HSR Studienarbeit

IBM SmartPower

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2013

Autor(en): Giuseppe Aquino / Simon Brouwer
Betreuer: Prof. Dr. Markus Stolze
Projektpartner: IBM

gaquino, sbrouwer
2.12.2013

Inhaltsverzeichnis

1 Management Summary 3
1.1 Ausgangslage 3
1.2 Vorgehen/Technologie 3
1.3 Ergebnis 3
2 Aufgabenstellung 4
2.1 Auftraggeber und Betreuer 4
2.2 Ausgangslage & Zielsetzung 4
2.3 Funktionale Anforderungen 4
2.3.1 Anforderungen für Entwicklungsphase 4
2.3.2 Personas 5
2.3.3 Szenarien 5
2.3.4 Use Cases (Brief) 6
2.4 Nicht Funktionale Anforderungen 6
2.4.1 Funktionalität 6
2.4.2 Benutzbarkeit 6
2.5 Wartbarkeit 7
2.5.1 Übertragbarkeit 7
3 Lösungskonzept 7
3.1 Dokumentation Architektur und Design 7
3.1.1 FERN Client GUI 8
3.1.2 FERN Proxy Webservice 9
3.2 Architekturentscheidungen und Begründungen 9
3.2.1 Eingesetzte Frameworks 10
4 Umsetzung 10
4.1 XML Antworten und Backbone JS Collections 10
4.2 Critical Sensor 10
4.2.1 Problem 11
4.2.2 Lösung 11
4.3 Schedule Collection 11
5 Ergebnis 12
5.1 Stärken und Schwächen des Produktes 12
5.1.1 Critical State in der Übersicht 12
6 Zusammenfassung und Ausblick 13
6.1 Zeitauswertung 13
6.1.1 Iterationen 13

6.1.2 Aktivitäten 13
6.2 Erfahrungsbericht & Lessons Learned 14
7 Literaturverzeichnis 16
8 Abbildungsverzeichnis 16

1 Management Summary

1.1 Ausgangslage

Momentan ist die Energie-Diskussion in aller Munde, vor allem auch beim Hausbau. Es stellt sich die Frage wie man ein Haus am Energie- und damit auch am Kostengünstigsten beheizen kann. Die IBM hat sich nun dieser Frage angenommen. Im Rahmen des Smart Grid Projektes der IBM wurden in Häusern Systeme installiert welche es erlauben, essentielle Werte, wie zum Beispiel die Temperatur des Hauses oder den momentanen Energieverbrauch des Hauses, zu überwachen und mit einem von der IBM entwickelten Algorithmus zu analysieren. Anhand der Analyse der Messwerte wird dann zum Beispiel die Heizung des Hauses gesteuert und sichergestellt, dass möglichst wenig Energie verbraucht wird ohne, dass das Haus unterkühlt ist.
Da die algorithmische Steuerung noch Gegenstand der Forschung ist, ist es wichtig möglichst schnell Häuser zu identifizieren welche die gesetzten Ziele nicht einhalten und dann manuell eingreifen zu können.
Um dies zu realisieren wurde ein rudimentäres Monitoring System entwickelt.
Die Aufgabe dieser Arbeit war ein Monitoring System zu entwickeln, welches das bisherige erweitert und verbessert. Das Ganze soll mit JavaScript sowie HTML5 realisiert werden und zudem in der Lage sein mit den vorhandenen Webservices zu arbeiten.

1.2 Vorgehen/Technologie

Da das bereits bestehende System unter einer gemischten Windows/Linux Umgebung genutzt wurde, soll das neue System mit Browser wie Firefox und/oder MS Internet Explorer genutzt werden können. Weil statische HTML Seiten die Anforderungen nicht erfüllen wurde JavaScript eingesetzt um eine dynamische Single-Page Webapplikation zu erstellen.
Das System welches von der IBM entwickelt wird befindet sich noch in der Aufbauphase, aus diesem Grund sollte das neue System einfach erweiterbar sein und mit den bereits vorhandenen Services kompatibel sein. Auch das optische Bild des aktuellen Systems sollte verbessert werden, so dass das System an Attraktivität für den späteren Kunden, wie z.B. die EKZ, gewinnt.

1.3 Ergebnis

Das Produkt der Studienarbeit „SmartPower GUI" ist ein schlichtes und einfach zu bedienendes User Interface welches leicht erweiterbar ist.
Das Web Interface bietet einen Überblick über das gesamte System und hebt die wichtigsten Informationen für den Kunden hervor. Die grafische Darstellung war dabei einer der wichtigsten Kriterien. Der Benutzer ist ausserdem in der Lage frühere Messungen einzusehen und zu prüfen.

2 Aufgabenstellung

2.1 Auftraggeber und Betreuer

Diese Studienarbeit wurde von der IBM ZRL in Rüschlikon in Auftrag gegeben.

- Ansprechpartner Auftraggeber: Dr. Douglas Dykeman, IBM Rüschlikon
- Betreuer HSR: Prof. Dr. Markus Stolze, Institut für Software

2.2 Ausgangslage & Zielsetzung

Das verbessern des Energieverbrauchs, sei es für Privathaushalte oder Firmen, ist bereits ein Schritt in einer moderneren, effizienteren Welt. Damit das SmartPower-Projekt ein Erfolg wird, musste ein GUI her welches den heutigen Vorstellungen entspricht. Mobile Applikationen sind heutzutage nicht mehr weg zu denken und da praktisch jedes Gerät über ein Browser verfügt ist es naheliegend für die Stromüberwachung ebenso einen Web Frontend zur Verfügung zu stellen, welches von überall und jederzeit zur Verfügung steht. Das System welches vor der Studienarbeit vorhanden war, war als Zwischenlösung gedacht und verfügte nicht über die Monitoring Funktionalität welche vom Auftraggeber (IBM) benötigt war und welche es ermöglicht das System auf eine effiziente Art und Weise zu überwachen bzw. zu kontrollieren. Ebenso war ein Redesign des Interfaces nötig um die Attraktivität für diesen Service zu steigern. Dabei sollte auch die Usability des Produktes unter die Lupe genommen werden, um so ein verbessertes Benutzererlebnis zu ermöglichen.

2.3 Funktionale Anforderungen

Die funktionalen Anforderungen wurden Schritt für Schritt mit der IBM erarbeitet. Da das Projekt noch in einer Pilotphase läuft, war es oft schwierig eine Entscheidung bezüglich realisierbaren Features zu fällen. Eine umfangreichere Dokumentation der Anforderungen wurde zusätzlich in einem separaten Dokument festgehalten.

2.3.1 Anforderungen für Entwicklungsphase

Da die Webservices welche von der IBM zur Verfügung gestellt wurden auf einer anderen Domain abgelegt waren, musste eine Lösung gefunden werden, welche es erlaubt Cross Domain AJAX Anfragen zu stellen. Dafür musste ein Proxywebservice entwickelt werden welches lokal gestartet werden konnte und die Anfragen an den IBM Webservice stellte und weiterleitete.

2.3.2 Personas

Für die Entwicklung von den Use Cases wurden Personas erstellt, welche den späteren Benutzern der Anwendung entsprechen.

Kurt Kontrolle

Name:	Kurt Kontrolle
Alter:	32
Funktion:	Software Entwickler
Kenntnisse:	Experte

Beschreibung:

Kurt Kontrolle arbeitet bereits seit längerer Zeit bei der IBM ZRL in Rüschlikon.
Zusammen mit seinem Team, entwickelt er Software die dazu beiträgt die Energieeffizienz zu verbessern.

Kurt Arbeitet hauptsächlich an seinem 13" Notebook auf welchem Windows 7 läuft. Als Browser wird von ihm Firefox in der jeweils aktuellsten Version genutzt.

Durch seine Arbeit als Software Entwickler kennt sich Kurt mit Computern sehr gut aus und kann somit auch komplexere Anwendungen bedienen, jedoch ist auch er über eine simple und schnelle Bedienung der zu anwendenden Applikationen dankbar, da diese seine Produktivität in anderen Bereichen steigern könnte.

„Das System an dem gerade gearbeitet wird ist noch nicht in einem Stabilen Zustand, weshalb Fehlerfälle schnell erkannt und nachvollziehbar sein müssen."

2.3.3 Szenarien

Szenario 1.0: Anzeigen von Warnungen in einem fehlerhaften Gerät

Kurt hat immer alles im Griff, manchmal passiert es jedoch trotzdem, dass das eine oder andere Geräte (Device) Schwierigkeiten macht. Für das Monitoring der Devices benutzt Kurt das Tool FERN. Schnell sortiert Kurt in der Geräteübersicht nach den Warnungen und schon sind ihm die Geräte ersichtlich welche Warnungen enthalten. Kurt nimmt sich in diesem Fall das Gerät vor, welches die meisten Warnungen enthält und klickt diesen erstmals an um zu schauen welche Probleme auf dem Device bestehen. Nun sieht er eine Liste von den Warnungen für das Device und kann mit Szenario 2.1 weiterfahren.

Szenario 1.1: Anzeigen wie lange ein Gerät schon fehlerhaft läuft

Kurt bemerkt kurz vor der Mittagspause, dass eines der Devices Warnungen anzeigt. Weil es sich um nicht gravierende Warnungen, welche schnell wieder einen unkritischen Zustand einnehmen können handelt, beschliesst Kurt trotzdem seine Mittagspause zu beziehen. Als er vom Mittag zurückkehrt, bemerkt Kurt dass die Warnungen immer noch vorhanden sind. Er lässt sich besorgt die Details des Devices anzeigen und sieht sofort der Fehlerhafte Sensor welches die Warnung produziert. Dank dem Graphen ist Kurt in der Lage zu sagen wie lange dieses Gerät bereits fehlerhaft ist.

Szenario 2.0: Anzahl Registered / Deregistered Geräte

Kurt's wöchentliches Teammeeting steht an. Sein Chef hat ihn letzte Woche gebeten zu überprüfen wie viele Geräte registriert und wie viele nicht registriert sind. Kurt möchte den Chef beeindrucken und ihm kurzfristig die aktuellsten Werte vorlegen. Deshalb geht er kurz vor der Sitzung auf die FERN Seite und öffnet die Device Overview. Dort findet Kurt ein Diagramm mit der gesamten Anzahl der Devices und die Unterscheidung zwischen registriert und nicht registriert. Diese Statistik kann er nun ausdrucken oder direkt auf dem Bildschirm presentieren.

Szenario 2.1: Status des Gerätes anzeigen/ändern

Kurt bemerkt dass eines der Geräte immer wieder in einem fehlerhaften Zustand fällt und nach kurzer Zeit wieder OK ist. Ihm ist in diesem Fall nicht gleich klar was das Problem mit dem Gerät ist. Um den Kunden des Devices nicht zu verärgern entscheidet sich Kurt das Device zu deregistrieren bis das Problem behoben ist. Kurt wählt das Device aus und stellt dessen Status auf „deregistered". Zusätzlich gibt er einen Grund an wieso diese Entscheidung gefällt wurde. Kurt kann so immer den Statusverlauf nachvollziehen.

2.3.4 Use Cases (Brief)

Anhand der Personas und der User Stories wurden dann die Use Cases Brief sowie auch in Fully Dressed form geschrieben (Siehe Anforderungsspezifikation.pdf). Hier werden die wichtigsten Use Cases in Brief Format aufgelistet.

UC 01: View devices/groups/objectives

Der User lässt sich die aktuelle Device-Liste anzeigen. Dabei kann er die Devices sortieren und beim Anklicken die wichtigsten Informationen anzeigen lassen. Dies funktioniert gleichermassen für die groups und objectives.

UC 02: View devices with state «critical»

Der User sieht auf einem Blick welche Devices in einem Kritischen Zustand sind. Dazu sind die Devices optisch gekennzeichnet.

UC 03: View planed/actual measurements

Nachdem der User einen Device oder eine Gruppe von Devices ausgewählt hat, kann er sich den geplanten bzw. den aktuellen Stromverbrauch anzeigen lassen. Dies wird mit einem Graphen grafisch dargestellt.

UC 04: View device

Nachdem der User in der Device Overview ein Device ausgewählt hat, wird diese mit den gängigen Informationen angezeigt. Zusätzlich hat der User eine Liste der eingetragenen Sensoren für das Device.

2.4 Nicht Funktionale Anforderungen

Folgend sind die wichtigsten Nicht Funktionale Anforderungen aufgelistet. Weitere NFAs sind im Anforderungsspezifikation.pdf zu finden.

2.4.1 Funktionalität

2.4.1.1 Interoperabilität

Das System wird während der Entwicklung mit Testdaten arbeiten. Später soll es in der Lage sein, mit wenigen Anpassungen, mit den zur Verfügung stehenden Webservices zu arbeiten. Es soll möglich sein mit Anpassung einer Variable pro Webservice die richtigen

2.4.2 Benutzbarkeit

2.4.2.1 Erlernbarkeit

Das Produkt wird hauptsächlich von Personen mit technischem Hintergrund genutzt. Der Benutzer soll in der Lage die Funktionalität des User Interfaces schnell zu erlernen (ca. 5-10 Minuten). Dafür sollen alle Felder sowie Knöpfe verständlich und nachvollziehbar beschriftet sein und wenn nötig mit Tooltips ergänzt werden.

2.4.2.2 Bedienbarkeit

Das User Interface soll so gestaltet werden, dass es auch auf mobile Geräte, sowie Tablets oder Smartphones, korrekt angezeigt und möglichst einfach bedient werden kann.
Weil im System, je nach Operation, mit längeren Rechenzeiten gerechnet werden muss, müssen Länger dauernden Anfragen optisch gekennzeichnet werden z.B. mit einem Ladebalken.

2.5 Wartbarkeit

Erweiterungen des Systems sollen nicht eine neue Seite erfordern. Das MVC Konzept soll erlauben Views in das System hinzuzufügen welche die neue gewünschte Funktionalität mitbringen.

2.5.1 Übertragbarkeit

2.5.1.1 Anpassbarkeit

Die Software wird während der Entwicklungsphase mit Daten gefüllt die aus einem statischen Umfeld stammen. Der Zugriff auf Daten muss deshalb zwingend variabel gestaltet werden, sodass später mit Anpassung einiger Variablen ein Zugriff auf die produktiven Webservices möglich ist.

3 Lösungskonzept

3.1 Dokumentation Architektur und Design

Die Architektur des GUIs wurde sehr stark von den bereits von der IBM zur Verfügung gestellten Backend Services beeinflusst. Das System wurde hauptsächlich mit dem Einsatz von JavaScript und HTML5 realisiert. Dank dem Einsatz von Frameworks wie Backbone JS, konnte eine Single Page Applikation realisiert werden.

> *AJAX (asynchronous JavaScript and XML) has enabled modern web applications to provide rich functionality to Internet users. AJAX based web applications avoids full page reloads and updates relevant portion of a page.*
> *(Usman Shaukat Qurashi, 2013)*

Der Proxy web service, welcher benötigt wurde, um Cross Domain Anfragen auf den IBM Web Service durchführen zu können, wurde in Java geschrieben. Dieser wird jedoch im Endsystem nicht mehr benötigt und diente somit nur als Stütze während der Entwicklung. Zusammenfassend wurden also im Verlauf der Studienarbeit das FERN Client UI und der Proxy Webservice erstellt (siehe Umrahmung).

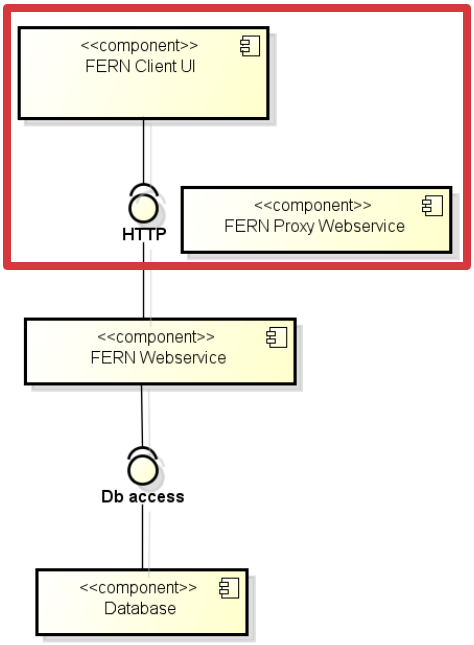


Abbildung 1: Systemübersicht

3.1.1 FERN Client GUI

Da bereits ein Web Interface vorhanden war konnten einige Plug-Ins direkt übernommen werden. Eine der wichtigsten Änderungen welche auch die Architektur des Interface beeinflusste, war der Einsatz von Backbone JS als Framework. Dieses Framework erlaubte es eine Singlepage Architektur zu realisieren und bot ausserdem nützliche Features für das abbilden der REST Service Antworten auf einem Model, so wie es bei einer Native Applikation der Fall ist. Ebenso konnten so Views erstellt werden, wodurch „duplicated code" massiv reduziert werden konnte.

Ordnerstruktur des FERN Client GUIs

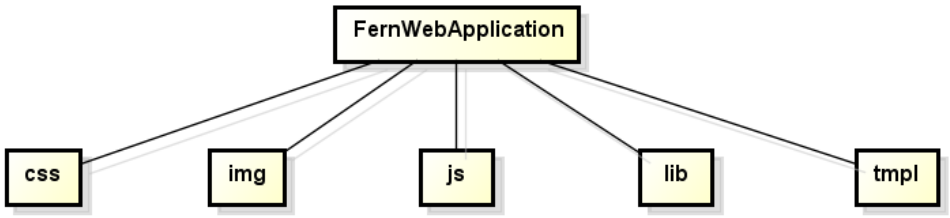


Abbildung 2: Ordnerstruktur

Folder	Beschreibung
css	Alle CSS Dateien enthalten sowie benötigte Fonts für Bootstrap.
img	Logos und sonstige Bilder.
js	Core des GUIs, alle Dateien welche die Funktionalität implementieren sind hier entehalten (Views, Collections, Models).
lib	Frameworks & Plug-Ins sind hier abgelegt.
tmpl	Templates die für die Darstellung der Views benötigt werden.

JS Ordner

Das JS Ordner beinhaltet alle Views sowie Collections und Models. Aus verschiedenen Gründen wurden einige Collections und Models in einer Datei implementiert:

- Zu viele Includes in der index.html sorgten für Unübersichtlichkeit
- Einige Models erweisen ähnliche Funktionalität, so dass diese „gruppiert" werden konnten.

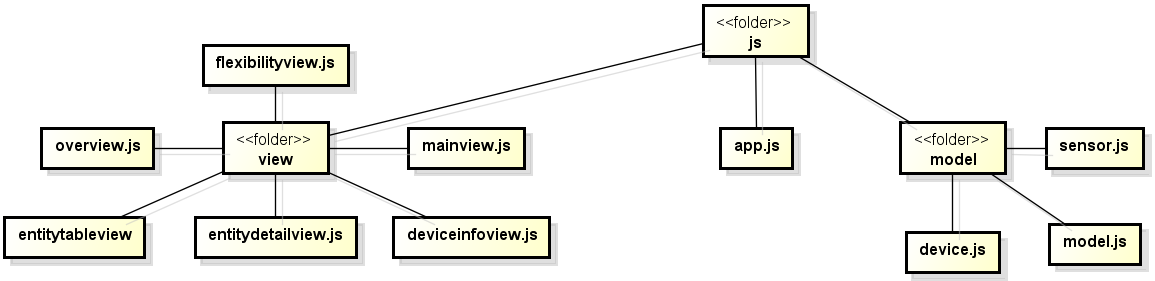


Abbildung 3: Ordnerstruktur js Ordner

3.1.2 FERN Proxy Webservice

Während der Entwicklungsphase war der Zugriff auf die IBM Webservices nur über VPN möglich ausserdem liefen die Webservices für welche das GUI geschrieben werden sollte auf einem IBM Server.

> *Inline frames can be used to download rich HTML documents from outside sources, but if the content comes from a different domain, the browser will not allow the JavaScript in the containing page to read or manipulate the document inside the frame, and vice versa. The XMLHttpRequest can be used to download arbitrary XML documents without rendering them in a browser pane, but it cannot be used to download files that are not from the same domain as the page making the request. (Jackson & Wang, 2013)*

Um das Cross-Domain Problem zu lösen, welches durch den Einsatz von AJAX und Backbone JS zustande kam zu lösen, wurde ein Proxywebservice entwickelt, welcher die Antworten des richtigen Webservice spiegelt und dem Client weiterleitet. Dieser Webservice wird für den Einsatz in der Umgebung vom Auftraggeber nicht mehr benötigt und diente nur zur Entwicklung ausserhalb der Einsatzumgebung. Folgend ein kleines Beispiel.

