Versionsverwaltung festgelegt, inklusive GitHub-Issues

### 4.1.2. Iteration 2 - 12.12.2019-25.12.2019

#### Ziele / Aktivitäten

- Anforderungen im Kundengespräch aufnehmen → Analyst, Entwickler, Projektmanager
- Stakeholder finden → Alle

#### Ergebnisse

- Anforderungen aufgenommen und einige Stakeholder gefunden

### 4.1.3. Iteration 3 - 26.12.2019-09.01.2020

#### Ziele / Aktivitäten

- Use-Cases ableiten → Analyst, Architekt, Tester
- Vision bearbeiten → Architekt, Projektmanager

- Glossar anlegen → Alle
- Basisarchitektur skizzieren → Architekt

## Ergebnisse

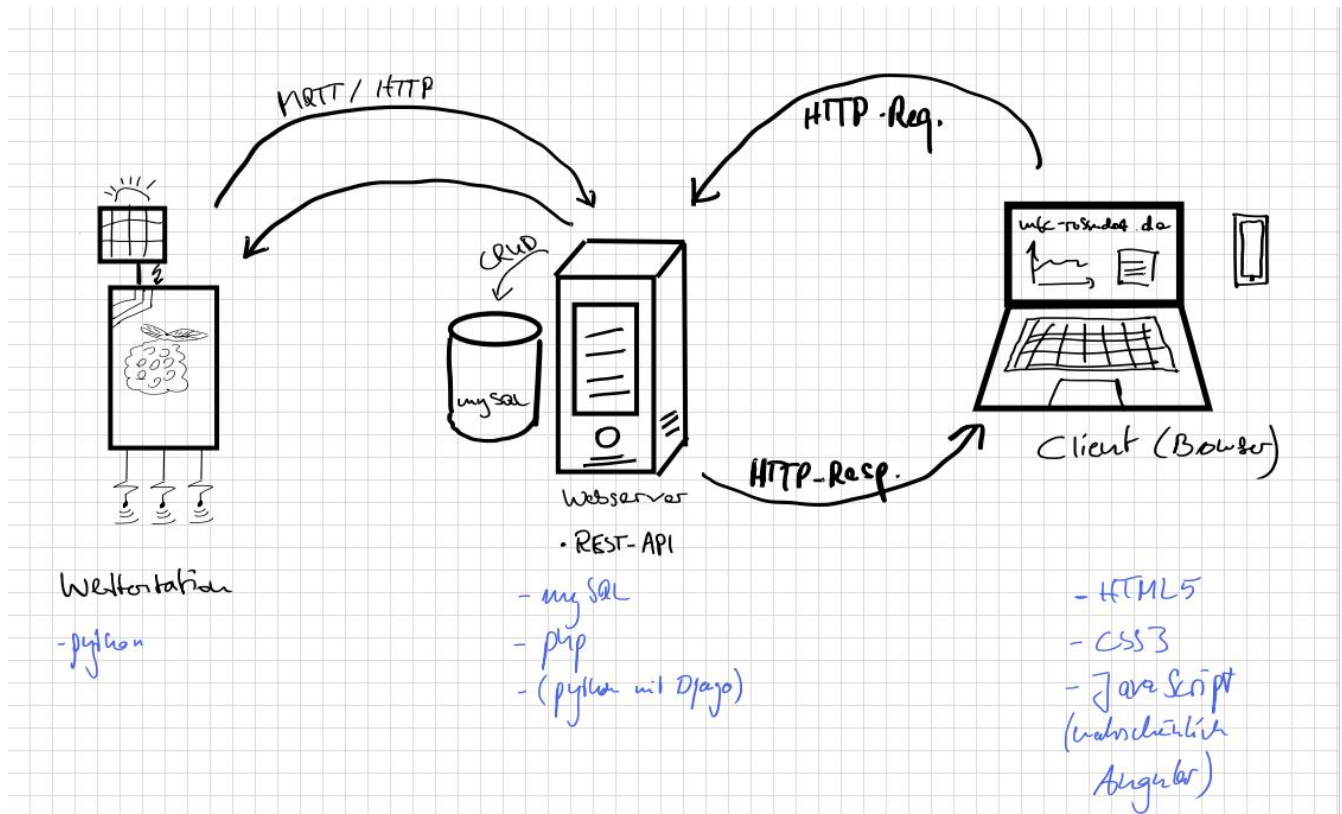


Abbildung 5. Skizze der Architektur

## Probleme

- Teaminterne Kommunikation mangelhaft, d.h. es wurde sich nicht über offene Aufgaben informiert und die Erledigung von Aufgaben wurde nicht kenntlich gemacht (wer macht gerade was, Issues wurden nicht genutzt)

### 4.1.4. Iteration 4 - 10.01.2020-22.01.2020

#### Ziele / Aktivitäten

- Regelmäßige Kommunikation im Team durchsetzen → Projektmanager
- Use Cases ableiten und teils detailliert ausarbeiten → Analyst, Architekt, Entwickler, Tester
- ERM erarbeiten → Architekt, Entwickler
- Aktivitätsdiagramme erarbeiten → Architekt, Entwickler

## Ergebnisse

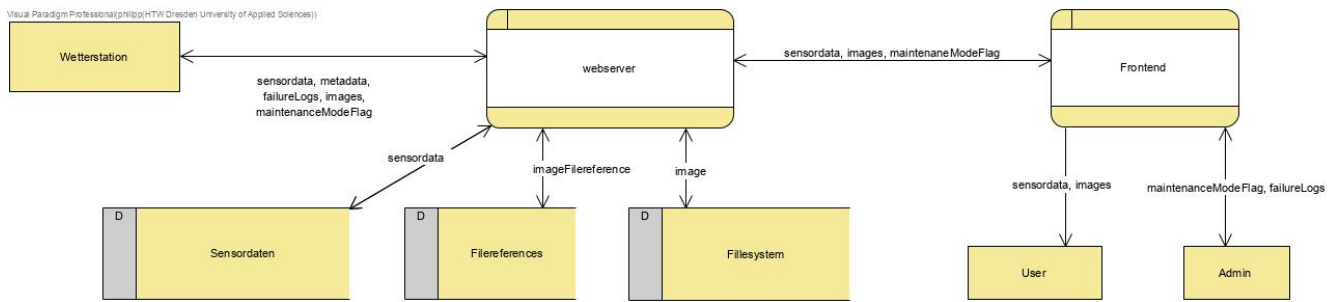


Abbildung 6. Kommunikation der Komponenten

## Probleme

- Work Items List wurde von den Teammitgliedern nicht angenommen

## 4.1.5. Iteration 5 - 23.01.2020-05.02.2020

### Ziele / Aktivitäten

- Use Cases überarbeiten → Analyst, Architekt, Entwickler, Tester
- Wireframes erstellen und Kunden vorzeigen
- Belegabgabe vorbereiten/durchführen → Projektmanager

## Ergebnisse

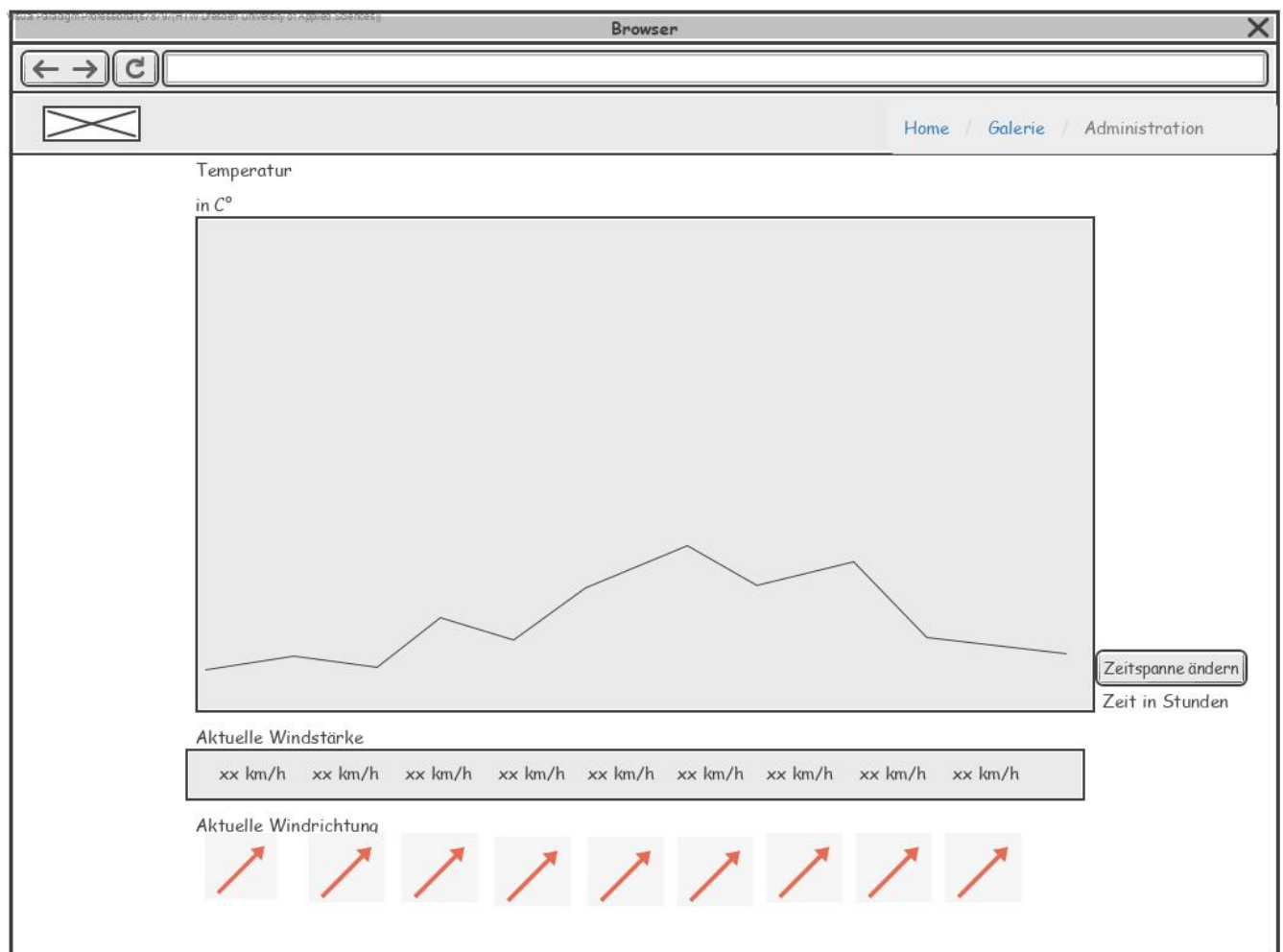


Abbildung 7. Wireframe für die Graphenansicht



## **Probleme**

- Work Items List wurde von den Teammitgliedern nicht angenommen

### **4.1.6. Iteration 6 - 06.02.2020-19.02.2020**

- Pause aufgrund der Prüfungszeit

### **4.1.7. Iteration 7 - 20.02.2020-04.03.2020**

- Pause aufgrund der Prüfungszeit und Winterferien

### **4.1.8. Iteration 8 - 05.03.2020-18.03.2020**

- Pause aufgrund von Winterferien

### **4.1.9. Iteration 9 - 19.03.2020-01.04.2020**

## **Probleme**

- Pandemie

### **4.1.10. Iteration 10 - 02.04.2020-21.04.2020**

## **Ziele / Aktivitäten**

- Rollenverteilung anpassen → Projektmanager
- Erste Programmierungen vornehmen

## **Ergebnisse**

- Rollenverteilung im Team geregelt
- Werte können in eine Datenbank gespeichert und über einen Browser ausgelesen werden mittels Rest-API

## **Probleme**

- Mangelnde Kommunikation mit GitHub-Issues

### **4.1.11. Iteration 11 - 22.04.2020-06.05.2020**

## **Ziele / Aktivitäten**

- Front-End und Back-End entwickeln → Architekt, Entwickler
- Test auf Grundlage von Use Cases beschreiben und entwickeln → Tester

## **Ergebnisse**

## **Bewertung**

Das Front-End wurde mit Grundfunktionalitäten ausgestattet, lediglich die Bildergalerie fehlt. Das Back-End und das Front-End sind soweit zusammengeführt, dass eine Kommunikation dazwischen möglich ist. Das Datenformat der Wetterdaten wurde uns vom Kunden noch nicht vorgelegt. Die Rest-API wurde ausgebaut, allerdings ist die Implementierung einer Authentifizierung in der Priorität nach hinten gerückt. Die Überarbeitung der Dokumentation wurde angefangen. Ein Testdokument wurde erstellt, bei welchem die Grundsätze des Testens für die Wetterstation aufgeschrieben wurden. Die Projektabgabe wurde nicht weiter spezifiziert.

