SE I - Belegabgabe

T15 - Fernabfrage von Kamerabild und Wetterdaten

Martin Großmann, Josefin Hähne, Philipp Barth, Justin Schirdewahn, Alexander Schoch, Agustin Calvimontes, Clemens Kujus

31. Januar 2020

Inhaltsverzeichnis

Technische Spezifikation . . . 1
1. Vision T15 - Fernabfrage von Kamerabild und Wetterdaten . . . 2
1.1. Einführung . . . 2
1.2. Positionierung . . . 2
1.3. Stakeholder Beschreibungen . . . 3
1.4. Produkt-/Lösungsüberblick . . . 4
1.5. Zusätzliche Produktanforderungen . . . 5
2. Use-Case Model T15 - Fernabfrage von Kamerabild und Wetterdaten . . . 6
2.1. Use-Case: Bildergalerie abrufen . . . 6
2.2. Use-Case: Webdiagramm . . . 9
2.3. Use-Case: Konfigurationstabelle . . . 10
2.4. Use-Case: Read Logfiles . . . 13
2.5. Use-Case: Flag entfernen . . . 14
2.6. Use-Case: Flag setzen . . . 15
2.7. Use-Case: Wetterdaten Posten . . . 17
3. Wetterstation System-Wide Requirements Specification . . . 19
3.1. Einführung . . . 19
3.2. Systemweite funktionale Anforderungen . . . 19
3.3. Qualitätsanforderungen für das Gesamtsystem . . . 19
3.4. Zusätzliche Anforderungen . . . 20
4. Glossar -Wetterstation- . . . 21
4.1. Einführung . . . 21
4.2. Begriffe . . . 21
4.3. Abkürzungen und Akronyme . . . 21
4.4. Verzeichnis der Datenstrukturen . . . 22
Projektdokumentation . . . 24
5. Projektplan Wetterstation . . . 25
5.1. Einführung . . . 25
5.2. Das Team . . . 25
5.3. Kommunikationswege . . . 25
5.4. Projektablauf . . . 25
5.5. Meilensteine und Ziele . . . 25
5.6. Fortschritte nach Sim4Seed . . . 27
5.7. Eingesetzte Tools/Techniken . . . 28
5.8. Risk List . . . 28
6. Risk List . . . 29
6.1. Klasse 1 Projektgefährdende Risiken 29
6.2. Klasse 2 Kostenintensive Risiken 29

6.3. Klasse 3 Moderate Risiken 29
6.4. Gegenmaßnahmen 29
7. Iterationsplan 12.12.2019 - 26.12.2019 30
7.1. Risiken 30
7.2. Bewertung 30
8. Iterationsplan 26.12.2019 - 09.01.2020 31
8.1. Work Items List 31
8.2. Risiken 31
8.3. Bewertung 31
Entwurfsdokumentation 32
9. Architecture Notebook "Wetterstation MFCR" 33
9.1. Zweck 33
9.2. Architekturziele und Philosophie 33
9.3. Annahmen und Abhängigkeiten 34
9.4. Architektur-relevante Anforderungen 35
9.5. Entscheidungen, Nebenbedingungen und Begründungen 35
9.6. Architekturmechanismen 35
9.7. Wesentliche Abstraktionen 36
9.8. Schichten oder Architektur-Framework 36
9.9. Architektursichten (Views) 36
10. Design Documentation 37
10.1. Entity-Relationship-Modell 37

Technische Spezifikation

- Vision
- Use Case Model (inkl. Wireframes, sofern vorhanden)
- System-wide Requirements
- Glossar
- Domänenmodell

1. Vision T15 - Fernabfrage von Kamerabild und Wetterdaten

1.1. Einführung

Der Zweck dieses Dokuments ist es, die wesentlichen Bedarfe und Funktionalitäten des Wetterstation-Projekts zu sammeln, zu analysieren und zu definieren. Der Fokus liegt auf den Fähigkeiten, die von Stakeholdern und adressierten Nutzern benötigt werden, und der Begründung dieser Bedarfe. Die Details, wie das Wetterstation-Projekt diese Bedarfe erfüllt, werden in der Use-Case und Supplementary Specification beschrieben.

1.1.1. Zweck

Der Zweck dieses Dokuments ist es, die wesentlichen Anforderungen an das System aus Sicht und mit den Begriffen der künftigen Anwender zu beschreiben.

1.1.2. Gültigkeitsbereich (Scope)

Dieses Visions-Dokument bezieht sich auf das Projekt "Wetterstation", das von Team "Wetter" entwickelt wird. Das System wird es erlauben, eingelesene Wetterdaten zu sammeln, lokal zu speichern und an den Webserver des mfcR zu senden. Dort können die Daten dann von jedem abgerufen werden.

1.1.3. Definitionen, Akronyme und Abkürzungen

siehe Glossary Definitionen
Glossary Abkürzungen und Akronyme

1.1.4. Referenzen

Architecture Notebook
System-Wide Requirements Specification

1.2. Positionierung

1.2.1. Fachliche Motivation

Der mfcR wünscht sich eine vereinseigene Wetterstation mit einer Webcam, wobei diese Daten übermittelt und auf der Vereinswebseite angezeigt werden sollen. Zweck des Systems ist die Daten für alle Nutzer des Modellflugplatzes bzw. Vereinsmitglieder bereitzustellen. Nutzer können somit nachvollziehen wie die aktuelle Wetterlage ist und wer sich auf dem Flugplatz befindet (ggf. zum Schutz des Eigentums).

1.2.2. Problem Statement

Tabelle 1. Problem

Das Problem	Dem mfcR stehen keine aktuellen und örtlich genauen Wetterdaten + Bild(er) zur Verfügung.
betrifft	die Nutzer des Modellflugplatzes des mfcR
die Auswirkung davon ist	Schlechte Einschätzung der Wetterlage auf dem Flugplatz und daraus resultierende Ungewissheit ob sich ein Besuch des Flugplatzes lohnt.
eine erfolgreiche Lösung wäre	Nutzer können schnell aktuelle Wetterdaten des Platzes abrufen, sowie sich ein Bild von der Lage machen.

1.2.3. Positionierung des Produkts

Tabelle 2. Endprodukt

Für	Nutzer der Webseite/ des Modellflugplatzes
die	die aktuellen Wetterdaten abfragen wollen
Das Produkt / die Lösung ist eine	Webanwendung auf der Vereinswebseite
Die	die aktuellen (Wetter)Verhältnisse auf dem Flugplatz zeigen
Im Gegensatz zur	jetzigen Webseite
Unser Produkt	zeigt aktuelle und örtliche genaue Informationen an.

1.3. Stakeholder Beschreibungen

1.3.1. Zusammenfassung der Stakeholder

Tabelle 3. Stakeholder

Name	Beschreibung	Verantwortlichkeiten
Thomas Brenner	Vorsitzender des Vereins MODELLFLUGCLUB ROSSENDORF e.V.	Ansprechpartner, überwacht den Projektfortschritt
Admins	Berechtigte für den internen Bereich	Informationen zum Systemzustand abfragen, Raspberry Pi in Wartungsmodus versetzen und verbinden
User	Besucher der Webseite	aktuelle Informationen und gespeicherte Bilder abrufen
Gesetzgeber	geltende Gesetze und Richtlinien der BRD	Schutz personenbezogener Daten

Name	Beschreibung	Verantwortlichkeiten
jweiland.net	Hosting-Dienstleister des mfcR	Bereitstellung der physischen webserverseitigen Infrastruktur

1.3.2. Benutzerumgebung

1. Beschreiben Sie die Arbeitsumgebung des Nutzers:
    - Der Nutzer muss auf einem internetfähigem Gerät online sein. Um neue Informationen abzurufen muss er die Seite nicht manuell aktualisieren. Bedienung ist auf unterschiedlichen Endgeräten möglich.
2. Gibt es besondere Umgebungsbedingungen, z.B. mobil, offline, Außeneinsatz, Touchbedienung, Nutzung durch seh- oder hörbeeinträchtigte Personen?
    - mobil abrufbar
    - Touchbedienung auf Smartphpones
3. Welche anderen Anwendungen sind im Einsatz? Muss ihre Anwendung mit diesen integriert werden?
    - TYPO3-CMS auf dem die Website beruht
        - Die Anwendung wird dort eingebunden

1.4. Produkt-/Lösungsüberblick

1.4.1. Bedarfe und Hauptfunktionen

Tabelle 4. Bedarfe

Bedarf	Priorität	Features	Geplantes Release
Bilder abrufen	mittel	stündlich aktualisierte Bildergalerie mit Auswahl nach Zeitstempel, Vollbildmöglichkeit	xx
Diagramme abrufen	hoch	Temperatur, Wind (Stärke+Richtung), Böenhaftigkeit, Feuchtigkeit nach Zeitstempel abrufbar	xx
Daten speichern	hoch	in Datenbank und lokal	xx
interner Bereich	hoch	Raspberry Pi in Wartungsmodus versetzen, Informationen über Systemzustand	xx

Bedarf	Priorität	Features	Geplantes Release
Remote-Zugriff	hoch	sofern der Raspi im Wartungsmodus ist, kann über ssh eine Remote-Verbindung hergestellt werden	xx