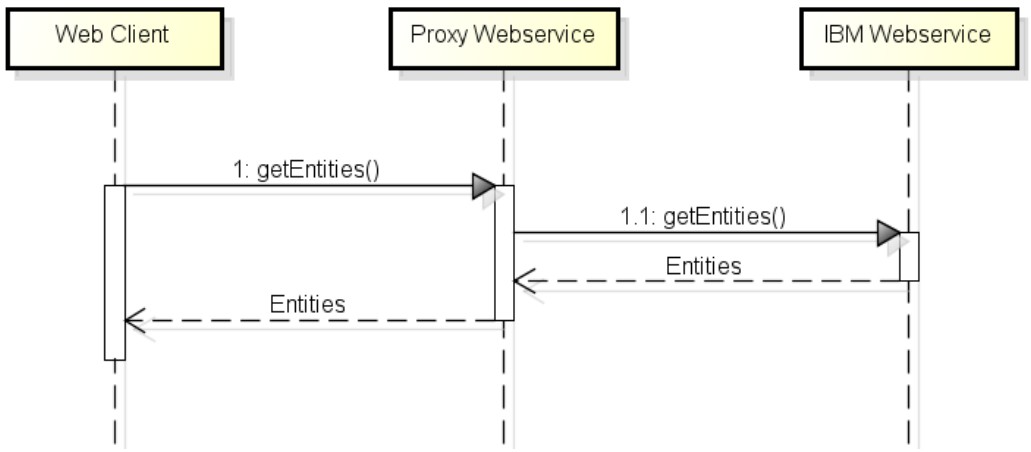


Abbildung 4: Sequenzdiagramm Proxy Webservice

3.2 Architekturentscheidungen und Begründungen

Zu Anfang stellte sich die Frage ob eine Single-Page Architektur oder eine Multi-Page Architektur umgesetzt werden soll. Für die Single-Page Variante sprach, dass sie schnell und modern ist und das sie ohne das erneute Laden der Seite auskommt. Also wenn auf einen Link geklickt wird, wird nicht die gesamte Seite neu geladen, sondern nur ein Teil der Seite durch dynamisch geladenen Inhalt ersetzt.
Gegen eine Single-Page Architektur sprach, dass man beim neu laden der Seite jedes Mal wieder auf der Startseite landet und keine Sprechenden URLs verwendet werden können, dies ist bei einer Multi-Page Architektur nicht der Fall.
Schlussendlich entschieden wir uns für die Single-Page Variante, da sich heraus stellte, dass mit dem Backbone.js Framework sprechende URLs erstellt werden können und so auch beim neu laden der Seite wider am gleichen Ort wie zuvor weitergearbeitet werden kann. Ausserdem erlaubte uns der Einsatz des Backbone.js Frameworks das MVC Pattern zu implementieren und so die Applikation besser strukturieren zu können, bzw. wie eine gewöhnliche native App aufzubauen.

3.2.1 Eingesetzte Frameworks

Für den Grundaufbau der Applikation wurde Backbone JS als Framework eingesetzt. Backbone JS erlaubt es das MVC Pattern zu implementieren und so Model, Views sowie Controls für die Applikation zu realisieren. Für das SmartPower IBM Projekt wurden folgende Komponenten von Backbone JS verwendet:

- Backbone.View
- Backbone.Model
- Backbone.Collection
- Backbone.Router

Die Views, Models und Collections sind Konzepte die bereits bekannt sein sollten, sei es von Java oder anderen nativen Sprachen. Das Router Konzept findet hauptsächlich Nutzung in der Webentwicklung. Erst durch den Router ist es möglich eine Single Page App zu realisieren, da dieser erlaubt eine Aktion, also sprich eine Methode, für eine bestimmte URL aus zu führen. Ausserdem regelt der Router das Handling der Historie im Browser.

> *Most programming languages contain good and bad parts, but JavaScript has more than its share of the bad, having been developed and released in a hurry before it could be refined. (Crockford, 2008)*

4 Umsetzung

4.1 XML Antworten und Backbone JS Collections

Durch den Einsatz von Backbone JS wird das Portieren von Devices/Aggregationen/Objektive auf dem Web Client vereinfacht weil es bereits die Funktion anbietet Collections via AJAX aufzubauen. Diese Collections erwarten jedoch einen JSON String. Weil die IBM Webservices jedoch oft XML als Antwort lieferten mussten die Collections meistens geparst werden.

4.2 Critical Sensor

Ein Feature welches mit Priorität behandelt wurde, war die Anzeige von kritischen Devices auf der Übersichtsseite. Der Webservice für die effiziente Implementierung war jedoch zum Zeitpunkt der Entwicklung noch nicht verfügbar, daher musste man für die Abfrage des Critical Sensor mehrere AJAX Anfrage starten. Um das Problem zu veranschaulichen wurde eine Annäherung der Domäne gemacht.

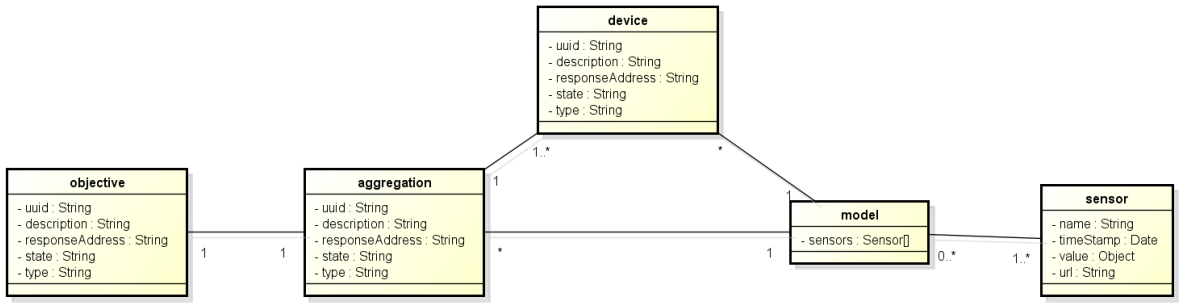


Abbildung 5: Domain Model

Wie man im Domänenmodell sehen kann muss für die Abfrage eines Sensors, das Model eines Devices bzw. einer Aggregation abgefragt werden. Das Model gibt Auskunft über die für das Device/Aggregation verfügbaren Sensoren, wobei für diesen Fall nur das „eg_critical" Sensor benötigt wurde.

4.2.1 Problem

Will man mit diesem Aufbau also einen Device bzw. eine Aggregation abfragen benötigt man 2 AJAX Anfragen, nämlich einmal auf das Model des Devices um die URL des Sensors herauszufinden und eine um den tatsächlichen Sensor anzufragen. Das Problem entsteht bei einer Liste von mehreren hundert Devices, was zur Folge hat das die Seite zu lange braucht um dargestellt zu werden.

4.2.2 Lösung

Das Problem wurde gelöst indem man zuerst eine Liste der Devices lädt, was mit einer Anfrage möglich ist und die benötigte Übersichtstabelle auf der Seite anzeigt. Nachdem diese geladen ist werden alle Devicemodelle angefragt um die URL des Sensors herausfindig zu machen. Diese Information wird dann der Liste der Devices hinzugefügt und die Übersichtstabelle wird mit neu gezeichnet. In folgender Grafik wird gezeigt wie die Übersichtstabelle aufgebaut wird.

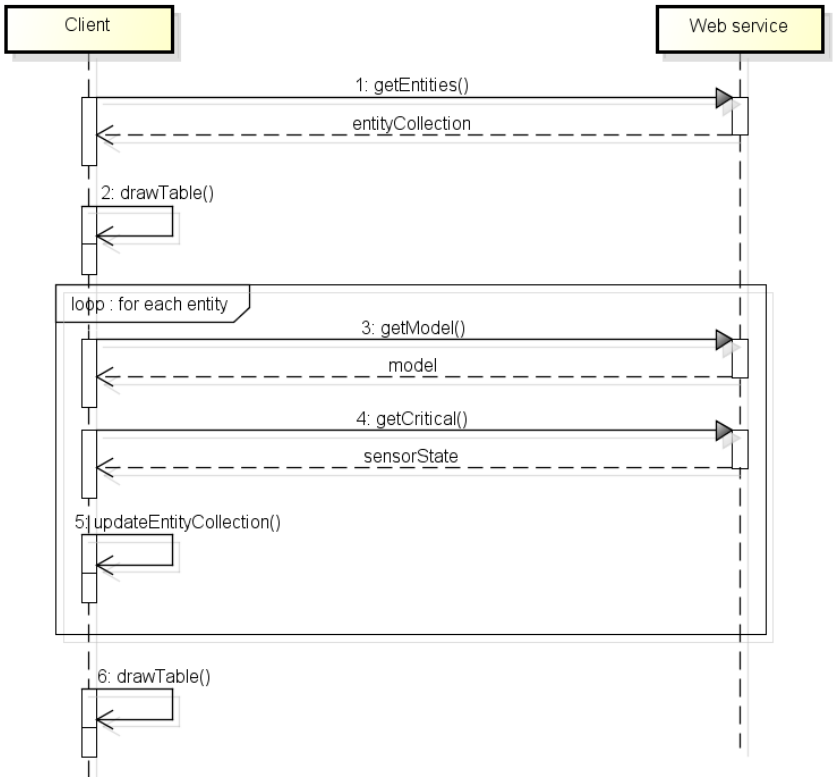


Abbildung 6: Sequenzdiagramm Critical Sensor

4.3 Schedule Collection

Die Schedule Collection wird benötigt um den Tatsächlichen Stromverbrauch eines Devices, Aggregation oder eines Objective anzuzeigen. Die Antwort des Webservice auf diese Anfrage war eine XML Datei welche für eine Backbone JS Collection geparst werden musste. Das Berechnen des Stromverbrauchs wurde somit gleich beim Parsen erledigt. Dafür wurde folgende Formel benutzt:

- $E_0 = 0$
- $E_n = P_{n-1} * (T_n - T_{n-1}) + E_{n-1}$

Wobei E_n für die aktuelle Energie steht P_{n-1} für Power (in Watt) und T_n, der Timestamp für den Zeitpunkt der Messung in Sekunden darstellt.

5 Ergebnis

Das Ergebnis ist ein funktionsfähiges Web Interface das den Anforderungen der IBM gerecht wird. Die gestellten Aufgaben konnten gelöst werden und werden bereits produktiv eingesetzt.

5.1 Stärken und Schwächen des Produktes

Das System erweist vor allem wegen der Backend Architektur kleine Schwächen in der Performance, diese werden jedoch vom Auftraggeber geduldet und zu einem späteren Zeitpunkt verbessert.
Durch die Verwendung von Backbone ist die Applikation gut strukturiert, einfach erweiterbar und gut skalierbar. Das GUI wurde vereinfacht und übersichtlicher gestaltet. Die durchdachte Planung und dank dem Erstellen von Wireframes konnte das GUI schlanker und besser strukturiert werden, so dass es einfacher und schneller zu bedienen ist und mit deutlich weniger Seiten auskommt.
Durch die Verwendung von Bootstrap konnte zudem ein schönes und einheitliches Design realisiert werden.
Die Verbesserung des GUI wurde durch Tests in einem

5.1.1 Critical State in der Übersicht

Durch die viele Anfragen an den Webservice benötigt das Laden der Übersichtstabelle 5-10 Sekunden. Deshalb werden die Geräte beim Laden der Seite bereits angezeigt und zu einem späteren Zeitpunkt mit der Critical Information versehen. Durch dieses Verfahren soll der Benutzer fortlaufend arbeiten können.

6 Zusammenfassung und Ausblick

6.1 Zeitauswertung

Für das gesamte Projekt wurden 475.25 Stunden aufgewendet, dies entspricht etwa dem Rahmen der für eine Studienarbeit aufgewendet werden sollte. Die Zeit ist gleichmässig unter den Teammitgliedern verteilt, 238.5 Stunden für Simon Brouwer und 236.75 Stunden für Giuseppe Aquino.

6.1.1 Iterationen

Project	Version	User	2013	Total time
IBM SmartPower			*475.25*	*475.25*
	Inception 1		*26.00*	*26.00*
		Simon Brouwer	**12.00**	**12.00**
		Giuseppe Aquino	**14.00**	**14.00**
	Elaboration 1		*51.50*	*51.50*
		Simon Brouwer	**25.00**	**25.00**
		Giuseppe Aquino	**26.50**	**26.50**
	Elaboration 2		*69.00*	*69.00*
		Simon Brouwer	**34.00**	**34.00**
		Giuseppe Aquino	**35.00**	**35.00**
	Construction 1		*107.00*	*107.00*
		Simon Brouwer	**53.50**	**53.50**
		Giuseppe Aquino	**53.50**	**53.50**
	Construction 2		*93.25*	*93.25*
		Simon Brouwer	**48.50**	**48.50**
		Giuseppe Aquino	**44.75**	**44.75**
	Construction 3		*87.50*	*87.50*
		Simon Brouwer	**44.50**	**44.50**
		Giuseppe Aquino	**43.00**	**43.00**
	Transition 1		*41.00*	*41.00*
		Simon Brouwer	**21.00**	**21.00**
		Giuseppe Aquino	**20.00**	**20.00**
Total time			**475.25**	**475.25**

Abbildung 7: Zeiterfassung Iterationen

6.1.2 Aktivitäten

Activity	2013-9	2013-10	2013-11	2013-12	Total time
Design		**5.00**			**5.00**
Development	**1.50**	**63.50**	**121.00**	**38.50**	**224.50**
Meeting	**6.50**	**21.00**	**47.50**	**8.00**	**83.00**
Documentation	**16.00**	**20.25**	**23.00**	**57.00**	**116.25**
Selfstudy	**2.00**	**37.00**	**7.50**		**46.50**
Total time	**26.00**	**146.75**	**199.00**	**103.50**	**475.25**

Abbildung 8: Zeiterfassung Aktivitäten

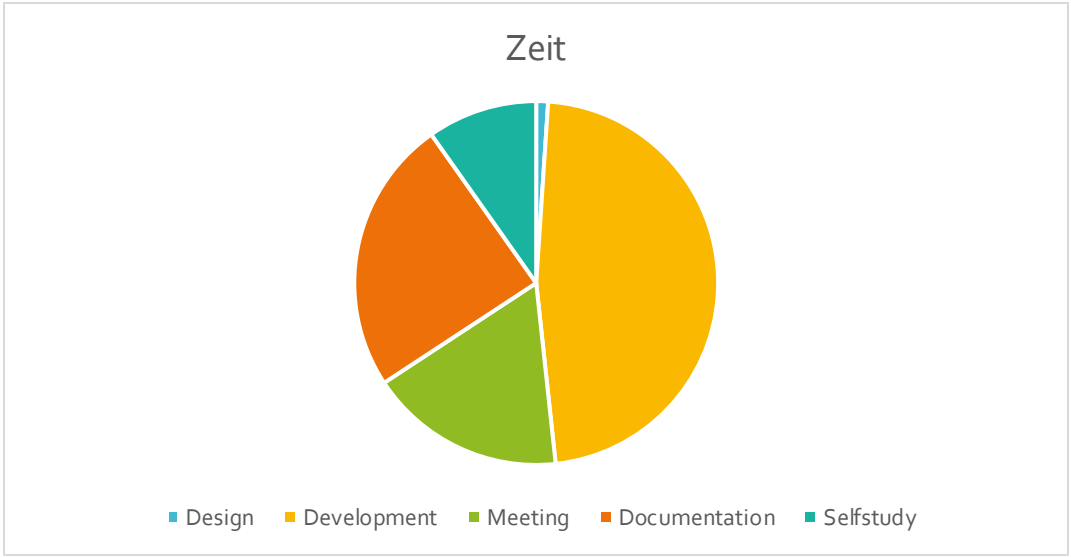


Abbildung 9: Zeiterfassung Aktivitäten Diagramm

6.2 Erfahrungsbericht & Lessons Learned

Simon Brouwer

Von Anfang an freute ich mich darauf die Chance zu bekommen mit einem grossen Industriepartner wie der IBM zusammenarbeiten zu können. Anfangs wussten wir noch nicht so genau was eigentlich unsere Aufgabe war. Dies klärte sich jedoch nach dem ersten Treffen bei der IBM. Dort lernten wir das Team kennen und bekamen einen Überblick über das gesamte Projekt. Wir hatten beide schon etwas Erfahrung mit JavaScript, jedoch nie in solch einem Ausmass, weshalb ich zu Anfang ziemlichen Respekt vor dem Projekt hatte. Nach der Entscheidung Backbone einzusetzen wurde das Bild schon einiges klarer. Als wir dann noch die Laptops mit Zugang zu den produktiven Web Services bekamen wussten wir genau was zu tun war und konnten so richtig los legen.
Die Zusammenarbeit mit dem Team der IBM verlief gut, bei Fragen konnten wir uns jeder Zeit melden und bekamen immer kompetente Antwort. Allgemein verlief die Kommunikation sehr angenehm, wir tauschten uns häufig aus um sicher zu stellen, dass alle genau wussten was gerade Läuft, so kam es nie zu einer grossen Überraschung.
Auch die Zusammenarbeit mit Giuseppe verlief gut. Wir waren beide begeistert vom Projekt und gaben deshalb viel Einsatz. Wir konnten die Arbeit gut untereinander Aufteilen und konnten einander bei Problemen immer gut helfen.
Die Zusammenarbeit mit dem Betreuer, Herrn Stozle, verlief gut und die Wöchentlichen Meetings halfen sehr dabei Probleme zu lösen und den Überblick über das Projekt zu behalten.

Da ich noch nie ein Projekt dieser Grösse mit JavaScript erarbeitet hatte war viel neu und ich konnte sehr viel neues lernen. Auch die Zusammenarbeit mit einem Industriepartner war für mich etwas Neues und ich habe viel daraus gelernt. Zum Beispiel wie wichtig gute Kommunikation für das gelinge eines Projektes ist. Wieder einmal war ich überrascht wie viel Zeit für die Dokumentation aufgewendet werden muss und es stellte sich heraus, dass ein gutes Zeitmanagement sehr von Vorteil war.

Alles in allem hat mir die Studienarbeit sehr gefallen und ich konnte viel dazu lernen.

Giuseppe Aquino

Als es darum ging eine Semesterarbeit auszusuchen, war für Simon und mich bereits klar, was das Themen gebiet sein sollte. Als begeisterte Webentwickler wollten wir unbedingt eine Arbeit welche sich mit dem Thema Webtechnologien befasste. Schnell war klar, dass wir die Arbeit bei der IBM schreiben durfte was mich persönlich noch mehr für die Arbeit begeisterte.
Die ersten 2 Wochen des Semester vergingen ohne dass wir klar wussten was genau zu tun war, weshalb wir nur mit der Arbeitsbeschreibung arbeiteten. Wir begannen die Dokumentation und ein Projektplan zu erstellen. Nach der ersten Sitzung mit der IBM, wussten wir bereits mehr über das Projekt und ich schreckte etwas zurück weil es sich grösser anhörte als ich gedacht hatte. Simon und ich hatten wie bereits erwähnt schon Erfahrung im Bereich Webentwicklung, jedoch hatte keiner von beiden, ein so grosses Webprojekt mit JavaScript entwickelt. Dies bedeutete für uns ein anderer Aufbau und andere Frameworks, alles was nötig ist um eine Enterprise Applikation zu realisieren. Weil das System relativ komplex und sich ausserdem noch in der Aufbauphase befindet, brauchte es etwas Zeit bis wir klar verstanden wie der Aufbau aussieht. Das Team der IBM war jedoch stets bereit uns Fragen zu beantworten und somit konnte immer schnell eine Lösung bzw. eine Antwort gefunden werden. Dank wöchentlichen Meetings mit der IBM, sei es über Telefon oder Persönlich konnten wir immer feststellen ob wir auf dem richtigen Weg waren. Die IBM gab uns immer kompetente und sachliche Feedbacks. Auch waren Sie offen für Änderungsvorschläge welche die

Zusammenarbeit zwischen Ihrem und unseren Teil der Software verbesserten. Im Allgemeinen habe ich die Zusammenarbeit mit der IBM sehr angenehm empfunden.
Ebenso die Arbeit mit meinem Kommilitonen Simon Brouwer war sehr angenehm. Wir konnten uns die Arbeit gut aufteilen und ergänzten uns sehr gut.