*Abbildung 8. Bewertung Iteration 11*

## **4.1.12. Iteration 12 - 07.05.2020-20.05.2020**

### **Ziele / Aktivitäten**

- Front-End, Back-End und Bildergalerie weiterentwickeln → Entwickler, Architekt -Testsuite aufbauen und damit erste Tests implementieren → Tester

### **Ergebnisse**

- Bilder in der Bildergalerie können geladen und dargestellt werden
- Front-End und Back-End erweitert
- Der momentane Projektstand/-verlauf wird vom Kunden für zufriedenstellend befunden

## **4.1.13. Iteration 13 - 21.05.2020-03.06.2020**

### **Ziele / Aktivitäten**

- Rest Endpunkt images/recent erstellen → Architekt
- Rest Endpunkt erweitern für StromLogging → Architekt
- Authentifizierung einrichten → Architekt
- Model für config und configsession erstellen → Architekt

### **Ergebnisse**

- das Front-End wurde gut weiterentwickelt, neu dazu gekommen ist ein Mockup des Adminpanel
- Testfälle stehen zwar, sind aber aufgrund von Problemen mit der Rest-API noch nicht ausführbar (siehe Issue#66)
- die Authentifizierung wurde noch nicht angegangen

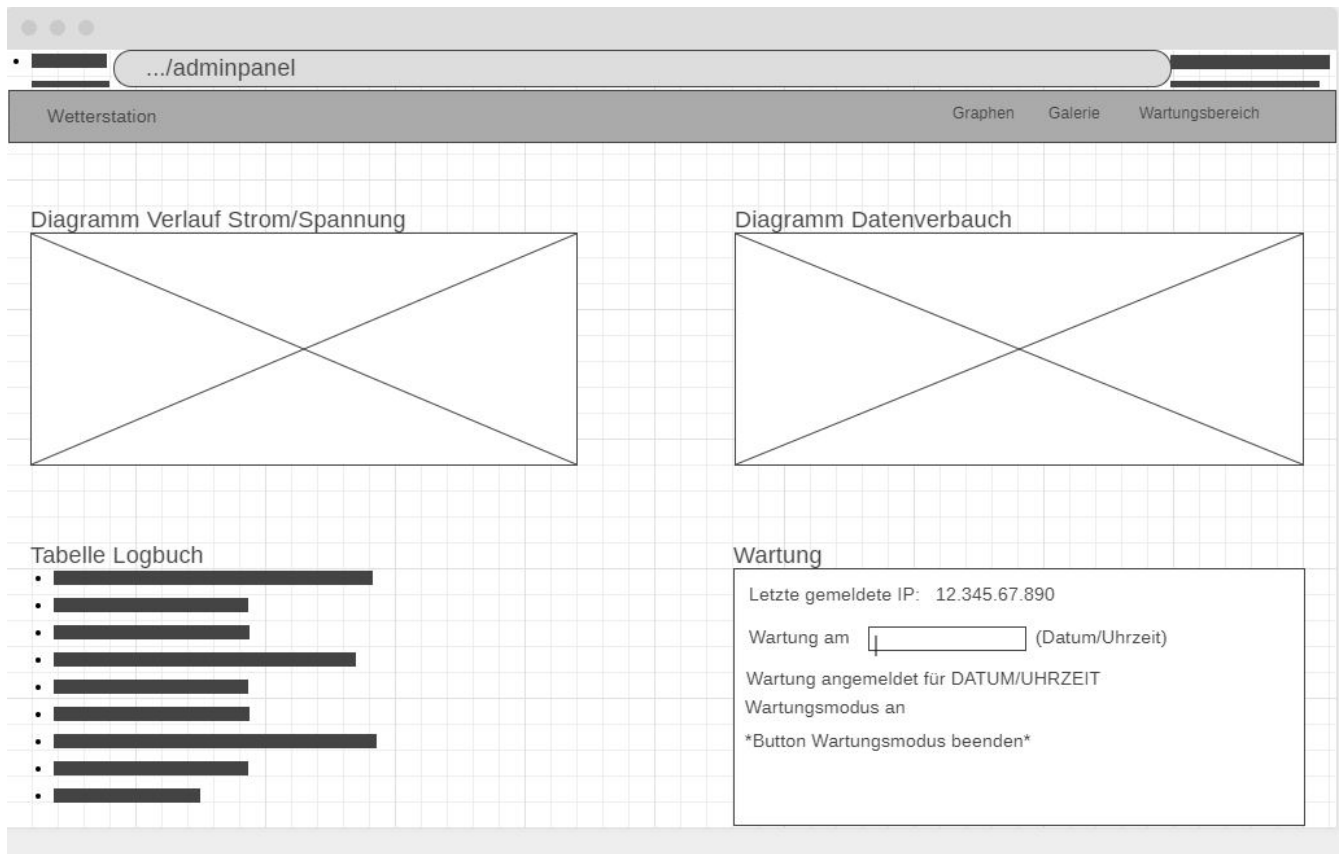


Abbildung 9. Mockup des Adminpanels

#### 4.1.14. Iteration 14 - 04.06.2020-17.06.2020

##### Ziele / Aktivitäten

- Rest Endpunkt erweitern für StromLogging → Architekt
- Model für Datenverbrauch → Architekt
- Model für config und configsession erstellen → Architekt
- Authentifizierung einrichten → Architekt

##### Ergebnisse

- Model für config und configsession erstellt

#### 4.1.15. Iteration 15 - 18.06.2020-01.07.2020

##### Ziele / Aktivitäten

- Model für config und configsession verfeinern (WIP) → Architekt
- Adminpanel bearbeiten und mit Back-End zusammenführen → Projektmanager
- Authentifizierung einrichten → Architekt

##### Ergebnisse

- Adminpanel ist mit Back-End zusammengeführt

## Probleme

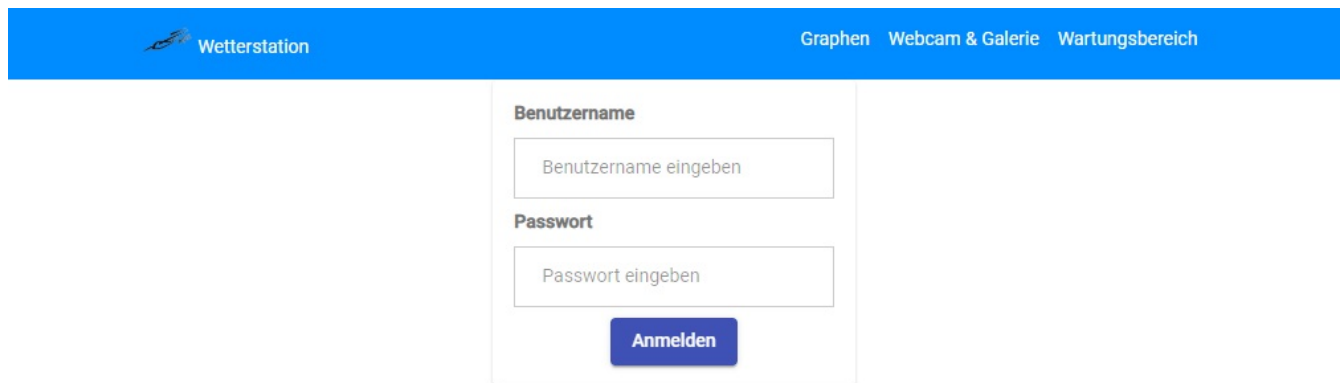
- Projektbericht und Dokumentation wurden nicht überarbeitet
- Modell für Config und Configsession wurde nicht verfeinert

### 4.1.16. Iteration 16 - 02.07.2020-15.07.2020

#### Ziele / Aktivitäten

- Tests erstellen, überarbeiten und durchführen → Tester, Architekt
- Model für config und configsession verfeinern → Architekt
- Galerie bearbeiten → Entwickler
- Adminpanel Diagramm Daten mit entsprechenden Datumsangaben zusammenführen → Projektmanager
- Authentifizierung einrichten → Architekt

#### Ergebnisse



The screenshot shows a web application interface with a blue header bar. On the left of the header is a logo and the text 'Wetterstation'. On the right are links: 'Graphen', 'Webcam & Galerie', and 'Wartungsbereich'. Below the header is a white login form. The form has two input fields: 'Benutzername' with placeholder text 'Benutzername eingeben' and 'Passwort' with placeholder text 'Passwort eingeben'. Below these fields is a blue button labeled 'Anmelden'.

Abbildung 10. Loginfenster

#### **4.1.17. Iteration 17 - 16.07.2020-31.07.2020**

##### **Ziele / Aktivitäten**

- Adminpanel verfeinern und an neue Anforderungen anpassen
- Deployment vorbereiten
- Übergabe der aktuellsten Version am 31.07.2020

##### **Ergebnisse**

- Software ist als as-is den Kunden übergeben und wurde von diesen dankend angenommen
- weitere Arbeit an der Software im Team über das Modul hinaus erfolgt freiwillig

# 5. Hauptaktivitäten zusammengefasst

## 5.1. Anforderungserhebung und -analyse

Autoren: Clemens Kujus

Die Anforderungserhebung und -analyse wurde während des Projektverlaufs durchgeführt. Die Anforderungen wurden am Projektanfang vor allem in Use-Cases festgehalten. In der Vision wurden im Gegensatz zu den Use-Cases keine Abläufe beschrieben, sondern die grundsätzlichen Anforderungen und Rahmenbedingungen festgehalten. In den systemwide requirements wurden systemweite Anforderungen an die Wetterstation-Software definiert.

## 5.2. Entwurf

Autoren: Alexander Schoch, Justin Schirdewahn

Im Zuge der Anforderungsanalyse wurden erste Entwürfe für das zu entwickelnde System erstellt. Anfang Januar entwickelten Philipp Barth (Architekt), Justin Schirdewahn (Entwickler) und Alexander Schoch (Entwickler) erste Designs für eine potentielle Datenbank- und Systemarchitektur. Diese wurden als Richtlinie für Prototypisierung verwendet, während der Entwicklung weiter verfeinert und mit neuen Anforderungen der Auftraggeber weiterentwickelt.

Einen ersten Entwurf des Projektes zu erstellen war doch komplexer als zunächst gedacht. Dank der in Software Engineering 1 erlernten Techniken an komplexe Probleme heran zu gehen haben wir es doch geschafft.

Zunächst haben wir die Aufgabe in verschiedene Use Cases aufgeteilt, um das Problem in kleinere verständlichere Abläufe aufzuteilen. Natürlich haben sich die Anforderungen im Laufe des Projektes geändert und die Use Cases mussten überarbeitet oder entfernt werden, weil diese einfach nicht mehr den gewonnen Informationen entsprachen.

Dank dieser kleinen Abläufe konnten wir eine erste grobe Architektur erstellen, um uns und dem Auftraggeber zu visualisieren, wie das Projekt aussehen könnte.

