|  |
| --- |
| ildergebnis fÃ¼r klima |
| Abschlussdokumentation  Klima-Viewer  **Projektteam KlimaViewer**  Ivo Kozina  Ohran Mujkic  Version 1.0  Bern, 23. Dezember 2018 |
|  |

Inhaltsverzeichnis

[1 Zweck des Dokuments 3](#_Toc533366448)

[2 Projektkontext 3](#_Toc533366449)

[3 Rahmenbedingungen, organisatorischer Kontext 3](#_Toc533366450)

[4 Bewertung der Funktionalen Anforderungen 4](#_Toc533366451)

[5 Design 5](#_Toc533366452)

[6 Datenmodell 6](#_Toc533366453)

[6.1 Zusammenspiel der Komponenten 6](#_Toc533366454)

[6.2 Herausforderungen 6](#_Toc533366455)

[7 Projektstruktur 7](#_Toc533366456)

[8 Namenkonventionen 7](#_Toc533366457)

[9 Ausblick 8](#_Toc533366458)

[11 Glossar 9](#_Toc533366459)

[12 Abildungsverzeichnis 10](#_Toc533366460)

[13 Anhang 11](#_Toc533366461)

[13.1 Projektbeschreibung 11](#_Toc533366462)

[14 Versionskontrolle 12](#_Toc533366463)

# Zweck des Dokuments

Der Hauptzweck dieses Dokument ist es die erreichten Ziele sowie die Anforderungen für das Projekt „KlimaViewer“ aufzuführen und zu vergleichen, sowie die verschiedenen Arbeitsschritte und Entscheidungen näher zu bringen.

# Projektkontext

Mit dem Projekt wird eine Webapplikation entwickelt, die es möglich macht Klimaveränderungen über einen bestimmten Zeitraum für die Nutzer verständlich, informativ sowie attraktiv darzustellen. Dabei soll der Verlauf der Klimaveränderung in grafischen Darstellungen erfolgen. Die grafischen Darstellungen können ohne bestimmte Vorkenntnisse interpretiert werden. Die Nutzer dieser Webapplikation können mit echten Klimadaten verschiedene Auswertungen und Trends erstellen. Zudem ermöglicht die Webapplikation das Aufrufen von aktuellen Wetterdaten. Somit wird ein neues Tool entwickelt, welches auf den bisherigen gesammelten Daten aufbaut. Das Tool soll als Prototyp dienen und keine produktive und vollendete Software darstellen.

# Rahmenbedingungen, organisatorischer Kontext

Die Realisierung für dieses Projekt erfolgt im Rahmen des Moduls „Projekt 1“. Die Umsetzung erfolgte hauptsächlich mit der Technologie JavaScript, da wir uns auf das MEAN Stack stützten. Die Lösung ist auf einer virtuellen Maschine im BFH Netz angesiedelt und steht über eine URL zur Nutzung.

# Bewertung der Funktionalen Anforderungen

Hier werden noch einmal alle funktionalen Anforderungen angezeigt, um aus diesen ein Fazit zu ziehen.

Attribute:

ID: eindeutige Identifikation

Status: Entwurf / Geprüft / Freigegeben

Priorität: Muss / Optional P1, P2, P3 / Wunsch (Nice to have)

