Projektarbeit

Entwicklung einer Wetterapp mit OpenUI5



|  |  |
| --- | --- |
| Entwickler: | Dominic Holzwarth,  Max Krupp,  Tanja Weiser |
| Team: | „Team Jacob“ |
| Lehrer: | Herr Schlütter |
| Klasse: | E3FIAE |
| Projektzeitraum: | 29.09.2015 – 21.01.2016 |

Inhaltsangabe

[1. Zustandsvergleich 3](#_Toc440616487)

[2. Projektumsetzung 3](#_Toc440616488)

[Projektplanung 3](#_Toc440616489)

[Entwicklung der App 3](#_Toc440616490)

[Entwicklung des Servers: 4](#_Toc440616491)

[Übersicht der wichtigsten Dateien 4](#_Toc440616492)

[Testen 5](#_Toc440616493)

[OPA5 5](#_Toc440616494)

[QUnit 6](#_Toc440616495)

[3. Struktogramm 9](#_Toc440616496)

[4. Lessons Learned 9](#_Toc440616497)

[5. Installation der App 10](#_Toc440616498)

[Github Repository 10](#_Toc440616499)

[Installation von node.js 10](#_Toc440616500)

[6. Starten des Servers und der App 10](#_Toc440616501)

[Starten des Servers per Konsole 10](#_Toc440616502)

[Starten der App 10](#_Toc440616503)

[7. Einstellungen der App 10](#_Toc440616504)

# 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 (Celsius und Fahrenheit) möglich

# Projektumsetzung

### Projektplanung

Zu Beginn der Projektplanung wurde ein Konto bei [**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 Daten wurden ebenfalls als Mockdaten abgespeichert und letztendlich zur Entwicklung und auch zum Testen der App genutzt. Weiterhin wurden Icons[[1]](#footnote-1)  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. Als Entwicklungsumgebung wurde Eclipse mit dem Plugin EGit verwendet. Benötigt wurden außerdem eine lokale OpenUI5 Bibliothek und ein Node.js Server zum Ausführen der App.

### Entwicklung der App

Die Wetter-Webapp wurde mit der kostenlosen Version des JavaScript-Frameworks SAPUI5 entwickelt. Sie ist aufgebaut mit einer index.html Seite, die das UI5-Framework lädt. Darauf folgt ein Script, das mithilfe des UI5-Framework eine UI5-App in den HTML-Body einfügt. Eine App kann aus einer oder mehreren Views bestehen. Dem Model-View-Controller Konzept entsprechend hat eine View einen Controller und ein oder mehrere Modelle. Eine View kann als JavaScript- oder XML-View umgesetzt werden. Entsprechend der aktuellen SAPUI5 Best Practice entschieden wir uns für die Verwendung von XML-Views. In XML-Views können verschiedene UI5-Controls definiert und ihre Attribute gesetzt werden. Die Attribute können fest definiert werden oder aus einem Model  ausgelesen werden. Ändert sich das Attribut im Model, so wird das Control automatisch aktualisiert.   Ein UI5-Control ist ein JavaScript-Objekt, das mithilfe der Attribute DOM-Strukturen generiert, diese mit CSS ausstattet und einfache Funktionalitäten zur Verfügung stellt. Ein einfaches UI5-Control ist z.B. ein Button. Wenn er angeklickt wurde, ruft er eine Funktion auf, die im XML-View angegeben wurde und sich im JavaScript-Controller der View befindet.

### Entwicklung des Servers:

Um den Verbrauch der durch den Wunderground-Service vorgeschriebenen Request-Anzahl für das kostenlose Konto zu minimieren, wurde zusätzlich zur App selbst ein Node.js-Server entwickelt, der die Daten vom Wetterservice zwischenspeichert und auch die Einstellungen der Anwendung verwaltet.

Der Server besteht aus zwei Dateien:

Die Datei server.js ist der Einstiegspunkt der Anwendung. Hier wird der Express-Server gestartet, um auf dem Port 3000 zu horchen, und es wird die Datei service.js eingebunden und instanziiert, um die anwendungsspezifische Logik auszulagern.

Der Service selbst besitzt als Membervariablen die folgenden Einstellungen:

|  |  |
| --- | --- |
| SERVICE\_URL | An diese URL wird das HTTP-Request geschickt, um Daten zu holen. |
| DEBUG\_SERVICE\_URL | An diese URL wird das HTTP-Request geschickt, wenn der Server im Debug-Modus gestartet wird. |
| APIKEY\_FILE | Diese Datei enthält den API-Key vom Wunderground-Service |

Der Server nutzt das Node.js-Package „Express“, das es erlaubt, relativ einfach einen HTTP-Webservice mit benutzerdefinierten Routen zu erstellen.