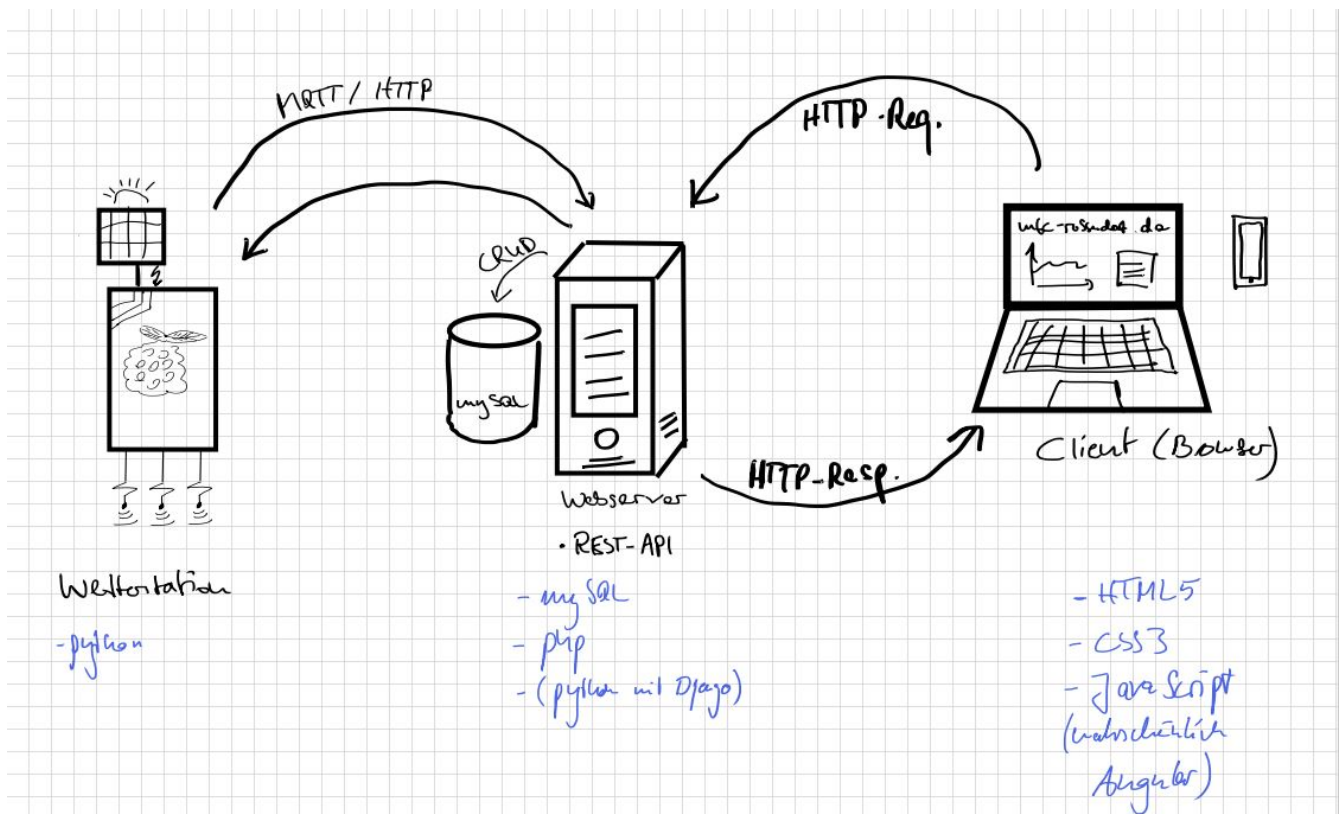


Abbildung 11. Erste Skizze der Architektur

Diese grobe Skizze war ein guter erster Anhaltspunkt auf dem alles nach und nach aufbaute. In dieser Skizze sind die verschiedenen Komponenten mit ersten Ideen, welche Technologien auf den jeweiligen Komponenten laufen könnten, zu sehen. Nach der Erstellung dieser groben Architektur wurde es nun Zeit genauere erste Entwürfe zu gestalten.

Ein wichtiger Entwurf war die Erstellung eines Data-Flow-Diagrammes.

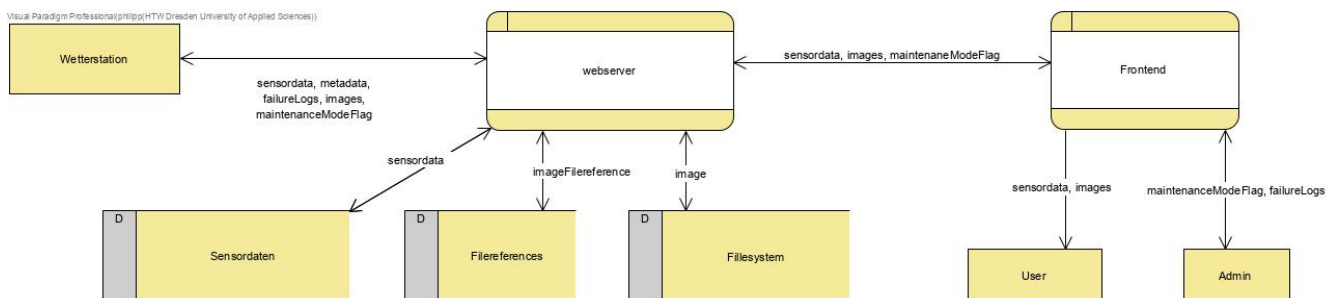


Abbildung 12. Data-Flow-Diagramm

Dieser Entwurf stellt unsere ersten Gedanken und Ideen dar, welche Daten an welchen Stellen benötigt werden und in welcher Art der Kommunikation die Komponenten die Daten liefern müssen. Es zeigt also nicht nur welche Daten zur Verfügung stehen sollen, sondern auch, ob die Komponenten Daten empfangen oder versenden. Erstellt wurde er mit Visual Studio Paradigm, da uns die Arbeit mit diesem Tool dank den Praktika in Software Engineering 1 nicht fremd war.

Einer der letzten Entwürfe, bevor wir überhaupt an die Implementierung gedacht haben, war der Entwurf eines Entity-Relationship-Modells. Auch hier mussten im Laufe des Projektes Anpassungen vorgenommen werden, aber unser erster Entwurf sah wie folgt aus.

## 1. Entity-Relationship-Modell

In dem von uns erstellten ERM wird der Aufbau der Datenbank dargestellt, welche später für die Verarbeitung der Daten essenziell wichtig ist. Jede Tabelle besitzt eine Primary Key ID (kurz PID), welche zur eindeutigen Identifizierung der Daten dient und unabhängig von diesen erstellt wird. Die Tabellen 'Raspi-Log' und 'Config' besitzen jeweils einen Fremdschlüssel auf die UID der Tabelle 'Admin-Credentials'.

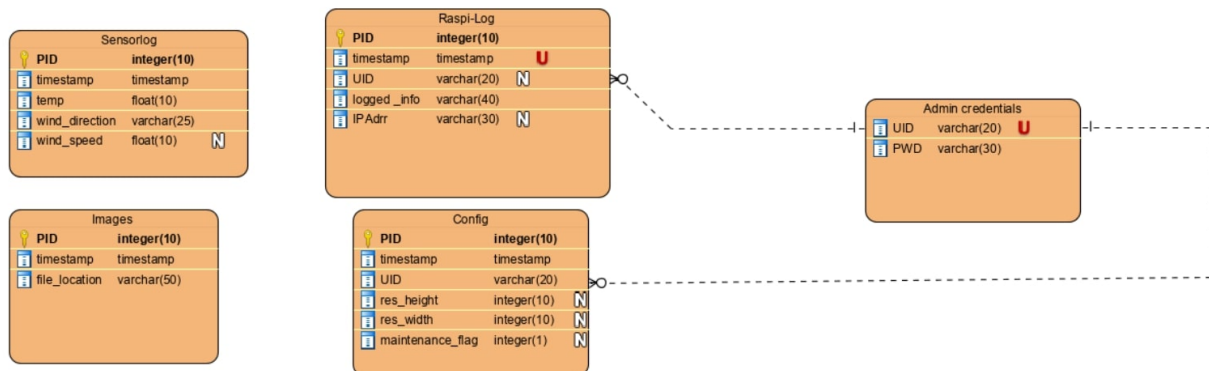


Abbildung 13. Erstes ERM

Vergleichen wir es mit unserem aktuellen Entwurf von unserem ERM sind deutliche Unterschiede zu sehen.

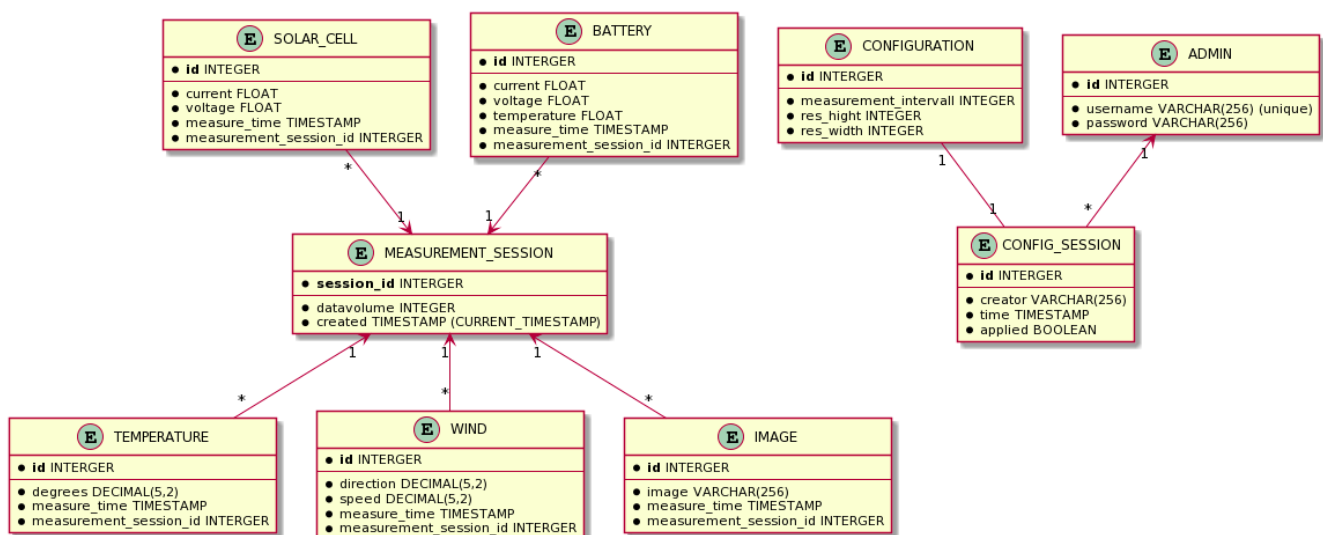


Abbildung 14. Aktuelles ERM

In Meetings mit den Kunden konnten die Anforderungen an die Datenbank nach und nach immer mehr heraus kristallisiert werden und in verschiedenen Entwürfen bis zum aktuellen Entwurf dargestellt werden.

Wichtig für die Kommunikation mit dem Kunden waren auch die Entwürfe von Wireframes für die Visualisierung unserer Vorstellungen für das Frontend. In verschiedenen Meetings konnten wir den Kunden einige Wireframes vorstellen und die Kunden konnten Ihre Wünsche und Anmerkungen zu diesen machen. So konnten wir am Ende ein Design entwickeln mit dem alle zufrieden waren.

Auch die Wireframes wurden mit Visual Studio Paradigm erstellt, da die Erstellung solcher Thema in den Praktika aus Software Engineering 1 war.



