(schönes Bild hier hin? ☺ )

PowerPoint nicht vergessen!

Projektarbeit

Projekt: Wetterapp

Teamname: Team Jacob

Lehrer: Bernd Schlütter

Klasse: E3FIAE

Tanja Weiser

Dominic Holzwarth

Max Krupp

Inhaltsangabe

[1. Zustandsvergleich 3](#_Toc437262315)

[2. Projektumsetzung 4](#_Toc437262316)

[3. Lessons Learned 6](#_Toc437262317)

[4. Schritte zum Starten des Servers 6](#_Toc437262318)

# 1. Zustandsvergleich

Soll-Zustand

* Aktuelles Wetter anzeigen
* Wettervorhersage anzeigen
* Customizing des Ortes

Ist-Zustand

* Aktuelles Wetter anzeigen
* Wettervorhersage anzeigen
* Customizing des Ortes
* Customizing der angezeigten Elemente möglich
* Customizing der Temperatureinheit (C° und Fahrenheit) möglich

# 2. Projektumsetzung

**Projektplanung**

Zu Beginn der Projektplanung wurde ein Konto bei [www.wunderground.com](http://www.wunderground.com) erstellt um einen API Key zu erhalten. Der API Key wird benötigt um aktuelle Wetterdaten von der Internetseite zu beziehen.

Nachdem es möglich war ein Request an den Wunderground-Server zu senden wurde die Struktur der in der Response enthaltenen JSON Daten studiert.

Diese JSON Daten wurden ebenfalls als Mockdaten abgespeichert und letztendlich zur Entwicklung und auch zum Testen der App genutzt.

Weiterhin wurden Icons ausgewählt, die zur Darstellung des Wetters dienen sollen.

Zur Versionsverwaltung soll Github verwendet werden, somit war es notwendig ein Repository zu erstellen.

Abschließend wurden erste Aufgaben zur Entwicklung des UI aufgeteilt.

**Entwicklungsumgebung**

Als Entwicklungsumgebung wurde Eclipse mit einem Git Plugin verwendet,

Benötigt wurde außerdem eine lokale OpenUI5 Bibliothek und ein Node.js Server zum Ausführen der App.

**Entwicklung der App**

**Testen**

Um die Qualität und die Funktionalität der App zu testen wurden OPA5 und QUnit Tests implementiert.

OPA5

OPA5 (One Page Acceptance Tests) ist eine API für OpenUI5 Steuerelemente. Es wird genutzt um Benutzer Interaktionen, Navigation und Data Binding in Verbindung mit OpenUI5 zu testen.

OPA5 Tests werden, wie die App, mit JavaScript entwickelt. Das bietet den Vorteil schnell und einfach auf JavaScript Funktionen zugreifen zu können. Weiterhin ist somit auch eine gute Integration mit OpenUI5 gegeben.

Ein OPA5 Test besteht aus Arrangements, Actions und Assertions. Für die Actions und die Assertions wurde jeweils eine JavaScript Datei angelegt und die benötigten Funktionen implementiert. Unter die Kategorie der Actions fallen hier beispielsweise Funktionen wie das Klicken auf einen Button, wogegen eine Assertion eine Behauptung, wie z.B.: den Text auf einem Button überprüft.

|  |  |
| --- | --- |
| **OPA5 Test** | **Beschreibung** |
| WEATHER TILE - Check Labels |  |
| CURRENT WEATHER ICON: Find icon and check image name |  |
| GUST TILE - Check Labels |  |
| GUST COMPASS - Find compass |  |
| PRECIP HUMIDITY TILE - Check Labels |  |
| HUMIDITY - Find humidity circle progress |  |
| RAINOMETER - Find rainometer |  |
| FORECAST 1 - Check Labels |  |
| FORCAST WEATHER ICON: Find icon and check image name |  |
| FORECAST 2 - Check Labels |  |
| FORCAST WEATHER ICON: Find icon and check image name |  |
| FORECAST 3 - Check Labels |  |
| FORCAST WEATHER ICON: Find icon and check image name |  |
| CUSTONMIZE BUTTON- Find customize button by id, click on it and search for dialog by id |  |
| CUSTOMIZE DIALOG - Open dialog, change town and save |  |
| CUSTOMIZE DIALOG - Open dialog, click on radiobutton Fahrenheit and save |  |
| CUSTOMIZE DIALOG - Open dialog, click on radiobutton Celcius and save |  |
| CUSTOMIZE DIALOG - Open dialog, change state of forecast switch and save |  |
| CUSTOMIZE DIALOG - Open dialog, change state of humidity switch and save |  |
| CUSTOMIZE DIALOG - Intervall |  |

QUnit

Um fehlerhafte Änderungen des Codes zu erkennen wurden mehrere Unit Tests entwickelt. Diese wurden mit QUnit, einem Framework zum Testen von JS Code, umgesetzt. Es wurden die Formatter Funktionen getestet. Als Formatter Funktionen bezeichnet man Funktionen, die Daten aus dem Model für die Anzeige auf dem UI aufbereiten. Zu diesen zählen folgende Funktionen:

WeatherIcon: Diese Funktion nimmt den Namen des Wetters entgegen z.B. *fog* und gibt den Pfad des svg Icons zurück z.B. *./icons/SVG/sw-09.svg* . Bei dieser Funktion wurden alle möglichen Wettersituationen durchgetestet.