Weil es sich um eine fast reine JavaScript Arbeit handelte und ich noch nicht sehr erfahren in JavaScript war, konnte ich sehr von dieser Arbeit profitieren. Ich lernte Frameworks kennen wie z.B. Backbone JS und werde sie vermutlich auch im späteren Projekte wieder einsetzen.

Allgemein war die Zeit während der Studienarbeit sehr intensiv und voller neuer Eindrücke. An dieser Stelle will ich mich nochmals bei Herrn Prof. Dr. Stolze für die Zusammenarbeit bedanken und natürlich bei dem Industriebpartner, die IBM ZRL in Rüschlikon.

7 Literaturverzeichnis

Crockford, D. (2008). *JavaScript: The Good Parts.* O'Reilly.

Jackson, C., & Wang, H. (18. 12 2013). *ACM.* Von ACM Digital Library: http://dl.acm.org/citation.cfm?id=1242572.1242655&coll=DL&dl=ACM&CFID=390050573&CFTOKEN=63429584 abgerufen

Usman Shaukat Qurashi, Z. A. (18. 12 2013). *IEEE.* Von IEEE Xplore: http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=6375436 abgerufen

8 Abbildungsverzeichnis

Abbildung 1: Systemübersicht 7
Abbildung 2: Ordnerstruktur 8
Abbildung 3: Ordnerstruktur js Ordner 8
Abbildung 4: Sequenzdiagramm Proxy Webservice 9
Abbildung 5: Domain Model 10
Abbildung 6: Sequenzdiagramm Critical Sensor 11
Abbildung 7: Zeiterfassung Iterationen 13
Abbildung 8: Zeiterfassung Aktivitäten 13
Abbildung 9: Zeiterfassung Aktivitäten Diagramm 13

HSR Studienarbeit

IBM SmartPower

Projektplan

gaquino, sbrouwer
16.9.2013

Änderungsgeschichte

Datum	Version	Änderung	Autor
16.09.2013	1.0	Erstellung Projektplan	Team
02.10.2013	1.1	Risikomanagement in eigenes Dokument verschoben	Team
19.12.2013	1.2	Abschliessendes Gantt-Diagramm eingefügt und letzte Korrekturen	Team

Inhaltsverzeichnis

1 Einführung 3
1.1 Zweck 3
1.2 Gültigkeitsbereich 3
2 Projekt Übersicht 4
2.1 Zweck und Ziel 4
2.2 Datenverwaltung 4
2.3 Richtlinien für Code und Dokumentation 4
2.4 Lieferumfang 4
2.5 Annahmen und Einschränkungen 4
3 Projektorganisation 5
3.1 Team 5
3.1.1 Giuseppe Aquino 5
3.1.2 Simon Brouwer 5
3.2 Organisationsstruktur 5
3.3 Externe Schnittstellen 5
4 Management Abläufe 6
4.1 Zeitliche Planung 6
4.1.1 Phasen / Iterationen 6
4.1.2 Meilensteine 7
4.2 Besprechungen & Meetings 7
4.3 Planänderungen und Probleme 7
5 Arbeitspakete 8
6 Infrastruktur 9
6.1 Software 9
6.2 Kommunikation 9
7 Qualitätsmassnahmen 10
7.1 Dokumentation 10
7.2 Projektmanagement 10
7.2.1 Login Betreuer 10
7.3 Entwicklung 10
7.3.1 Vorgehen 10
7.3.2 Code Reviews 10
7.4 Tests 10
7.4.1 Systemtests 10

1 Einführung

1.1 Zweck

Dieses Dokument beinhaltet die detaillierte Beschreibung des Projektes, dessen Aufteilung in Phasen, deren Arbeitspakete und die Festlegung von Meilensteinen.

1.2 Gültigkeitsbereich

Dieses Dokument gilt als Grundlage des Projektes und beschränkt sich auf die Dauer der Studienarbeit im HS 2013/2014.

2 Projekt Übersicht

Im Projekt „SmartPower GUI" soll ein anpassungsfähiges User-Interface entwickelt werden, mit dem Regelleistungsanbieter für Einzelne und Gruppen von Lasten die Schaltvorgänge und zugehörigen relevanten Prozessparameter beobachten und nachvollziehen können. Für den Anbieter insgesamt wie auch für Gruppen oder einzelne Stromkonsumenten sollen Qualitätsmerkmale – auch in ihrem Zeitverlauf - einsehbar sein.

2.1 Zweck und Ziel

Das im Projekt zu entwickelnden User-Interface soll dem Regelleistungsanbieter, wie auch dem Technologielieferant helfen, die Akzeptanz und Attraktivität von „SmartPower" zu unterstützen.

2.2 Datenverwaltung

Die Daten für die Applikation werden direkt von den Webservices der IBM bezogen. Diese Webinterfaces werden ebenso während der Studienarbeit von der IBM an den neuen Bedürfnissen der Applikation angepasst.

2.3 Richtlinien für Code und Dokumentation

Damit die Dokumentation bzw. Code einheitlich sind werden für Dokumente Templates verwendet und für Code CodingConventions festgelegt, die von den Entwicklern als Guideline war genommen werden sollen.

2.4 Lieferumfang

- Benutzeroberfläche
- Dokumentation

2.5 Annahmen und Einschränkungen

Folgende Bedingungen gelten während des Projektes:

- Es dürfen für das Projekt keine virale OpenSource Libraries verwendet werden
    - Soll heissen keine Libraries die dazu führen, dass die Software als OpenSource gehandelt werden muss.

3 Projektorganisation

Das Projektteam besteht aus 2 sich gleichgestellten Mitgliedern.

3.1 Team

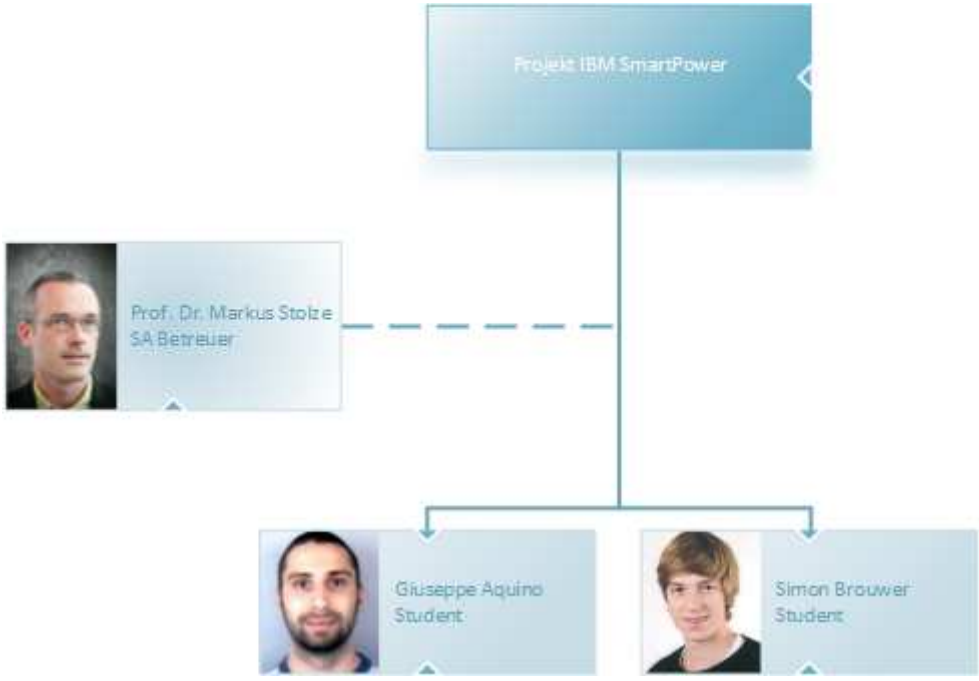
3.1.1 Giuseppe Aquino

Kenntnisse in:	Java, C++, PHP, SQL, JavaScript
Motivation:	Das Arbeiten mit Webtechnologien war schon immer ein grosses Hobby von mir. Das Vertiefen dieser Kenntnisse mit einem grossen Partner wie die IBM ist daher eine super Gelegenheit.

3.1.2 Simon Brouwer

Kenntnisse in:	Java, C++, PHP, SQL, JavaScript
Motivation:	Webentwicklung war schon seit Anfang meiner Ausbildung ein Hobby von mir, welches ich in meiner Freizeit und später bei der Arbeit, ausgeübt habe. Deshalb freue ich mich umso mehr nun mit einem grossen Industriepartner wie IBM zusammenzuarbeiten und meine Kenntnisse zu Vertiefen.

3.2 Organisationsstruktur



3.3 Externe Schnittstellen

Betreuer:	Prof. Dr. Markus Stolze
Externe Projekt Betreuer:	Dr. Douglas Dykeman
	Dr. Michael Nidd
	Dr. Olle Sundstroem

Es werden Regelmässig Meetings mit dem Betreuer geführt. Die Meetings sind auf jeden Freitag geplant. Traktanden der Meetings werden vorzüglich vor dem Meeting dem Dozenten gemailt. Die Meetings werden protokolliert und auf der Studienarbeit Dropbox abgelegt. Eine Kopie der Protokolle wird bis spätestens ein Tag nach der Sitzung an dem Betreuer, via Email gesendet. Ebenso finden wöchentliche Meetings mit dem Industriepartner IBM statt. Diese können als konventionelles Meeting behandelt werden, jedoch auch als Präsentation oder normaler Austausch.

4 Management Abläufe

Für das Projekt steht ein Semester à 14 Wochen zur Verfügung. Bis zur Deadline (20.12.2013) werden folgende Deliverables erwartet:

- Projektdefinition (Vision) mit dem Auftraggeber vereinbart (in Form von Use Cases und/oder Szenarien)
- Anforderungsanalyse inkl. Benutzerprofil, Usability Zielbestimmung und getestetes UI Desing (Papier Prototyp)
- Architektur Dokumentation
- Lauffähiger Code entsprechend der Anforderungsanalyse und der Architektur-Dokumentation (Sinnvoll im Code dokumentiert und mit Systemtests)
- Abschlussbericht und Abstract sowie weitere Dokumentationen entsprechend den Anforderungen der Abteilung Informatik der HSR.

Jeder Student leistet während des Semesters ca. 17 Stunden / Woche für die Arbeit. Insgesamt entsteht daraus ein Arbeitsaufwand von ca. 480 Stunden.

4.1 Zeitliche Planung

4.1.1 Phasen / Iterationen

4.1.1.1 Inception (16.09.2013 – 29.09.2013)

Während der Inception Phase wird mit dem Partner IBM die Vision diskutiert und allfälligen Fragen geklärt. Das Team erhält einen ersten, konkreten Einblick in das Projekt. Es wird ein Projektplan erstellt sowie die Rahmenbedingungen definiert. Das Team beginnt mit der Erarbeitung der Requirements. Ausserdem wird die Entwicklungsumgebung eingerichtet, sowie verschiedene Management Tools.

4.1.1.2 Elaboration I (30.09.2013 – 13.10.2013)

In der ersten Elaborationsphase werden die Requirements genau ausgearbeitet und verfeinert. Zusätzlich wird versucht die Risiken zu erkennen und einzugrenzen oder zu entfernen. Die Evaluation eines GUIs wird ebenfalls vorgenommen mittels Paper-Prototypen. Die Datenbank bzw. der Zugriff auf die Daten für die Anwendung wird getestet.

4.1.1.3 Elaboration II (14.10.2013 – 27.10.2013)

Die während der Elaborationsphase erarbeiteten Requirements werden nochmals verfeinert. Es wird ein Architektur Dokument mit den technischen Rahmenbedingungen erstellt. Für einen sicheren Start in der Construction werden allenfalls Prototypes erstellt, sowie verschiedene POCs (Proof of Concept).

4.1.1.4 Construction I (28.10.2013 – 10.11.2013)

Mit fortlaufenden Code-Reviews beginnt die Implementierung der Lösung. Dabei liegt der Fokus auf den wesentlichen Bestandteilen der Softwarelösung.

4.1.1.5 Construction II (11.11.2013 – 24.11.2013)

Software wird mit zusätzlichen Features erweitert. Microtests (falls sinnvoll) inklusive Durchführung von Systemtests und Bug-Fixing aus vorangegangenem Testing. In dieser Phase sollte eine Definitive Architektur zustande kommen und ein Finales UI vorhanden sein.

4.1.1.6 *Construction III (25.11.2013 – 08.12.2013)*

Systemtests sowie Microtests erweitern und Bug-Fixing. Systemtests erfolgen in dieser Phase etwas früher damit noch genug Zeit vorhanden ist um diese zu fixen.

4.1.1.7 *Transition (09.12.2013 – 20.12.2013)*

In der letzten Phase wird ein definitives Release erstellt. Die Schlusspräsentation wird ebenfalls während dieser Phase vorbereitet. Die Dokumente welche für die Abgabe benötigt werden, sowie die Abgabe CD werden während dieser Phase fertig gestellt.

4.1.2 Meilensteine

Meilenstein	**Resultate**	**Termin**
MS1 Projektplan	Projektplan Coding-Conventions	27.09.2013
MS2 Anforderungen und Analyse	Requirements Dokumentation Paperprototypes	11.10.2013
MS3 Ende Elaboration	Grobe Architektur-Dokumentation Architekturprototyp UI Prototyp	25.10.2013
MS4 Architektur/Design	Detaillierte Architektur-Dokumentation Finales UI	22.11.2013
MS5 Work Done	Final Release Kurzfassung der Arbeit und Ao-Poster	13.12.2013

4.2 Besprechungen & Meetings

Zusätzliche Meetings innerhalb des Teams: Jeden Mittwoch findet jeweils ein kurzes Stand-Up Meeting statt (kann nach Bedarf mehrere geben). Im Stand-Up wird jeweils der aktuelle Stand der Dinge besprochen. Zeit Slot ist ca. 5-10 Minuten/Meeting.

Themen für das Stand-Up Meeting:

- Was habe ich gemacht?
- Wo habe ich zurzeit Probleme?
- Was steht noch an?
- …Wichtige Informationen austauschen

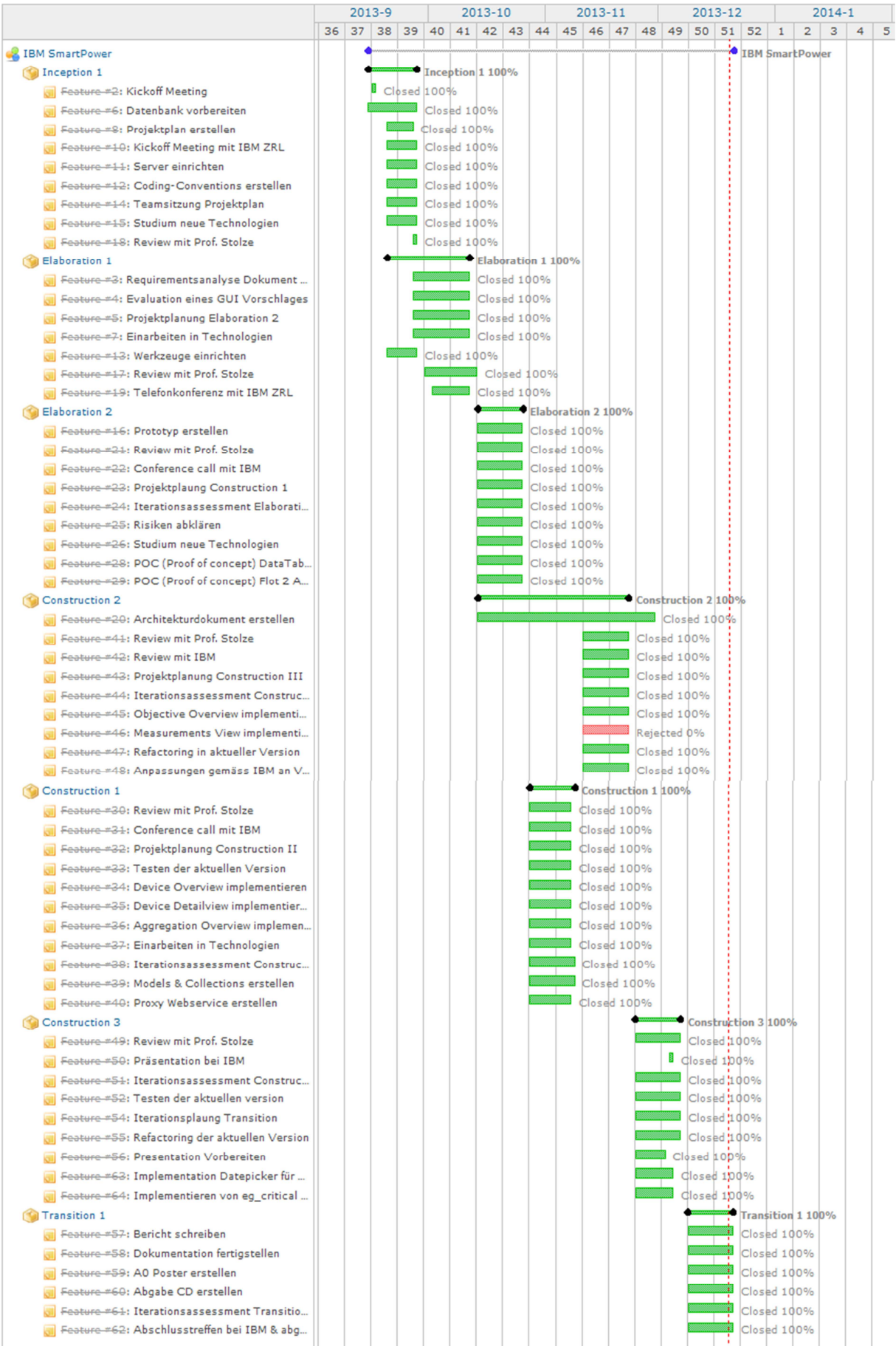
Kann ein Problem nicht während eines Stand-Up Meetings gelöst werden, so wird ein Individueller Termin gemacht an dem man gemeinsam das Problem in Angriff nehmen kann.

4.3 Planänderungen und Probleme

Bei einer Planänderung wird folgendermassen vorgegangen:

- Krisensitzung wird einberufen
- Problemstellung besprechen
- Mögliche Lösungsvorschläge werden ausdiskutiert (evtl. Betreuer mit einbeziehen)
- Einigung auf einen Lösungsvorschlag
- Änderung umsetzen

5 Arbeitspakete



6 Infrastruktur

Für das gemeinsame Arbeiten sowie für Besprechungen werden die verfügbaren Räume der HSR genutzt. Die Projektmitglieder können jeweils am eigenen Notebook oder an den von der HSR zur Verfügung gestellten PCs arbeiten. Die Verwaltung der Arbeitspakete erfolgt hierbei über Redmine welches auf dem HSR Virtual Server installiert wurde. Die Versionsverwaltung für den produzierten Code wird von der HSR zur Verfügung gestellt (HSR GIT).

6.1 Software

Während dem Projekt „IBM Smart Power" wird mit folgender Software gearbeitet:

- IDE (Eclipse für Web und Java (falls nötig))
- Redmine
- GIT
- Skype
- MS Office

6.2 Kommunikation

Die Kommunikation zwischen den Projektmitarbeiter wird bevorzugt mündlich erfolgen. Sollte dies nicht möglich sein stehen folgende Mittel zur Verfügung:

- E-Mail
- Sykpe
- Sametime

7 Qualitätsmassnahmen

7.1 Dokumentation

Die Dokumentation des Projektes ist stets nachzuführen und wird bei jeder Teamsitzung kurz besprochen. Bei Gelegenheit sollte die Projektdokumentation auch durch den Betreuer reviewed werden. Ablageort für die Dokumentation ist das Dropbox-Share der Studienarbeit.

7.2 Projektmanagement

Das Projektmanagement wird mit Redmine erfolgen. Alle Arbeitspakete werden erstellt und in die einzelnen Phasen eingeteilt. Verantwortlichkeiten werden mündlich besprochen.