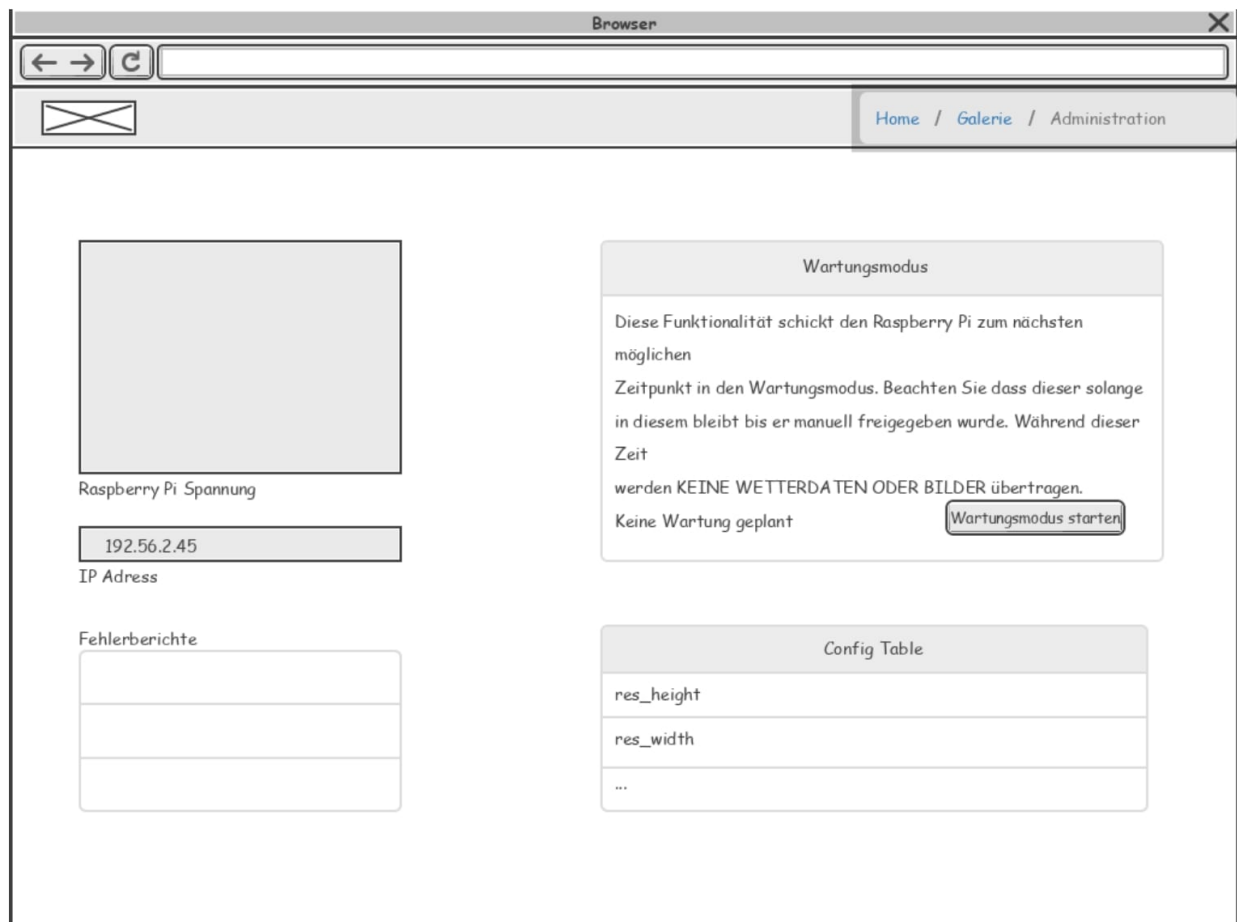


Abbildung 15. Erster Wireframe zum Adminpanel

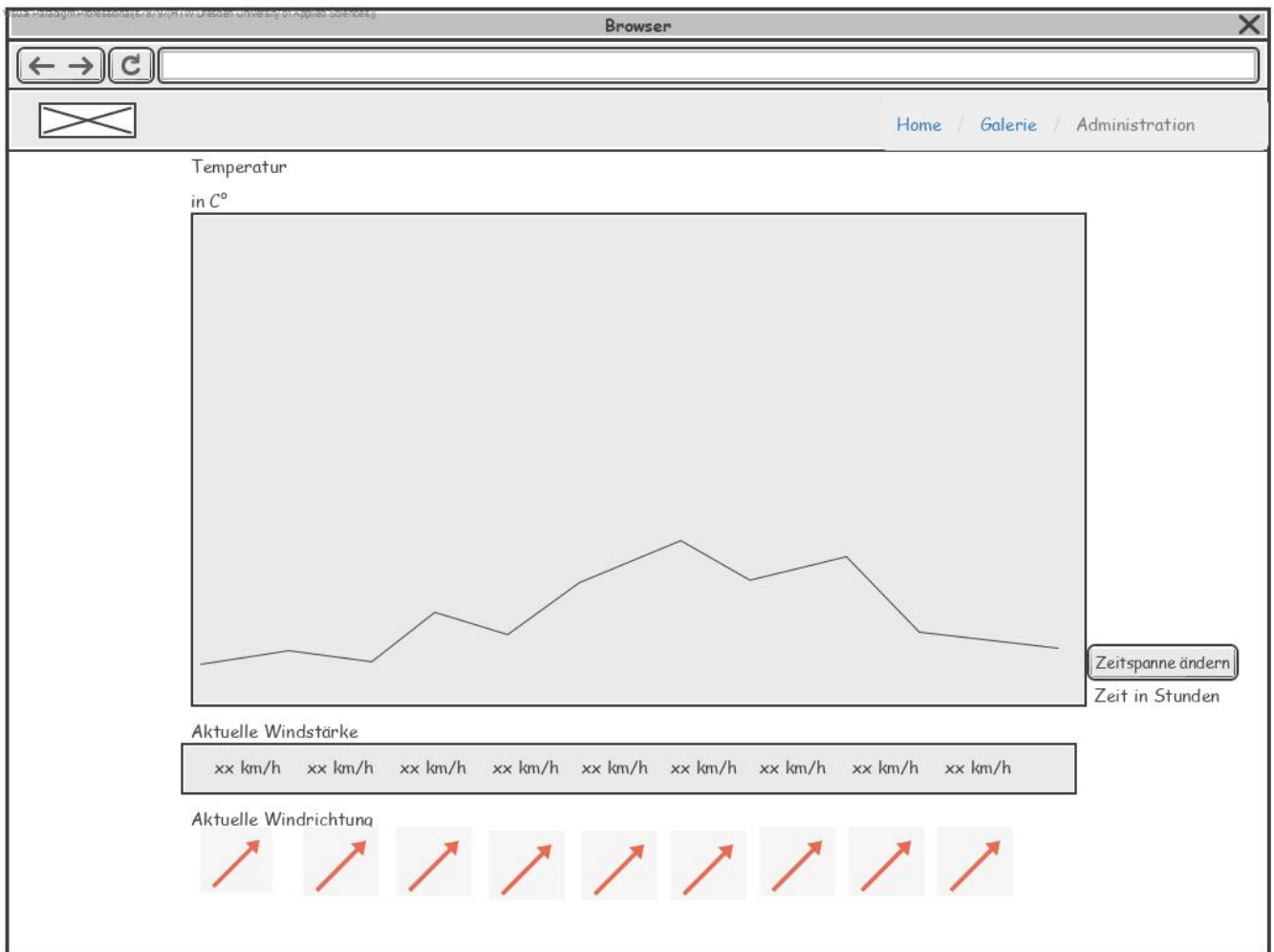


Abbildung 16. Wireframe zur Graphenanzeige

Der letzte Entwurf der hier gezeigt wird entstand am Anfang des Projektes aus den Anforderungen der Kunden. Da unsere Kunden diese Anforderungen nicht mehr stellen ist der folgende Entwurf veraltet und wird bei der Entwicklung nicht mehr beachtet. Im folgenden Aktivitätsdiagramm haben wir den Ablauf der Umstellung des Raspberry Pi in den Wartungsmodus visualisiert.

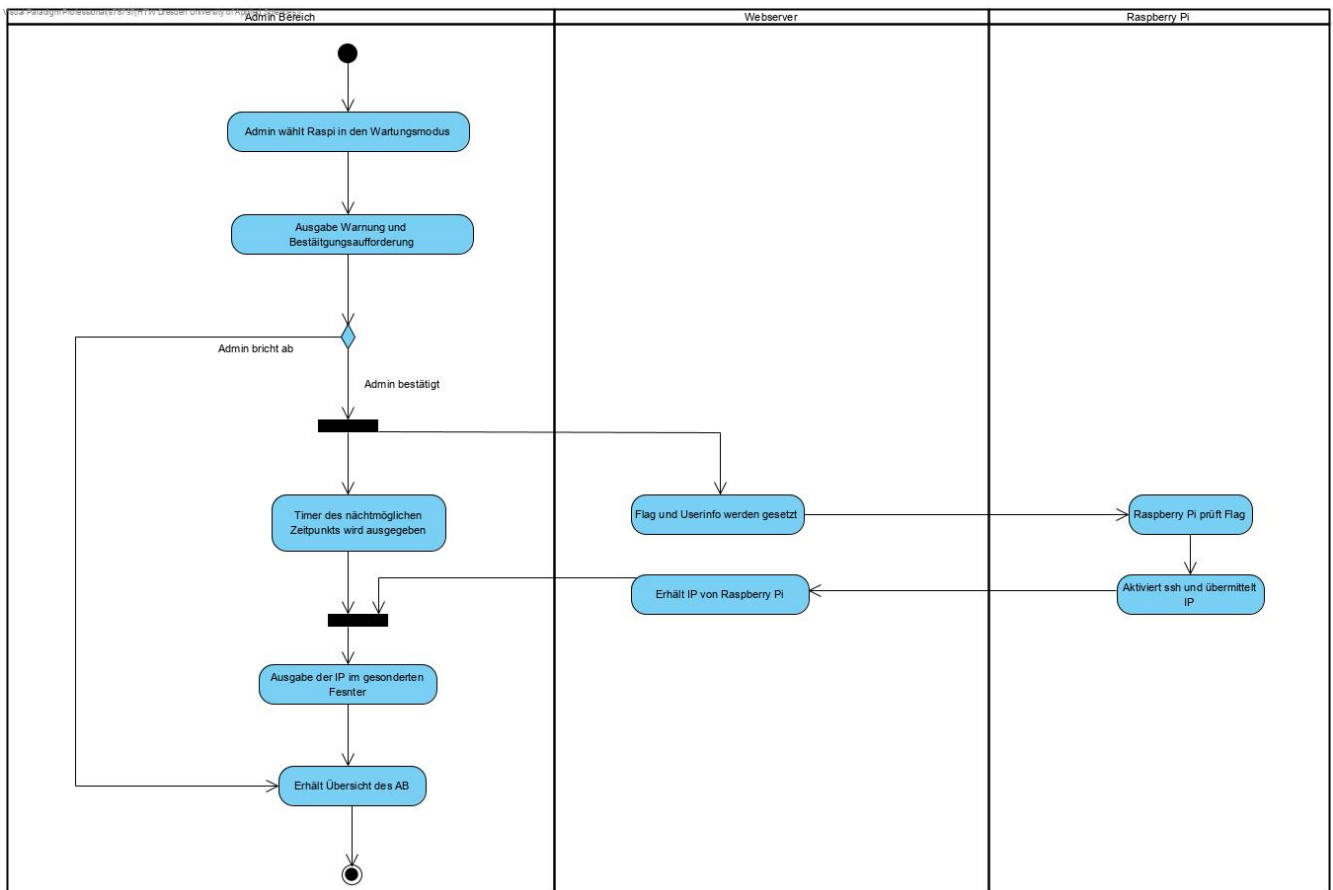


Abbildung 17. Verworfenen Wartungsmodus

## 5.3. Implementierung

Autoren: Alexander Schoch, Philipp Barth

Die Implementierung des Back-Ends begann Ende März, während die Implementierung des Frontends etwas später Ende April begann. Die Entwicklung des Back-Ends und der Skripte des Raspberry Pi's übernahm Philipp Barth, wobei ihn stellenweise Justin Schirdewahn unterstützte. Das Frontend wurde zunächst von Alexander Schoch entwickelt und später von Clemens Kujus unterstützt.

Das Frontend wurde in drei Bereiche unterteilt- Graphen, Webcam und Admin-Bereich.

Der Graphenbereich wurde von Alexander Schoch ab dem 22.04. entwickelt und im Verlaufe des Projektes, mit den sich ändernden Anforderungen der Auftraggeber in der Funktionalität erweitert.

Die Webcam wurde zunächst von Justin Schirdewahn begonnen ohne die Verwendung von weiteren Bibliotheken. Dies erwies sich als nicht sehr erfolgreich und nach einigen Prototypen unter der Verwendung verschiedener Bibliotheken für die Galerie erwies sich die "ngx-gallery" als eine Alternative. Diese wurde ab dem 29. Juni von Alexander Schoch implementiert.

Clemens Kujus implementierte ab Mitte Juni den Adminbereich und wurde bei Problemen von Alexander Schoch unterstützt.

Der Großteil des Back-Ends unterlag in der Implementierung Philipp Barths Verantwortung. Die erste Version der Rest-API, welche die Kernfunktionalitäten, Bilder und Sensordaten posten und

abfragen, beinhaltete, war Mitte April fertiggestellt. Bis Ende Mai wurde das Backend den neuen Anforderungen entsprechend erweitert. Es gab nun auch eine Schnittstelle zum Abfragen des verbrauchten Datenvolumens, sowie die Logik um diese zu berechnen. Nach der Einrichtung des Raspberry Pis seitens der Kunden kamen zusätzliche Anforderungen hinzu. Der Raspberry Pi sollte nun nicht mehr halbstündlich durch einen externen Trigger eingeschaltet werden und messen, sondern durchgehend eingeschaltet sein und minütlich die Sensoren abfragen. Desweiteren wurde auf Kundenwunsch das Backend so erweitert, dass auch die Informationen Spannung und Leistung der Solarzelle, des Akkus und des Ladecontrollers gepostet und abgefragt werden können. Aufgrund des nun viel höheren Datenaufkommens (1 Datensatz pro Minute) wurde zusätzlich ein Feature implementiert um die Daten zu aggregieren. Die aggregierten Daten konnten dann über eine Schnittstelle abgefragt werden. Diese Funktionalitäten wurden bis Ende Juni implementiert. In diesem Zeitraum wurde auch das Skript für den Raspi fertiggestellt um die Daten entsprechend an den Webserver zu übermitteln. In der letzten Phase des Projektes wurde noch die Authentifizierung für das Admin Panel hinzugefügt, sowie Logging hinzugefügt.