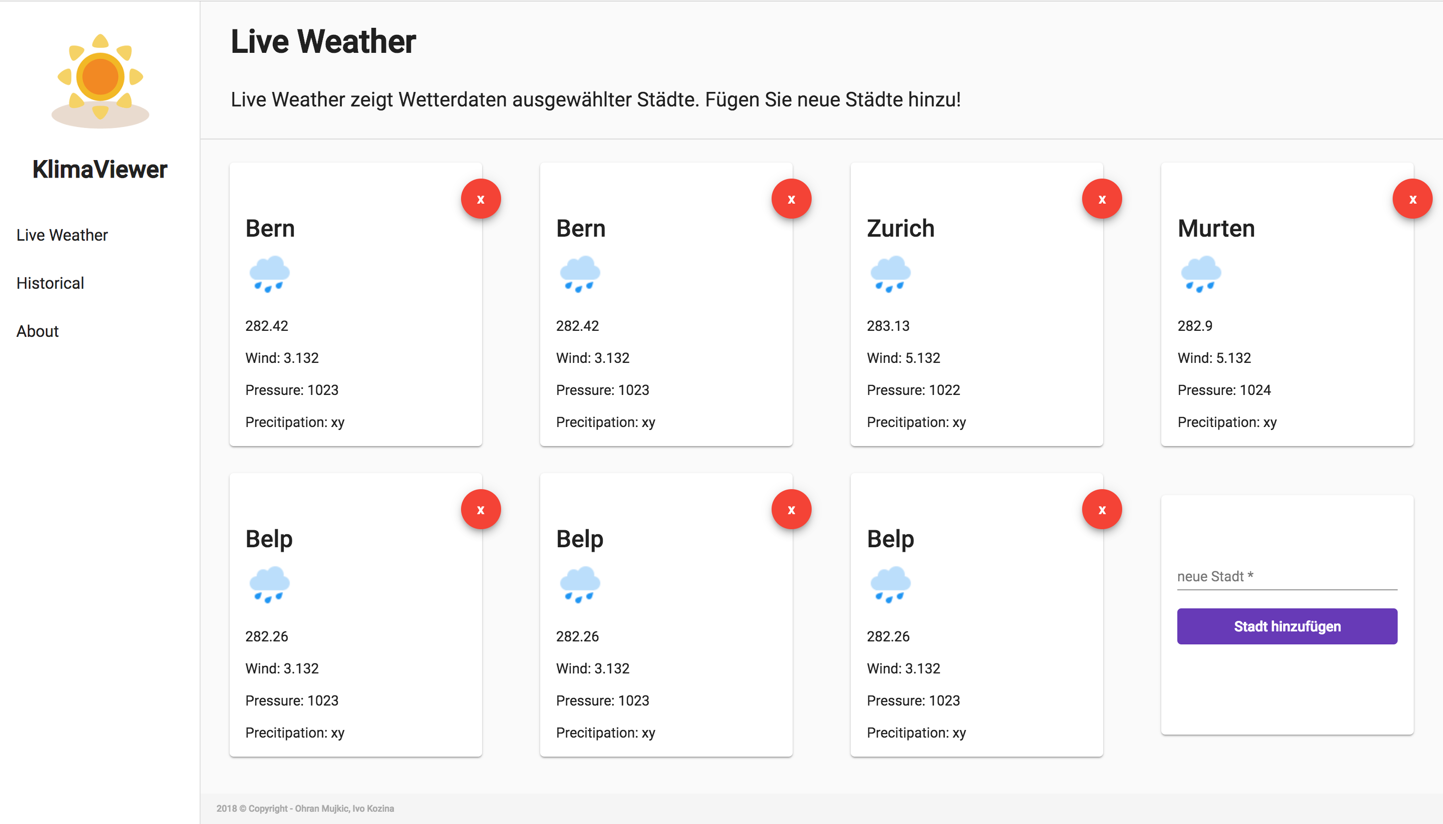
|  |  |  |  |
| --- | --- | --- | --- |
| ID | Status | Priorität | Beschreibung |
| F1 | Freigegeben | M | Der Benutzer kann die Applikation in einem Webbrowser öffnen. |
| F2 | Freigegeben | M | Der Benutzer kann für verschiedene Kennzahlen (Temperatur, Windgeschwindigkeit, Niederschlag sowie Luftdruck) über einen bestimmten Zeitraum in der Vergangenheit eine graphische Darstellung erhalten, welche sich hauptsächlich auf die Veränderung fokussiert. Diese Daten kann er auch als CSV-Datei exportieren. |
| F3 | Freigegeben | M | Der Benutzer kann für verschiedene Städte die aktuellen Wetterdaten (Temperatur, Windgeschwindigkeit, Niederschlag sowie Luftdruck) erfahren. |
| F4 | Freigegeben | M | Der Benutzer kann eine neue Stadt hinzufügen und diese wird in der Webapplikation hinterlegt und weiter getrackt. |