1.5. Zusätzliche Produktanforderungen

äußere Faktoren:

- keine Infrastruktur vorhanden
  - somit kein Strom am Modellflugplatz
- System wird nach Fertigstellung im Boden eingegraben → keine leichte Wartung möglich → Remote-Zugriff + hohe Stabilität wichtig

eingesetzte Hardware:

- Raspberry Pi → Raspi 4 (sollte auch auf Raspi Zero laufen)
- diversere Sensoren (via I2C/SPI)
- Webcam (via Raspi on-board Camera Connector)
- UMTS-Modul (via USB)
- Akku (LiPo)
- Solarzelle
- Lade-Management
- ggf. externer Wake-Up-Timer
- Gehäuse

Software (Raspi/Webserver)

- Verwendung einer Skriptsprache (aktuell Python)
- Backend vorzugsweise php
- Zugang zu Webserver wird gestellt (10 GB)
- Zugang zu einer mySQL-DB wird gestellt
- Einbindung auf Webseite (TYPO3-CMS)

Tabelle 5. Anforderungen

Anforderung	Priorität	Geplantes Release
Verwendung einer Skriptsprache (aktuell Python)	mittel	xx
Backend vorzugsweise mittels PHP	mittel	xx

2. Use-Case Model T15 - Fernabfrage von Kamerabild und Wetterdaten

2.1. Use-Case: Bildergalerie abrufen

2.1.1. Kurzbeschreibung

In diesem Use-Case wird die Anzeige und das Bearbeiten der aufgenommenen Bilder behandelt.

2.1.2. Kurzbeschreibung der Akteure

User

User können sich die Bilder anschauen, welche in einer Galerie auf der Website des MFCR zur Verfügung gestellt werden.

Admins

Admins können sich diese Bilder ebenfalls anzeigen lassen, haben aber erweiterte Rechte, um Bilder zu bearbeiten, zu kopieren, zu verschieben oder zu löschen.

2.1.3. Vorbedingungen

1. Die Kamera hat bereits Bilder aufgenommen
2. Kamera wird korrekt vom Rasperry Pi angesteuert wenn sich dieser aktiviert/gestartet wird.
3. Die Bilder werden korrekt auf den Server gepusht und dazugehörige Meta-Daten in der DB gespeichert
4. Die Bilder werden vom Server richtig zum Client übertragen
5. Die Bilder werden auf der Website korrekt angezeigt

2.1.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn der User/Admin die Website mit der Bildergalerie aufruft
2. Die Bilder werden in aufgenommener Reihenfolge als Galerie angezeigt, das aktuellste Bild wird größer dargestellt
3. Beim Klicken auf ein anderes Bild wird die Auswahl an die Bildschirmgröße des Nutzers angepasst dargestellt
4. Die Bilder laden nach unten weiter, wenn der User weiter nach unten scrollt
5. Der User/Admin kann einen Zeitraum angeben, in dem die aufgezeichneten Bilder angezeigt werden sollen
6. Der Use Case ist abgeschlossen.

2.1.5. Alternative Abläufe

Bildfehler

Wenn der User/Admin im Schritt 1 des Standardablauf die Website aufruft, dann kann es sein, dass ein oder mehrere Bilder nicht richtig verarbeitet wurden . Der Admin loggt sich ein und kann die entsprechenden Bilder bearbeiten oder löschen . Der Use Case wird im Schritt 1 des Standartablaufes fortgesetzt.

Zeitraumfehler

Wenn der User/Admin im Schritt 4 des Standardablauf einen ungültigen Zeitraum angibt, dann wird er gebeten den Zeitraum zu ändern . Der User/Admin gibt einen Zeitraum an, in dem keine Bilder aufgezeichnet wurden . Der Server greift auf die Datenbank zu und stellt fest, dass es keine Bilder im angegebenen Zeitraum gibt . Dem User/Admin wird die Meldung angezeigt, dass er den Zeitraum ändern soll und wann das erste und das letzte Bild aufgezeichnet wurden . Der Use Case wird im Schritt 4 Standardablaufes fortgesetzt.

2.1.6. Unterabläufe (subflows)

Zeiträume

1. Der User/Admin kann einen Zeitraum angeben, um nach Bildern in diesem zu filtern
2. Der Server greift auf die Datenbank zu
3. Die Website generiert die Galerie dem entsprechenden Zeitraum nach neu
4. Der Subflow ist abgeschlossen

2.1.7. Wesentliche Szenarios

Bilder bearbeiten/löschen/kopieren

Wenn der Admin sich eingeloggt hat, hat er die Möglichkeit in Schritt 1 die Bilder von der Website zu löschen oder diese herunterzuaden, um sie lokal zu bearbeiten oder zu sichern.

2.1.8. Nachbedingungen

Galerieanzeige

Die Website hat die Galerie entsprechend der User/Admin-Anforderungen geladen und wartet auf neue Änderungen (weitere Bilder laden, Zeitraumänderung, Bilder löschen), um die Galerie dementsprechend neu zu generieren

2.1.9. Besondere Anforderungen

Qualitätsanforderung

Sollte ein Admin ein Bild löschen, muss die Website die Galerie ohne dieses Bild weiterhin ordnungsgemäß und lückenlos anzeigen können.