## 5.4. Test

Autoren: Clemens Kujus

Die Rolle des Testers wurde erst von Hannes Fogut ausgeführt. Mitte Mai 2020 wurden die ersten dynamischen Tests implementiert, wobei es noch Probleme gab diese mit Django zusammen auszuführen. Als diese gelöst wurden, liefen die Tests problemlos. Angestrebt wurde ein concurrent testing. Die Testdokumentation wurde vom Tester selbst bearbeitet. Der Architekt Philipp Barth half dem Tester bei der Erstellung und Dokumentation der Tests. Die Tests sind vor allem Positivtests und dienen der Verifikation. Die Validierung der Zwischenergebnisse erfolgte durch die regelmäßigen Treffen mit den Kunden. Akzeptanztests, d.h. die Abnahmetests, werden in Absprache mit den Kunden ausgeführt, sobald die Software den von den Kunden gewünschten Anforderungen entspricht.

Statische Tests wurden nicht mit Tools ausgeführt, sondern manuell durch die Teammitglieder. Dies wurde unter anderem durch die Nutzung von Pull Requests ermöglicht, bei dem eine zweite Person den Quellcode und die veränderten Dateien einsehen konnte und somit früh feststellen konnte ob offensichtliche Fehler oder Konventionsverstöße begangen wurden.

## 5.5. Übergabe und Dokumentation

Autoren: Clemens Kujus

Die Übergabe der Software erfolgte am 31.07.2020 an den Kunden Herr Thomas Brenner im Rahmen eines Meetings und Iterationsabschlusses. Dazu wurde ihm der Zugang zum Git-Repository des Projekts gewährt, welches er sich mit dem Release "Übergabeversion v1.0" auf seinen Computer geladen hat. Diese Vorgehensweise der Übergabe geschah im Einverständnis mit den Kunden. Zusätzlich zu der Software im "src" Ordner hat der Projektmanager die Anwenderdokumentation und die Betriebsdokumentation an Herr Brenner und Herr Professor Iwe per Mail übergeben. Die Software wurde vom Kunden dankend angenommen. Im Rahmen der Übergabe wurden noch einmal alle Funktionalitäten per Bildschirmübertragung auf einem Computer eines Entwicklers vorgeführt und abgenommen. Abnahmetests erfolgen im beidseitigen Interesse erst mit der Vollendung der Software durch das Projektteam, welche über den eigentlichen Rahmen

(Abgabetermin Software, Dokumentation) hinaus geht. Das Protokoll zum Treffen ist [hier](#) zu finden.

Die Dokumentation erfolgte während der Entwicklung soweit möglich. Anwenderdokumentation und Betriebsdokumentation wurden mit am 31.07.2020 übergeben. Der Projektbericht und die Entwicklerdokumentation wurden erst nach der Übergabe an den Kunden angefertigt. Die Testdokumentation entstand mit den ersten Testmöglichkeiten und wurde nach und nach erweitert.

# 6. Wesentliche Entscheidungen

Autoren: Alexander Schoch

Für die Entwicklung des Frontends haben wir uns für Angular entschieden. Von den Kunden war die Entwicklung einer Webapplikation gewünscht. Angular ist eines der größten JavaScript Frameworks, wird fortlaufend weiterentwickelt, bietet eine komponentenbasierte Model View Controller Architektur und eine ausführliche Dokumentation. Ebenso die hohe Stabilität des Frameworks

## 6.1. Chart.js für die Darstellung der Graphen

Um die Graphen darzustellen haben wir uns für die kostenlose, Open-Source JavaScript Bibliothek chart.js entschieden. Sie ist zwar nicht so flexibel und umfangreich wie z.B. die Bibliothek D3, ist aber deutlich einsteigerfreundlicher und bietet eine mehr als ausreichende Dokumentation. Chart.js hat somit unseren Funktionsansprüchen mehr als genügt und sich mit seinem flachen, einfachen Design in die Material Design UI gut eingefügt.

## 6.2. ngx-gallery für die Galerie und Lightbox

Ursprünglich war die selbstständige Entwicklung einer Galerie geplant. Da der Aufwand für die Implementierung jedoch sehr groß gewesen wäre, haben wir die Angular Bibliothek ngx-gallery verwendet.

## 6.3. Angular-JWT für die Authentifizierung

Da die API unseres Back-Ends hauptsächlich im JSON-Format mit dem Front-End kommuniziert haben wir uns für die Implementierung einer tokenbasierten Authentifizierung mit Angular-JWT entschieden. JWT (JSON Web Tokens) sind ein Industriestandard für die Generierung von Tokens. Die Verwendung von JSON Web Tokens ist zustandslos und daher sicherer als die Verwendung von Session-Variablen und Cookies. Ein weiterer Vorteil ist die hohe Zuverlässigkeit bei gleichzeitiger Minimierung des Netzwerk-Overheads.

## 6.4. Django als Webframework für das Back-End

In der Anforderungsspezifikation wurde die Entwicklung eines Back-Ends auf Grundlage einer Skriptsprache gewünscht. Wir haben uns für Django und das Django REST Framework entschieden. Diese basieren auf der Programmiersprache Python mit der einige der Entwickler schon erste Erfahrungen sammeln konnten. Des Weiteren kommt Django mit einer Vielzahl von Paketen für das Entwickeln von performanten und effizienten Webanwendungen. Ein weiterer Vorteil war der eingebaute Support für fast alle Datenbanksysteme, der uns die Integration der vom Kunden gewünschten MySQL-Datenbank vereinfachte. Django bot zudem einen hohen Sicherheitsstandard an. Während man bei der Entwicklung in PHP einen hohen Wissensstand und Erfahrung mit dieser Sprache braucht (die keiner von uns hatte), bot bereits Django's integriertes Authentifizierungsframework eine hohe Sicherheit.

# 7. Aufgetretene Probleme

Autoren: Clemens Kujus

## 7.1. Aufgabenverteilung

Die Aufgabenverteilung war am Anfang des Projekts so geplant, dass sich jeder aus einem Aufgabenpool (im Iterationsplan) seine Work Items rauszieht, Issues anlegt und bearbeitet. Das hat geklappt, auch wenn der ein oder andere sich erst nach Aufforderung Aufgaben rausgesucht hat. Nach der Projektpause Februar/März 2020 lief diese Methode allerdings nur noch sehr schwer an. Die Kenntlichmachung der gerade bearbeiteten Aufgaben wurden kaum vorgenommen, ebenso das Anlegen von Issues. Als ein Grund dafür ist sicherlich die etwas unübersichtliche Situation allgemein und im Hochschulbetrieb zu nennen, zum anderen die in den Iterationsplänen bis dahin ungenaue Rollenzuweisung.

Die Gegenmaßnahmen gegen die unzureichende Aufgabenverteilung wurden der Risk-List entnommen. Zum einen haben wir uns ab Mai öfter getroffen, meist wöchentlich (auch mit Kunden). Außerdem hat der Projektmanager Ende April/Anfang Mai die Aufgaben verteilt, damit jeder genau weiß was er bis wann zu erledigen hat. Da diese Aufgabenzuweisung allerdings unserer eher agilen Arbeitsweise widerspricht, wurde sie im Interesse aller nicht weiter fortgeführt. Des Weiteren wurde nun in den Iterationsplänen bei der Planung eine vorläufige Aufgabenverteilung vorgenommen, damit jeder schon mal seine auf ihn zugeschnittenen Aufgaben sieht.

### Priorisierte Aufgaben

- Front-End weiterentwickeln → Alex, Justin, Clemens
- Dokumentation überarbeiten und von den Teammitgliedern fordern → Josefin
- Testfälle spezifizieren und entsprechend implementieren → Hannes
- Rest Endpunkt images/recent erstellen → Philipp
- Rest Endpunkt erweitern für StromLogging → Philipp
- Model für Datenverbrauch → Philipp
- Authentifizierung einrichten → Philipp
- Model für config und configsession erstellen → Philipp

*Abbildung 18. Vorläufige Aufgabenverteilung in den Iterationsplänen*

## 7.2. Git-Banches und Commitabstände

Für die Anforderungserhebung und den Entwurf (Inception-/Elaborationphase) wurde hauptsächlich auf dem master-Branch in Git gearbeitet. Dies hat ausreichend gut funktioniert, da

die Aufgaben voneinander getrennt lösbar waren, d.h. wenn einer an einem Use-Case gearbeitet hat konnte ein anderer ohne Versionskonflikte erwarten zu müssen am Architecture Notebook arbeiten. Dadurch waren auch große Commits mit vielen Änderungen möglich, da die zu bearbeitenden Dokumente im Vorhinein ersichtlich waren.

Mit dem Anfang der Implementierung war es allerdings abzusehen dass diese Vorgehensweise nicht lange gut geht, weshalb wir uns für die Arbeit auf verschiedenen (vorerst allgemeinen) Branches wie "projectmanagement" und "frontend" entschieden haben. Auf dem master-Branch sollten nur lauffähige Versionen liegen, was durch Pull-Requests und damit einem 4-Augen-Prinzip sichergestellt werden sollte.

Das Problem der Branches war die Allgemeinheit. Ein Branch "frontend" beinhaltet die Arbeit am gesamten Front-End, was merge-Konflikte geradezu provoziert. Große Commits haben es außerdem sehr erschwert, den Code nachzuvollziehen. In einem Gastvortrag im Modul "Geschäftsprozessmodellierung" wurde nebenläufig eine Arbeitsweise in der Softwareentwicklung vorgestellt, die auf möglichst kleine Änderungen zum nächsten build setzt. Diese Vorgehensweise haben wir versucht zu adaptieren, indem wir möglichst detaillierte Branches und möglichst kleine Commits einsetzten. Dadurch hat sich die Frequenz an neuen Funktionalitäten im lauffähigen master-Branch erhöht und damit die Präsentation der Zwischenstände dem Kunden gegenüber vereinfacht, die Nachvollziehbarkeit der Commits ist gestiegen und die Anzahl an merge-Konflikten ist sehr gering, bzw. sind diese oft einfach zu beheben.

## **7.3. Vorstellungen und Ideen der Kunden**

Im Verlaufe des Projekts, und vor allem gegen Ende, haben sich die Vorstellungen der Kunden oft geändert. Zum einen in erleichternde Art und Weise, z.B. wurde uns die Datenakquise von den Sensoren abgenommen und das Datenvolumen der mobilen Internetverbindung wird an unsere Bedürfnisse angepasst. Zum anderen gab es allerdings auch durchaus widersprüchliche und erschwerende Anforderungen. So wurde uns z.B. an einem Meeting die Abkehr von PHP freigestellt, beim nächsten Meeting sollten wir (trotz vorangegangener Entwicklung) den Einsatz von PHP dann wieder in Betracht ziehen. Auch wurde teilweise auf in unserer Entwicklung nicht benötigte Implementierungen bestanden. Unsere Meinung zu diesen Vorstellungen und Eingriffen in unsere Entwicklungsarbeit haben wir dann immer wieder in Treffen deutlich gemacht und natürlich auch unter Berücksichtigung der Kundenwünsche so umgesetzt, dass ein sinnvoller Arbeitsablauf möglich war.

Ein größeres Problem waren die stetig wachsenden Anforderungen und Änderungen. Diese sind vor allem dem Umstand geschuldet, dass die Wetterstation vom Kunden parallel zu unserer Wetterstation-Software gebaut wurde. Dadurch waren diverse Messwerte und Messgrößen der Sensoren bis einige Tage vor Projektende nicht bekannt. Einige Features wurden und werden daher nach offizieller Abgabe der Software am 31.07.2020 freiwillig noch weiter bearbeitet, damit am Ende auch ein sinnvoller Realbetrieb möglich ist.

## **7.4. Mangelnde Erfahrung in der Entwicklung mit Angular**

Autor: Alexander Schoch



Ein Problem in der Entwicklung des Front-End war die fehlende Erfahrung im Umgang mit Angular. Keiner von uns hat zuvor eine Webapplikation entwickelt oder mit Typescript und Angular gearbeitet. Da von Anfang an mit großen Datenmengen und einer späteren Weiterentwicklung der Wetterstation zu rechnen war (durch z.B. weitere Sensoren), wäre ein Umstieg auf ein anderes Framework oder Bibliothek wie React oder Vue wenig sinnvoll gewesen. Wir mussten uns alles Nötige zeitaufwändig aneignen und best-practices lernen. Angular hat zwar eine steile Lernkurve, der Workload durch die sich ständig ändernden Kundenwünsche und Anforderungen war dennoch sehr hoch, weshalb Clemens Kujus bei der Implementierung mithalf und die Arbeit erleichterte.