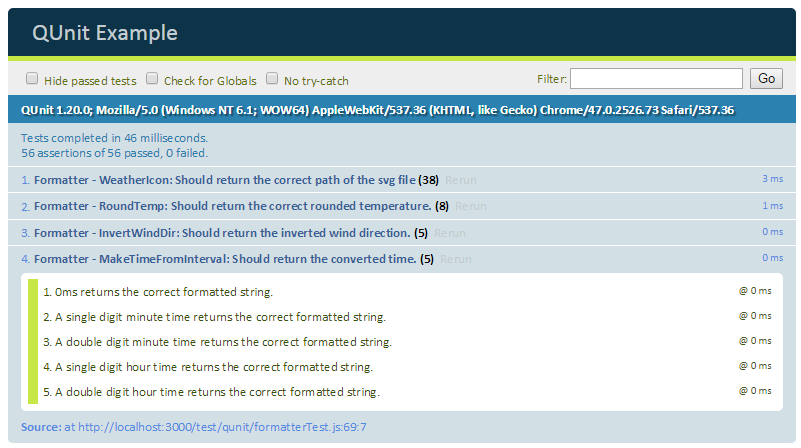
RoundTemp: Diese Funktion nimmt einen String, der eine Dezimalzahl enthält, entgegen und gibt eine gerundete Zahl zurück. Bei diesem Test wurden die Testdaten so gewählt, dass pro mögliche Länge (zweistellig, einstellig, einstellig negativ und zweistellig negativ) eine Test Zahl aufgerundet und eine abgerundet wird.

InvertWindDir: Die Funktion nimmt eine Windrichtung in ° entgegen und gibt die gegenüberliegende Position zurück. Dies wird vom „Windanzeigekompass“ benötigt um aus der Windrichtung die Richtung in die der Wind weht zu ermitteln. Bei diesem Test wurde als Testdaten…

* eine Zahl die < 180° ist gewählt, da das Ergebnis normal berechnet werden kann
* eine Zahl die > 180° ist gewählt, da das Ergebnis wieder bei 0° anfangen muss und nicht > 359°.
* die Zahl 0°, da sie die kleinste mögliche Zahl ist.
* die Zahl 180°, da das Ergebnis 0° und nicht 360° sein soll.
* Die Zahl 360°, da sie die größte mögliche Zahl ist.

MakeTimeFromInterval: Die Funktion nimmt eine ms Zahl entgegen und rechnet diese in das Format Stunden:Minuten um. Sie wird benötigt für die Eingabe des Refresh-Intervalls im Konfigurationsbereich. Bei diesem Test wurden als Testdaten…

* die Zahl 0, da sie die kleinstmöglichste Zahl ist
* eine ms Zeit, die 0 Stunden und eine einstellige Minutenzeit zurückgibt.
* eine ms Zeit, die 0 Stunden und eine zweistellige Minutenzeit zurückgibt.
* eine ms Zeit, die eine einstellige Stundenzeit zurückgibt.
* eine ms Zeit, die eine zweistellige Stundenzeit zurückgibt.

Die Tests können über den Browser aufgerufen werden und geben Details zu den einzelnen Tests (und eventuellen Fehlern).

Struktogramm der Methode „**checkDataChanged**“ der **Main.controller.js**



# 3. Lessons Learned

* Ausarbeitung der Projektstruktur und Konzepte hat zu lange gedauert
* Aufgabenverteilung (?)
* Anders als geplant verlaufen, bzgl. Änderungen während der Entwicklung
* Keine Designvorgabe, daher lange Vorbereitung nötig

# 4. Schritte zum Starten des Servers

1. Öffnen der Kommandozeile .
2. Mit der Kommandozeile in den Ordner der App navigieren.
3. Vor dem ersten Starten des Servers muss der Befehl „npm install“ ausgeführt werden.
4. Der Server kann jetzt mit dem Befehl „node server.js“ gestartet werden.

# 5. Quellen

[One Page Acceptance Tests (OPA5)](http://help.sap.com/saphelp_scm700_ehp03/helpdata/en/26/96ab50faad458f9b4027ec2f9b884d/content.htm?frameset=/en/22/f175e7084247bc896c15280af9d1dc/frameset.htm&current_toc=/en/5c/be4e5b4a19479a92b1d32ff23b7b63/plain.htm&node_id=451&show_children=false)