2.1.10. Wireframes

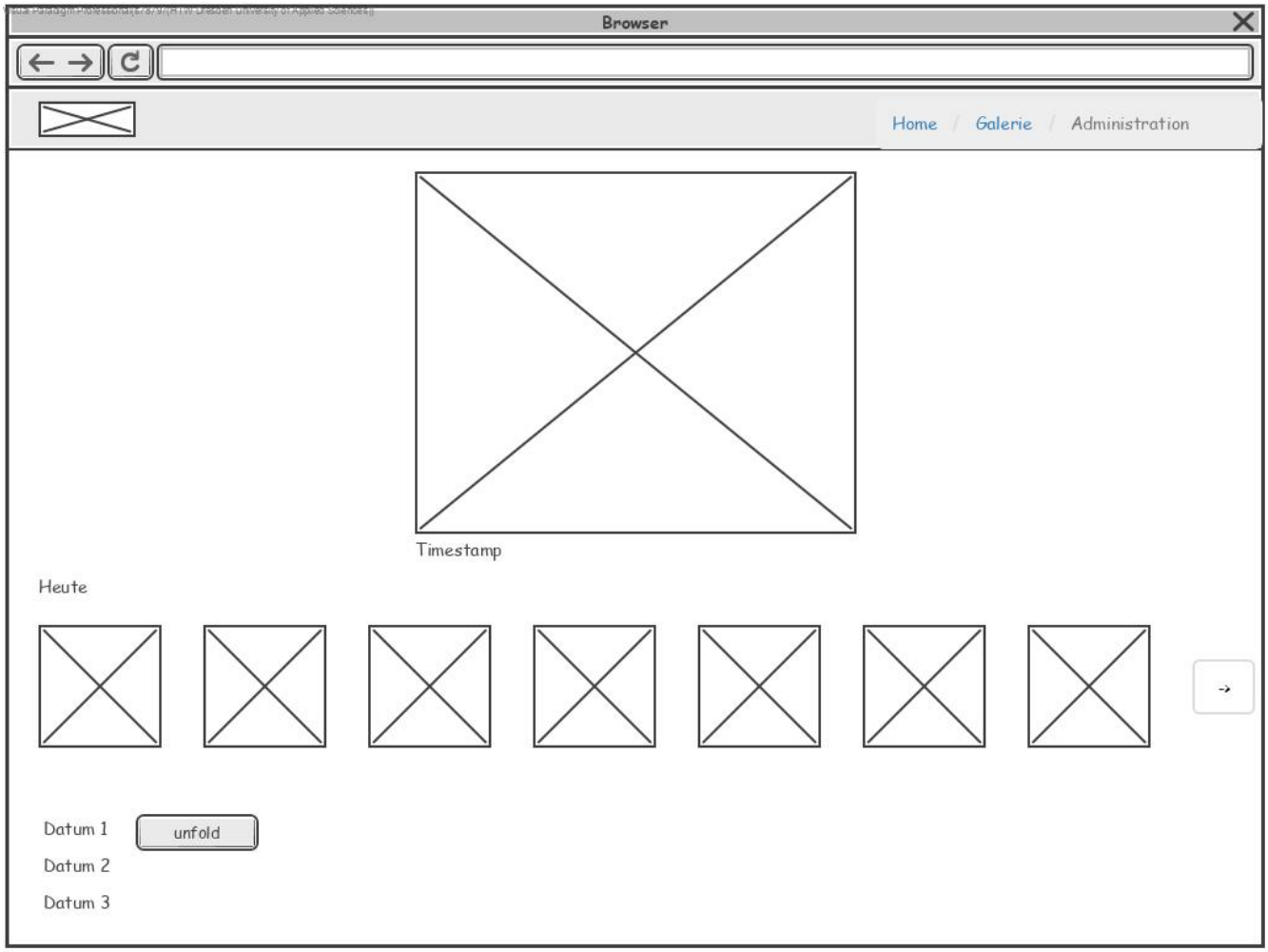

Abbildung 1. Galeriemodi

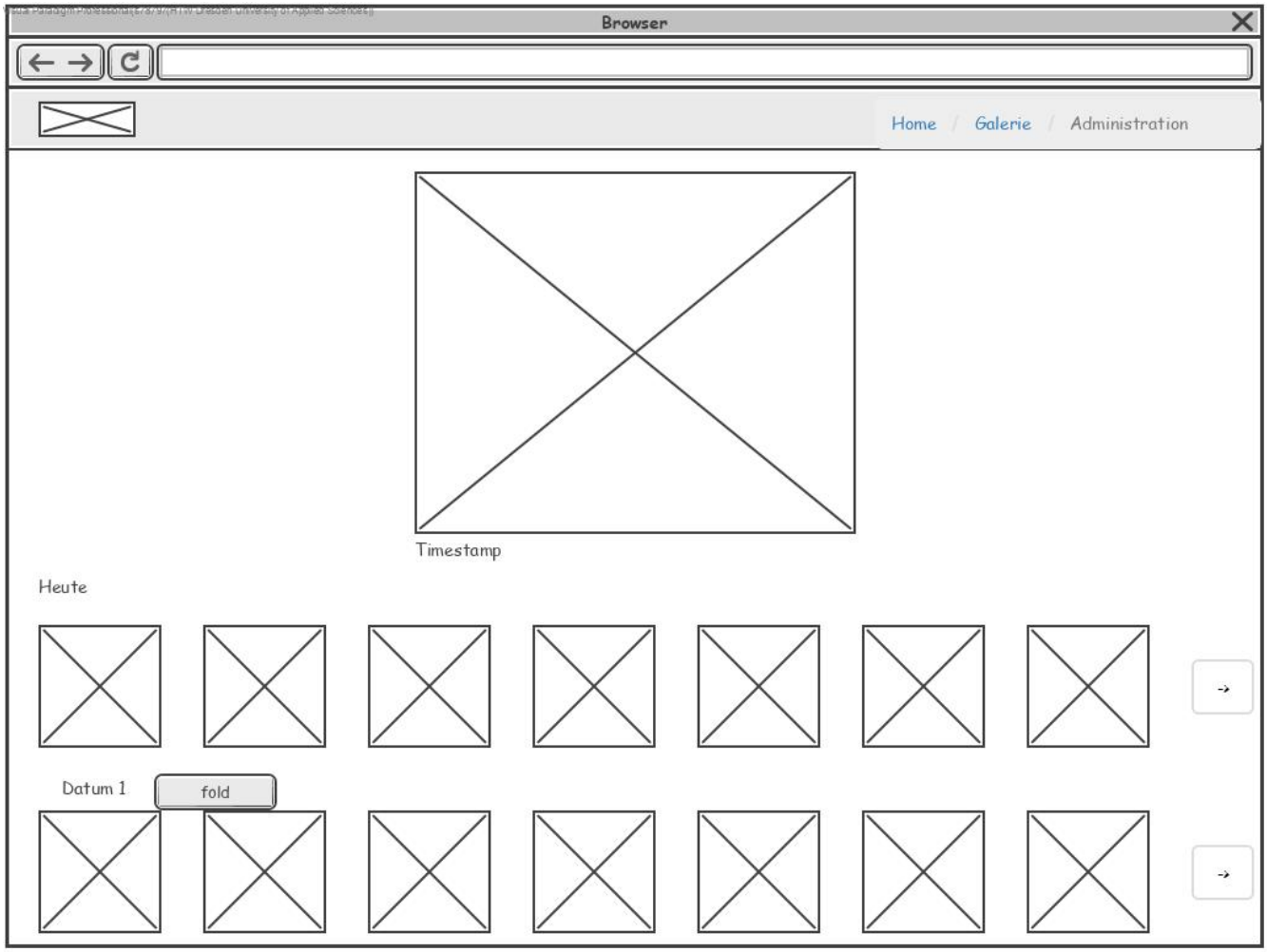

Abbildung 2. Galeriemodi

2.2. Use-Case: Webdiagramm

2.2.1. Kurzbeschreibung

Es geht um die Anzeige der gesammelten Daten, die Funktionen, die durch die Anzeige gewährleistet werden sollen und mögliche Schwierigkeiten.

2.2.2. Kurzbeschreibung der Akteure

User, Admins

2.2.3. Vorbedingungen

- Daten werden gesammelt
- Daten werden in Datenbank ordnungsgemäß gespeichert
- Daten werden vom Server richtig verarbeitet

2.2.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn ein User oder ein Admin die gesammelten Daten aufrufen und auswerten möchte

2. Der User/Admin ruft die Seite auf, in der die Daten angezeigt werden sollen
3. Die gesammelten Daten werden als Diagramm angezeigt
4. Der User/Admin wählt die Datenart aus, die er sehen möchte
5. Der User/Admin wählt den Zeitraum aus in dem die gesammelten Daten angezeigt werden sollen
6. Der Server greift auf die Datenbank, in der die Daten gespeichert werden, zu und nutzt die Daten, die angegeben wurden
7. Dem User/Admin wird ein Diagramm aus den gewählten Daten angezeigt
8. Der Use Case ist abgeschlossen.

2.2.5. Alternative Abläufe

Zeitraum ohne Daten

Wenn der User/Admin im Schritt 7 des Standardablauf die Datenart oder den Zeitraum ändert, dann wird das Diagramm neu generiert . Der User/Admin wählt einen Zeitraum, in dem keine Daten aufgezeichnet wurden . Der Server zeigt dem User/Admin, dass es zu diesem Zeitraum keine Daten gibt . Der Use Case wird im Schritt 5 fortgesetzt.

2.2.6. Unterabläufe (subflows)

Datenart Änderung

1. Der User/Admin wählt eine andere Datenart
2. Der Server greift erneut auf die Datenbank zu, um die erforderlichen Daten zu erhalten
3. Der Subflow wird in Schritt 5 des Usecase fortgesetzt

Zeitraum Änderung

1. Der User/Admin wählt einen anderen Zeitraum für die momentan ausgewählte Datenart
2. Der Server greift erneut auf die Datenbank zu, um die erforderlichen Daten zu erhalten
3. Der Usecase wird in Schritt 6 des Usecase fortgesetzt

2.2.7. Nachbedingungen

Diagramm

Der Server zeigt dem User/Admin ein Diagramm, was auf den von ihm/ihr ausgewählten Daten und dem ausgewählten Zeitraum generiert wurde, um eine schnelle und einfache Auswertung der aufgezeichneten Daten zu ermöglichen.

2.3. Use-Case: Konfigurationstabelle

2.3.1. Kurzbeschreibung

Administratoren wird es ermöglicht bestimmte Werte, welche in einer Tabelle der Datenbank gespeichert sind, zu ändern. Diese Tabelle wird nach jedem Neustart des Raspis ausgelesen und die Werte werden übernommen.

2.3.2. Kurzbeschreibung der Akteure

Administratoren

Nur Administratoren, sprich User mit einem gültigen Login und entsprechenden Zugriffsrechten auf erweiterte Funktionen, haben Zugriff auf die Funktionalitäten, die in diesem Use-Case definiert sind

2.3.3. Vorbedingungen

1. Verbindung zwischen Website und Datenbankserver wird erfolgreich hergestellt
2. Datenbank ist aktiv
3. Verbindung zwischen Raspi und Datenbankserver wird erfolgreich hergestellt

2.3.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn der Administrator auf den Adminbereich zugreift
2. Der Administrator sieht den aktuellen Inhalt der Config-Tabelle
3. Der Administrator kann die Werte über die Website ändern
4. Die Website übermittelt die neuen Werte an den Datenbankserver
5. Der Datenbankserver speichert die neuen Werte in der Config-Tabelle
6. Der Raspi baut nach jedem Neustart eine Verbindung zum Datenbankserver her
7. Der Raspi fragt den Inhalt der Config-Tabelle nach jedem Neustart ab
8. Der Raspi übernimmt die Werte aus der Config-Tabelle
9. Der Use Case ist abgeschlossen.

2.3.5. Alternative Abläufe

Verbindungsfehler

Wenn die Website im Schritt 4 des Standardablauf keine Verbindung zum Datenbankserver aufbauen kann, dann wird dem Admin eine Fehlermeldung ausgegeben . Bei Verbindungsfehler wird Fehlermeldung im Adminbereich angezeigt . Der Use Case wird im Schritt 9 fortgesetzt.

Unzulässige Werte

Wenn der Admin im Schritt 3 des Standardablaufes unzulässige Werte in die Config-Tabelle eintragen möchte wird eine Warnung ausgegeben . Bei Änderungen der Werte in unzulässige Werte wird eine Warnmeldung ausgegeben . Der Use Case wird im Schritt 3 fortgesetzt.