# Besprechungsprotokolle

Verlinkungen wurden aufgrund besserer Übersichtlichkeit für den Projektbericht durch Pfadangaben in unserem GitHub-Repository <https://github.com/philippBa13/Wetterstation> ersetzt. Diese sind mit "Siehe GitHub-Repository: [...]" gekennzeichnet.

# 8. Protokoll zum Treffen am 09.01.2020

## 8.1. Zeit

17.00 bis 18.00 Uhr

## 8.2. Teilnehmer

- ☑ Alexander S.
- ☑ Justin S.
- ☑ Josephin
- ☑ Agustin
- ☑ Martin
- ☑ Auftraggeber

## 8.3. Iterationsplan

Siehe GitHub-Repository: [/project\\_docs/Iterationsplan\\_3.adoc](#)

## 8.4. Themen

### 8.4.1. Vorbereitete Fragen

Siehe GitHub-Repository: [/project\\_docs/questions/Fragen\\_kommendes\\_Kundengespraech.adoc](#)

#### Besprochene Inhalte

- Abgabezeitraum: Auf Juni geeinigt, wenn es nicht eher fertig werden sollte
- Technische Fragen
  - Logins für uns
    - Bekommen die Zugänge in den nächsten zwei bis drei Tage
    - Datenbank wurde noch nicht aufgesetzt
    - Nächste Woche neues Treffen, für die Mitglieder die Zugänge benötigen
  - Fragen technischer Art können an beide Auftraggeber gestellt werden
  - Softwarelösung, die Wetter simuliert bis reale Sensoren einsatzbereit sind
  - Backend zum Schluss in PHP!
  - Daten, welche verarbeitet werden sollen
    - Windstärke
    - Windrichtung
    - Böenhaftigkeit
    - Temperatur
  - Hardwarekonfiguration wird von Auftraggeber übernommen

- Funktion des automatischen Hoch- und Herunterfahrens wird von Auftraggeber übernommen
- Automatisches aktualisieren der Daten für User, wenn möglich
- Keine Livebildübertragung, wenn Raspi an ist
- Netzanbindung ausreichend vorhanden
- Adminbereich als Stand-Alone für uns implementieren
- Keinen Extraaufwand für hohe Sicherheit
- Bei Fehler in den Wartungsmodus, aber mit Infonachricht(Fehlertext)
- Windrichtung mit Pfeilen dargestellt, immer nur aktuelle Windrichtung
- Für Windrichtung am Windfinder orientieren (Kompass und daneben Richtungspfeil)
- IP wird sich wahrscheinlich dynamisch ändern (Kosteneffizient denken)
- Wireframes
  - Admin-Panel Spannungsanzeige darstellen
  - Wartungsmodus Terminal mit allen aktuellen Einstellungen zum Raspi anzeigbar
  - Wireframes ansonsten in Ordnung
- **Für nächstes Treffen**
  - Schnittstelle zwischen Pi und Webserver herstellen, damit Daten übertragen werden können
  - Wireframe für Diagrammanzeige
  - Datenbank einrichten

## 8.5. Offenes

- DSGVO Richtlinien im Umgang mit den Bildern von Personen auf dem Flugplatz
  - Zettel zum unterschreiben im Verein austeilen

# 9. Protokoll zum Treffen am 08.04.2020

## 9.1. Zeit

17.00 bis 18.15 Uhr

## 9.2. Teilnehmer

### **intern:**

- ☒ Clemens
- ☒ Alexander
- ☒ Justin
- ☒ Philipp
- ☒ Josephin
- ☐ Agustin
- ☒ Hannes

**extern:** Herr Professor Iwe Herr Thomas Brenner

## 9.3. Iterationsplan

Siehe GitHub-Repository: `/project_docs/Iterationsplan_10.adoc`

## 9.4. Themen

- Python & Django sind ok, allerdings ist eine Skriptsprache mit ordentlichen Kommentaren Voraussetzung
- Raspberry ist wieder erreichbar
- config UND configsession als Sitzungseinstellungen für Datenbank (Nutzer und Sitzung den Einstellungen zuordnen)
- Configwerte nicht in eine einzelne Tabelle schreiben, sondern weitere Tabelle mit Configeinträgen mit Verweis auf Tabelle mit Configart und Wert
- noch eine Spalte in Configtabellen (resultspalte), in der steht ob eine Einstellung geklappt hat
- Backend kommt auf gleiches Gerät wie die DB

Was übernehmen die Auftraggeber?

- Eine Datei pro Sensor, die wir in Zeitabschnitten abfragen (liegen in einem def. Ordner, Textformat)
- Raspberry im Wartungsmodus (wir sollen nur Information geben, dass sich der Raspberry nicht runter fahren soll)

# 10. Protokoll zum Treffen am 20.05.2020

## 10.1. Zeit

17.00 bis 18.30 Uhr

## 10.2. Teilnehmer

### intern:

- ☒ Clemens
- ☒ Alexander
- ☒ Justin
- ☒ Philipp
- ☒ Josefin
- ☐ Agustin
- ☒ Hannes

**extern:** Herr Professor Iwe, Herr Brenner

## 10.3. Iterationsplan

Siehe GitHub-Repository: `/project_docs/Iterationsplan_12.adoc`

## 10.4. Themen

1. Vorstellung der Ergebnisse (vor allem Front-End mit den Datengraphen)
2. Klärung weiterer Fragen

## 10.5. Ergebnisse

1. Front-End
  - a. Die Art der Darstellung ist OK
  - b. Die Darstellung der Windrichtung soll ähnlich der von "Windfinder" erfolgen (s. Website mfcR)
  - c. Admin-Bereich hat niedrige Priorität, sollte aber bald angegangen werden
  - d. Die Weiterentwicklung des Front-End wird vor allem von Alexander, Justin und Clemens übernommen
2. Back-End
  - a. Die Eintragung und Abfrage der Werte ist in der groben Struktur (JSON) OK
  - b. Die Werte können auch über eine View dargestellt werden, was bedeutet dass die Daten welche zu einem Messpunkt gehören auch durch eine ID in der DB diesem zugeordnet

werden können → verschönert nur, daher niedrige Priorität

- c. Daten wie der Ladezustand (bzw. Stromverbrauch) des Akkus und der Datenverbrauch (Internet) sollen mit geloggt und in der DB gespeichert werden, wobei uns die Struktur dieser Daten noch zugeschickt wird
- d. Der Adminbereich sollte wie im Front-End auch im Back-End bald angegangen werden (z.B. Config-Tabelle erstellen)

### 3. Sonstiges

- a. Der momentane Projektstand/-verlauf wird vom Kunden für zufriedenstellend befunden
- b. Das nächste Treffen mit dem Kunden ist am 27.05.2020 17 Uhr geplant

# 11. Protokoll zum Treffen am 27.05.2020

## 11.1. Zeit

17.00 bis 18.10 Uhr

## 11.2. Teilnehmer

### **intern:**

- ☒ Clemens
- ☒ Alexander
- ☒ Justin
- ☒ Philipp
- ☒ Josefin
- ☒ Agustin
- ☒ Hannes

**extern:** Herr Professor Iwe, Herr Brenner

## 11.3. Iterationsplan

Siehe GitHub-Repository: [/project\\_docs/Iterationsplan\\_13.adoc](/project_docs/Iterationsplan_13.adoc)

## 11.4. Themen

1. Vorstellung der Ergebnisse (vor allem Front-End mit den Datengraphen)
2. Klärung weiterer Fragen

## 11.5. Ergebnisse

1. Front-End
  - a. Die Art der Darstellung ist OK
  - b. Das Aussehen des Front-End muss noch überarbeitet (verschönert) werden, aber die Funktionalität passt
  - c. Luftbild des Flugplatzes für die Windrichtung wird noch gegeben
  - d. Für Windrichtung/-geschwindigkeit sind nur die letzten 3-4 Stunden interessant
  - e. Datenvolumen/Spannungsverbrauch soll im Admin-Panel in Tages-Intervallen dargestellt werden
2. Back-End
  - a. Die Session ID sollte auf Anfrage vom Raspberry vom Datenbankserver generiert und dann dem Raspberry übergeben werden, so ist sichergestellt das Daten und Bild die gleiche ID



bekommen

b. Für das Datenvolumen ist die Größe des Bildes ausreichend

3. Sonstiges

a. Realdaten werden in den kommenden Tagen noch gegeben

b. Bilder könnten bei geringer Größe und nützlichem Inhalt (d.h. man erkennt auch etwas) auch bei Nacht verschickt werden

c. Die Definition der Datenstruktur kann in einer XSD-Datei vorgenommen werden, welche uns zugeschickt wird

d. Das nächste Treffen mit dem Kunden ist am 27.05.2020 17 Uhr geplant

# Ergebnisse

# 12. Projektergebnisse

Autoren: Clemens Kujus

## 12.1. Ziel am Projektanfang

Das Ziel des Projektes war eine funktionstüchtige und in den regulären Betrieb eingebundene Wetterstation-Software, welche die von der Wetterstation gemessenen Werte/aufgezeichneten Bilder zyklisch an eine Datenbank auf einem Webserver sendet, diese dort speichert und dem User durch Zugriff vom Browser aus anzeigt. In einem durch Login geschützten Adminbereich soll der Administrator Informationen über den Systemzustand erhalten und den Raspberry in einen Wartungsmodus versetzen können. Messwerte sollen interpoliert, Bilder in einer Galerie und diese Daten allgemein mit Zeitraumauswahl dynamisch dargestellt werden.

## 12.2. Zielerreichung

### 12.2.1. Was haben wir erreicht

Erreicht haben wir mit der Übergabeversion eine lauffähige Software, die Werte aus einer lokalen Datenbank ausliest und dem Nutzer auf einer Weboberfläche anzeigt. Dabei wurden für die Daten logisch sinnvolle Werte benutzt, wie z.B. Windrichtung nur von 0°-359°. Die Software kann alle geforderten Messwerte (Temperatur, Windgeschwindigkeit, Windrichtung) interpoliert, dynamisch und mit Zeitraumbezug anzeigen. Die Galerie zeigt die Bilder anforderungsgemäß an, d.h. das aktuellste Bild wird hervorgehoben als erstes angezeigt und die Bilder werden für jeden gesuchten Zeitraum (tagesgenau) angezeigt. Im durch ein tokenbasiertes Authentifizierungsverfahren geschützten Adminbereich kann der Administrator systemrelevante Informationen durch Graphen einsehen. Dabei sieht er durch eigene Auswahl entweder Strom/Spannung von Raspi und Akku, Leistungsaufnahme des Raspi und Ladestrom des Akkus. Dabei werden Leistungsaufnahme und Ladestrom nur auf Grundlage von hart implementierten Beispielwerten angezeigt.

Die Kunden sind mit der übergebenen Software zufrieden. Zwar ist diese nicht für den Realbetrieb geeignet, aber die Hauptfunktionalitäten sind vorhanden. Die für einen Einsatz nötigen Funktionalitäten und Deployment sind in ihrer Menge absehbar und werden aufgrund gegenseitigen Interesses vom Projektteam noch durchgeführt.

### 12.2.2. Was haben wir nicht erreicht

Nicht erreicht haben wir die Inbetriebnahme der Software. Das liegt daran, dass wir durch die stetigen neuen/geänderten Anforderungen uns immer wieder neu darauf konzentriert haben dass das Produkt überhaupt den Anforderungen der Kunden entspricht. Das Deployment wurde außerdem dadurch schwierig, dass der finale Ort der Software, d.h. der Webserver, uns erst kurz vor Abgabe der Übergabeversion durch den Kunden bekannt gemacht wurde. Wir haben diesen schon deutlich früher angefragt, aber die Kunden mussten selbst erst noch einen Webserver wählen.

Des Weiteren ist kein Wartungsmodus verfügbar. Die Notwendigkeit eines solchen Wartungsmodus fiel schlichtweg aus den Anforderungen heraus, da der Raspi entgegen der ursprünglichen

Aufgabenstellung nicht einschläft sondern immer an ist.

Die Anzeige der Informationen über den Systemzustand ist schwierig zu bewerten, da die Kunden selbst erst mitten in der Implementierungsphase entschieden haben was da ungefähr alles möglich sein soll und diese Anforderungen final erst kurz vor Abgabe der Übergabeversion spezifiziert hat. Es sind also Systeminformationen vorhanden, welche allerdings nicht den finalen Anforderungen der Kunden entsprechen.