7.2.1 Login Betreuer

User: mstolze
Passwort: mstolze123

7.3 Entwicklung

Für die Verwaltung und Versionierung vom Code wird GIT eingesetzt. Anhand von Coding-Conventions wird die Codequalität garantiert. Es sollten alle Teilnehmer der SA mit demselben Entwicklungstool arbeiten.

7.3.1 Vorgehen

Anhand der zugeteilten Arbeitspakete, kann jedes Teammitglied seine Aufgaben erledigen. Der implementierte Code wird zunächst nur lokal commited und erst in einem lauffähigen Zustand auf das GIT gepushed.

7.3.2 Code Reviews

Einmal in der Woche wird der neu produzierte Code reviewed und allenfalls angepasst. Dies hat den Vorteil, dass jeder den Code kennt und bei Ausfall auch übernehmen kann.

Möglicher Ablauf eines Tickets:

- Feature wird von der Gruppe für nötig und sinnvoll erklärt = Ticket wird erstellt
- Implementierung des Features wird in Angriff genommen
- Feature ist fertig gestellt
- Review für Feature wird gemacht

Durch diesen Ablauf wird ebenso die Codequalität fortlaufend überprüft.

7.4 Tests

7.4.1 Systemtests

Systemtests werden fortlaufend gemacht. Gegen Ende der Construction – Iteration III wird ebenfalls ein summativer Systemtest durchgeführt. Dieser Test wird hauptsächlich Teamintern erfolgen, da so die Fehler gleich behoben werden können oder bei grösseren Problemen gleich nach einer Lösung gesucht werden kann.

HSR Studienarbeit

IBM SmartPower

Codingconvention

gaquino, sbrouwer

23.9.2013

Änderungsgeschichte

Datum	Version	Änderung	Autor
23.09.2013	1.0	Erstellung Codingconventions	Team

Inhaltsverzeichnis

1. Coding Conventions 3
1.1 Allgemein 3
1.2 CSS 3
1.3 Übersicht 3
2. Comments 4
3. User Interface 4

1. Coding Conventions

1.1 Allgemein

- Aussagekräftige Namen verwenden.
- Wir verwenden den Standard Eclipse Coding-Style. Vor dem einchecken mit Tastenkombination ctrl + shift + f den Code formatieren!
- Englische Bezeichnungen verwenden (für Code und Kommentar).
- Keine unnötigen Abkürzungen.
- Nicht Code auf Zeile verschachteln
    - Nicht: function A() {get {return 5;}}

1.2 CSS

Bei häufig wiederverwendeten Elementen dem Element ein „class" Attribut definieren, z.B. .
Bei einmalig verwendeten oder speziellen Elementen eine „id" Attribut definieren, z.B. .

1.3 Übersicht

Code	Beispiel	Bemerkung
Lokale Variablen	parameter	Name ausschreiben
Membervariable	itemCount	Name ausschreiben
Methoden	readWordsFromFile(...)	Funktionsname ausschreiben
Ressource-Files	planGraph	Ausgeschrieben Optional bei Bildern: Grösse mitangeben bspw. mit 256x256

HTML GUI Elemente	Beispiel	Bemerkung
Textfield (input type="text")	name	%Beschreibung%
Textarea	comment	%Beschreibung%
Button	ok/abort	%Beschreibung%
Checkbox (input type="checkbox")	something	%Beschreibung%
Radiobutton (input type="radio")	something	%Beschreibung%

File	Beispiel	Bemerkung
Views	index.html	%Beschreibung%.html
Control	skript.js	%Beschreibung%.js

2. Comments

- Nur sinnvolle Kommentare (Zusatzinfos/Gedanken zum Code) nicht à la „increment i"
- Kommentar der keinen zusatznutzen bringt lieber weglassen
- Kommentar auf Englisch
- Kommentar beschreibt die darunterliegende Zeile. Kommentar nicht auf gleicher Zeile wie Code

3. User Interface

- GUI für aktuelle Browser optimiert
- Keine Virale Libraries verwenden die veranlassen dass die Software als Opensource gehandelt werden muss.

HSR Studienarbeit

IBM SmartPower

Risikomanagement Dokumentation

HSR

25.9.2013

Inhaltsverzeichnis

1 Risikomanagement 2
1.1 Risiken Inception 1 2
1.2 Risiken Elaboration 1 3
1.3 Risiken Elaboration 2 4
1.4 Risiken Construction 1 5
1.5 Risiken Construction 2 6
1.6 Risiken Construction 3 7
1.6.1 Risiko Behandlung 7
2 Umgang mit Risiken 8

1 Risikomanagement

1.1 Risiken Inception 1

Folgend die Risiken die während der Inception 1 analysiert und erkannt wurden.

ID	Titel	Beschreibung	Schaden	Eintrittswarscheinlichkeit	Vorbeugung
R01	Performance Probleme (Netzwerk)	Webservice liefert Resultat sehr langsam	Klein	Klein	Fortlaufend testen
R02	Performance Probleme (Applikation)	User Interface ist nicht flüssig	Klein	Klein	Fortlaufend testen
R03	Datenverlust	Sourcecode bzw. Konfiguration gehen verloren	Sehr gross	Klein	Backups
R04	Kompatibilitätsprobleme	Alte Browser werden nicht unterstützt	Mittel	Mittel	Fortlaufend auf allen Browsern Testen, Library-Abhängigkeiten überprüfen
R05	Projektumfang unterschätzt	Projekt erweist sich als grösser, zeitaufwendiger als erwartet	Mittel	Mittel	Genügend Zeitreserven einplanen
R06	Änderung der Voraussetzungen	Zusätzliche Voraussetzungen entstehen um Projekt zu erfüllen	Klein – Mittel	Klein – Mittel	Regelmässige Sitzungen und Zeitreserven einplanen
R07	Webservice	Simulierter Webservice verhält sich nicht wie tatsächlicher Webservice	Gross	Klein	Gut in IBM Technologien einarbeiten
R08	Usability	Tests bzw. UI Bewertung fallen beim Kunden schlecht aus	Gross	Klein	Paper Prototypen sowie oft mit Kunden absprechen
R09	Informationen fehlen	Relevante Information für die Applikation fehlen, evtl. muss Webservice erweitert werden	Mittel	Mittel	Abklären ob Information relevant für Kunde ist.

1.2 Risiken Elaboration 1

Folgend die Risiken die während der Elaboration 1 analysiert und erkannt wurden.

ID	Titel	Beschreibung	Schaden	Eintrittswarscheinlichkeit	Vorbeugung
R01	Performance Probleme (Netzwerk)	Webservice liefert Resultat sehr langsam	Klein	Klein	Fortlaufend testen
R02	Performance Probleme (Applikation)	User Interface ist nicht flüssig	Klein	Klein	Fortlaufend testen
R03	Datenverlust	Sourcecode bzw. Konfiguration gehen verloren	Sehr gross	Klein	Backups
R04	Kompatibilitätsprobleme	Alte Browser werden nicht unterstützt	Mittel	Mittel	Fortlaufend auf allen Browsern Testen, Library-Abhängigkeiten überprüfen
R05	Projektumfang unterschätzt	Projekt erweist sich als grösser, zeitaufwendiger als erwartet	Mittel	Mittel	Genügend Zeitreserven einplanen
R06	Änderung der Voraussetzungen	Zusätzliche Voraussetzungen entstehen um Projekt zu erfüllen	Klein – Mittel	Klein – Mittel	Regelmässige Sitzungen und Zeitreserven einplanen
R07	Webservice	Simulierter Webservice verhält sich nicht wie tatsächlicher Webservice	Gross	Klein	Gut in IBM Technologien einarbeiten
R08	Usability	Tests bzw. UI Bewertung fallen beim Kunden schlecht aus	Gross	Klein	Paper Prototypen sowie oft mit Kunden absprechen
R09	Informationen fehlen	Relevante Information für die Applikation fehlen, evtl. muss Webservice erweitert werden	Mittel	Mittel	Abklären ob Information relevant für Kunde ist.
R10	DataTables Framework erfüllt die Kriterien nicht	DataTables Framework unterstützt die gewünschten Funktionalitäten nicht	Mittel	Mittel	Gut in Framework einarbeiten
R11	JsFlot Framework erfüllt die Kriterien nicht	JsFlot Framework unterstützt die gewünschten Funktionalitäten nicht	Mittel	Klein	Gut in Framework einarbeiten

1.3 Risiken Elaboration 2

Folgend die Risiken die während der Elaboration 2 analysiert und erkannt wurden.

ID	Titel	Beschreibung	Schaden	Eintrittswarscheinlichkeit	Vorbeugung
R01	Performance Probleme (Netzwerk)	Webservice liefert Resultat sehr langsam	Klein	Klein	Fortlaufend testen
R02	Performance Probleme (Applikation)	User Interface ist nicht flüssig	Klein	Klein	Fortlaufend testen
R03	Datenverlust	Sourcecode bzw. Konfiguration gehen verloren	Sehr gross	Klein	Backups
R04	Kompatibilitätsprobleme	Alte Browser werden nicht unterstützt	Mittel	Mittel	Fortlaufend auf allen Browsern Testen, Library-Abhängigkeiten überprüfen
R05	Projektumfang unterschätzt	Projekt erweist sich als grösser, zeitaufwendiger als erwartet	Mittel	Mittel	Genügend Zeitreserven einplanen
R06	Änderung der Voraussetzungen	Zusätzliche Voraussetzungen entstehen um Projekt zu erfüllen	Klein – Mittel	Klein – Mittel	Regelmässige Sitzungen und Zeitreserven einplanen
R07	Webservice	Simulierter Webservice verhält sich nicht wie tatsächlicher Webservice	Gross	Klein	Gut in IBM Technologien einarbeiten
R08	Usability	Tests bzw. UI Bewertung fallen beim Kunden schlecht aus	Gross	Klein	Paper Prototypen sowie oft mit Kunden absprechen
R09	Informationen fehlen	Relevante Information für die Applikation fehlen, evtl. muss Webservice erweitert werden	Mittel	Klein	Abklären ob Information relevant für Kunde ist.
R10	DataTables Framework erfüllt die Kriterien nicht	DataTables Framework unterstützt die gewünschten Funktionalitäten nicht	Mittel	Klein	Gut in Framework einarbeiten
R11	JsFlot Framework erfüllt die Kriterien nicht	JsFlot Framework unterstützt die gewünschten Funktionalitäten nicht	Mittel	Klein	Gut in Framework einarbeiten

- Risiko konnte ausgeschlossen werden

1.4 Risiken Construction 1

ID	Titel	Beschreibung	Schaden	Eintrittswarscheinlichkeit	Vorbeugung
R01	Performance Probleme (Netzwerk)	Webservice liefert Resultat sehr langsam	Klein	Klein	Fortlaufend testen
R02	Performance Probleme (Applikation)	User Interface ist nicht flüssig	Klein	Klein	Fortlaufend testen
R03	Datenverlust	Sourcecode bzw. Konfiguration gehen verloren	Sehr gross	Klein	Backups
R04	Kompatibilitätsprobleme	Alte Browser werden nicht unterstützt	Mittel	Klein	Fortlaufend auf allen Browsern Testen, Library-Abhängigkeiten überprüfen
R05	Projektumfang unterschätzt	Projekt erweist sich als grösser, zeitaufwendiger als erwartet	Mittel	Klein	Genügend Zeitreserven einplanen
R06	Änderung der Voraussetzungen	Zusätzliche Voraussetzungen entstehen um Projekt zu erfüllen	Klein – Mittel	Klein – Mittel	Regelmässige Sitzungen und Zeitreserven einplanen
R07	Webservice	Simulierter Webservice verhält sich nicht wie tatsächlicher Webservice	Gross	Klein	Gut in IBM Technologien einarbeiten
R08	Usability	Tests bzw. UI Bewertung fallen beim Kunden schlecht aus	Gross	Klein	Paper Prototypen sowie oft mit Kunden absprechen
R09	Informationen fehlen	Relevante Information für die Applikation fehlen, evtl. muss Webservice erweitert werden	Mittel	Klein	Abklären ob Information relevant für Kunde ist.
R10	DataTables Framework erfüllt die Kriterien nicht	DataTables Framework unterstützt die gewünschten Funktionalitäten nicht	Mittel	Klein	Gut in Framework einarbeiten
R11	JsFlot Framework erfüllt die Kriterien nicht	JsFlot Framework unterstützt die gewünschten Funktionalitäten nicht	Mittel	Klein	Gut in Framework einarbeiten

1.5 Risiken Construction 2

ID	Titel	Beschreibung	Schaden	Eintrittswarscheinlichkeit	Vorbeugung
R01	Performance Probleme (Netzwerk)	Webservice liefert Resultat sehr langsam	Klein	Klein	Fortlaufend testen
R02	Performance Probleme (Applikation)	User Interface ist nicht flüssig	Klein	Klein	Fortlaufend testen
R03	Datenverlust	Sourcecode bzw. Konfiguration gehen verloren	Sehr gross	Klein	Backups
R04	Kompatibilitätsprobleme	Alte Browser werden nicht unterstützt	Mittel	Klein	Fortlaufend auf allen Browsern Testen, Library-Abhängigkeiten überprüfen
R05	Projektumfang unterschätzt	Projekt erweist sich als grösser, zeitaufwendiger als erwartet	Mittel	Klein	Genügend Zeitreserven einplanen
R06	Änderung der Voraussetzungen	Zusätzliche Voraussetzungen entstehen um Projekt zu erfüllen	Klein – Mittel	Klein – Mittel	Regelmässige Sitzungen und Zeitreserven einplanen
R07	Webservice	Simulierter Webservice verhält sich nicht wie tatsächlicher Webservice	Gross	Klein	Gut in IBM Technologien einarbeiten
R08	Usability	Tests bzw. UI Bewertung fallen beim Kunden schlecht aus	Gross	Klein	Paper Prototypen sowie oft mit Kunden absprechen
R09	Informationen fehlen	Relevante Information für die Applikation fehlen, evtl. muss Webservice erweitert werden	Mittel	Klein	Abklären ob Information relevant für Kunde ist.
R10	DataTables Framework erfüllt die Kriterien nicht	DataTables Framework unterstützt die gewünschten Funktionalitäten nicht	Mittel	Klein	Gut in Framework einarbeiten
R11	JsFlot Framework erfüllt die Kriterien nicht	JsFlot Framework unterstützt die gewünschten Funktionalitäten nicht	Mittel	Klein	Gut in Framework einarbeiten

- Risiko konnte ausgeschlossen werden

1.6 Risiken Construction 3

ID	Titel	Beschreibung	Schaden	Eintrittswarscheinlichkeit	Vorbeugung
R01	Performance Probleme (Netzwerk)	Webservice liefert Resultat sehr langsam	Klein	Klein	Fortlaufend testen
R02	Performance Probleme (Applikation)	User Interface ist nicht flüssig	Klein	Klein	Fortlaufend testen
R03	Datenverlust	Sourcecode bzw. Konfiguration gehen verloren	Sehr gross	Klein	Backups
R04	Kompatibilitätsprobleme	Alte Browser werden nicht unterstützt	Mittel	Klein	Fortlaufend auf allen Browsern Testen, Library-Abhängigkeiten überprüfen
R05	Projektumfang unterschätzt	Projekt erweist sich als grösser, zeitaufwendiger als erwartet	Mittel	Klein	Genügend Zeitreserven einplanen
R06	Änderung der Voraussetzungen	Zusätzliche Voraussetzungen entstehen um Projekt zu erfüllen	Klein – Mittel	Klein – Mittel	Regelmässige Sitzungen und Zeitreserven einplanen
R07	Webservice	Simulierter Webservice verhält sich nicht wie tatsächlicher Webservice	Gross	Klein	Gut in IBM Technologien einarbeiten
R08	Usability	Tests bzw. UI Bewertung fallen beim Kunden schlecht aus	Gross	Klein	Paper Prototypen sowie oft mit Kunden absprechen
R09	Informationen fehlen	Relevante Information für die Applikation fehlen, evtl. muss Webservice erweitert werden	Mittel	Klein	Abklären ob Information relevant für Kunde ist.
R10	DataTables Framework erfüllt die Kriterien nicht	DataTables Framework unterstützt die gewünschten Funktionalitäten nicht	Mittel	Klein	Gut in Framework einarbeiten
R11	JsFlot Framework erfüllt die Kriterien nicht	JsFlot Framework unterstützt die gewünschten Funktionalitäten nicht	Mittel	Klein	Gut in Framework einarbeiten

- Risiko konnte ausgeschlossen werden
- Risiko ist eingetreten

1.6.1 Risiko Behandlung

Einer der geplanten Sensoren wurde auf dem Backend anders implementiert als erwartet, was zur Folge hatte, dass mehr Anfragen an den Webservice gestellt werden mussten als geplant. Dies führte zu längeren Ladezeiten der Seite.

Mit der IBM wurde eine Lösung ausgehandelt. Der entstandene Schaden war minimal.

2 Umgang mit Risiken

Bei Eintritt eines der oben genannten Risiken, wird dies zunächst Teamintern behandelt. Sollte keine brauchbare Lösung zustande kommen, wird umgehend der Betreuer oder die entsprechende Person kontaktiert.

HSR Studienarbeit

IBM SmartPower

Iterationsassessment

gaquino, sbrouwer
16.9.2013

Änderungsgeschichte

Datum	Version	Änderung	Autor
16.09.2013	1.0	Erstellung Iterationsassessment	Team
24.10.2013	1.1	Anpassung Elaboration 2	Team
28.10.2013	1.2	Iterationsassessment für Elab 2 hinzugefügt	gaquino
08.11.2013	1.3	Iterationsassessment für Construction 1 hinzugefügt	sbrouwer
22.11.2013	1.4	Iterationsassessment für Construction 2 hinzugefügt	Team
06.12.2013	1.5	Iterationsassessment für Construction 3 hinzugefügt	Team
19.12.2013	1.6	Iterationsassessment für Transition hinzugefügt	Team

Inhaltsverzeichnis

1 Einführung 3
1.1 Zweck 3
1.2 Referenzen 3
2 Inception 1 4
2.1 Iterationsplanung 4
2.2 Iterationsassessment 4
2.3 Fazit 4
3 Elaboration 1 5
3.1 Iterationsplanung 5
3.2 Iterationsassessment 5
3.3 Fazit 5
4 Elaboration 2 6
4.1 Iterationsplanung 6
4.2 Iterationsassessment 6
4.3 Fazit 6
5 Construction 1 7
5.1 Iterationsplanung 7
5.2 Iterationsassessment 7
5.3 Fazit 7
6 Construction 2 8
6.1 Iterationsplanung 8
6.2 Iterationsassessment 8
6.3 Fazit 8
7 Construction 3 9
7.1 Iterationsplanung 9
7.2 Iterationsassessment 9
7.3 Fazit 9
8 Transition 1 10
8.1 Iterationsplanung 10
8.2 Iterationsassessment 10
8.3 Fazit 10

1 Einführung

1.1 Zweck

Dieses Dokument beinhaltet die detaillierte Beschreibung des Projektes in den verschiedenen Projektphasen und deren Arbeitspakete.

1.2 Referenzen

- Projektplaln.pdf

2 Inception 1

2.1 Iterationsplanung

Ticket
Kickoff Meeting
Datenbank vorbereiten
Projektplan erstellen
Kickoff Meeting mit IBM ZRL
Server einrichten
Coding Conventions erstellen
Teamsitzung Projektplan
Studium neue Technologien
Review mit Prof. Stolze

2.2 Iterationsassessment

Ticket	Fortschritt
Kickoff Meeting	100%
Datenbank vorbereiten	100%
Projektplan erstellen	100%
Kickoff Meeting mit IBM ZRL	100%
Server einrichten	100%
Coding Conventions erstellen	100%
Teamsitzung Projektplan	100%
Studium neue Technologien	100%
Review mit Prof. Stolze	100%

2.3 Fazit

Die Inception war vor allem fürs einlesen ins Projekt gedacht und zu erfahren um was es genau geht. Die Iteration lief wie geplant und es konnte alles eingerichtet werden, was für die Arbeit gebraucht wurde. Teilweise wurden einige Tools eingerichtet die wie sich später rausstellte nicht notwendig waren.