2.3.6. Wesentliche Szenarios

Änderung der Wartungs-Flag

1. Der Admin ändert den Wert der Wartungs-Flag
2. Der neue Wert wird in der Config-Tabelle gespeichert
3. Der Raspi fragt den Inhalt der Config-Tabelle beim nächsten Neustart ab
4. Der Raspi übernimmt die Änderung des Wartungs-Flag-Wertes
5. Der Raspi fährt nicht automatisch herunter, sondern befindet sich nun im Wartungsmodus

2.3.7. Nachbedingungen

Änderungen des Tabelleninhaltes

1. Der Raspi übernimmt die Änderungen und reagiert dementsprechend auf diese

2.3.8. Besondere Anforderungen

Einhalten der Datentypen

1. Jede Zeile der Tabelle Config kann nur bestimmte Werte speichern
2. Der Admin darf die Werte nur so ändern, dass diese Ordnungsgemäß gespeichert und verarbeitet werden können

2.3.9. Wireframes

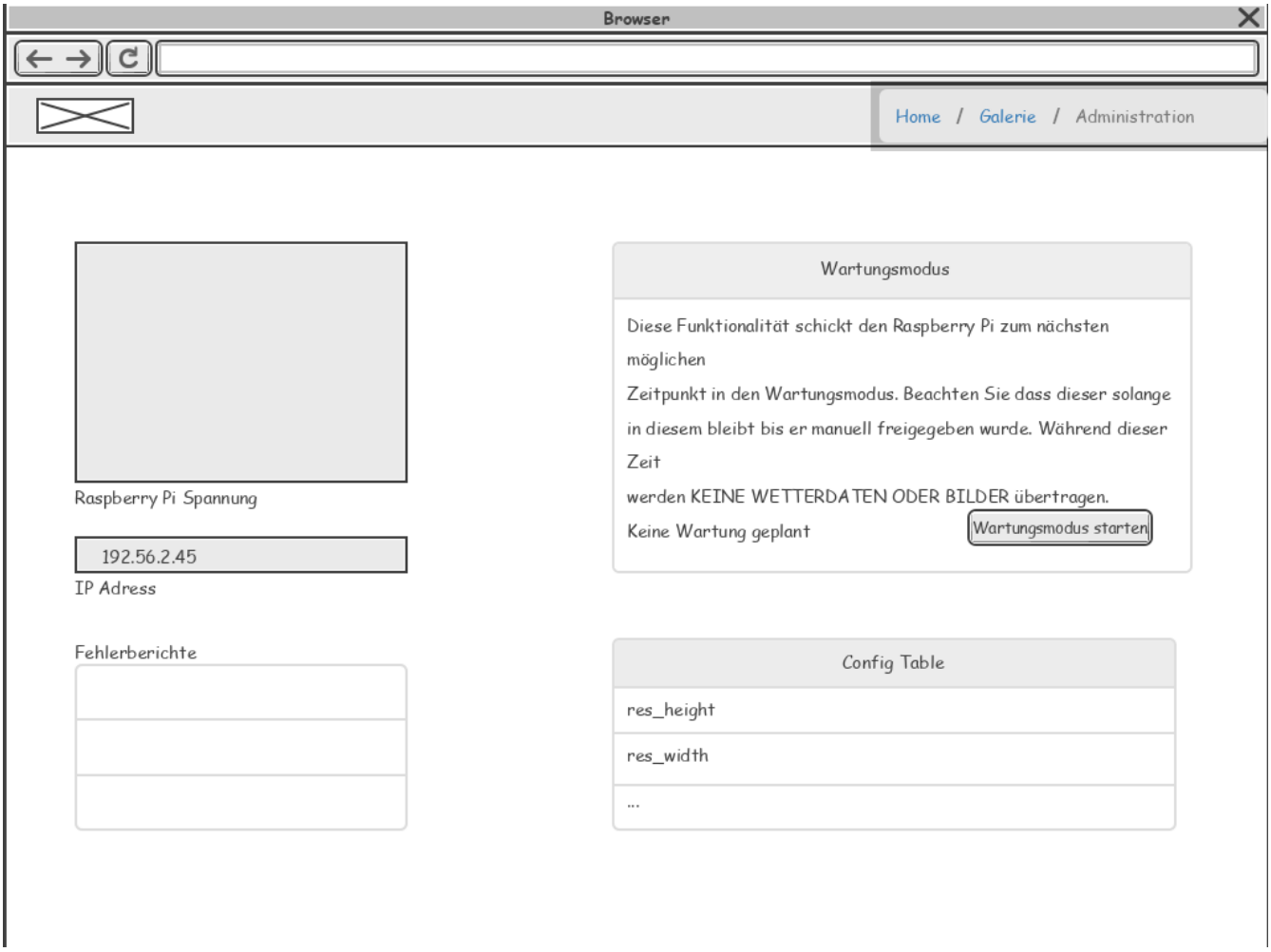

Abbildung 3. Admin Panel

2.4. Use-Case: Read Logfiles

2.4.1. Kurzbeschreibung

In diesem Usecase wird die Auswertung der Daten, welche in den Logfiles gespeichert werden, beschrieben.

2.4.2. Kurzbeschreibung der Akteure

Administratoren

Nur Administratoren, sprich User mit einem gültigen Login und entsprechenden Zugriffsrechten auf erweiterte Funktionen, haben Zugriff auf die Funktionalitäten, die in diesem Use-Case definiert sind

2.4.3. Vorbedingungen

1. Speichern bestimmter Daten in einer Logfile
2. Speichern der Logfile in der Datenbank
3. Verbindung zwischen Website und Datenbank wird ordnungsgemäß hergestellt

2.4.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn der Admin auf den Adminbereich zugreift
2. Der Inhalt der Logfile wird dem Admin im Adminbereich angezeigt
3. Der Use Case ist abgeschlossen.

2.4.5. Alternative Abläufe

2.4.6. Unterabläufe (subflows)

Überschreiben der Logfile

1. Die Logfile wird in der Datenbank gespeichert
2. Beim erneuten Start des Raspi wird eine neue Logfile an die Datenbank übermittelt
3. Die neue Logfile überschreibt die schon gespeicherte Logfile in der Datenbank

2.5. Use-Case: Flag entfernen

2.5.1. Kurzbeschreibung

Der Adminbereich beinhaltet eine Möglichkeit die Wetterstation bzw. den Raspberry Pi in einen sogenannten Wartungsmodus zu versetzen. Dies soll über einen Eintrag(Flag), der in die Datenbank geschrieben wird, realisiert werden. Der Eintrag wird beim Datenaustausch zwischen dem Pi und dem Webserver abgefragt und weiterverarbeitet. Dieser Use-Case beschreibt den Vorgang des Entfernens einer bereits gesetzten Flag bzw. einer geplanten Wartung

2.5.2. Kurzbeschreibung der Akteure

Administratoren

Nur Administratoren, sprich User mit einem gültigen Login und entsprechenden Zugriffsrechten auf erweiterte Funktionen, haben Zugriff auf die Funktionalitäten, die in diesem Use-Case definiert sind

2.5.3. Vorbedingungen

1. Admin greift auf AB zu
2. Admin ist eingeloggt
3. Ein Eintrag für eine geplante Wartung ist in der Datenbank und aktiv

2.5.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn der Admin die Funktionalität "Abbrechen" wählt.
2. Admin wird gefragt ob er die geplante Wartung abbrechen will und erhält Auswahlmaske mit "Ja" und "Nein"

3. Admin wählt ja
4. System entfernt

2.5.5. Alternative Abläufe

Admin bricht Vorgehen ab

Wenn der Admin im Schritt 3 des Standardablauf "Abbrechen" wählt, dann

- wird der Vorgang abgebrochen
- Der Use Case wird im Schritt 7 fortgesetzt.

2.5.6. Unterabläufe (subflows)

Für diesen Use-Case sind keine Unterabläufe definiert

2.5.7. Wesentliche Szenarios

2.5.8. Besondere Anforderungen

Raspberry Pi wird nicht automatisch heruntergefahren

1. Der Raspberry Pi schaltet sich nicht von selbst wieder aus bzw. kehrt nicht in seinen Ruhezustand zurück, damit die Wartungen ordnungsgemäß durchgeführt werden können

2.6. Use-Case: Flag setzen

2.6.1. Kurzbeschreibung

Der Adminbereich beinhaltet eine Möglichkeit die Wetterstation bzw. den Raspberry Pi in einen sogenannten Wartungsmodus zu versetzen. Dies soll über einen Eintrag/Flag, der in die Datenbank geschrieben wird, realisiert werden. Der Eintrag wird beim Datenaustausch zwischen dem Pi und dem Webserver abgefragt und weiterverarbeitet. Dieser Use-Case beschreibt den Vorgang des Setzens einer Flag.

2.6.2. Kurzbeschreibung der Akteure

Administratoren

Nur Administratoren, sprich User mit einem gültigen Login und entsprechenden Zugriffsrechten auf erweiterte Funktionen, haben Zugriff auf die Funktionalitäten, die in diesem Use-Case definiert sind

2.6.3. Vorbedingungen

1. Admin greift auf AB zu
2. Admin ist eingeloggt

2.6.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn der Admin die Funktionalität "Raspberry Pi in den Wartungsbereich versetzen" wählt.
2. Admin erhält Maske für die Eingabe eines Zeitpunkts, wann der Wartungsmodus geplant ist (Standard: nächste volle Stunde), Möglichkeit zum Bestätigen und Abbrechen.
3. Website schreibt Flag in die Datenbank.
4. Dem Admin wird nun ein Timer bis zum nächsten geplanten Wartungsmodus angezeigt und die Möglichkeit gegeben die Flag wieder zu entfernen und den Vorgang abzubrechen.
5. Anschließend wird dem Admin wieder die Übersicht des Adminbereichs angezeigt.