# 13. Reflexion Philipp Barth

## 13.1. Teamrolle: Architekt / Entwickler

Im Projekt "Wetterstation" habe ich in der Rolle des Architekten mitgewirkt und das Vorhaben seit Beginn von Software Engineering 1 an begleitet. Auch wenn das Projekt retrospektivisch nicht immer nach Plan lief und von vielen äußeren Einflüssen, wie den sich oft ändernden Anforderungen stellenweise hin und her geschubst wurde, war es aus meiner Sicht ein Erfolg. Das Projekt hat bei mir auf verschiedenen Ebenen zu positiven Lerneffekten beigetragen. Zum einen ist es eine sehr wertvolle Erfahrung in einem echten bzw. sehr realitätsnahen Projekt mitgewirkt zu haben. Wie eingangs schon angedeutet, wurde uns schnell klar warum es Sinn macht eine agile Vorgehensweise im Software Engineering zu nutzen, da sich unsere Anforderungen oft änderten. So entwickelte sich unser Raspberry Pi, der zunächst nur jede halbe Stunde über einen externen Trigger aufwachen sollte und somit 48 Messwerte pro Tag generiert, zu einem dauerhaft aktiven, minütlich messenden Datengenerator mit 1000 Messwerten pro Tag. Des Weiteren hat sich gezeigt wie wichtig regelmäßiger Kontakt und Feedbacksessions mit dem Kunden sind, da nur so sichergestellt werden kann, dass am Ende ein Produkt entsteht was den Anforderungen genügt.

Auch auf technischer Ebene habe ich als Architekt und Entwickler so einiges dazu gelernt. So zum Beispiel die Fähigkeit sich in unbekannte Technologien einzuarbeiten, wie in unserem Fall python mit Django und Django-Rest, RESTful Webservices und Single Page Applications. Zusätzlich konnte ich meine SQL Fähigkeiten auffrischen und vertiefen. Wir benötigten im Backend eine Funktion um die große Menge an Daten auf eine fixe Anzahl zu aggregieren, so dass das Frontend diese vernünftig verarbeiten kann. Nach einiger Recherchezeit und vielen Trial and Error SQL Befehlen auf der Konsole hatte ich dann endlich eine funktionierende Lösung.

Neben den regelmäßigen Treffen innerhalb des Teams, möchte ich auch die zweiwöchentlichen Kundentreffen positiv hervorheben. Dieser Rhythmus hatte sich nach ersten Anlaufschwierigkeiten sehr gut eingepegelt, wodurch eine regelmäßige Planung und Rücksprache mit den Kunden möglich war. Es wurden jeweils die Ergebnisse der letzten zwei Arbeitswochen vorgestellt, Feedback gesammelt und schließlich die Ziele für das nächste Treffen vereinbart. Diese Arbeitsweise hat sich gut integriert und meiner Ansicht nach wesentlich dazu beigetragen das Produkt zu verbessern.

Auch die Kommunikation innerhalb des Teams, speziell zwischen den Entwicklern hat gut funktioniert. Wir haben die Aufgaben grob in Frontend und Backend unterteilt und die APIs vor Beginn der Entwicklung eines neuen Features definiert. Dadurch konnten Abhängigkeiten verringert und Unklarheiten im Vorhinein beseitigt werden.

Sollte ich wieder an einem ähnlichen Kundenprojekt mitwirken, werde ich mehr Acht darauf legen einen Featurestopp frühzeitig in die Wege zu leiten. Hier haben wir zu oft nachgegeben und zu viel bejaht. Somit kamen wir mit einer gewissen Verzögerung in die Schlussphase des Projekts. Es kann dadurch mehr Wert auf Qualitätssicherung gelegt werden und somit die Robustheit des Systems gesteigert werden.

Zusammenfassend hat mir das Projekt nicht nur Spaß gemacht, sondern auch noch einen Lerneffekt mit sich gebracht, der mir in zukünftigen Projekten sicher weiter hilft. Auch das Thema war spannend, und hat die Arbeitsfreude am Projekt für mich bis zum Schluss aufrechterhalten.

# 14. Reflektion Hannes Fogut

## 14.1. Teamrolle: Tester, Entwickler

Der aller erste und in meinen Augen wichtigste Punkt war das Arbeiten miteinander. Ich kam erst im zweiten Teil des Projekts in das Team und muss sagen, dass ich super aufgenommen und integriert wurde. Man gab mir Zeit um mich in das Projekt einzuarbeiten und stand mir bei jeglichen Fragen zur Seite. Dieser Punkt ist mir persönlich besonders wichtig gewesen, da es mir das komplette Gegenteil zum letzten SE2 Projekt aufgezeigt hat. Es war zudem eine Herausforderung fuer mich, sich in ein neues Thema herein zu finden, welches aus dem Testen(neu) in Python(neu) bestand. Meine Erfahrungen aus dem letzten Jahr wurden dazu auch gut aufgenommen und ich glaube, dass ich mich damit auch sehr gut in das Projekt einbringen konnte.

Rueckblickend stell ich mir nun folgende Fragen:

- Was habe ich Gelernt?

Durch meine Rolle im Team habe ich vertieft Einblicke in die Welt des Testens bekommen. Ich habe zahlreiche Moeglichkeiten kennengelernt, wie man Software testen kann. Ausserdem ist mir bewusst geworden, wie maechtig doch eigentlich dieses "Tool" ist (Beispiel: Test Driven Development). Dazu kamen Erfahrungen in der Programmiersprache Python, die ich zuvor noch nie verwendet hatte. Leider sind es noch keine fundierten Kenntnisse und das erzeugen von Quellcode faellt mir nach wie vor schwer, jedoch habe ich nun ein grobes Verstaendniss und kann den Quellcode verstehen.

- Worauf bin ich stolz?

Tatsaechlich weniger auf das Endprodukt, als auf den Weg dahin. Am Anfang des Semesters dachte ich, dass es aehnlich wie das vorherige Jahr wird, da ich ab und zu parallelen erkannt habe. Jedoch ist das Projekt trotz Corona gut angelaufen und wurde gut beendet. Unsere Kunden sind mit unserem Produkt zufrieden, worauf man sicherlich auch stolz sein kann.(Wobei man hier ein riesen Lob an die Entwickler aussprechen muss)

- Was hat gut funktioniert?

Sowohl die Kommunikation zwischen dem Kunden und dem Team, als auch die Kommunikation untereinander.

- Was wuerde ich beim naechsten Projekt anders machen?

Wir hatten sehr eifrige Kunden, die immer und immer wieder noch eine zusätzliche Idee fuer das Projekt hatten. Wir haben diese alle angenommen und probiert zu realisieren, obwohl an manchen Stellen die Zeit knapp war. Es ist zwar gut, dass man den Kunden zufriedenstellen will, jedoch hat uns das an der ein oder anderen Stelle viel Zeit gekostet.

# 15. Reflexion Clemens Kujus

## 15.1. Teamrolle: Projektmanager / Analyst

Meine Rolle im Projekt war primär die des Projektmanagers und nebenläufig die des Front-End Entwicklers. Im großen und ganzen habe ich hier gelernt, dass es doch recht unterschiedliche Auffassungen von Teamarbeit gibt und es schwierig ist, den Aufwand im Team gleichmäßig zu verteilen. Außerdem ist es äußerst hilfreich, immer wieder von Teammitgliedern den aktuellen Status ihrer Arbeit zu erfragen und falls nötig mit Nachfragen nachzuhaken. Die Arbeit an einem realen Projekt hat auch dazu beigetragen, dass wir nicht wie in anderen Belegprojekten das Rad neu erfinden und alle die gleiche Aufgabe lösen, sondern dass es für mich auch sehr motivierend ist wenn die Software Aussicht hat am Ende tatsächlich im Realbetrieb zu landen und genutzt zu werden. Das es sich um ein komplexes reales Problem handelt hat bei mir auch dazu beigetragen, dass ich den Nutzen der Vorgehensweisen in der Softwareentwicklung besser verstanden habe und die Grenzen des Wasserfallmodells doch schnell erreicht sind. Des Weiteren habe ich festgestellt, dass der Nutzen den Aufwand sowohl bei der Protokollierung von Meetings, dem Verteilen von Aufgaben als auch technischen Skizzen, Kommentaren usw. deutlich überwiegt.

In meiner Arbeit als Front-End Entwickler habe ich viel über die Arbeit im Team mit Software zur Versionsverwaltung und die Programmierung im Web-Bereich gelernt. So habe ich z.B. die Nutzung kleiner Commits und Git-branches kennen und nutzen gelernt. Außerdem habe ich einen Einblick in die große Menge an vorhandenen Frameworks und Tools bekommen und gemerkt dass es gar nicht so einfach ist, eine Website nach den Vorstellungen des Kunden zu gestalten.

Besonders stolz bin ich zum einen darauf, dass wir am Ende des Projekts eine Software abgeben, die sehr gute Aussicht hat tatsächlich genutzt zu werden und den Nutzern letztendlich Freude bereiten soll. Zum anderen möchte ich die Leistung des Teams hervorheben, welches trotz Unterschiede im Arbeitsaufwand auch als Team agierte und als Ziel die fertige Software vor Augen hat.

Gut funktioniert hat vor allem in der zweiten Projekthälfte die Kommunikation untereinander und mit den Kunden. Durch regelmäßige Onlinetreffen sowohl im gesamten Team (spätestens alle 2 Wochen, oft wöchentlich) als auch in kleineren Kreisen, z.B. unter den Entwicklern, waren immer mehrere Beteiligte über den aktuellen Stand und akute Probleme informiert. Da ab Mai 2020 die Treffen fast immer wöchentlich mit den Kunden statt fanden, waren diese auch immer über Probleme und den aktuellen Stand informiert und konnten uns direkt Feedback geben. Die Kommunikation auf technischer Sicht war mit den Kunden aufgrund derer Fachexpertise in dem Bereich ebenfalls sehr angenehm. Gut funktioniert hat aus meiner Sicht des Projektmanagers vor allem die Implementierung. Mit Angular und Django haben die Entwickler meiner Meinung nach sehr gute Frameworks herausgesucht, mit welchen sie sehr schnell neue Funktionalitäten implementieren konnten und es mir ermöglicht haben mich recht schnell einzuarbeiten und mitentwickeln zu können.

Beim nächsten Projekt würde ich als Projektmanager mehr darauf achten dass alle Teammitglieder regelmäßig am Projekt arbeiten und ihre Aufgaben auch im geforderten Zeitraum versuchen zu erledigen. Außerdem sollten einheitliche "Richtlinien" für den Umgang mit Git und anderen genutzten Tools von Anfang an etabliert und auch geprüft werden, da ich festgestellt habe dass der richtige Umgang mit Hilfsmitteln wie Git und AsciiDoc bei manchen Teammitgliedern kaum



bekannt ist. Des Weiteren sollte ich beim nächsten mal mehr darauf achten, dass sich das Projekt auch in einem für uns machbaren Rahmen hält. Die Kunden hatten stetig neue Ideen für die Software. Auf der einen Seite möchte ich die Entwickler da nicht aufhalten diese Funktionalitäten zu implementieren, da die Wetterstation-Software dadurch nur brauchbarer wird. Zum anderen wurde es dadurch nicht möglich die Software bis zur Abgabe am 31.07.2020 nach den letzten Wünschen der Kunden fertig zu stellen. So wurden z.B. die Diagramme im Adminpanel 2 Tage vor Abgabe vom Kunden nochmal umgeworfen, da doch noch andere Daten vom Raspberry gesammelt werden. Die neuen Diagramme haben es natürlich nicht in die Übergabeverision geschafft. Dadurch ist diese Version für den Kunden allerdings auch nicht praktisch nutzbar. Ich würde dem Kunden beim nächsten Projekt daher wahrscheinlich eine Deadline für neue Features geben, damit sich die Entwickler nach der Deadline auf die bis dahin geforderten Funktionalitäten konzentrieren können.

Zusammenfassend werte ich das Projekt für mich persönlich als Erfolg. Ich konnte Einblicke in das Projektmanagement und die Entwicklerarbeit gewinnen. Durch die "realen" Kunden konnte ich auch gut die Entwicklung der Anforderungen durchspielen und konnte teilweise auch erkennen, dass man den Kunden manchmal eher vorschlagen sollte was er braucht und nicht immer genau das umsetzen sollte was er sich in dem Moment denkt.