3 Elaboration 1

3.1 Iterationsplanung

Ticket
Review mit Prof. Stolze
Einarbeiten in Technologien
Projektplanung Elaboration 2
Requirementanalyse Dokument erstellen
Evaluation eines GUI Vorschlages
Werkzeuge einrichten
Telefonkonferenz mit IBM ZRL

3.2 Iterationsassessment

Ticket	Fortschritt
Review mit Prof. Stolze	100%
Einarbeiten in Technologien	100%
Projektplanung Elaboration 2	100%
Requirementanalyse Dokument erstellen	100%
Evaluation eines GUI Vorschlages	100%
Werkzeuge einrichten	100%
Telefonkonferenz mit IBM ZRL	100%

3.3 Fazit

Die Elaboration 1 Phase lief wie geplant. Wir haben uns einen Überblick über das Projekt verschafft und es kann nun plangemäss in die Elaboration 2 starten und mit den Prototypen beginnen.

4 Elaboration 2

4.1 Iterationsplanung

Ticket
Prototyp erstellen
Architekturdokument erstellen
Review mit Prof. Stolze
Conference call mit IBM
Projektplaung Construction 1
Iterationsassessment Elaboration 2
Risiken abklären
Studium neue Technologien
POC (Proof of concept) DataTables
POC(Proof of concept) Flot 2 Achsen in einem Graph

4.2 Iterationsassessment

Ticket	Fortschritt
Prototyp erstellen	100%
Architekturdokument erstellen	10%
Review mit Prof. Stolze	100%
Conference call mit IBM	100%
Projektplaung Construction 1	100%
Iterationsassessment Elaboration 2	100%
Risiken abklären	100%
Studium neue Technologien	100%
POC (Proof of concept) DataTables	100%
POC(Proof of concept) Flot 2 Achsen in einem Graph	100%

4.3 Fazit

Die Iteration ist gelungen und wichtige Technische Anforderungen konnten geklärt werden. Leider war sich das Team bis Ende Iteration nicht bewusst wie die Architektur aussehen soll, da noch wenig Erfahrung in Bereich JS Applikationen vorhanden war. Das Team entschloss sich dennoch für den Einsatz eines Application Framework für JS, weshalb Zeit für die Einarbeitung investiert werden musste. Das Arbeitspaket „Architekturdokument erstellen" musste aus diesem Grund auf die Construction I verschoben werden.

5 Construction 1

5.1 Iterationsplanung

Ticket
Review mit Prof. Stolze
Conference call mit IBM
Projektplanung Construction II
Testen der aktuellen Version
Device Overview implementieren
Device Detailview implementieren
Aggregation Overview implementieren
Einarbeiten in Technologien
Architekturdokument erstellen

5.2 Iterationsassessment

Ticket	Fortschritt
Review mit Prof. Stolze	100%
Conference call mit IBM	100%
Projektplanung Construction II	100%
Testen der aktuellen Version	100%
Device Overview implementieren	100%
Device Detailview implementieren	80%
Aggregation Overview implementieren	100%
Einarbeiten in Technologien	100%
Architekturdokument erstellen	80%

5.3 Fazit

Die Iteration verlief gut. Es konnten die ersten wichtigen Bausteine der Applikation implementiert werden. Durch die Änderungen der Anforderungen musste die Architektur für gewisse Teile der Applikation angepasst werden und es konnte deshalb das Architekturdokument noch nicht fertiggestellt werden. Auch die Detailview konnte auf Grund von Änderungen an den Anforderungen noch nicht ganz fertiggestellt werden. Zudem wurde auf Seiten der IBM die Datenbank auf welche der FERN Webservice zugreift migriert, was dazu führte das der Webservice über mehrere Tage nur sporadisch oder gar nicht erreichbar war, weshalb zu diesen Zeiten nicht an der Applikation gearbeitet werden konnte, da das Team auf die Daten des Webservice angewiesen ist.

6 Construction 2

6.1 Iterationsplanung

Ticket
Anpassungen gemäss IBM an Views vornehmen
Refactoring in aktueller Version
Measurements View implementieren
Objective Overview implementieren
Iterationsassessment Construction II
Projektplanung Construction III
Review mit IBM
Review mit Prof. Stolze
Device Detailview implementieren
Architekturdokument erstellen

6.2 Iterationsassessment

Ticket	Fortschritt
Anpassungen gemäss IBM an Views vornehmen	100%
Refactoring in aktueller Version	100%
Measurements View implementieren	0%
Objective Overview implementieren	100%
Iterationsassessment Construction II	100%
Projektplanung Construction III	100%
Review mit IBM	100%
Review mit Prof. Stolze	100%
Device Detailview implementieren	100%
Architekturdokument erstellen	80%

6.3 Fazit

Die Construction 2 verlief wie geplant. Es stellte sich heraus, dass durch die Anpassungen, welche von der IMB gewünscht wurden, die Measurements View überflüssig machte, weshalb diese gestrichen wurde. Da auch in dieser Iteration auf Grund des geplanten Refactorings noch viele Änderungen an der Architektur gemacht wurden wurde das Architekturdokument noch nicht fertiggestellt. Nun steht die Architektur jedoch fest und das Architekturdokument wird in der Construction 3 fertiggestellt werden können.

7 Construction 3

7.1 Iterationsplanung

Ticket
Review mit Prof. Stolze
Präsentation bei IBM
Iterationsassessment Construction 3
Testen der aktuellen version
Iterationsplaung Transition
Refactoring der aktuellen Version
Präsentation Vorbereiten
Implementation Datepicker für Time Series
Implementieren von eg_critical Sensor in Overview & Detail View

7.2 Iterationsassessment

Ticket	Fortschritt
Review mit Prof. Stolze	100%
Präsentation bei IBM	100%
Iterationsassessment Construction 3	100%
Testen der aktuellen version	100%
Iterationsplaung Transition	100%
Refactoring der aktuellen Version	100%
Präsentation Vorbereiten	100%
Implementation Datepicker für Time Series	100%
Implementeiren von eg_critical Sensor in Overview & Detail View	100%

7.3 Fazit

Die Construction 3 Iterationsphase verlief gut. Die Implementierung der Applikation konnte erfolgreich abgeschlossen werden. Da der „eg_critical" Sensor erst ab dieser Iteration zur Verfügung stand und nicht so implementiert war wie zu Anfang angenommen führte dies zu etwas mehr Aufwand als geplant, konnte jedoch termingerecht fertiggestellt werden. Die Arbeit wurde dem IBM Team präsentiert und an der IBM in Betrieb genommen. Somit ist die Applikation von der IBM erfolgreich abgenommen.

8 Transition 1

8.1 Iterationsplanung

Ticket
Bericht Schreiben
Dokumentation fertigstellen
Ao Poster erstellen
Abgabe CD erstellen
Iterationsassessment Transition 1
Abschlusstreffen bei IBM und Abgabe der Laptops

8.2 Iterationsassessment

Ticket	Fortschritt
Bericht Schreiben	100%
Dokumentation fertigstellen	100%
Ao Poster erstellen	100%
Abgabe CD erstellen	100%
Iterationsassessment Transition 1	100%
Abschlusstreffen bei IBM und Abgabe der Laptops	100%
Architekturdokument erstellen	100%

8.3 Fazit

Die Transition Iteration verlief wie geplant. Die Programmierung konnte Ende Construction III abgeschossen werden und es konnte die volle Aufmerksamkeit der Dokumentation gewidmet werden.

HSR Studienarbeit

IBM SmartPower

Anforderungsspezifikation

gaquino, sbrouwer
30.9.2013

Änderungsgeschichte

Datum	Version	Änderung	Autor
30.09.2013	1.0	Erstellung Anforderungsspezifikation	Team
22.10.2013	1.1	Anpassung der Szenarien	Team
17.12.2013	1.2	Anpassung für Abgabe der SA	Team

Inhaltsverzeichnis

1 Einführung 4
1.1 Gültigkeitsbereich 4
1.2 Referenzen 4
1.3 Übersicht 4
2 Allgemeine Beschreiung 4
2.1 Produkt Perspektive 4
2.2 Produkt Funktion 4
2.3 Benutzer Charakteristik 4
2.4 Annahmen und Einschränkungen 5
3 Personas 6
4 Szenarien 6
5 Use Cases 8
5.1 Use Case Diagramm 8
5.2 Aktoren & Stakeholder 8
5.3 Beschreiung (Brief) 8
5.4 Beschreiung (Fully Dressed) 9
5.4.1 UC 01: View devices/aggregation/objectives 9
5.4.2 UC 02: View devices with state «critical» 10
5.4.3 UC 03: View planed/actual measurements 10
5.4.4 UC 04: View specific device 11
5.4.5 UC 05: View current model 11
5.4.6 UC 06: Edit current model 12
5.4.7 UC 07: Add new model 13
6 GUI Evaluation 14
7 Nicht Funktionale Anforderungen (NFR) 15
7.1 Funktionalität 15
7.1.1 Angemessenheit 15
7.1.2 Interoperabilität 15
7.1.3 Sicherheit 15
7.2 Zuverlässigkeit 15
7.2.1 Fehlertoleranz 15
7.3 Benutzbarkeit 15
7.3.1 Erlernbarkeit 15
7.3.2 Bedienbarkeit 16
7.4 Effizienz 16

7.4.1 Zeitverhalten 16
7.5 Wartbarkeit 16
7.6 Übertragbarkeit 16
7.6.1 Anpassbarkeit 16
8 Anhang 16
9 Abbildungsverzeichnis 16

1 Einführung

1.1 Gültigkeitsbereich

Dieses Dokument gilt als Grundlage des Projektes und beschränkt sich auf die Dauer der Studienarbeit im HS 2013/2014.

1.2 Referenzen

- Projektplan_SA.pdf

1.3 Übersicht

Mit diesem Dokument werden die Use Cases erwähnt und erklärt sowie auch die Szenarien und die Personas. Die wichtigsten Use Cases werden im Format „Fully Dressed" ausführlich behandelt. Zusätzlich sind in diesem Dokument die Nicht-Funktionalen-Anforderungen (NFRs) aufgelistet.

2 Allgemeine Beschreiung

2.1 Produkt Perspektive

Das Smart Power GUI soll ein Produkt darstellen, welches es den IBM Mitarbeitern erleichtert eine Übersicht über die vom SmartPower System verwalteten Geräte(Devices) zu gewinnen. Es soll auf eine einfache und übersichtliche Weise die Sensoren, und deren Status, der jeweiligen Geräte anzeigen. Als Ausgangslage diente folgendes UI:

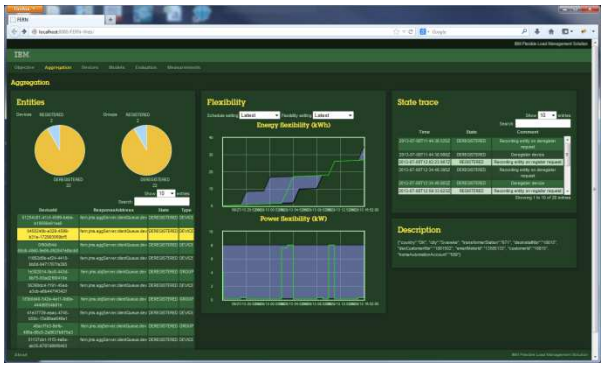


Abbildung 2: Ausgangslage UI 1

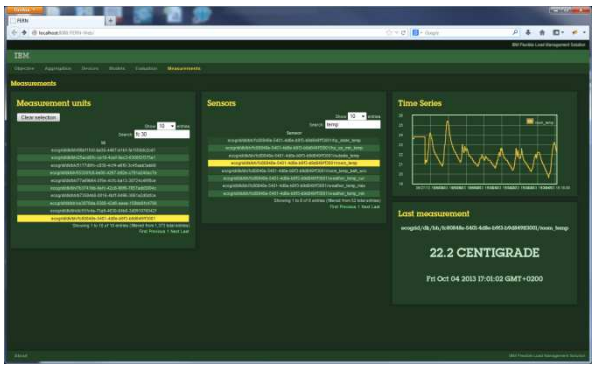


Abbildung 1: Ausgangslage UI 2

Das bisher benutzte UI ist sehr allgemein gehalten. Die Sensoren der Devices werden nicht durch das Model bestimmt, somit werden auch Sensoren angesprochen welche für das Device nicht existieren. Ebenso fehlt die Monitoring Funktionalität, welche von der IBM dringend benötigt wird, um Fehlerhafte Geräte ausfindig zu machen.
Das neue UI soll die Usability sowie die Funktionalität des UIs verbessern bzw. erweitern. Es soll einfach anpassbar sein, damit es auch unter einer anderen Arbeitsumgebungen einsetzbar ist. Da die IBM auch später mit dem neuen UI arbeiten will, soll es skalierbar sein um so neue Features mit möglichst wenig Aufwand implementieren zu können.

2.2 Produkt Funktion

Der Benutzer kann sich mit dem Smart Power GUI auf einfache Weise eine Übersicht über die vom SmartPower System verwalteten Geräte verschaffen. Das User Interface bereitet die vorhandenen Daten auf und stellt sie möglichst einfach und übersichtlich dar. Es ermöglicht das Anzeigen aller Geräte sowie auch die Einschränkung auf eine bestimmte Gruppe von Geräten (Aggregation/Objective) oder einzelne Geräte und dessen Sensoren. Es erlaubt auf den ersten Blich festzustellen welche Geräte in einem kritischen Zustand sind und diese zu verwalten.

2.3 Benutzer Charakteristik

Die Zielgruppe sind Mitarbeiter der IBM, welche am Smart Power Projekt mitwirken.

2.4 Annahmen und Einschränkungen

Für das Smart Power UI wird vorausgesetzt dass die Kompatibilität zu älteren Browser gewährleistet wird, dies wird bis zu einem gewissen Grad erfüllt auf Grund von verwendeten Frameworks wie z.B. jQuery. Das Testing dieser Vorgabe wird jedoch hauptsächlich auf neueren Browsern durchgeführt. Die Daten welche in der Entwickelten Applikation verwendet werden, werden von den IBM Web Services geliefert. Das testen mit den richtigen Webservices liegt in der Hand der IBM, da der Zugriff auf den Servern nur Ihnen gestattet ist.

3 Personas

Das Ziel der Personas ist, zu Zeigen welche Anwender diese Applikation verwenden und auf welche Art und Weise. Somit soll verhindert werden Features einzubauen welche später vom Endbenutzer nicht benötigt werden.

Kurt Kontrolle

Name:	Kurt Kontrolle
Alter:	32
Funktion:	Software Entwickler
Kenntnisse:	Experte

Beschreibung:

Kurt Kontrolle arbeitet bereits seit längerer Zeit bei der IBM ZRL in Rüschlikon.
Zusammen mit seinem Team, entwickelt er Software die dazu beiträgt die Energieeffizienz zu verbessern.

Kurt Arbeitet hauptsächlich an seinem 13" Notebook auf welchem Windows 7 läuft. Als Browser wird von ihm Firefox in der jeweils aktuellsten Version genutzt.

Durch seine Arbeit als Software Entwickler kennt sich Kurt mit Computern sehr gut aus und kann somit auch komplexere Anwendungen bedienen, jedoch ist auch er über eine simple und schnelle Bedienung der zu anwendenden Applikationen dankbar, da diese seine Produktivität in anderen Bereichen steigern könnte.

„Das System an dem gerade gearbeitet wird ist noch nicht in einem Stabilen Zustand, weshalb Fehlerfälle schnell erkannt und nachvollziehbar sein müssen."

4 Szenarien

Szenario 1.0: Anzeigen von Warnungen in einem fehlerhaften Gerät

Kurt hat immer alles im Griff, manchmal passiert es jedoch trotzdem, dass das eine oder andere Geräte (Device) Schwierigkeiten macht. Für das Monitoring der Devices benutzt Kurt das Tool FERN. Schnell sortiert Kurt in der Geräteübersicht nach den Warnungen und schon sind ihm die Geräte ersichtlich welche Warnungen enthalten. Kurt nimmt sich in diesem Fall das Gerät vor, welches die meisten Warnungen enthält und klickt diesen erstmals an um zu schauen welche Probleme auf dem Device bestehen. Nun sieht er eine Liste von den Warnungen für das Device und kann mit Szenario 2.1 weiterfahren.

Szenario 1.1: Anzeigen wie lange ein Gerät schon fehlerhaft läuft

Kurt bemerkt kurz vor der Mittagspause, dass eines der Devices Warnungen anzeigt. Weil es sich um nicht gravierende Warnungen, welche schnell wieder einen unkritischen Zustand einnehmen können handelt, beschliesst Kurt trotzdem seine Mittagspause zu beziehen. Als er vom Mittag zurückkehrt, bemerkt Kurt dass die Warnungen immer noch vorhanden sind. Er lässt sich besorgt die Details des Devices anzeigen und sieht sofort der Fehlerhafte Sensor welches die Warnung produziert. Dank dem Graphen ist Kurt in der Lage zu sagen wie lange dieses Gerät bereits fehlerhaft ist.

Szenario 1.2: Fehlerhafte Geräte überwachen

Kurt ist seit letzter Woche immer wieder mit einem Geräte konfrontiert das Probleme macht. Er möchte dieses Device gerne öfters überprüfen. Weil die IDs der Devices relativ lange und komplex sind ist das Suchen eine mühsame Angelegenheit. Kurt beschliesst deshalb das Device als Favorit zu notieren. Nun findet er das Gerät immer am Anfang der Liste und erkennt ihn durch die optische Hervorhebung. Kurt ist froh die Suche nicht mehr so oft betätigen zu müssen.

Szenario 2.0: Anzahl Registered / Deregistered Geräte

Kurt's wöchentliches Teammeeting steht an. Sein Chef hat ihn letzte Woche gebeten zu überprüfen wie viele Geräte registriert und wie viele nicht registriert sind. Kurt möchte den Chef beeindrucken und ihm kurzfristig die aktuellsten Werte vorlegen. Deshalb geht er kurz vor der Sitzung auf die FERN Seite und öffnet die Device Overview. Dort findet Kurt ein Diagramm mit der gesamten Anzahl der Devices und die Unterscheidung zwischen registriert und nicht registriert. Diese Statistik kann er nun ausdrucken oder direkt auf dem Bildschirm presentieren.

Szenario 2.1: Status des Gerätes anzeigen/ändern

Kurt bemerkt dass eines der Geräte immer wieder in einem fehlerhaften Zustand fällt und nach kurzer Zeit wieder OK ist. Ihm ist in diesem Fall nicht gleich klar was das Problem mit dem Gerät ist. Um den Kunden des Devices nicht zu verärgern entscheidet sich Kurt das Device zu deregistrieren bis das Problem behoben ist. Kurt wählt das Device aus und stellt dessen Status auf „deregistered". Zusätzlich gibt er einen Grund an wieso diese Entscheidung gefällt wurde. Kurt kann so immer den Statusverlauf nachvollziehen.

5 Use Cases

Aufgrund der Mitarbeit mit der IBM sind wichtige Diagramme in Englischer Sprache.