2.6.5. Alternative Abläufe

Admin bricht Vorgehen ab

Wenn der Admin im Schritt 3 des Standardablauf "Abbrechen" wählt, dann

- wird der Vorgang abgebrochen
- Der Use Case wird im Schritt 7 fortgesetzt.

2.6.6. Unterabläufe (subflows)

Für diesen Use-Case sind keine Unterabläufe definiert

2.6.7. Besondere Anforderungen

Raspberry Pi wird nicht automatisch heruntergefahren

1. Der Raspberry Pi schaltet sich nicht von selbst wieder aus bzw. kehrt nicht in seinen Ruhezustand zurück, damit die Wartungen ordnungsgemäß durchgeführt werden können

2.6.8. Aktivitätsdiagramm

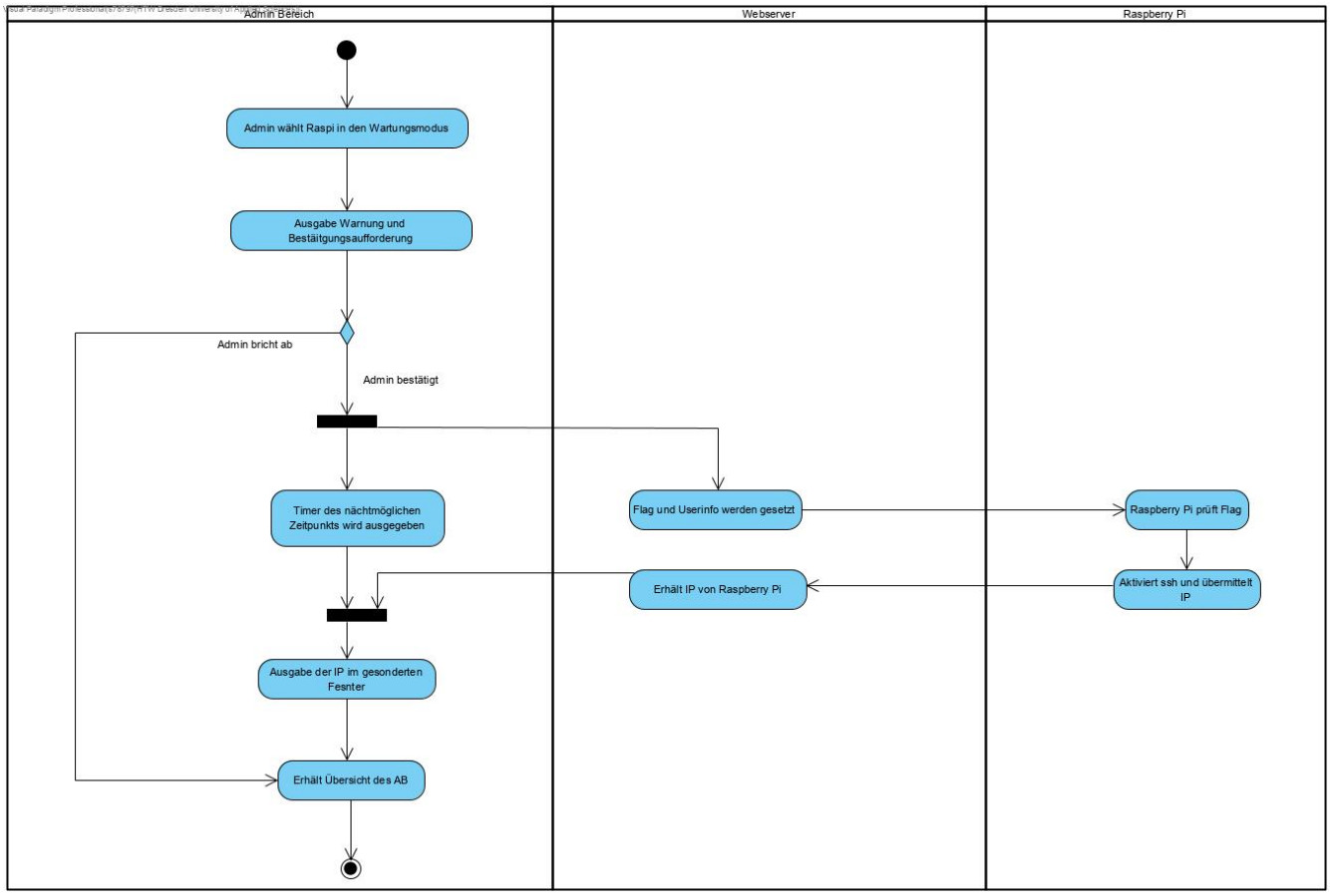

Abbildung 4. Aktivitätsdiagramm

2.7. Use-Case: Wetterdaten Posten

2.7.1. Kurzbeschreibung

Dieser Use Case beschreibt die in regelmäßige Übertragung der Wetterdaten an den Datenbank-Server.

2.7.2. Kurzbeschreibung der Akteure

RaspberryPi

- er sammelt die Wetterdaten und postet sie an den DB-Server

Datenbank-Server

- er empfängt die Daten und schreibt diese in die Datenbank

2.7.3. Vorbedingungen

- Raspi ist mit dem Internet verbunden
- Raspi wurde aus dem sleep modus geweckt
- DB-Server ist online

2.7.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn der Raspi aus dem sleep modus erwacht und die aktuellen Wetterdaten ermittelt hat
2. Der Raspi ruft die einzelnen Wetterdaten aus dem Dateisystem ab
3. Der Raspi baut eine Verbindung zum Server auf
4. Der Raspi überträgt die Daten an den Server
5. Der Use Case ist abgeschlossen.

2.7.5. Alternative Abläufe

Verbindung vom Raspi und DB-Server bricht kurzzeitig ab

Wenn Raspi im Schritt 3 des Standardablauf keine Verbindung aufbauen kann, dann

1. wird einige Sekunden gewartet und der Verbindungsaufbau erneut angestrebt
2. Der Use Case wird im Schritt 4 fortgesetzt.

Verbindung vom Raspi und DB-Server bricht längerfristig ab

Wenn Raspi im Schritt 3 des Standardablauf keine Verbindung aufbauen kann und auch nach erneutem Versuch keine Verbindung hergestellt werden kann, dann

1. speichert der Raspi die Wetterdaten lokal ab
2. schreibt er in die logfile, dass keine Daten übertragen wurden
3. überträgt diese logfile beim nächsten erfolgreichen Post

3. Wetterstation System-Wide Requirements Specification

3.1. Einführung

In diesem Dokument werden die systemweiten Anforderungen für das Projekt **Wetterstation** spezifiziert. Die Gliederung erfolgt nach der FURPS+ Anforderungsklassifikation:

- Systemweite funktionale Anforderungen (F),
- Qualitätsanforderungen für Benutzbarkeit, Zuverlässigkeit, Effizienz und Wartbarkeit (URPS) sowie
- zusätzliche Anforderungen (+) für technische, rechtliche, organisatorische Randbedingungen

Die funktionalen Anforderungen, die sich aus der Interaktion von Nutzern mit dem System ergeben, sind als Use Cases in einem separaten Dokument festgehalten. [siehe [usecase_model.adoc]]

3.2. Systemweite funktionale Anforderungen

- Übertragung der Daten geschieht in gleichmäßigen Abständen (einmal pro Stunde)
- Admins haben Berechtigung um sich im internen Bereich anzumelden

3.3. Qualitätsanforderungen für das Gesamtsystem

3.3.1. Benutzbarkeit (Usability)

- Der Nutzer kann sich in wenigen Schritten (max 5 Klicks) ein Diagramm anzeigen lassen und gelangt mit maximal 2 Klicks zur Bildergalerie
- Der Nutzer benötigt keine weiteren Vorkenntnisse um das System zu benutzen (auch für Gelegenheitsnutzer zu verstehen)
- Fehlertexte werden verständlich angezeigt
- Die Sprache für die Benutzeroberfläche ist deutsch

3.3.2. Zuverlässigkeit (Reliability)

- Die Wetterdaten werden genau angezeigt
- Das System soll eine relativ hohe Verfügbarkeit von 99% haben
- Wenn ein Fehler auftritt werden die Daten nach der Fehlerbehebung wieder aus der Datenbank gezogen

3.3.3. Effizienz (Performance)

- Die Bilder bzw. Diagramme werden innerhalb von 5 Sekunden angezeigt
- Es werden mindestens einmal pro Stunde neue Werte an den Server geschickt
- Es können mindestens 2 Nutzer parallel mit dem System arbeiten
- Das System muss innerhalb von 10 Sekunden starten und herunterfahren

3.3.4. Wartbarkeit (Supportability)

- Das System muss kompatibel mit der Webseite des mfcR sein
- Das System muss auch auf mobilen Oberflächen laufen (z.B. Smartphones)
- Anwendung von gängigen Design-Patterns zur besseren Erweiterbarkeit

3.4. Zusätzliche Anforderungen

3.4.1. Einschränkungen

- zu nutzende Komponenten: mySQL-Datenbank, Angular
- Vorgaben für die Programmiersprache auf Raspi: Python
- Hardwareeinschränkungen durch Raspi

3.4.2. Organisatorische Randbedingungen

- Logo des mfcR soll auf der Oberfläche sichtbar sein

3.4.3. Rechtliche Anforderungen

- Datenschutz (Bilder)

4. Glossar -Wetterstation-

4.1. Einführung

In diesem Dokument werden die wesentlichen Begriffe aus dem Anwendungsgebiet (Fachdomäne) der **Wetterstation** definiert. Zur besseren Übersichtlichkeit sind Begriffe, Abkürzungen und Datendefinitionen gesondert aufgeführt.