|  |  |
| --- | --- |
| ID | Fazit |
| F1 | Da die Applikation auf einer VM im BFH Netz läuft und mit einem Webserver aufgebaut wurde, ist es dem Benutzer möglich, die Applikation in einem Webbrowser aufzurufen. |
| F2 | Diese Anforderung wurde im Teil „Historical“ der Applikation umgesetzt. Die nötigen Daten dazu kann man bei OpenWeatherMap herunterladen. Da dies jedoch kostenpflichtig ist, haben wir auf unserer VM einen Cronjob erstellt, welcher diese Daten holt und in unserer mongoDB speichert. In unserer Applikation rufen wir dann diese Daten auf und verwenden sie im Chart. |
| F3 | Diese Anforderung wurde im Teil „Live Weather“ der Applikation umgesetzt. In der Übersicht Live Weather ist es möglich die Wetterinformationen von selbst hinzugefügten Städten anzuzeigen. \*\*Ohran\*\* beschrib hie wie das die städt azeigt werde |
| F4 | \*\*Ohran\*\* hie genau glich |

# Design

Beim Design war die Anforderung, das Applikation ansprechend und leicht zu bedienen sein. Deshalb werden hier kurz die Gedanken beim Erstellen der Ansicht der Grundapplikation beschrieben. Einfachheitshalber wurde die Rubrik LiveWeather gewählt:

1



2

3

4

|  |  |
| --- | --- |
| Nr. | Beschreibung |
| 1 | Als Navigation war im Mockup ein Menu gewählt, auf der Seite oben und zentriert. Beim Umsetzen merkten wir jedoch, dass sich eine Sidebar an der Seite viel mehr eignet. Dadurch wurde die Seite in zwei Teile geteilt was dazu führte, dass man stets den Überblick über die anderen Links hatte und zum anderen konnte dadurch der Inhalt übersichtlicher dargestellt werden, |
| 2 | Auf jeder Seite steht oben, wo man sich gerade befindet. |
| 3 | Unter dem Titel der Seite wird auch die Funktion dieser kurz beschrieben. |
| 4 | Die gewählten Städte befinden sich in Kartenähnlichen Boxen. Diese wurden so programmiert, dass sie skalierbar sind und sich an die Seite anpassen. |

Für die Frontend Umsetzung stützen wir und auf das Material Framework. Es bietet viele vorgegebene Elemente, welche auch zum Teil ein vordefiniertes Aussehen besitzen. Dies haben wir uns zu Nutze gemacht und an unser Projekt angepasst.

# Datenmodell

Da hier die Probleme und Lösungen wichtiger sind und bereits in den Anforderungsspezifikationen das Datenbankmodell und das dazugehörige Klassendiagramm gezeigt wurden, wird es nur noch kurz aufgezeigt, um den Gesamtkontext nochmal zu sehen.



Abbildung 1: ERD

## Zusammenspiel der Komponenten

\*\*Ohran\*\* schrib hie wen Api u wen Datebnk brucht werde bli bla blu

## Herausforderungen

Die grösste Herausforderung war, das historische Daten (Daten in der Vergangenheit) bei OpenWeatherMap kostenpflichtig sind. Da dieses Projekt aber nur zur Vorzeige dient, konnten wir uns einig werden, dass wir Daten aus einer kürzeren Zeitperiode selber speichern. Dazu wurde ein Cronjob erstellt, welcher Wetterinformationen von OpenWeatherMap sammelte und in die Datenbank speicherte. Diese Daten wurden dann als « historische » Daten verwendet.