5.1 Use Case Diagramm

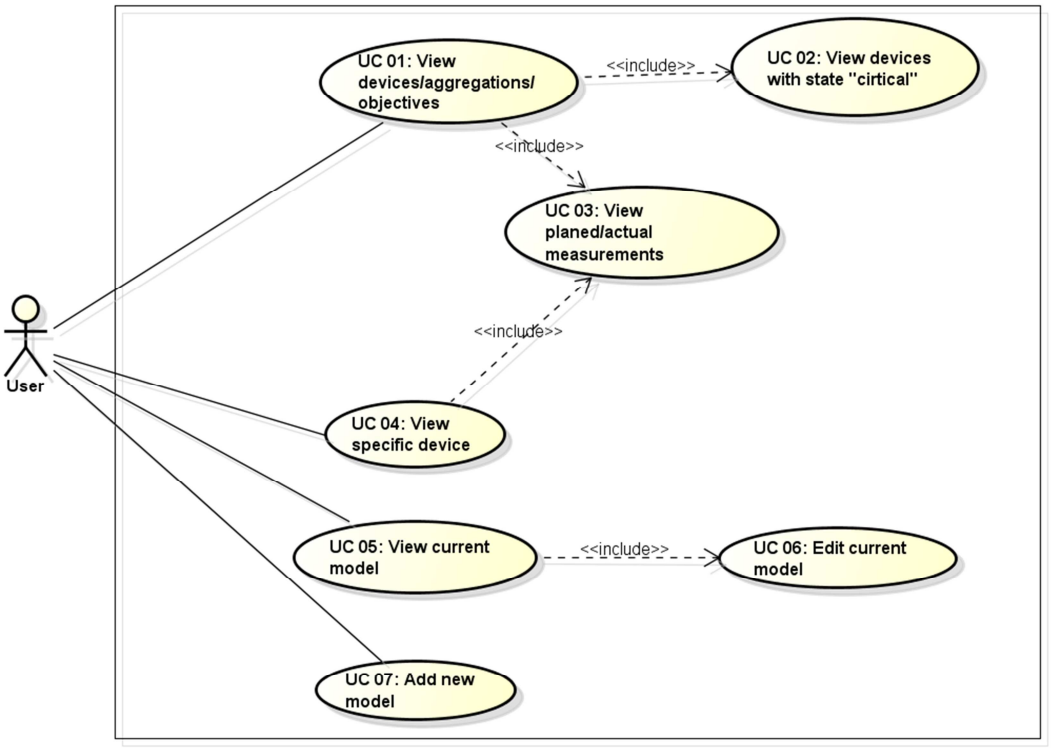


Abbildung 3: Use Case Diagramm

5.2 Aktoren & Stakeholder

- Benutzer

5.3 Beschreiung (Brief)

UC 01: View devices/aggregation/objectives

Der User lässt sich die aktuelle Device-Liste anzeigen. Dabei kann er die Devices sortieren und beim Anklicken die wichtigsten Informationen anzeigen lassen.

UC 02: View devices with state «critical»

Der User sieht auf einem Blick welche Devices in einem Kritischen Zustand sind. Dazu sind die Devices optisch gekennzeichnet.

UC 03: View planed/actual measurements

Nachdem der User einen Device oder eine Gruppe von Devices ausgewählt hat, kann er sich den geplanten bzw. den aktuellen Stromverbrauch anzeigen lassen. Dies wird mit einem Graphen grafisch dargestellt.

UC 04: View specific device

Nachdem der User in der Device Overview ein Device ausgewählt hat, wird diese mit den gängigen Informationen angezeigt. Zusätzlich hat der User eine Liste der eingetragenen Sensoren für das Device.

UC 05: View current model

Der User wählt einen Device aus, und lässt sich das Model des Devices anzeigen.

UC 06: Edit current model

Der User wählt einen Device aus und lässt sich das Model anzeigen. Dieses kann vom User editiert werden.

UC 07: Add new model

Der User kann einem Device ein neues Model zuweisen. Dazu muss er ein Device auswählen und diesem ein neues Model zuweisen.

5.4 Beschreiung (Fully Dressed)

5.4.1 UC 01: View devices/aggregation/objectives

Use-Case Name	**View devices/aggregation/objectives**
Umfang	Nachdem der User die Seite im Browser geöffnet hat, wird dem User eine Übersicht der Devices angezeigt. Dabei ist gleich ersichtlich ob ein Device sich in einem kritischen Zustand befindet.
Primärakteur	User
Stakeholder und Interessen	User: Schnelle, Klare und möglichst Informative Übersicht über den Status der Systeme
Vorbedingungen	User ist berechtigt die Webapplikation zu benutzen. Das System hat Zugriff auf den Devices (Netzwerk ist vorhanden)
Erfolgsgarantie, Nachbedingungen	Die Devices werden angezeigt. Devices mit Status „Critical" sind ersichtlich (falls welche vorhanden).
Standardablauf	1. Benutzer öffnet die Webapplikation 2. Liste wird angezeigt.
Erweiterungen	• 2a. Ist keine Netzwerkverbindung vorhanden soll eine Failure-Seite angezeigt werden.
Spezielle Anforderungen	
Häufigkeit des Auftretens	Mehrmals täglich

5.4.2 UC 02: View devices with state «critical»

Use-Case Name	UC 02: View devices with state «critical»
Umfang	Der User sieht eine Übersicht über alle Devices. Der User erkennt welche Devices mit dem Status „critical" versehen sind.
Primärakteur	User
Stakeholder und Interessen	User: Schnelle Übersicht über Devices die in einem kritischen Zustand sind.
Vorbedingungen	User ist berechtigt die Webapplikation zu benutzen. Das System hat Zugriff auf den Devices (Netzwerk ist vorhanden) Device mit Zustand „critical" ist vorhanden.
Erfolgsgarantie, Nachbedingungen	Devices mit Zustand „critical" sind in der Liste vorhanden und optisch erkennbar.
Standardablauf	1. User öffnet die Webapplikation 2. User lässt sich die Liste der kritischen Geräte anzeigen.
Erweiterungen	• 2a. Sind keine kritische Devices vorhanden steht dieser Use-Case nicht zur Verfügung.
Spezielle Anforderungen	
Häufigkeit des Auftretens	Bei vorhanden sein von Critical Devices

5.4.3 UC 03: View planed/actual measurements

Use-Case Name	UC 03: Show planed/actual measurements
Umfang	Der User möchte eine Übersicht über den aktuellen Status des Gerätes. Dazu ist er in der Lage mehrere Sensoren eines Devices „abzutasten". Die Werte der Sensoren lässt sich der User grafisch darstellen in Form eines Graphen.
Primärakteur	User
Stakeholder und Interessen	User: Verständliche und nicht überladene grafische Darstellung der Sensoren
Vorbedingungen	User ist berechtigt die Webapplikation zu benutzen. Das System hat Zugriff auf den Devices (Netzwerk ist vorhanden) Plan wurde bereits erstellt.
Erfolgsgarantie, Nachbedingungen	Graphen zeigt die richtige Daten an
Standardablauf	1. User öffnet die Webapplikation 2. User wählt ein Device aus. 3. Sensoren-Liste wird angezeigt 4. User wählt Sensor aus 5. Graphische Darstellung des Sensor wird angezeigt 6. Weiter bei Punkt 4.
Erweiterungen	
Spezielle Anforderungen	
Häufigkeit des Auftretens	Gelegentlich

5.4.4 UC 04: View specific device

Use-Case Name	UC 04: View specific device
Umfang	Der User wählt in der Device Übersicht ein Device aus. Sämtliche Informationen über das ausgewählte Device werden angezeigt. Dies beinhaltet die Sensoren und deren Werte sowie Grafiken zum Stromverbrauch sowie Status Updates des Devices.
Primärakteur	User
Stakeholder und Interessen	User: Möglichst vollständige Informationsseite über ein spezifisches Device
Vorbedingungen	Device existiert.
Erfolgsgarantie, Nachbedingungen	Device Informationen sind dem User ersichtlich
Standardablauf	1. User wählt Device aus 2. Informationen über ausgewähltes Device werden angezeigt.
Erweiterungen	• 2a. Sensor liefert keine Daten. Graphen wird leer angezeigt.
Spezielle Anforderungen	-
Häufigkeit des Auftretens	Mehrmals Täglich

5.4.5 UC 05: View current model

Use-Case Name	UC 05: View current model
Umfang	Der User wählt ein Device aus und lässt sich für diesen das aktuelle Model anzeigen.
Primärakteur	User
Stakeholder und Interessen	User: Übersichtliche Darstellung des Models.
Vorbedingungen	Es ist ein Model für das ausgewählte Device vorhanden. Model abrufbar.
Erfolgsgarantie, Nachbedingungen	-
Standardablauf	1. Gerät wird ausgewählt 2. User klickt auf die Schaltfläche zum anzeigen des Models 3. Model wird angezeigt.
Erweiterungen	• 3a. Model ist nicht vollständig gefüllt oder die Struktur des Models ist fehlerhaft. Vorhandene Daten werden angezeigt.
Spezielle Anforderungen	
Häufigkeit des Auftretens	Mehrmals pro Woche

5.4.6 UC 06: Edit current model

Use-Case Name	UC 06: Edit current model
Umfang	Der User befindet sich im UC 05, Show current Model. Der User klickt auf die Schaltfläche zum Bearbeiten des Models. Das Model ist nun editierbar. Der User kann die vorgenommenen Änderungen abspeichern und die Änderungen werden im System registriert.
Primärakteur	User
Stakeholder und Interessen	User: Einfache und schnelle Anpassung des vorhandenen Models
Vorbedingungen	UC 05 wurde abgeschlossen
Erfolgsgarantie, Nachbedingungen	Das geänderte Model ist im System abgespeichert.
Standardablauf	1. User klickt auf die Schaltfläche zum Bearbeiten des Models 2. Der User bearbeitet die vorhandenen Daten 3. Der User Speichert die gemachten Änderungen 4. Das System speichert die eingegebenen Daten ab. 5. User kehrt in den UC 05 zurück
Erweiterungen	• 2a. User gibt fehlerhafte Daten in ein.
Spezielle Anforderungen	-
Häufigkeit des Auftretens	Mehrmals pro Woche

5.4.7 UC 07: Add new model

Use-Case Name	**UC 07: Add new model**
Umfang	Der User hat ein Device ausgewählt und klickt auf die Schaltfläche zum Erstellen eines neuen Models. Der User gibt füllt die benötigten Daten in ein bereits vorhandenes Template ein und speichert diese im System ab.
Primärakteur	User
Stakeholder und Interessen	User: Einfache und sichere Erstellung eines Models.
Vorbedingungen	Template für das Model ist bereits vorhanden.
Erfolgsgarantie, Nachbedingungen	- Das neue Model wurde dem Gerät hinzugefügt und im System abgespeichert. - UC 05 wird mit dem neuen Model ausgeführt.
Standardablauf	1. User klickt auf die Schaltfläche zum Erzeugen eines neuen Models. 2. Das Template des Models wird angezeigt. 3. Der User passt das vorhandene Template an und klickt auf die Schaltfläche zum Speichern des Models 4. Das Model wird dem Device zugewiesen und im System abgespeichert. 5. Es wird UC 05 ausgeführt
Erweiterungen	• 3a. Der User gibt fehlerhafte Daten in das Formular ein.
Spezielle Anforderungen	
Häufigkeit des Auftretens	Mehrmals pro Woche

6 GUI Evaluation

Anhand der Use Cases wurden die Wireframes entwickelt, die sicherstellen sollen, dass sämtliche Use Cases erfüllt werden können. Einmal in der Woche fand ein Meeting mit der IBM statt, so konnte immer festgestellt werden ob das neue UI den Vorstellungen des Auftraggebers entspricht. Bevor mit der Entwicklung begonnen wurde, wurden die Wireframes von der IBM abgenommen.

Wireframe Overview:

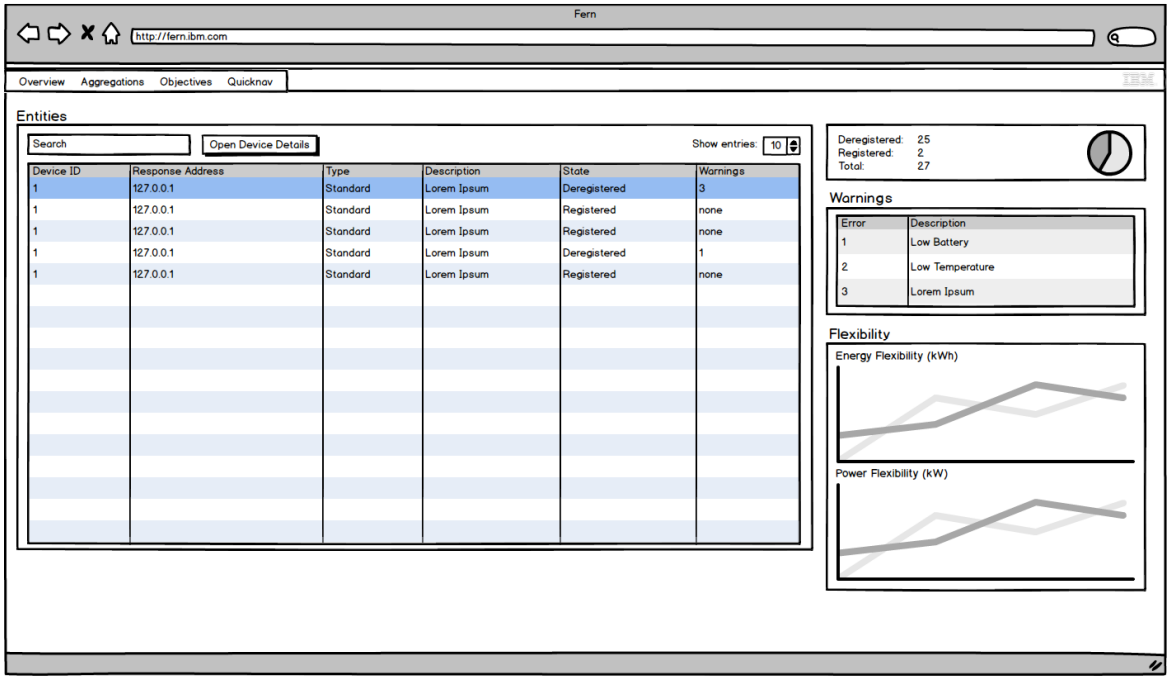


Abbildung 4: Wireframe overview

Wireframe Detailview:

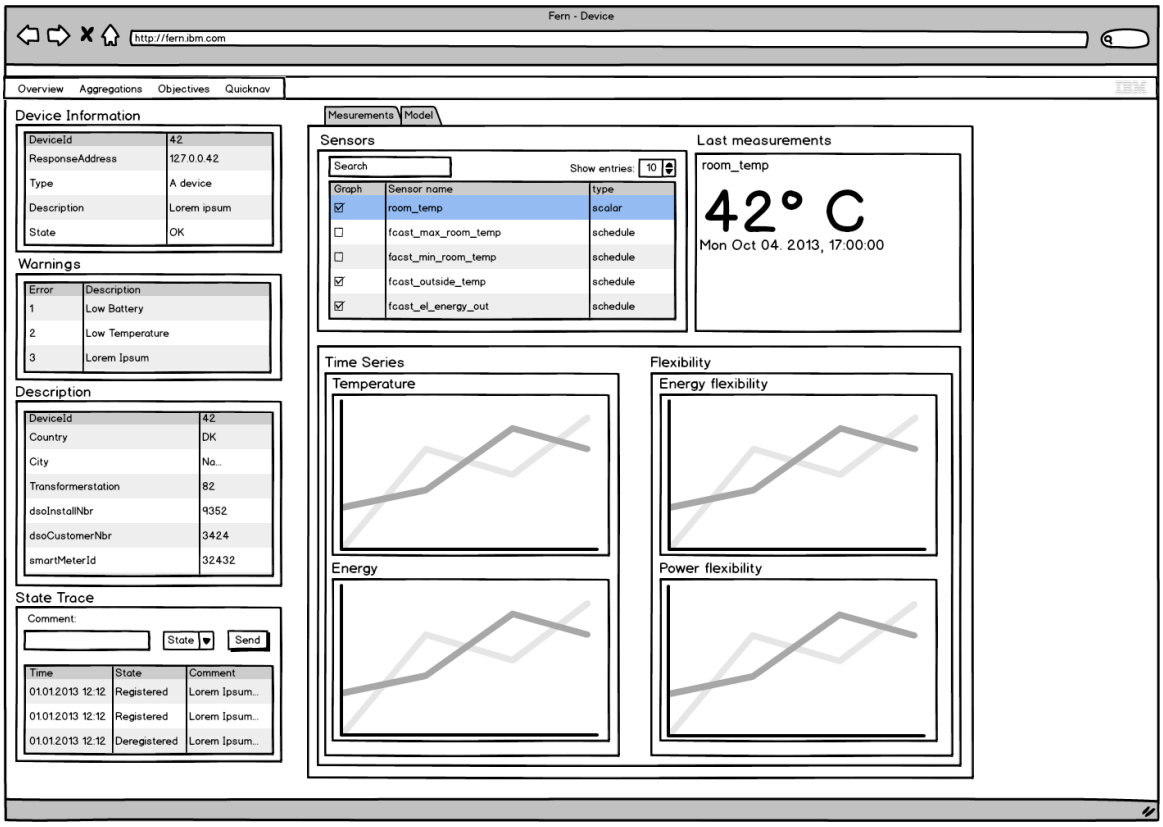


Abbildung 5: Wireframe detailview

7 Nicht Funktionale Anforderungen (NFR)

7.1 Funktionalität

7.1.1 Angemessenheit

Das vorhandene IBM Smart Power Interface befindet sich zurzeit noch in einer „Test Version". Ziel der SA ist es das User Interface benutzerfreundlicher und Effizienter zu gestalten. Der Benutzer soll die bisherige Funktionalität und allenfalls zusätzliche Features, mit weniger Klicks als bisher nutzen können.

7.1.2 Interoperabilität

Das System wird während der Entwicklung mit Testdaten arbeiten. Später soll es in der Lage sein, mit wenigen Anpassungen, mit den zur Verfügung stehenden Webservices zu arbeiten. Es soll möglich sein mit Anpassung einer Variable pro Webservice die richtigen

7.1.3 Sicherheit

Da die Software hauptsächlich von der IBM Intern genutzt wird, sind soweit keine Anforderungen in diesem Bereich vorhanden.

7.2 Zuverlässigkeit

7.2.1 Fehlertoleranz

Trotz fehlenden oder fehlerhaften Daten, soll das User Interface immer noch in der Lage sein die vorhandenen Daten korrekt darzustellen.

7.3 Benutzbarkeit

7.3.1 Erlernbarkeit

Das Produkt wird hauptsächlich von Personen mit technischem Hintergrund genutzt. Der Benutzer soll in der Lage die Funktionalität des User Interfaces schnell zu erlernen (ca. 5-10 Minuten). Dafür

sollen alle Felder sowie Knöpfe verständlich und nachvollziehbar beschriftet sein und wenn nötig mit Tooltips ergänzt werden.

7.3.2 Bedienbarkeit

Das User Interface soll so gestaltet werden, dass es auch auf mobile Geräte, sowie Tablets oder Smartphones, korrekt angezeigt und möglichst einfach bedient werden kann.
Weil im System, je nach Operation, mit längeren Rechenzeiten gerechnet werden muss, müssen Länger dauernden Anfragen optisch gekennzeichnet werden z.B. mit einem Ladebalken.

7.4 Effizienz

7.4.1 Zeitverhalten

Das Laden einer Seite darf nicht länger als 3 Sekunden in Anspruch nehmen. Vorgänge die Länger dauern, sollten optisch gekennzeichnet werden z.B. mit einem Ladebalken.

7.5 Wartbarkeit

Erweiterungen des Systems sollen nicht eine neue Seite erfordern. Das MVC Konzept soll erlauben Views in das System hinzuzufügen welche die neue gewünschte Funktionalität mitbringen.

7.6 Übertragbarkeit

7.6.1 Anpassbarkeit

Die Software wird während der Entwicklungsphase mit Daten gefüllt die aus einem statischen Umfeld stammen. Der Zugriff auf Daten muss deshalb zwingend variabel gestaltet werden, sodass später mit Anpassung einiger Variablen ein Zugriff auf die produktiven Webservices möglich ist.

8 Anhang

Die Tests wurden in einem Separaten Dokument festgehalten.

- Nichtfunktionaletests_SA.pdf
- Systemtestprotokoll_SA.pdf
- Systemtestspezifikation_SA.pdf

9 Abbildungsverzeichnis

Abbildung 2: Ausgangslage UI 1 4
Abbildung 1: Ausgangslage UI 2 4
Abbildung 3: Use Case Diagramm 8
Abbildung 4: Wireframe overview 14
Abbildung 5: Wireframe detailview 15

HSR Studienarbeit

IBM SmartPower

Architecture Document

gaquino, sbrouwer

10.10.2013

Change History

Datum	Version	Änderung	Autor
10.10.2013	1.0	Creating Document	Team
06.11.2013	1.1	Beschreibung der Architektur	Team
19.12.2013	1.3	Änderungen für Abgabe	Team

Inhaltsverzeichnis

1 Introduction 4
1.1 Purpose 4
1.2 Extent of validity 4
1.3 References 4
1.4 Overview 4
2 System overview 5
2.1 Implemented Components 5
2.1.1 Client 5
2.1.2 Proxy Webservice 5
3 Architectural goals & limitations 6
3.1 Goals 6
3.2 Limitations 6
3.3 Software requirements 6
3.3.1 Security 6
3.3.2 Distribution 6
4 Logical architecture 7
4.1 Frameworks 7
4.1.1 Backbone JS 7
4.1.2 JQuery 7
4.1.3 Bootstrap 7
4.2 Plugins 7
4.2.1 Flot 7
4.2.2 DataTables 7
4.3 Web application 8
4.3.1 css 8
4.3.2 img 8
4.3.3 js 8
4.3.4 lib 10
4.3.5 tmpl 10
4.4 Proxy webservice 11
4.4.1 Aggregation 11
4.4.2 Device 11
4.4.3 Model 11
4.4.4 Sensor 11
4.4.5 ProxyDataManager 11

4.4.6 Example web service call 11
4.5 Important sequences 12
4.5.1 Fetching data from the "eg_critical" sensor for the overview 12
5 Deployment 13
5.1 On server 13
5.2 For local development 13
6 Datenspeicherung 13
7 Grössen und Leistung 13
8 List of figures 14

1 Introduction

1.1 Purpose

This document is meant to give an overview for the architecture of the „SmartGrid (FERN)" system and specially for its UI.