4.2. Begriffe

Tabelle 6. Begriffe

Begriff	Definition und Erläuterung	Synonyme
User	Mitglied oder Interessent des "MFCR", welcher das System zur Informationsabfrage nutzt	Benutzer, Endnutzer
Administrator	Mitglieder des "MFCR" mit besonderen Rechten. Sie sind in der Lage über einen Login Änderungen am System vorzunehmen (bspw. Wartungsmodus des Raspi zu aktivieren).	Admin
Wartungsmodus	Zustand der sicherstellt, dass der Raspi beim nächsten Einschalten nicht zurück in den Ruhenzustand wechselt sonder aktiv bleibt. Um dann über ssh erreichbar zu sein.	
Auftraggeber	Die Vertreter des "MFCR" (Heino Iwe, Thomas Brenner)	Kunden

4.3. Abkürzungen und Akronyme

Tabelle 7. Ablürzugnen und Akronyme

Abkürzung	Bedeutung	Erläuterung
AB	Admin-Bereich	dedizierter Bereich für die Administration der Wetterstation
MFCR	Modellflugclub Rossendorf e.V.	Modellflugverein, der das Projekt "Wetterstation" stellt

Abkürzung	Bedeutung	Erläuterung
SW	Software	Gegenstück zu Hardware, alle Programme und zugehörigen Daten des Systems
Raspi	Raspberry-Pi (gesamte Baureihe, aber wahrscheinlich Raspi 4)	Einplatinen-Computer mit Linux-Distribution zur Datensammlung und Übermittlung an den Webserver

4.4. Verzeichnis der Datenstrukturen

Tabelle 8. Datenstrukturen

Bezeichnung	Definition	Format	Gültigkeitsregeln	Aliase
Anmeldedaten	Zusammensetzung von Benutzername und Passwort.	String	Emailadresse muss @-Zeichen und Punkt enthalten.	Login
Wetterdaten	auf dem Raspi gesammelte Sensordaten zu Windstärke, Windrichtung, Temperatur	JSON	-	Sensordaten
Konfigurtationsparameter	Liste von Parametern die Einstellungen des Raspi spezifizieren (Kamera-Auflösung, Wartungsmodusflag, Wartungszeitraum, Aquiseintervall)	*spezifieziert im ERM Modell*	-	Parameter

Ein Domänenmodell wurde noch nicht erstellt. Dies wird es aber in den nächsten Iterationen.

Projektdokumentation

- Projektplan
- Risikoliste
- Iteration Plan (für zwei ausgewählte Iterationen)

5. Projektplan Wetterstation

5.1. Einführung

Dieses Dokument gibt einen Überblick über das Projekt Wetterstation. Es wird das Projektteam, eingesetzte Tools/Techniken, Risiken und Iterationen vorgestellt.

5.2. Das Team

- Martin Großmann : Tester, Projektmanager
- Josefin Hähne : Analyst, Architekt
- Philipp Barth : Architekt, Entwickler
- Justin Schirdewahn : Entwickler, Tester
- Alexander Schoch : Entwickler, Analyst
- Clemens Kujus : Projektmanager, Entwickler
- Agustin Calvimontes: Deployment Engineer

Nach der ersten Iteration wurden die Hauptrollen von Martin Großmann (ehemals Projektmanager) und Clemens Kujus (ehmals Tester) getauscht

5.3. Kommunikationswege

1. WhatsApp für schnelle Kommunikation
2. GitHub-Issues für Themenbearbeitung
3. E-Mail für Kontakt mit den Kunden/Coach
4. Regelmäßige Treffen im Team (nach Bedarf mit Kunde/Coach)
5. Kontakt im Alltag an der HTW

5.4. Projektablauf

Das Projekt ist in einer Zeitspanne von Dezember 2019 bis Juni 2020 gestreckt. Dabei gibt es zwei große Etappenziele. Im ersten Semester steht die Analyse und Projektplanung im Vordergrund, im zweiten Semester die Implementierung, wobei natürlich im Sinne der iterativen Entwicklung keine strikte Trennung der Aufgaben erfolgt.

Die Iterationslängen betragen 2 Wochen. Diese werden geplant, durchgeführt und ausgewertet, wobei nach Abschluss einer Iteration im Team auf deren Basis die nächste Iteration geplant wird.

5.5. Meilensteine und Ziele

Tabelle 9. Meilensteine und Ziele

Iteration	Primary Objectives	Scheduled start
I-2	- Anforderungen des Kunden anhand des Gesprächs genauer spezifieren - Stakeholder finden - Use Cases ableiten und grob beschreiben - Vision bearbeiten	12.12.2019
I-3	- Stakeholder finden - Use Cases ableiten und einen Teil davon detailliert ausarbeiten - Vision bearbeiten - Gloassar anlegen/bearbeiten - 2. Kundengespräch vorbereiten/durchführen	26.12.2019
I-4	- Eine geregelte Kommunikation der Afugabenteilung durchsetzen und anwenden (Work-Items-List, Git-Issues) - Use Cases ableiten und einen großteil davon detailliert ausarbeiten - Wireframes erstellen/bearbeiten - Verbindung zum Raspberry Pi testen - Erste Funktionalität gemäß Auftraggeber, d.h. Simulation von Messwerten, erarbeiten - ERM (database model) bearbeiten - Aktivitätsdiagramme zu komplexen Abläufen erarbeiten - Links (Likliste vom Auftraggeber) durchgehen und relevante Inhalte raussuchen	09.01.2020
I-5	- Überarbeiten der Dokumentationsdokumente - Design-Dokument erstellen - Belagabgabe vorbereiten/durchführen - Git-Repo "aufräumen"	23.01.2020

Iteration	Primary Objectives	Scheduled start
I-6	Pause aufgrund der Prüfungszeit	06.02.2020

5.6. Fortschritte nach Sim4Seed

Iteration 2:

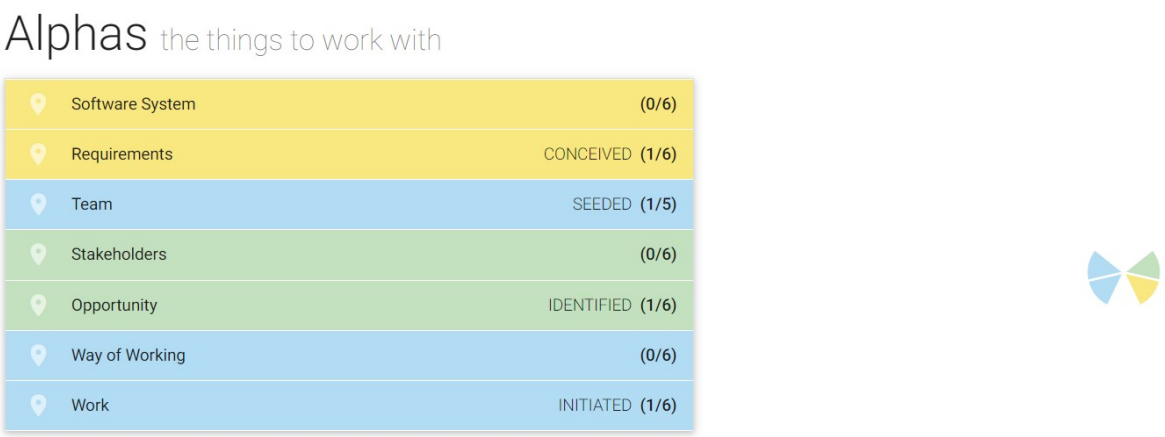

Abbildung 5. Fortschritt Iteration 2

Iteration 3:

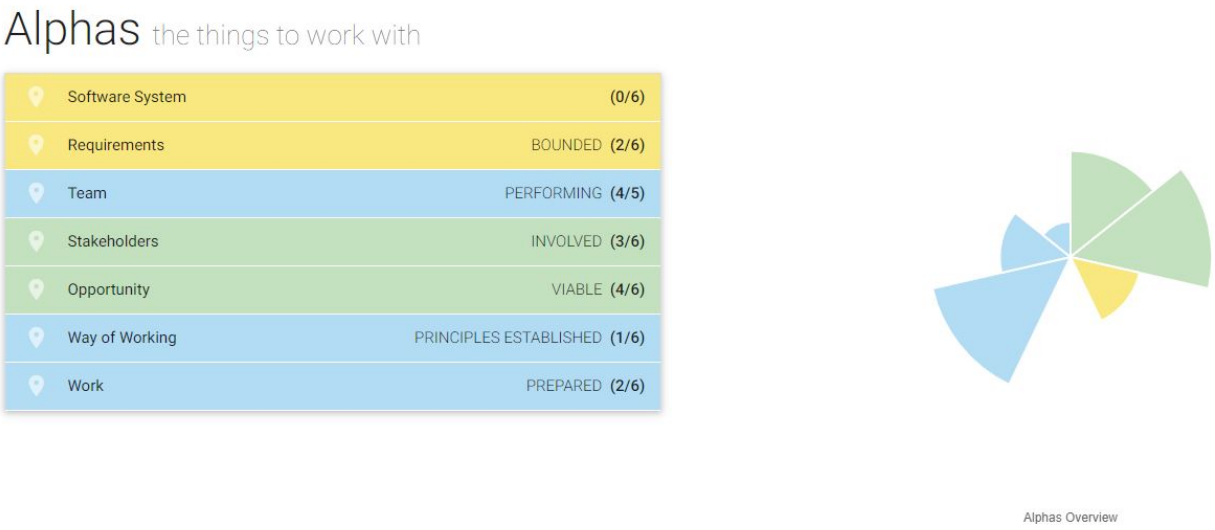

Abbildung 6. Fortschritt Iteration 3

Iteration 4:

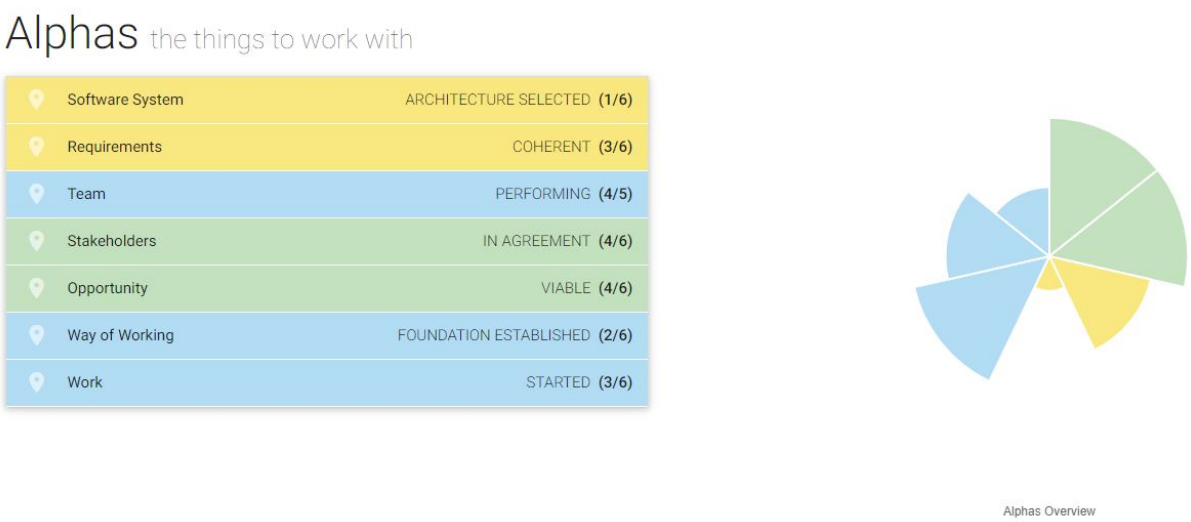

Abbildung 7. Fortschritt Iteration 4

5.7. Eingesetzte Tools/Techniken

- Arbeitsweise nach dem Open Unified Process
- GitHub für Versionsverwaltung/Issues
- Visual Paradigm für Diagramme

5.8. Risk List

Siehe Risikoliste

6. Risk List

6.1. Klasse 1 Projektgefährdende Risiken

1. Ungenügende Terminplanung wird zu spät erkannt
2. Anforderungsliste zu hoch
3. Nichterreichbarkeit Projektbeteiligter
4. Unzureichende Dokumentation
5. Entwicklung an den Anforderungen vorbei
6. Ungenügende Kommunikation
    a. innerhalb des Entwicklerteams
    b. mit dem Kunden

6.2. Klasse 2 Kostenintensive Risiken

1. Unklare Kompetenzzuweisung
2. Angespannte Arbeitsatmosphäre
3. Fehlende Hard-/Software
4. Zu hohe Komplexität der eingesetzen Technologie(n)

6.3. Klasse 3 Moderate Risiken

1. Ausfall Teammitglied
2. Unbeholfenheit eines Teammitlgiedes

6.4. Gegenmaßnahmen

1. Regelmäßige Treffen (spätestens alle 2 Wochen alle zusammen) im Team
2. Treffen mit den Kunden spätestens nach 2 Iterationen
3. Bei Eintritt unklarer Kompetenzzuweisung Zuordnung der Aufgaben durch den Projektmanager
4. Reduktion der Anforderungen auf Mögliches
5. Rollentausch (OpenUP-Rollen) im Team
6. Austausch über Möglichkeiten der Hard-/Software mit den Kunden
7. Zeitplan mit Kunden abstimmen

7. Iterationsplan 12.12.2019 - 26.12.2019

- Anforderungen des Kunden anhand des Gesprächs genauer spezifieren
- Stakeholder finden
- Use Cases ableiten und grob beschreiben
- Vision bearbeiten

7.1. Risiken

Risiken der Anforderungsanalyse und Planung identifizieren, analysieren, bewerten und Gegenmaßnahmen planen

Siehe Risikoliste

7.2. Bewertung

Die Bewertung der Iteration erfolgt beim Teamtreffen nach der Iteration anhand der Verständlichkeit der Aufgabenlösung gegenüber anderen Teammitgliedern und dem Aufkommen an neuen Fragen für das nächste Kundengespräch, wobei mehr Fragen eine bessere Lösung der Aufgaben andeuten können.

8. Iterationsplan 26.12.2019 - 09.01.2020

- Stakeholder finden
- Use Cases ableiten und einen Teil davon detailliert ausarbeiten
- Vision bearbeiten Issue
- Gloassar anlegen/bearbeiten
- 2. Kundengespräch vorbereiten/durchführen

8.1. Work Items List

https://docs.google.com/spreadsheets/d/1HxpITdcoMaGQRV4cEODZcJ2ZDh8JpjwfIRo5Z4ReBgc/edit?usp=sharing

8.2. Risiken

Risiken der Anforderungsanalyse und Planung identifizieren, analysieren, bewerten und Gegenmaßnahmen planen

Siehe Risikoliste

8.3. Bewertung

Die Bewertung der Iteration erfolgt beim Teamtreffen nach der Iteration anhand der Verständlichkeit der Aufgabenlösung gegenüber anderen Teammitgliedern und dem Aufkommen an neuen Fragen für das nächste Kundengespräch, wobei mehr Fragen eine bessere Lösung der Aufgaben andeuten können. Außerdem sollten einige Use-Cases so ausgearbeitet sein, dass wir diese im Kundengespräch erklären und uns Feedback holen können.

Bewertung: Die Aufgaben wurden gelöst und es entwickelten sich neue Fragen an den Kunden. Allerdings war die teaminterne Kommunikation unzureichend, was u.a. dazu führte dass einige Teammitglieder keine Aufgaben lösten. Es wurde weder in Git-Issues, noch in der Work-Items-List oder im WhatsApp-Chat über die Aufgabenlösung gesprochen.

Entwurfsdokumentation

- Architektur-Notizbuch
- Test Cases
- Design

9. Architecture Notebook "Wetterstation MFCR"

9.1. Zweck

Dieses Dokument beschreibt die Philosophie, Entscheidungen, Nebenbedingungen, Begründungen, wesentliche Elemente und andere übergreifende Aspekte des Systems, die Einfluss auf Entwurf und Implementierung haben.

9.2. Architekturziele und Philosophie

Hauptziel des Systems ist es ausgewählte Wetterdaten (Windstärke, Windrichtung, Böenhaftigkeit, Temperatur) mittels eines Raspberry Pi und diversen Sensoren zu sammeln, diese an einen Webserver zu übermitteln und schließlich auf der Website des MFC Rossendorf darzustellen.

Das System lässt sich aus Achitektur-Sicht in drei Teilbereiche kapseln. Ein **Backend** und zwei **Clients**:

- **Datensammlung und -übertragung auf dem Raspberry Pi** (*client 1*)

> Der Raspi sammelt durch das Auslesen von Sensoren zyklisch Daten. Diese werden auf dem Raspi lokal zwischengespeichert. (Umsetzung durch Auftraggeber). Anschließend postet der Raspi diese an eine REST-API.

- **Webserver** (*backend*)

> Der Webserver erhält zyklisch die Daten vom Raspi und speichert diese in einer Datenbank. Außerdem bietet er eine REST-API für den Raspi an, damit dieser seine Konfigurationsparameter ermitteln kann. Zudem kommuniziert er mit dem Web-Frontend (*client 2*) über eine REST-API zur Übertragung der Wetterdaten.

- **Web-Frontend** (*client 2*)

> Das Web-Frontend konsumiert über die REST-API die gespeicherten Wetterdaten. Die Darstellung dieser Daten findet dann auf der Website des MFC Rossendorf statt.