### Übersicht der wichtigsten Dateien

|  |  |
| --- | --- |
| Filename | Beschreibung |
| /webapp/  index.html | Lädt das SAPUI5 Framework und die „Display“ View. |
| /webapp/view/  Display.view.xml | Sier enthält die Struktur der SAPUI5-Controls. |
| /webapp/view/  Formatter.js | Er enthält Funktionen, die aufgerufen werden, um die Daten zum Anzeigen in der XML-View vorzubereiten. |
| /webapp/controller/  Main.controller.js | Er enthält Funktionen, die bei der Interaktion mit der App aufgerufen werden. Die onInit-Funktion wird automatisch aufgerufen und lädt das Konfigurations- und Wettermodel. |
| /webapp/view/  SettingsPopover.fragment.xml | Es enthält eine XML-Struktur der Controls, die im Konfigurations-Popover angezeigt werden. |
| /webapp/style/  style.css | Custom CSS zum Verbessern des Aussehens. |

### 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 Benutzerinteraktionen, 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.

|  |  |
| --- | --- |
| Weather Tile | |
| Check Labels | Überprüft alle Labels und deren Werte |
| Find current weather icon and check image name | Sucht das Icon per ID und überprüft den Icon Pfad |
| Gust Tile | |
| Check Labels | Überprüft alle Labels und deren Werte |
| Find compass | Überprüft die Sichtbarkeit des Kompasses |
| Precip / Humidity Tile | |
| Check Labels | Überprüft alle Labels und deren Werte |
| Find humidity circle progress | Überprüft die Sichtbarkeit des Humidity Circle Progress Controls und dessen Wert |
| Find rainometer | Überprüft die Sichtbarkeit des Rainometer Controls |

|  |  |
| --- | --- |
| Forecast 1-3 | |
| Check Labels | Überprüft alle Labels und deren Werte |
| Find weather icon and check image name | Sucht das Icon per ID und überprüft den Icon Pfad |
| Custom Settings | |
| Find customize button by id, click on it and search for dialog by id | Überprüft, ob der Dialog geöffnet wird, nachdem auf das Icon geklickt wurde |
| Open dialog, change town and save | Ändert in den Einstellungen die Stadt und überprüft die Änderungen nach dem Speichern |
| Open dialog, click on radiobutton Fahrenheit and save | Ändert die Einheit der Temperatur und überprüft nach dem Speichern die Änderungen auf den Tiles |
| Open dialog, click on radiobutton Celcius and save | Ändert die Einheit der Temperatur und überprüft nach dem Speichern die Änderungen auf den Tiles |
| Open dialog, change state of forecast switch and save | Ändert die Sichtbarkeit der Wettervorhersage und überprüft nach dem Speichern die Änderungen auf dem UI |
| Open dialog, change state of humidity switch and save | Ändert die Sichtbarkeit der Luftfeuchtigkeit und überprüft nach dem Speichern die Änderungen auf dem UI |
| Intervall | Ändert die Intervallzeit, speichert und überprüft die Änderungen nochmals in den Einstellungen |

Die verwendeten Testdaten sind Echtdaten, die als Antwort vom Wunderground-Server auf eine Anfrage kamen. Diese wurden im Projekt in einem JSON-Modell gespeichert.

Ausgeführt werden die Tests nach dem Start des Node.js Servers über folgende URL **http://localhost:3000/test/opa/index.html?test=true**