1.2 Extent of validity

This document will be valid for the „Studienarbeit 2013" which begins on the 16. of September 2013 and ends on 20. December 2013.

1.3 References

- Wireframes_FERN.pdf

1.4 Overview

The FERN application is divided in three parts:

- Database (with all the relevant data for the devices)
- Web services (used for the information exchange between UI and Database)
- User Interface

The User Interface, will be running on an Apache Tomcat Server 2.4 because of the already existing environment.
For the frontend of the application the classic MVC pattern is used. Thanks to the separation of the single parts into web service, GUI and database the interchangeability, maintainability, testability and scalability are given.

2 System overview

FERN is split into 3 logical layers. The client user interface as presentation layer, the database for the persistence of the data and the web services for the communication and logic between database and user interface. In addition to the existing web services an additional web service has been implemented. It is used as a proxy to enable cross-domain Ajax calls to the web services.
Marked in red; implemented pieces for the "Studienarbeit".

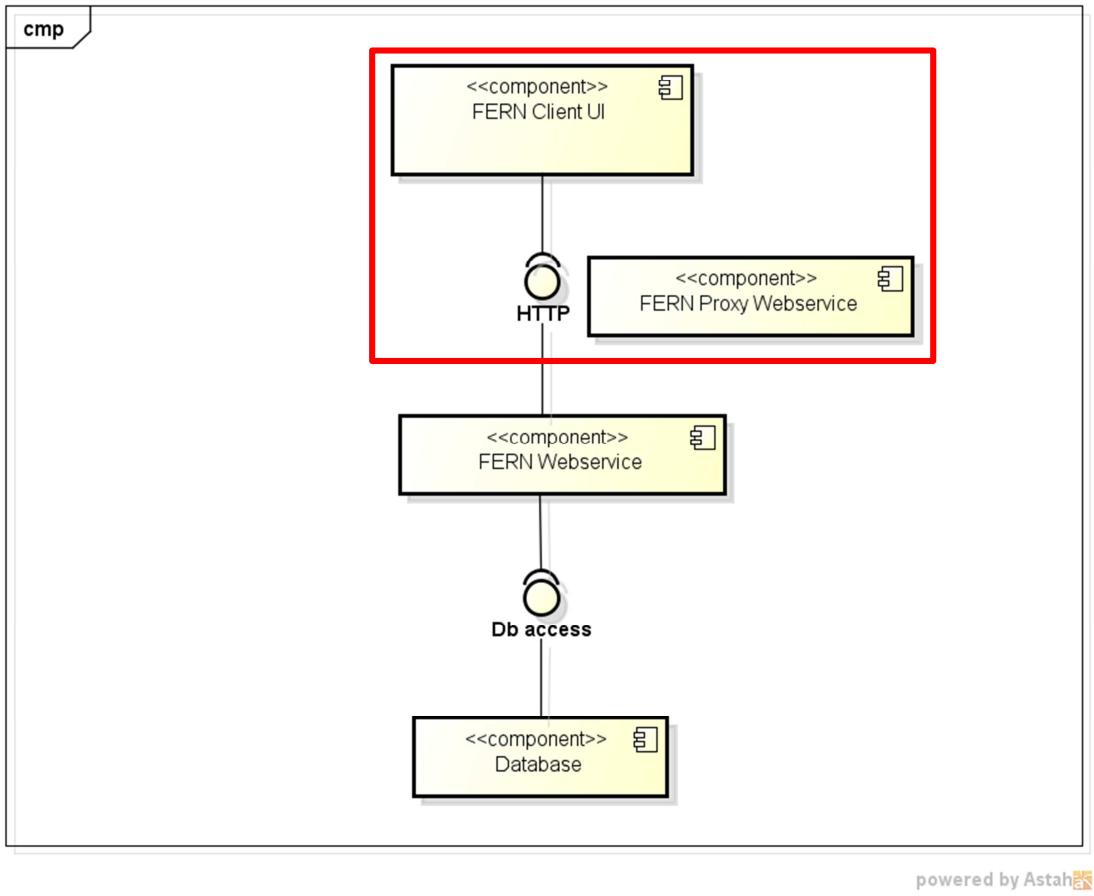


Figure 1: System overview

2.1 Implemented Components

As seen in the diagram above (Figure 1: System overview), only the Fern Client UI and the FERN Proxy Webserver will be implemented during the "Studienarbeit". All other components are already implemented and ready to use.

2.1.1 Client

During the "Studienarbeit" the Client will access the data through Proxy Webserver. The web application should be compatible with new browsers and also work for older browsers. The testing will be done on IBM Notebooks with Firefox 17.0.10 and MS Internet Explorer 10. For maintaining the MVC Pattern and a homogeneous web design, different JavaScript frameworks are being used, such as Backbone JS and Bootstrap.

2.1.2 Proxy Webservice

For the "Studienarbeit" the web application has been deployed on an external webserver that is part of the IBM infrastructure. Because of the use of ajax for the requests a proxy webserver has been built. This allows the use of web services which are placed in another domain/location. The proxy web service is an extension that could be used in future but isn't needed, if the web application is deployed on the same server as the web services.

3 Architectural goals & limitations

3.1 Goals

Listed below are the most important requirements, a full list can be found in the Anforderungsspezifikation_SA.docx document.

- The User Interface for the Smart Power System should be designed to be easy to use and clearly structured in order that the user is able to work efficiently and still have as much information as possible.
- The data displayed in the UI should always be up-to-date and display the most recent measurements. Older measurements should still be available.
- Client and server should be loosely coupled because of the variation of different uses the web application has to fulfill.

3.2 Limitations

- The system always needs a connection to the proxy web service when not working on the deployment server.
- The proxy web service needs a connection to the FERN web services

3.3 Software requirements

3.3.1 Security

The security aspect is not being considered for the "Studienarbeit" since the system is only used internally. The proxy web service, which replies the answer from the FERN web service is also only available in the IBM network.

3.3.2 Distribution

The web application will be distributed by copy deployment, also the proxy webserver. Because of the development environment used during the "Studienarbeit" some references to the web services might need some changes, if they are deployed to different locations on the server. For more detailed information see chapter 5.

4 Logical architecture

Here you can find the logical architecture of the web application and also some specifications for the proxy web service.

4.1 Frameworks

4.1.1 Backbone JS

The goal was to make a fast, easy to use and an appealing User Interface using JavaScript and HTML5. There were two approaches to this, a single page architecture where all the content is loaded dynamically with Ajax into one single page or a multi-page architecture where there is an individual page for each part of the UI. It was decided, that the single page variation was the more elegant and up to date solution. This had one major drawback, which was that if someone were to reload the page they would not stay where they were on the site but jump to the first page. Also it prevents the sharing of links, because there is no direct URL to a page.
To eliminate these drawbacks it was decided to use Backbone.js, an innovative JavaScript Framework, which enabled us to have descriptive URLs that enable a user to reload the page and stay on the same page.
It also enabled the use of the MVC pattern and many more features.

4.1.2 JQuery

JQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript. More information on jQuery can be found on their website.

4.1.3 Bootstrap

Sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development. It was decided to use this framework to improve the look and feel of the user interface and make sure a consistent GUI experience is provided. More information on bootstrap can be found on their website.

4.2 Plugins

4.2.1 Flot

Flot is a pure JavaScript plotting library for jQuery, with a focus on simple usage, attractive looks and interactive features. Because the plugin was already being used in the old FERN-GUI and is very flexible it was decided to continue using it. More information on Flot can be found on their website.

4.2.2 DataTables

DataTables is a plug-in for the jQuery Javascript library. It is a highly flexible tool, based upon the foundations of progressive enhancement, which will add advanced interaction controls to any HTML table. This plugin was also already in use at the old FERN-GUI and it was decided to keep using it for the current GUI. More information on DataTables can be found on their website.

4.3 Web application

The folders are structured as can be seen in the following diagram (Figure 2: Folder structure). Because of the dependencies between the different files in the folders it is highly recommended to keep the folder names.

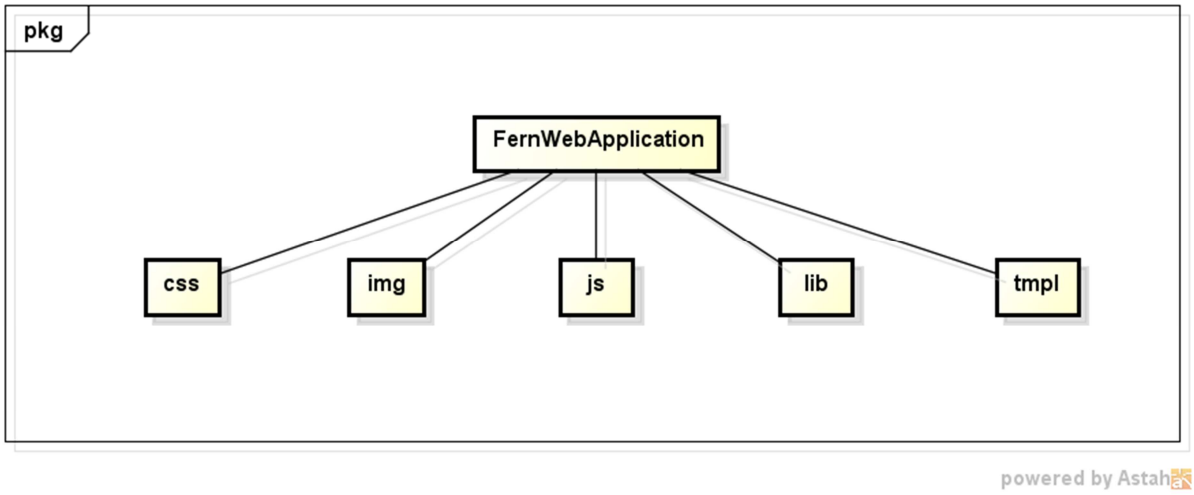


Figure 2: Folder structure

4.3.1 css

All style sheets needed for the application are placed in this folder, also other style sheets needed for the plugins and frameworks can be found in the css folder.

4.3.2 img

Images such as logo have been placed in the img folder.

4.3.3 js

This folder contains the models, the views and the main JavaScript file "app.js". The whole implementation of the web application can be found in the js folder.
The web application has been built using Backbone JS. Backbone JS is a framework which allows the implementation of the MVC Pattern on web applications. Because of that the files are structured as seen in the following diagram (Figure 3: js folder structure).

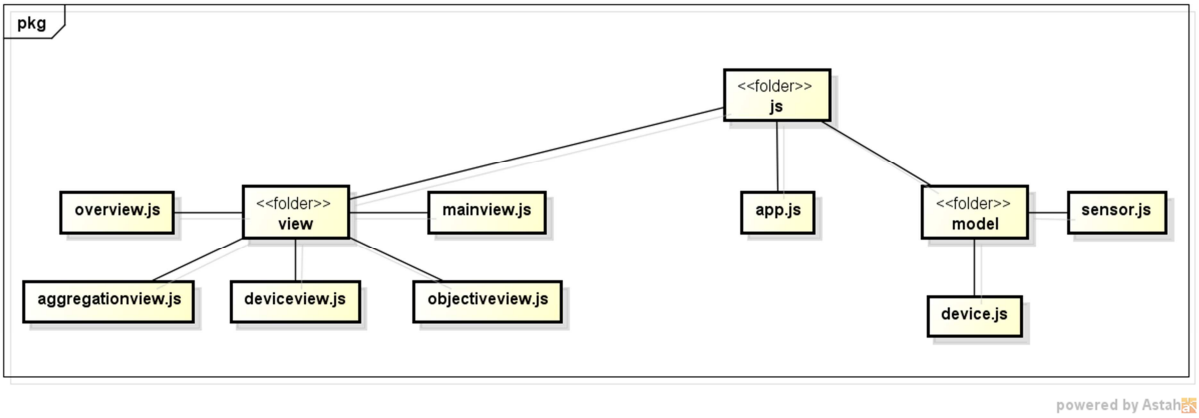


Figure 3: js folder structure

4.3.3.1 app.js

The app.js can be considered as the main executable. It instantiates the app and calls the method for loading all templates from the tmpl folder (mentioned in 4.1.5).

app.Router

The app.Router is also declared in this file. The Router is needed for the steering of the navigation flow.

4.3.3.2 view

The view folder is, as the name says, the container for the implementation of the views.

4.3.3.2.1 view/mainview.js

app.MainView

The mainview is used for showing the default layout without content. The content needed will be included when the right view is needed and initialized.

4.3.3.2.2 view/overview.js

app.OverView

The overview is the main landing page that will show up when starting the web application. It is meant to show the list of all entities (for each layer) and show the most important information such as warnings. It is able to handle all the entities from all the layers, but shows only the entities for one selected layer. The layer to be displayed is determined by the URL for example the "overview/dev" URL show only the entities for the Device-Layer. To draw the actual table with the data in it the OverView uses the EntityTableView(4.3.3.2.5). In order to show the flexibility for a device it also uses the FlexiblityView(4.3.3.2.4).

4.3.3.2.3 view/entitydetailview.js

app.EntityDetailView

The entitydetail view is meant to be the more detailed view for a certain device. It shows more information and offers the option to register or deregister a device. Also it's possible to take a look at the actual sensors used in the models of the device.

app.LastMeasurementsView

The LastMeasurementsView is used in the EntityDetailView to show the latest measurements of the selected devices. It shows a table with the name of the sensor, the date of the last measurement and the value of the last measurement.

app.EntityDetailView.SensorView

The SensorView is a sub view of the EntityDetailView and is responsible for showing a table with all the sensors that are registered in the model for a specific device.

4.3.3.2.4 View/flexibilityview.js

app.FlexiblityView

The flexibility view is meant to show the energy and power flexibility for a specific device or group. It is built in a way that it can easily be used in other views as a sub view. It is used both by the overview as well as the entitydetail view.

4.3.3.2.5 view/entitytableview.js

app.EntityTableView

The entitytable view is built to render the table with the entities form a specified layer in it. It is also built so that is can be easily used in other views as a sub view. Currently it is only used in the overview.

4.3.3.2.6 view/deviceinfoview.js

app.DeviceInfoVeiw

The deviceinfo view is a sub view of the entitydetail view and it shows detailed information about a specific devie, such as the description of the device, a state trace and the critical sensor for the device(if there is one).

4.3.3.3 model

4.3.3.3.1 model/device.js

The device model is needed for taking the data down from the webservice and put it in a collection or model that can be used in the web application as an object.

app.Entity

Model defined for the single device on the device layer. It could also be used for the aggregation or objective layer.

app.EntityCollection

The app. EntityCollection is responsible for taking all the entities of the device layer and organize them as objects in a collection. This way the handling with data becomes really comfortable and easy to use in the plugins used for the presentation of the web application.

app.EntityStateCollection

Gets the state trace

4.3.3.3.2 model/sensor.js

The sensor.js includes the measurements collection, the flexibility collection and the schedule collection.

app.MeasurementCollection

The measurements collection gets the values of a sensor, with the provided URL and stores it in the collection.

app.FlexibilityCollection

The flexibility collection gets the flexibility values for a device, with the provided URL. It gets both, the power and the energy flexibility for the device and stores them in the collection.

app.ScheduleCollection

The schedule collection gets the power and energy schedule for a device, with the provided URL and stores the data in the collection. The values for the energy schedule have to calculated with the values from the power schedule as followed:

$$E_0 = 0$$
$$E_1 = P_0 * (T_1 - T_0) + E_0$$
$$E_2 = P_1 * (T_2 - T_1) + E_1$$
$$E_3 = P_2 * (T_3 - T_2) + E_2$$

Where T is the timestamp of the slot value in seconds and P is the power in watts.

4.3.3.3.3 Model/model.js

app.DeviceModelCollection

The devicemodel collection queries the model of a specific device. It then gets all the Sensors that are registered for that device and saves them in the collection. This is used in the SensorView to show the Sensors that are available for the selected device.

4.3.4 lib

The lib folder is meant to contain all external javascript files, such as plugins/extensions and frameworks. The main framework Backbone JS in version 0.9.2 is also placed in this folder.

4.3.5 tmpl

The templates needed for the rendering of the views are placed in the tmpl folder.

4.4 Proxy webservice

JavaScript doesn't allow requests to other domains, which prevents accessing the FERN web service form a local machine. The only solution that could be found was to build an own local Proxy web service. It is a RESTfull web service with the same structure as the FERN web service. The web service is stated on a local machine, so that JavaScript is able to access it, the web service then analyses the URL and calls the according URL on the FERN web service. This enables us to test the application from a local environment over a VPN connection.

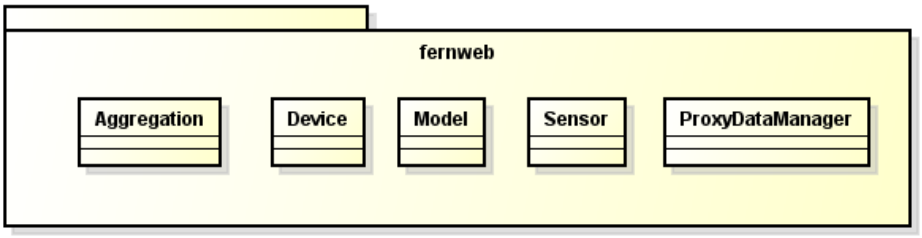


Figure 4: Proxy overview

4.4.1 Aggregation

In the Aggregation class all the calls to the aggregation layer of the FERN web service are implemented.

4.4.2 Device

In the Device class all the calls to the device layer of the FERN web service are implemented.

4.4.3 Model

In the Model class all the calls to the model of a device on the FERN web service are implemented.

4.4.4 Sensor

In the Sensor class all the calls to the sensor of the FERN web service are implemented.

4.4.5 ProxyDataManager

The ProxyDataManager is a static class which calls the actual web service. It only has one method that receives the URL to call as a parameter and returns the response as text. All the other classes use this class to make calls to the FERN web service.

4.4.6 Example web service call

The diagram below shows a sequence diagram of a call from the client to the web service, through our Proxy web service.

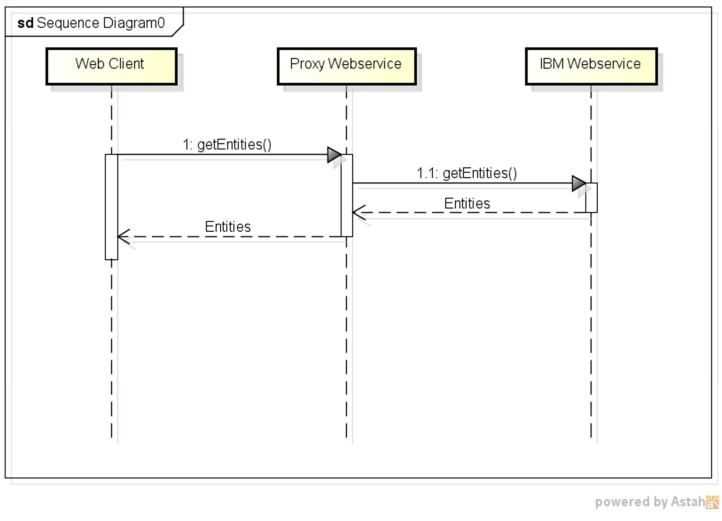


Figure 5: Proxy sequence diagram

4.5 Important sequences

4.5.1 Fetching data from the "eg_critical" sensor for the overview

In the overview there is a table, which shows all the entities for a layer. In that table the entities with a warning or critical state should be highlighted. Because the information, if an entity is critical or not, comes from another web service than the list of entities, the list of entities is fetched first and rendered in the table. Afterwards the model for each entity is fetched from the web service to make sure the URL for the "eg_cirtical" sensor is fetched dynamically and not hardcoded. Then, with the URL from the model, the actual sensor is queried and the data gets appended to the corresponding entity in the collection. After this is done for all the devices in the collection the table gets redrawn and the entities with a critical state are highlighted.