# 16. Reflexion Justin Schirdewahn

## 16.1. Teamrolle: Entwickler, Tester

Seit Software Engineering 1 bin ich im Projekt "Wetterstation" in der Rolle des Entwicklers tätig. Hauptsächlich habe ich bei der Arbeit an diesem Projekt gelernt, dass die Kommunikation zwischen dem Team und dem Kunden und die zwischen den einzelnen Rollen im Team extrem wichtig ist. Auch wenn viele externe Einflüsse das Projekt an einigen Stellen behinderte konnten alle Hindernisse durch regelmäßigen Informationsaustausch doch früher oder später überwunden werden. Im Nachhinein betrachtet hätten einige Dinge anders beziehungsweise besser gemacht werden können. So hätte ich mich mehr mit unserem anderen Entwickler und unserem Architekten in Verbindung setzen sollen, um mehr Überblick über benötigten noch offenen Aufgaben zu erlangen und die Arbeitsteilung besser gestalten zu können. Trotz dieser kleinen Probleme und ab und zu holprigen Momenten bin ich doch stolz darauf mit meinem Team am Ende ein funktionierendes System, welches zur Lösung eines realen Problems dient, erschaffen zu haben.

Bei der Bearbeitung des Projektes konnte ich Parallelen zwischen den erlernten Modellen und Vorgehensweisen aus Software Engineering 1 und der tatsächlichen Arbeit am Projekt sehen. So wurde uns als Team schnell bewusst, dass eine die Aufteilung in Iterationen besser geeignet war, als das einmalige definieren von Anforderungen und nicht funktionalen Anforderungen. Dies wurde besonders deutlich durch die sich immer wieder ändernden Wünsche unseres Kunden.

Wie schon erwähnt war ich über die Wichtigkeit von Kommunikation doch erstaunt. So konnten wir in Software Engineering 2 in regelmäßigen Meetings, sowohl untereinander, als auch mit dem Kunden, gut auf neue Wünsche und Anforderungen reagieren und die Zwischenergebnisse mit dem Kunden zusammen betrachten.

Doch nicht nur in Sachen Kommunikation, Planung und Teamarbeit habe ich neue Dinge gelernt. Auch aus technischer Sicht konnte ich viele neue Eindrücke gewinnen. Das einarbeiten in verschiedene Frameworks und neue Technologien war manchmal mühselig, aber am Ende doch Lohnenswert. Lohnenswert in der Hinsicht das Potential von Frameworks und Bibliotheken zu erkennen, im Gegensatz zu einer von Grundauf selbst erstellten und oft unnötig kopfzerbrechenden Lösung. So wurde die selbst programmierte Lösung für die grafische Anzeige von Bildern auf der Website, welche nicht richtig funktionierte, durch eine Bibliothek ersetzt, welche alle gewünschten Funktionalitäten erfüllte. Die Arbeit mit unbegrenzten Möglichkeiten, wie dieser, hat mich sehr fasziniert und begeistert.

Ein wichtiger und nicht zu unterschätzender Anteil an einem Projekt ist natürlich auch die Dokumentation. Bei diesem Punkt konnte ich Parallelen zwischen meiner Tätigkeit als Werkstudent und diesem Projekt sehen und habe wieder einmal gemerkt, wie wichtig eine gute Dokumentation von Entwicklerseite ist. Auf Arbeit betrachte ich diesen Punkt eher aus Anwenderseite und verstand oft nicht, wieso man gewisse Informationen nicht einfach aufgeschrieben hat. Das Projekt hat mir die Sicht eines Entwicklers gezeigt und ich verstehe nun, dass es doch viel schwerer ist als man denkt, eine gute Dokumentation zu schreiben.

Abschließend kann ich sagen, dass ich doch in vielen Aspekten neue Dinge gelernt habe und schon erlerntes vertiefen konnte. Die Arbeit am Projekt hat mir viel Spaß gemacht und ich bin stolz auf das was wir am Ende alles als Team erreicht haben.

## 16.2. Reflexion Agustin Calvimontes

### 16.2.1. Teamrolle: Deployment Engineer, Technical Writer

In unserem Projekt der "Wetterstation" habe ich vieles gelernt. Die Entdeckung und Verwendung mehrerer Tools hat Spaß gemacht. Wie zum Beispiel GitHub als unser wichtigstes Tool für die Arbeit am Projekt. Ich bin auch froh mich mit Python und AsciiDoc beschäftigt zu haben. Die Arbeit mit den Kommilitonen war auch sehr wertvoll und hilfreich. Ich habe schnell gelernt, dass die Kommunikation im Team sehr bedeutend ist. Der Umgang mit dem Kunden war auch eine gute Erfahrung. Seit Beginn hat es mir gefallen an etwas zu arbeiten, das wirklich von dem Kunden genutzt werden wird. Es hat mir einen großen Einblick gegeben wie die Arbeit im Bereich Software Engineering aussieht und somit wie der Alltag abläuft. Mit dem Endzustand der Software bin ich äußerst positiv beeindruckt, denn trotz der Probleme wie der Ausfall von unseren Treffen in Person, haben wir stets Lösungen gefunden und auch dank starkes Projektmanagements haben wir das Projekt abgeschlossen. Schließlich bin ich froh an diesem Projekt teilgenommen zu haben.

# 17. Reflexion Alexander Schoch

## 17.1. Teamrolle: Entwickler/ Projektmanager

Innerhalb des Projekts war ich seit Software Engineering I als Entwickler, dabei hauptsächlich für die Entwicklung des Front-Ends zuständig. Rückblickend habe ich gelernt, dass Kommunikation und Arbeitsverteilung ein kritischer Punkt in einem Projekt solchen Ausmaßes sind. So hat sich meine ursprüngliche Zweitrolle als Analyst in Software Engineering I zu einem Ersatzprojektmanager im Falle von Clemens Abwesenheit geändert. Ebenso wurde mir klar, wie wichtig der Einsatz von agilen Methoden im Projektverlauf sind. Viele Anforderungen an das System änderten sich mit dem physischen Bau der Wetterstation. Zum Beispiel sollte die Datenakquise nicht mehr einmal halbstündlich erfolgen, sondern minütlich und einmal halbstündlich an den Server geschickt werden und eine Vielzahl von Messwerten in Bezug auf das Adminpanel änderten sich. Die Umsetzung solch drastischer Änderungen wären im Wasserfallmodell kaum möglich gewesen. Insgesamt hat es mir viel Spaß gemacht an einem Softwaresystem zu arbeiten, von dem man ausgehen kann, das am Ende auch eingesetzt wird und für die Vereinsmitglieder des MFCR einen Mehrwert bietet. Auch wenn innerhalb des Teams nicht alles glatt lief, die angestrebten Ziele einzelner Iterationen nicht immer zum Ende dieser erreicht wurden und die Aufgabenverteilung teilweise sehr unausgeglichen war, sehe ich das Projekt als Erfolg an.

Auch technisch konnte ich viele neue Erfahrungen im Projektverlauf sammeln. Durch Angular konnte ich mich in meiner Rolle mit einer neuen Technologie beschäftigen. Auch wenn das Erlernen einer für mich neuen Programmiersprache viel Zeit in Anspruch genommen hat, habe ich es als eine gute Möglichkeit gesehen mich mit Webapplikationen tiefer zu beschäftigen und mich als Entwickler zu verbessern.

Besonders positiv möchte ich auch die Kommunikation innerhalb des Teams sowie mit den Kunden hervorheben. Auch wenn die geänderten Umstände aufgrund von Corona für anfängliche Schwierigkeiten sorgte, haben wir uns als Team schnell auf ein wöchentliches und mit dem Kunden auf einen mindestens zweiwöchentliches Meeting einigen können und kontinuierlich beibehalten. So konnten wir uns regelmäßig Feedback von den Kunden einholen, neue Ziele bis zum nächsten Treffen spezifizieren, uns über den Stand der Wetterstation erkundigen und auf Änderungen schneller reagieren. Auch die Einrichtung eines Discord-Servers für die Kommunikation zwischen einzelnen Teammitgliedern wurde gelegentlich genutzt und ermöglichte das gemeinsame Programmieren oder Fehlersuche im Code.

Auch die Arbeitsverteilung unter den Entwicklern und dem Architekt hat sehr gut funktioniert. Die Unterteilung in die zwei großen Bereiche Front-End und Back-End mit ihren jeweils zugeordneten Entwicklern hatte den Vorteil, dass sich die Entwickler wenig in die Quere kamen, Mergekonflikte minimiert und man sich anschließend lediglich um die Zusammenführung kümmern musste.

Im nächsten Projekt würde ich mehr Fokus darauf legen, dass der Kunde die vereinbarte Infrastruktur schnellstmöglich bereitstellt. Ebenso werde ich mehr darauf achten einen festen Featurestopp im Projekt zu definieren. So kann ein größeres Augenmerk auf Tests, Qualitätssicherung und Stabilität des Systems gelegt werden.

Abschließend lässt sich sagen, dass so ein großes Projekt im Verlauf von Software Engineering I

und II eine sinnvolle und wichtige Bereicherung war. Nicht nur um die erlernten Vorlesungsinhalte in einen praktischen Kontext zu setzen, sondern auch um die Kommunikation und Organisation eines Teams zu verbessern.

# 18. Reflexion Josefin Hähne

## 18.1. Teamrolle: Analyst / Architekt

Ich übernahm seit Anfang des Projekts in Software Engineering I die Rolle des Analysten und als Backup des Architekten. Zunächst viel es mir nicht ganz so leicht mich in die Rolle hineinzufinden, aber nach ein bisschen Zeit kam ich gut zurecht. Ich habe gelernt mit verschiedenen Tools, wie zum Beispiel mit GitHub, Visual Studio Code oder Visual Paradigm, umzugehen und zu arbeiten. Ich finde es sehr schön an einem realen Projekt mitgearbeitet zu haben, mit dem man anderen helfen kann und das auch eingesetzt wird, und so Erfahrungen zu sammeln.

Es war eine Herausforderung immer neue Anforderungen zu erhalten, diese zu analysieren und mit in das Projekt einzubinden, sowie auch eine komplette Konzeptänderung. Anfangs war es angedacht, dass der Raspi zyklisch (aller einer Stunde) aufwacht neue Daten postet und sich danach wieder schlafen legt. Im Laufe des Projekts änderte sich dies und der Raspi sollte nun die ganze Zeit aktiv bleiben. Dies lag auch daran, dass parallel zu unserer Software auch erst die Wetterstation auf dem Modellflugplatz errichtet wurde. Durch immer neue Anforderungen war es auch gut, dass durch die Pandemie der Zeitraum für die Projektbearbeitung verlängert wurde und wir so auch mehr Zeit hatten alle Wünsche der Kunden so gut es geht umzusetzen.

Auch wurde es in SE II schwieriger Termine für Teammeetings zu finden, bei denen alle anwesend sein können. Gut war, dass wir zwar einen vorgegebenen Slot im Stundenplan hatten, der für die Projektbearbeitung vorgesehen war, jedoch hat das nicht immer bei allen gepasst. Manche mussten zu der Zeit arbeiten. Bei mir war mein Kind zu Hause, um welches ich mich kümmern musste und das beschäftigt werden wollte. Somit erschwerte die Situation auch meine Arbeitsumgebung.

Dennoch hat die Kommunikation unter den Teammitgliedern gut funktioniert. Wir halfen uns gegenseitig bei Fragen oder Problemen und diskutierten, wenn nötig, darüber. Auch bezüglich der Auftraggeber funktionierte die Kommunikation gut, auch wenn sie manchmal etwas länger brauchten, um auf Emails zu antworten oder uns Informationen zukommen zu lassen, auf die wir warteten um in dem Projekt weiter zu kommen.

Beim nächsten Projekt würde ich schon eher beginnen mich genauer mit dem Thema auseinander zu setzen und einzuarbeiten. Des Weiteren nicht immer alles zu bejahen, sondern auch mehr zu hinterfragen, beispielsweise wenn es auf Projektende zugeht und von Seiten der Auftraggeber immer wieder Änderungen und neue Anforderungen gibt. Das erschwert das Projekt gut abzuschließen und auch vorher ausreichend zu testen.

Zusammenfassend war es aber eine schöne Erfahrung bei diesem Projekt mitgewirkt zu haben, ich habe viel dadurch gelernt und konnte daraus auch viel mitnehmen, nicht nur in Bezug auf Software Engineering, sondern auch in einem solchen Softwareteam zu arbeiten. Man nimmt durch die praktische Anwendung mehr in das Berufsleben mit als hätte man nur die Techniken theoretisch gelehrt bekommen.