# Projektstruktur

Backend

* Api
  + node\_modules : Speicherort ganzen Library-Package
  + DB .js : URL zur Datenbank
  + package.json : Projekt-Konfiguration
  + server.js : Backend, das eine Verbindung mit der Datenbank herstellt und die Abfragen erhält
* App
  + node\_modules : Speicherort ganzen Library-Package
  + package.json : Projekt-Konfiguration
  + e2e : End-to-End Testing
  + models : Enthält die Models, welche eine Abbildung der Tabellen in der Datenbank sind
  + routes : Der API-Service, welcher die Abfragen empfängt und weiter bearbeitet. Aufgeliedert nach Tabellen
  + src : Source Code der Webapp
    - app
      * services : Services beinhalten verschiedene Funktione, die immer wieder verwendetet werden.
      * Live-weather : Logik und View von der LiveWeather-Seite
      * Historical : Logik und View von der Historical-Seite
      * About: Logik und View von der About-Seite
      * Location.ts : Interface für Location-Objekt
      * WeatherRecord.ts : Interface für WeatherRecord-Objekt
      * App-routing.module.ts : Routing der Web-App
      * App.component.ts : Logik Appübergreifend
      * App.component.html : View Appübegreifend
      * App.module.ts : Alle importierten Libraries
      * App.component.scss : Stylesheet Appübergreifend
      * App.comonent.spec.ts : Unit-Test Appübergreifend
    - Assets : Ablage für Bilder
    - Environments : Konfiguration für verschiedene Enviroments
    - Tests.ts : Tests-Config

\*Screenshot von Package-Explorer von Visual Studio

# Versionisierung

Das Projekt wurde mithilfe von Github realisiert. Somit ist jeder Commit und damit jede Veränderung ersichtlich. Es wurden auch Issues erstellt, um Teilaufgaben zuteilen zu können und somit die offenen Tasks im Auge behalten zu können.