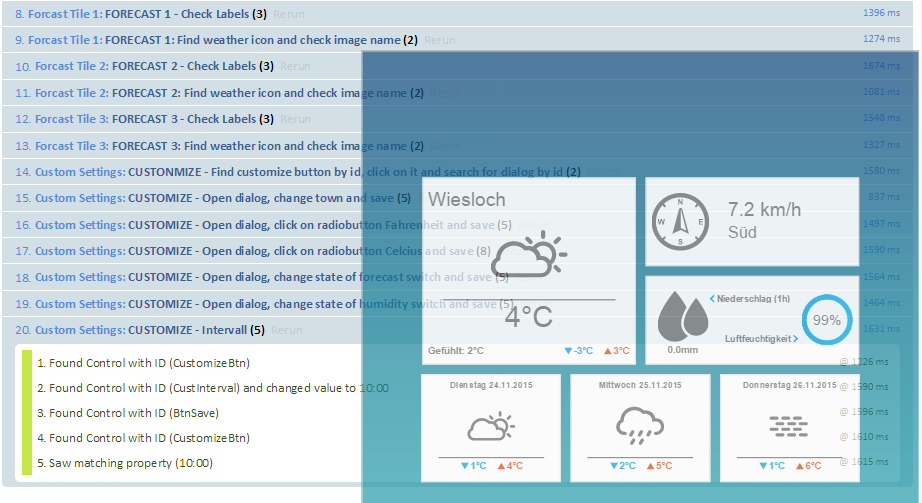


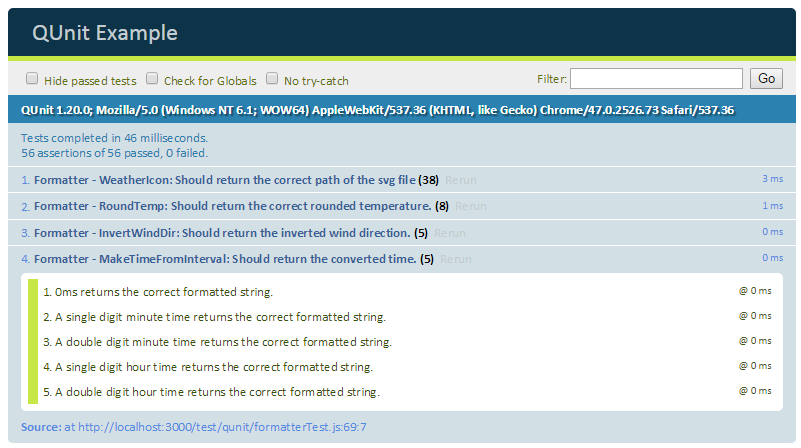
Abbildung 1 -  OPA5 Tests im Browser

### 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:

|  |  |
| --- | --- |
| Funktion | Beschreibung |
| 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 Testzahl aufgerundet und eine abgerundet wird |
| InvertWindDir | Die Funktion nimmt eine Windrichtung in Grad 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 kleinstmögliche 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).

Abbildung 2 – QUnit Tests im Browser

# Struktogramm

Methode: **checkDataChanged**

Controller: **Main.controller.js**



Abbildung 3 – Struktogramm der Methode CheckDataChanged

# Lessons Learned

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

# Installation der App

### Github Repository

Download: <https://github.com/d060270/openui5_weather/tree/master>

### Installation von node.js

Um die App benutzen zu können muss man node.js installieren.

Download: <https://nodejs.org/en/>

Vor dem ersten Start der App muss im Verzeichnis der App, in dem auch die server.js liegt,  per Konsole der Befehl npm install ausgeführt werden.

# 6. Starten des Servers und der App

### Starten des Servers per Konsole

1. Öffnen der Konsole im Verzeichnis der App
2. Starten des Servers mit dem Befehl node server.js

### Starten der App

1. Öffnen der URL **http://localhost:3000/**

# Einstellungen der App

Um Einstellungen in der App vorzunehmen kann man auf das Zahnrad in der rechten oberen Ecke klicken. Das Zahnrad wird erst sichtbar, wenn man mit dem Cursor darüberfährt.

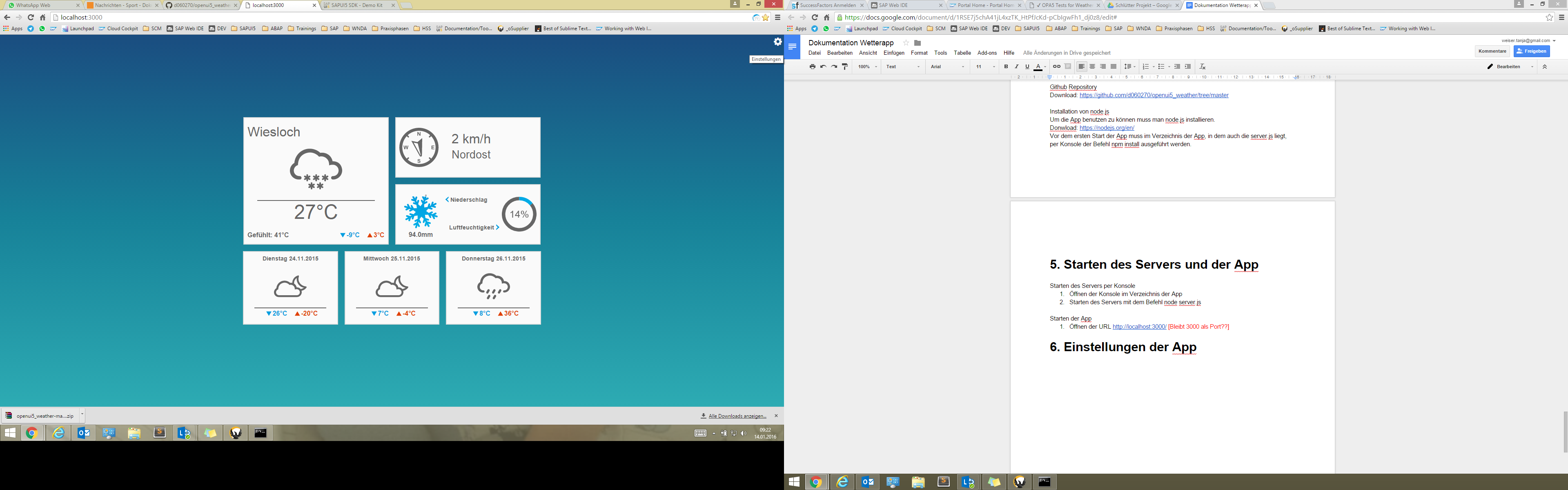


Abbildung 4 -  Einstellungsicon

Einstellungsmöglichkeiten

* Ändern der Stadt
* Ändern des Aktualisierungsintervalls
* Änderung der Temperatureinheit
* Anzeigen bzw. Ausblenden der Wettervorhersage und der Luftfeuchtigkeit

1. <http://www.danvierich.de/> ; <http://www.flaticon.com/free-icon/snowflake_95900#term=snowflake&page=5&position=49> [↑](#footnote-ref-1)