Wesentliche Architekturziele:

- Ressourcenschonendes Vorgehen aufgrund eingeschränkter Hardware der Wetterstation
  - Die Datenakquise und anschließende Übertragung an den Server erfolgt wahrscheinlich auf einem **Raspberry Pi 4**. Zwar hat dieser mittlerweile genügend Leistung um diesen Punkt zu vernachlässigen, jedoch sollte der externe Akku nicht unnötig beansprucht werden, da das

System autark auf dem Flugplatz des MFCR läuft. Zudem sind die Wetterdaten lokal zu speichern um sie bei evtl. Verbindungsabbruch später übertragen zu können.

- Robustheit gegen mögliche Fehler
  - Das System soll nach Fertigstellung auf dem Flugplatz des MFCR im Freien installiert werden, was den Wartungsvorgang deutlich erschwert. Fehlerfälle müssen demnach sorgfältig behandelt und ggf. geloggt werden.
- Gute Wartbarkeit und Erweiterbarkeit
  - Den Auftraggebern soll die Möglichkeit gegeben werden zu einem späteren Zeitpunkt weitere Funktionalitäten hinzuzufügen (bspw. zusätzlicher Sensor). Dafür sollte das System mit möglichst geringem Aufwand erweiterbar sein.
  - Um dies zu unterstützen und die Entwicklung zu vereinfachen sollte auf gängige Design-Patterns (Clean Code) und Dokumentation wertgelegt werden.
- Performance und Aktualität
  - Der User sollte bei der Abfrage der Daten auf der Website Anworten mit möglichst geringer Antwortzeit vom Backend erhalten.

9.3. Annahmen und Abhängigkeiten

1. eingeschränkte Ressourcen:
   a. die Wetterstation soll autark laufen
      - kein Anschluss ans Stromnetz, nur ein Akku mit Solarmodul → möglichst stromsparendes Vorgehen
      - Netzwerkanbindung über UMTS-Stick mit beschränktem Datenvolumen → Datenvolumen darf nicht überschritten werden
2. Nicht-funktionale Vorgaben der Auftraggeber:
   a. Auftraggeber hätten Backend des Webserver gern in einer Skriptsprache
3. Technische Abhängigkeiten vom Auftraggeber:
   a. Hardware lagert während der Entwicklungszeit beim Kunden → externer Zugang muss gegeben und am besten immer verfügbar sein
   b. Gewisse Funktionalität des Systems v.a. auf Hardwareebene übernimmt Kunde selbst, wie z.B. Integration des externen Wake-up Timers, Stromquellenmanagement, sowie das Auslesen der Sensoren. Wir nutzen in diesen Fällen nur die gestellten Schnittstellen.
4. Vorasugegange Entscheidungen der Auftraggeber:
   a. Hosting der Website wird extern betrieben von "jweiland.net"
      - dort soll auch die zusätzliche SW laufen
      - Zudem wird unser System dann in das aktuelle TYPO3 CMS eingebettet um es auf der Website darzustellen
        - Auftraggeber wollten sich darum kümmern
5. Unerfahrenheit des Teams:

a. keiner hat nennenswerte Erfahrungen mit den Webtechnologien die wir einsetzen werden
    - HTML, CSS, JS, REST, php → größerer Zeitaufwand da Einarbeitung notwendig

9.4. Architektur-relevante Anforderungen

1. Systemweite Anforderungen an Benutzbarkeit
2. Systemweite Anforderungen an Zuverlässigkeit
3. Systemweite Anforderungen an Performance
4. Systemweite Anforderungen an Wartbarkeit
5. Weitere Systemweite Anforderungen

9.5. Entscheidungen, Nebenbedingungen und Begründungen

1. Programmiersprache Python für den Raspi verwenden
    - Aufgrund der einfachen Wartbarkeit und guten Handhabbarkeit der Skriptsprache wurde diese Einschränkung von den Auftraggebern vorgegeben. Aufgrund der Vielzahl an verfügbaren Bibliothekten, riesigen Community und der schnell erlernbaren Syntax eine sinvolle Wahl.
2. Programmiersprache python mit dem Django Rest Framework für Backend verwenden
    - Einige Teammitglieder haben bereits Erfahrung mit python. Des Weiteren ist die Sprache relativ leicht zu erlernen. Das Django Rest Framework bietet zusätzlich ein ORM an.
    - Alternativ wurde vom Auftraggeber php als Backend vorgeschlagen → womit allerdings niemand im Team Erfahrung hat
3. Angular als Frontend Framework, da Entwickler bereits Erfahrungen damit haben
4. persistente Datenspeicherung in einer DB, um Daten langfristig zu sichern
    - das DBMS wurde vom Auftraggeber ausgewählt (mySQL)
5. Kommunikation zwischen Raspi und Webserver bzw. DB-Server erfolgt über eine Rest-Schnittstellen über HTTPS

9.6. Architekturmechanismen

Doku "Concept: Architectural Mechanism"

1. Webserver
    - kommuniziert mit dem Client sowie dem DB-Server um Kontent auszuliefern
2. DB-Server
    a. mit REST-Schnittstelle
        - für CRUD-Operationen
    b. mit mySQL-DBMS zur Speicherung der Daten

3. Ajax
    - Kommunikation zwischen Client und Server zur dynamischen Erstellung der Website
4. relationales DBMS
    - in einer mySQL-DB werden die Sensordaten vom Webserver gespeichert und abgerufen

9.7. Wesentliche Abstraktionen

9.8. Schichten oder Architektur-Framework

- Client-Server Model:
    - User und Raspi fungieren als Clients, die über eine REST-Schnittstelle mit dem Webserver kommuniziert (request and response via http)
- MVC-Pattern:
    - durch Django-REST Framework gegeben
        - Model = Speicherung der Entitäten in der DB (mySQL-DB)
        - View = Darstellung der Daten im Webbrowser des User (HTML, CSS, JS bzw. Angular)
        - Controller = Implementierung der Logik der Anwendung. Er empfängt die Requests der Clients, verarbeitet diese und antwortet diesen.

9.9. Architektursichten (Views)

Folgende Sichten werden empfohlen:

9.9.1. Logische Sicht

Dataflow-Diagram

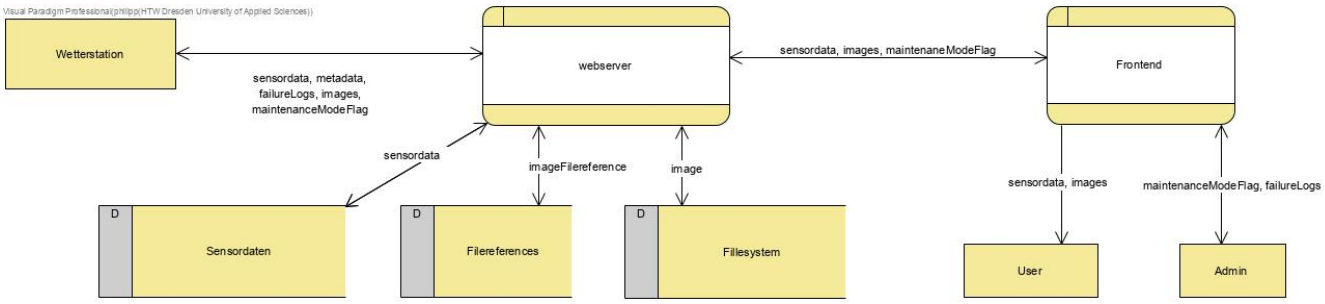

Abbildung 8. Datenflussdiagramm

10. Design Documentation

10.1. Entity-Relationship-Modell

In dem von uns erstelltem 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'.

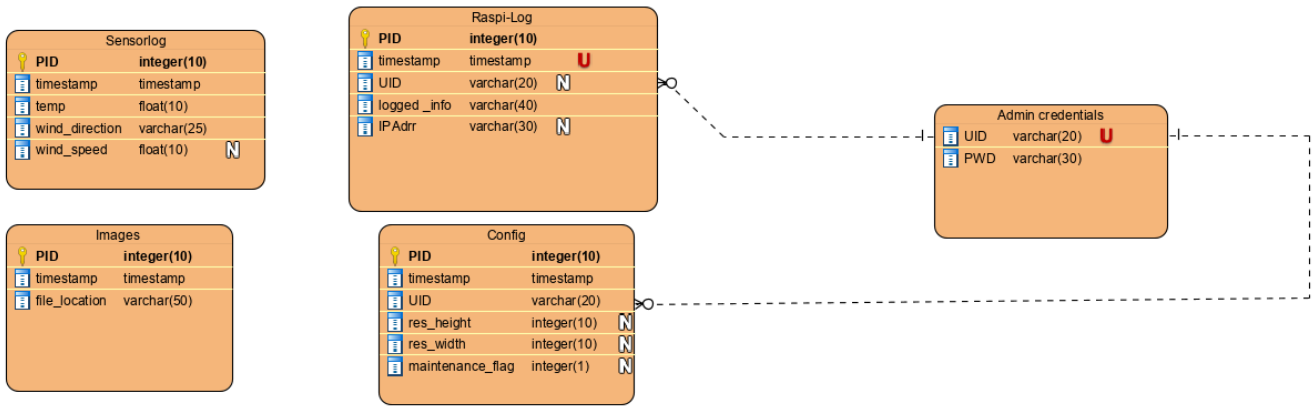

Abbildung 9. Entity-Relationship-Modell