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

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

1. Vision -Projektthema-

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

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

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

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
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

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
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)

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

2. Use-Case Model <Projektthema>

Unresolved directive in technical_specifications/usecase_model.adoc - include::usecase_wartungsmodus_aktivieren.adoc[]

2.1. Use-Case: Diagramm abrufen

2.1.1. Kurzbeschreibung

Dieser Usecase behandelt das Diagramm, welches auf der Website aus den gesammelten Daten erstellt werden soll und von den Nutzern der Website abgerufen werden soll.

2.1.2. Kurzbeschreibung der Akteure

User

Die normalen User, die sich die Daten der Wetterstation als Diagramm anzeigen lassen wollen.

Admins

Die Admins, die sich die Daten der Wetterstation anzeigen lassen wollen.

2.1.3. Vorbedingungen

1. Raspberry Pi hat valide Sensordaten gesammelt
2. Raspberry Pi übermittelt Daten an Webserver
3. Daten werden richtig in Datenbank gespeichert
4. Daten werden vom Webserver richtig aufgerufen
5. Daten werden ordnungsgemäß verarbeitet

2.1.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn User/Admins die Website aufrufen
2. Der User/Admin wählt eine Datenart aus
3. Der User/Admin wählt einen Zeitraum
4. Die Website greift auf die Datenbank zu und zieht sich die entsprechenden Daten aus der Datenbank
5. Die Website generiert das Diagramm den Daten entprechend
6. Der Use Case ist abgeschlossen.

2.1.5. Alternative Abläufe

Zeitraumfehler

Wenn der User/Admin im Schritt 3 des Standardablauf einen Zeitraum wählt in dem keine Daten aufgezeichnet wurden, dann wird der User/Admin darum gebeten einen anderen Zeitraum auszuwählen . Der User/Admin wählt einen Zeitraum . Der Server versucht auf die Datenbank zuzugreifen und stellt fest, dass im angegebenen Zeitraum keine Daten aufgezeichnet wurden . Dem User/Admin wird angezeigt, dass er bitte einen anderen Zeitraum auszuwählen soll . Der Use Case wird im Schritt 3 fortgesetzt.

2.1.6. Unterabläufe (subflows)

Andere Datenart

1. Der User/Admin möchte die Datenart ändern
2. Der Server greift erneut auf die Datenbank zu
3. Die Website generiert ein neues Diagramm, den angegebenen Daten entsprechend

Anderer Zeitraum

1. Der User/Admin möchte den Zeitraum ändern
2. Der Server greift erneut auf die Datenbank zu
3. Die Website generiert ein neues Diagramm, dem angegebenen Zeitraum entsprechend

2.1.7. Wesentliche Szenarios

2.1.8. Nachbedingungen

Diagrammanzeige

1. Dem User/Admin wird ein Diagramm entsprechend den ausgewählten Daten und dem Zeitraum angezeigt und wartet auf Änderungen von User-/Adminseite

2.1.9. Besondere Anforderungen

Gleichzeitige Änderung der Daten und des Zeitraumes

1. Der User/Admin hat die Möglichkeit die Datenart und den Zeitraum gleichzeitig zu ändern, sodass das Diagramm dementsprechend generiert wird

Unresolved directive in technical_specifications/usecase_model.adoc - include::usecase_bildergalerie.adoc[]

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

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

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

Abkürzung	Bedeutung	Erläuterung
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

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

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 1:

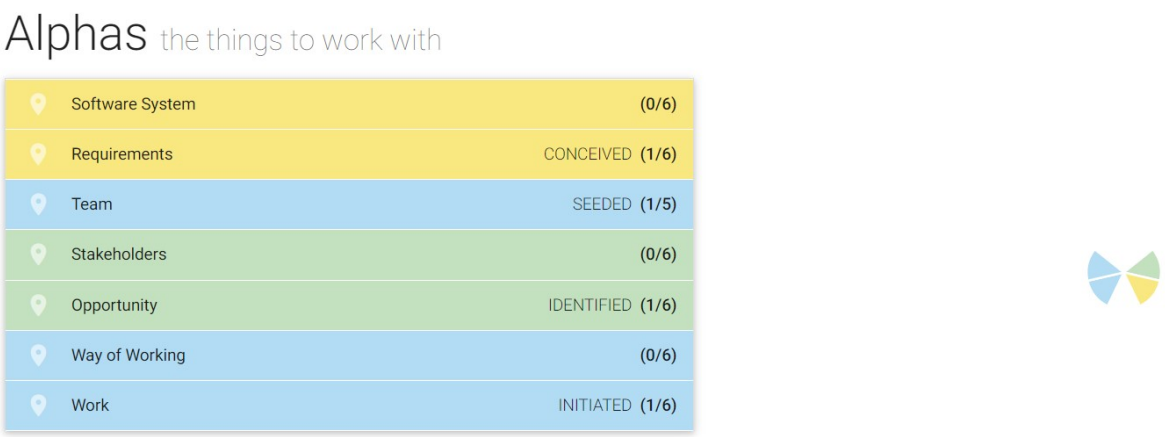

Abbildung 1. Fortschritt Iteration 2

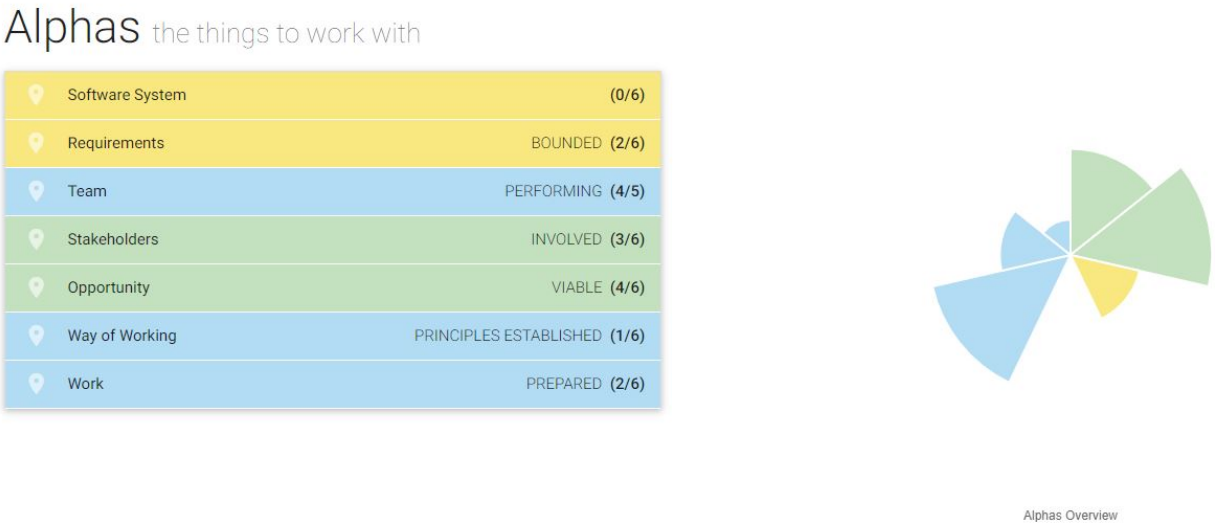

Abbildung 2. Fortschritt Iteration 3

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 12.12.2019 - 26.12.2019

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

8.1. Risiken

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

Siehe Risikoliste

8.2. 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.

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 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 Azftraggeber 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

9.9.2. Physische Sicht (Betriebssicht)

9.9.3. Use cases

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'.