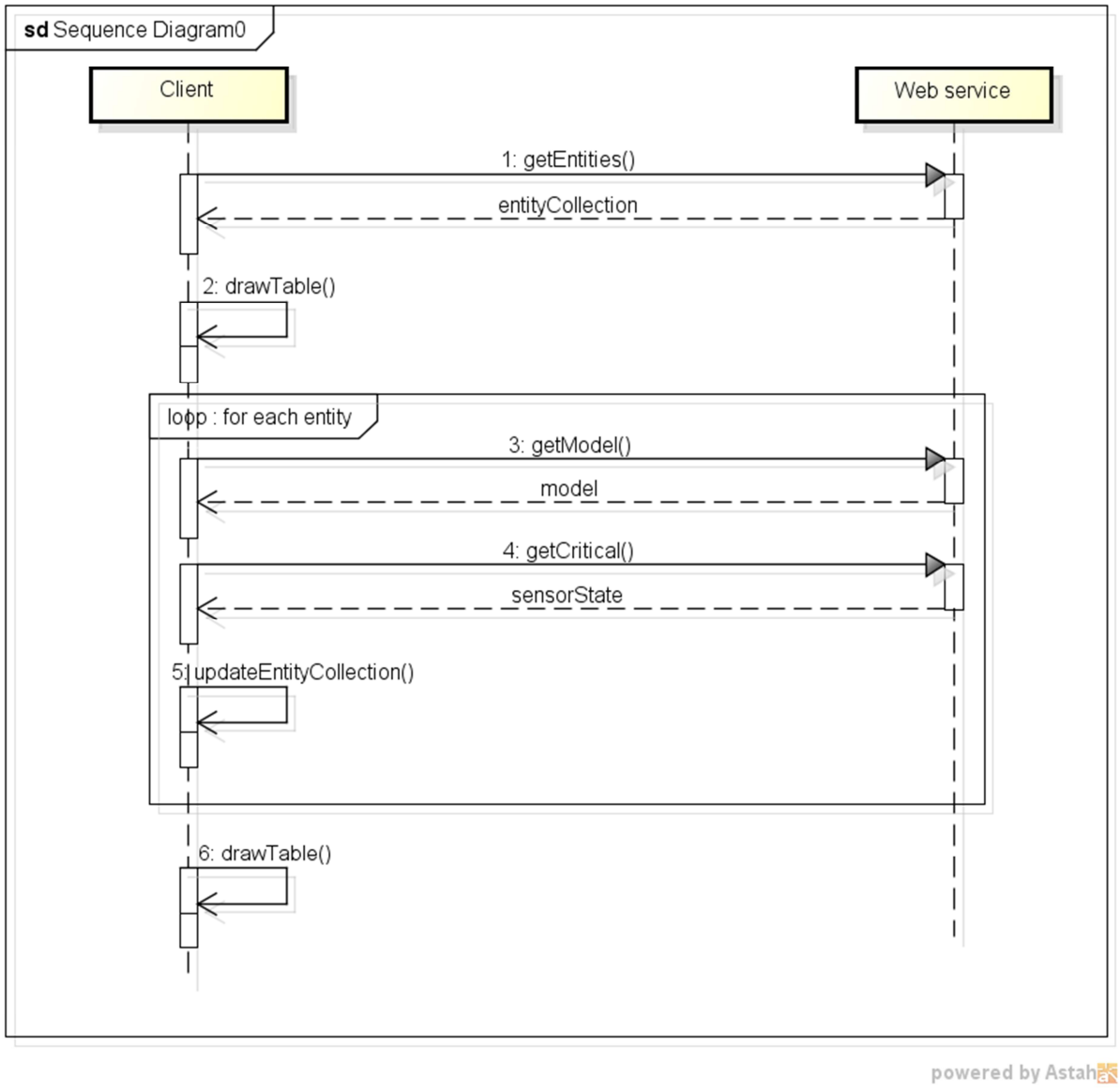


Figure 6: Critical sensor sequence diagram

5 Deployment

5.1 On server

To deploy the Smart Power GUI to a web server the FernWebApplication folder (0) has to be copied to a folder on the web server.
If there are changes in the web service URLs the following variables have to be updated in the app.js file.

- sensorBase : "[sensor web service root]"
- modelBase : "[model web service root]"

5.2 For local development

To deploy the Smart Power GUI to a local development environment the Eclipse Project, including the Smart Power GUI and the Proxy web service, has to be imported to the local Eclipse installation. Before the application can be run the proxy web service has to be started on a local Tomcat server. This can be done by simply selecting the project in the project view and then clicking on the "run" button. The GUI can then be accessed with the URL http://localhost:8080/FERN-Web/.

6 Datenspeicherung

The data that is shown in the Smart Power GUI is stored in a database. The database and all the values in it were provided by the IBM and are accessible through two different web services.
The data that is retrieved from the server is stored in collections or models, provided by the Bacbone.js framework, for the time the session is active.
The GUI is not meant to add, delete or update any data on the database and the web services do not provide any methods to do so.

7 Grössen und Leistung

The application is built to be scalable and should not run out of memory or computing power. The only limit is the time it takes to make calls to the web service.
With the current state of the critical sensor (described in 4.5.1) updating the overview with the critical data works in acceptable time up to 1000 entities. If there are more than 1000 entities it would take too long to retrieve the data and update the GUI to use the application efficiently.

8 List of figures

Figure 1: System overview 5
Figure 2: Folder structure 8
Figure 3: js folder structure 8
Figure 4: Proxy overview 11
Figure 5: Proxy sequence diagram 11
Figure 6: Critical sensor sequence diagram 12

HSR Studienarbeit

IBM SmartPower

Nichtfunktionale Tests

gaquino, sbrouwer

15.12.2013

Inhaltsverzeichnis

1 Einführung 2
1.1 Zweck 2
2 Voraussetzungen 2
3 Vorbereitungen 2
4 Nichtfunktionale Testspezifikation 2
5 Testdurchführung 3
5.1 Angaben zur Durchführung 3
5.2 Tests 3
5.2.1 Bekannte Einschränkungen 3
5.2.2 Mögliche Detailverbesserungen 3
6 Fazit 3

1 Einführung

1.1 Zweck

Dieses Dokument beschreibt die Nichtfunktionalen Tests für das SmartPower GUI und deren Durchführung.

2 Voraussetzungen

Für die Durchführung der Tests muss ein Browser (Internetexplorer 8 oder höher oder Firefox 20 oder höher) mit aktiviertem JavaScript vorhanden sein. Ausserdem muss eine Verbindung zu den IBM Webservices bestehen (direkt oder über VPN).

3 Vorbereitungen

Für das Testen auf einer lokalen Installation muss zuerst der Proxy Webservice gestartet werden, damit auf die IBM Webservices zugegriffen werden kann, da JavaScript keine Cross-Domain anfragen erlaubt.

4 Nichtfunktionale Testspezifikation

Kategorie	Nr.	Beschreibung
Angemessenheit	1	Vergleich Anzahl Klicks um die energy und power Flexibilität eines Gerätes anzuzeigen
	2	Anzahl Klicks um die Liste der Devices, Groups und Objectives anzuzeigen
	3	Anzahl Klicks um die Sensordaten einer Entity anzuzeigen
	4	Anzahl Klicks um die Warnungen einer Entity anzuzeigen
Interoperabilität	1	System kann mit Änderung von weniger als 100 Zeilen Code von Testdaten auf Livedaten umgestellt werden.
Fehlertoleranz	1	Werden noch Daten angezeigt wenn der Web-Service fehlerhafte Informationen liefert
	2	Es wird eine Meldung angezeigt falls der Web-Service keine Daten liefert
Erlernbarkeit	1	Mit dem Projekt vertrauter Benutzer kann die Applikation innert 5 – 10 Minuten vollständig bedienen
Bedienbarkeit	1	User Interface skaliert auch für kleinere Bildschirme und Tablets
	2	Bei langen lade-operationen wird ein Ladesymbol angezeigt
Zeitverhalten	1	Ladezeit der Seite ist kleiner als 3 Sekunden

5 Testdurchführung

5.1 Angaben zur Durchführung

Die Interoperabilität wurde nicht getestet da schon viel früher auf die Testdaten verzichtet werden konnte und mit den tatsächlichen Webservices gearbeitet werden konnte.
Die Erlernbarkeit wurde an der Präsentation der Applikation kurz besprochen.

5.2 Tests

Kategorie	Nr.	Erwartetes Resultat	Tatsächliches Resultat	Status
Angemessenheit	1	1 Click (wert altes GUI)	1 Click	PASS
	2	0 Clicks für Devices, 1 Click für Groups und Objectives (wert altes GUI)	0 Clicks für Devices, 1 Click für Groups und Objectives	PASS
	3	3 Clicks (wert altes GUI)	2 Clicks	PASS
	4	- (wert altes GUI)	1 Click	PASS
Interoperabilität	1	-	-	FAIL
Fehlertoleranz	1	Es werden die Korrekten Daten angezeigt oder eine Meldung angezeigt, dass die Daten fehlerhaft waren	Es werden die Korrekten Daten angezeigt.	PASS
	2	Es wird eine Meldung angezeigt	Wie erwartet	PASS
Erlernbarkeit	1	Es werden 5-10 Minuten benötigt	Es wurden 5 Minuten benötigt	PASS
Bedienbarkeit	1	User Interface wird auch auf kleinen Bildschirmen und Tablets korrekt angezeigt	Funktioniert bis auf einen kleinen Range in dem das responsive bootstrap noch nicht greift.	PASS
	2	Es wird ein Ladesymbol angezeigt	Wie erwartet	PASS
Zeitverhalten	1	Ladezeit ist kleiner als 3 Sekunden	Wie erwartet mit ausnahme der Critical Sensor Daten in der Overview	PASS

5.2.1 Bekannte Einschränkungen

Die Ladezeit beim Critical Sensor kann nicht eingehalten werden, da der Webservice nicht wie erwartet implementiert wurde.

5.2.2 Mögliche Detailverbesserungen

Responsive Bootstrap auch für diesen kleinen Range(z.B. für 13 Zoll Bildschirme) noch optimieren.
Critical Sensor Implementierung für schnellere Ladezeit optimieren.

6 Fazit

Die Tests verliefen sehr gut und die Applikation wurde von der IBM abgenommen.

HSR Studienarbeit

IBM SmartPower

Systemtestprotokoll

gaquino, sbrouwer

2.10.2013

Inhaltsverzeichnis

1 Einführung 2

1.1 Zweck 2

2 Angaben zur Durchführung 2

3 Protokoll 3

3.1 Construction I 3

3.1.1 Verbesserungsmöglichkeiten 3

3.1.2 Bekannte Einschränkungen 3

3.1.3 Mögliche Detailverbesserungen 3

3.2 Construction II 4

3.2.1 Verbesserungsmöglichkeiten 4

3.2.2 Bekannte Einschränkungen 4

3.2.3 Mögliche Detailverbesserungen 4

3.3 Construction III 5

3.3.1 Verbesserungsmöglichkeiten 5

3.3.2 Bekannte Einschränkungen 5

3.3.3 Mögliche Detailverbesserungen 5

1 Einführung

1.1 Zweck

Dieses Dokument beinhaltet sämtliche Testdurchführungen der Systemstest für die Studienarbeit „Smart Grid GUI" und gilt für das gesamte Projekt.

2 Angaben zur Durchführung

Die Tests werden nach jeder Construction durchgeführt und dokumentiert. Somit sind auch gewisse Features noch nicht implementiert, da sie erst für eine spätere Iteration geplant sind.

3 Protokoll

3.1 Construction I

UseCase	Nr.	implementiert	Fehler/Unschönheit	Status
UC 01: Show device overview	1	Ja	Wenn das Fenster verkleinert wird überlappt die Tabelle und die rechte Seite der View	PASS
	2	Ja		PASS
	3	Ja	Device Info kann noch nicht durchsucht werden	PASS
UC 02: Show devices with state „critical"	1	Nein		FAIL
UC 03: Show plaed/actual maesurements	1	Nein		FAIL
	2	Ja		PASS
	3	Nein		FAIL
	4	Nein		FAIL
	5	Ja		PASS
	6	Nein		FAIL
UC 04: Show device	1	Nein		FAIL
	2	Ja	Das schedule wird noch nicht angezeigt und der Teil zwischen den Graphen wird noch nicht ausgefüllt	PASS
UC 05: Show current model	1	Nein		FAIL
UC 06: Edit current model	1	Nein		FAIL
UC 07: Add new model	1	Nein		FAIL
	2	Nein		FAIL

3.1.1 Verbesserungsmöglichkeiten

In der Overview könnte der Tabelle noch ein „hidden-field" hinzugefügt werden, so dass die Device Info auch durchsuchbar wird, jedoch nicht angezeigt wird(soll nicht angezeigt werden!)

3.1.2 Bekannte Einschränkungen

Es wurde mit dem Stand der Software von ende Cunstruction I gearbeitet, es kann sein das noch weitere Anforderungen hinzu kommen oder das noch Anforderungen wegfallen.
Critical Sensor ist von der IBM noch nicht implementiert.

3.1.3 Mögliche Detailverbesserungen

-

3.2 Construction II

UseCase	Nr.	implementiert	Fehler/Unschönheit	Status
UC 01: Show device overview	1	Ja		PASS
	2	Ja		PASS
	3	Ja		PASS
UC 02: Show devices with state „critical"	1	Nein		FAIL
UC 03: Show plaed/actual maesurements	1	Ja	Schedule wird noch nicht angezeigt	PASS
	2	Ja		PASS
	3	Ja		PASS
	4	Nein		FAIL
	5	Ja		PASS
	6	Nein		FAIL
UC 04: Show device	1	Ja	HTML Formatierung ist noch nicht optimal	PASS
	2	Ja		PASS

3.2.1 Verbesserungsmöglichkeiten

Es wurde von der IBM bekannt gegeben, dass die Use Cases 05 – 07 nicht implementiert werden sollen, somit werden auch die Tests für diese Use Cases weggelassen.
Code besser an Backbone.js anpassen und refactoren damit die Applikation schneller und strukturierter wird.

3.2.2 Bekannte Einschränkungen

Es wurde mit dem Stand der Software von ende Cunstruction II gearbeitet, es kann sein das noch weitere Anforderungen hinzu kommen oder das noch Anforderungen wegfallen.
Critical Sensor ist von der IBM noch nicht implementiert.

3.2.3 Mögliche Detailverbesserungen

Für die Device Info das HTML anpassen, dass es übersichtlicher wird.

3.3 Construction III

UseCase	Nr.	implementiert	Fehler/Unschönheit	Status
UC 01: Show device overview	1	Ja		PASS
	2	Ja		PASS
	3	Ja		PASS
UC 02: Show devices with state „critical"	1	Ja		PASS
UC 03: Show plaed/actual maesurements	1	Ja		PASS
	2	Ja		PASS
	3	Ja		PASS
	4	Ja		PASS
	5	Ja		PASS
	6	Ja		PASS
UC 04: Show device	1	Ja		PASS
	2	Ja		PASS

3.3.1 Verbesserungsmöglichkeiten

-

3.3.2 Bekannte Einschränkungen

-

3.3.3 Mögliche Detailverbesserungen

-

HSR Studienarbeit

IBM SmartPower

Systemtestspezifikation

gaquino, sbrouwer

2.10.2013

Inhaltsverzeichnis

1 Einführung 2
1.1 Zweck 2
2 Voraussetzungen 2
3 Vorbereitungen 2
4 Systemtest 2

1 Einführung

1.1 Zweck

Dieses Dokument dient als Grundlage für die einzelnen Testdurchführungen. Da es mehrere Durchführungen geben wird, werden diese in einem separaten Dokument beschrieben und mit einem Datum versehen (Systemtestprotokoll_SA.docx).

2 Voraussetzungen

Für die Durchführung der Tests muss ein Browser(Internetexplorer 8 oder höher oder Firefox 20 oder höher) mit aktiviertem JavaScript vorhanden sein. Ausserdem muss eine Verbindung zu den IBM Webservices bestehen (direkt oder über VPN).

3 Vorbereitungen

Für das Testen auf einer lokalen Installation muss zuerst der Proxy Webservice gestartet werden, damit auf die IBM Webservices zugegriffen werden kann, da JavaScript keine Cross-Domain anfragen erlaubt.

4 Systemtest

Use Case	Nr.	Beschreibung
UC 01: Show device overview	1	Die device overview kann aufgerufen werden und es werden sämtliche devices in der Tabelle angezeigt
	2	Wenn mehr als 25 devices vorhanden sind werden nur die ersten 25 devices angezeigt und man kann mit Hilfe eines „Weiter" Knopfes die anderen devices anzeigen lassen
	3	Wenn etwas in das „Suchen" Feld eingegeben wird wird die Tabelle aktualisiert und nur noch jene devices angezeigt, weleche auf die Suchkriterien zutreffen
UC 02: Show devices with state „critical"	1	In der overview werden jene devices mit einem "critical" state herforgehoben, in dem die Zeile in der Tabelle farbig hinterlegt wird
UC 03: Show plaed/actual maesurements	1	Wenn in der overview auf ein device geklickt wird, werden die flexibility Messungen, power sowie auch energy flexiblity, für dieses device angezeigt. Ausserdem wir das schedule für das augewählte device im Diagramm angezeigt.
	2	Wenn in der detail view ein Sensor ausgewählt wird werden dessen Werte im Diagramm angezeigt.
	3	Wenn in der detail view mehrere Sensoren ausgewählt werden werden die Werte im Diagramm angezeigt und wenn unterschiedliche Einheiten vorkommen werden mehrere Achsen dargestellt
	4	Wenn in der detail view eine Zeitspanne definiert wird werden alle bereits ausgewählten Sensoren im Diagramm aktualisiert und alle welche später hinzugefügt werden, werden die Werte dieser Zeitspanne angezeigt.
	5	Wenn in der detail view mehr als 5 Sensoren ausgewählt werden wird eine Meldung angezeigt, dass maximal 5 Sensoren

		dargestellt werden können und der Graph wird dem Diagramm nicht hinzugefügt.
	6	Wenn in der detail view auf ein Sensor geklickt wird, wird ein das „last measurements"-Feld mit den letzen werten des Sensors gefüllt. Es werden das Datum der Messung, der Name des Sensors sowie dessen Wert angezeigt.
UC 04: Show device	1	Wenn in der overview auf ein device mit doppelklick ausgewählt wird, wird in die detail view gewechselt und die Informationen für das ausgewählte Gerät angezeigt. Dies beinhaltet eine Beschreibung, die Critical Sensoren sowie einen State Trace
	2	Wenn in der overview auf ein device mit doppelklick ausgewählt wird, wird in die detail view gewechselt und es werden die flexibility Messungen für dieses device für die letzen zwei Tage angezeigt, sowie das schedule für diese Zeitspanne
UC 05: Show current model	1	Wenn in der overview ein device ausgewählt wird und auf die „show model" Schaltfläche geklickt wird, wird das model des ausgewählten device angezeigt.
UC 06: Edit current model	1	Wenn in der model view auf die „edit model" Schaltfläche geklickt wird werden die Textfelder editierbar und das model kann angepasst werden.
	2	Wenn in der model view auf die „Speichern" Schaltfläche geklickt wird, werden die vorgenommenen Änderungen an den Server gesendet und gespeichert.
UC 07: Add new model	1	Wenn in der Navbar auf die „Add new model" Schaltfläche geklickt wird erscheint eine Ansicht mit den nötigen Textfeldern um eine Model zu erstellen
	2	Wenn in der addmodel view auf die „Speichern" Sachaltfläche geklickt wird, wird das model an den Server gesendet und gespeichert. Anschliessend wird in die showmodel view gewechselt