Gitfile: <https://github.com/mujko1/Klima-Viewer.git>

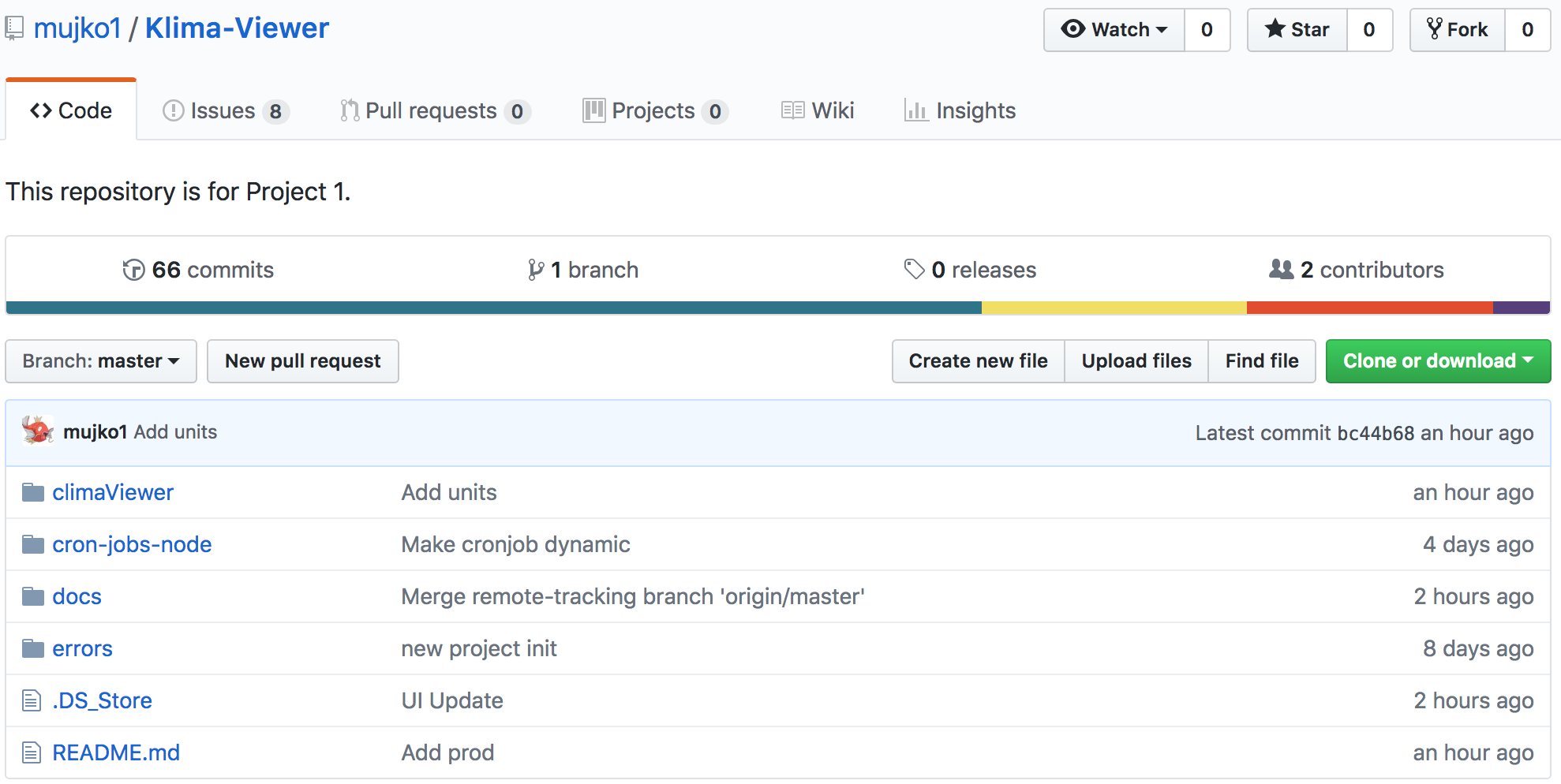
****

Abbildung 2: Auszug aus Github

# Namenskonventionen

Wir haben uns in der Namenskonvention an den üblichen Konventionen des Frameworks gehalten. Dazu haben wir die Styling Guides befolgt, welche von Angular zu Verfügung stehen. <https://angular.io/guide/styleguide>

# Ausblick

* Die Applikation könnte mit einem Userlogin erweitert werden. Somit hätte jeder seine persönliche Ansicht der eigenen Städte.
* Es könnten Anhand der Wetterdaten auch Wetterprognosen erstellt werden.
* Historische Daten könnten komplett verwendet werden, damit auch in weiter Vergangenheit Charts erstellt werden können.
* Die Applikation könnte flexibler umgebaut werden, damit auch die Datennutzung von anderen und verschiedenen Anbietern gleichzeitig möglich ist.

# Glossar

|  |  |
| --- | --- |
| **Begriffe** | **Erklärung** |
| Prototyp | Muster |
| Browser | Programm um nach Webseiten zu suchen, lesen oder verwalten. |
| UI | User Interface, Benutzeroberfläche |
| WorldClim | Plattform, welche Wetterdaten aus der ganzen Welt als Datei zu Verfügung stellt. |
| OpenWeatherMap | Plattform, welche Wetterdaten aus der ganzen Welt über einen API zu Verfügung stellt. |
| API | Programmierschnittstelle |
| REST | Representational state transfer – Software-Architekturestyle meistens für Webservices. |
| JSON | JavaScript Object Notation – Dateiformat meisens im Gebrauch für Webservices. |
| DB | Datenbank – Sammlung und Ablage von Daten. |
| Mean Stack | Kostenloser, Open-Source JavaScript Library zum Erstellen von dynamischen Webseiten. |
| URL | Adressierung einer Webseite |
| Entity-Relationship-Model | Zeigt anhand von einem Diagramm das Grobkonzept einer Datenbank. Hierbei spielen die Tabellen sowie deren Beziehung zueinander die Schlüsselrolle. |
| Cronjob | Ein Skript, welches im Hintergrund regelmässig zu einer vordefinierten Zeit, immer sich ausführt. |
| MongoDB | Das ist eine Datenbankprogramm. |
| CSV | Comma-separated Values – Ist ein Dateityp, dass sich besonders gut für den Export von Daten eignet. |

# Abildungsverzeichnis

[Abbildung 1: ERD 6](#_Toc533367509)

[Abbildung 2: Auszug aus Github 8](#_Toc533367510)

# Anhang

## Projektbeschreibung



# Versionskontrolle

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Beschreibung** | **Autor** |
| X0.1 | 15.12.2018 | Dokument erstellt | Ohran Mujkic |
| X0.2 | 16.12.2018 | Zweck des Dokuments, Projektkontext und Rahmenbedingungen erstellt | Ivo Kozina |
| X0.3 | 18.12.2018 | Bewertung der Funktionalen Anforderungen | Ohran Mujkic |
| X0.4 | 19.12.2018 | Design | Ivo Kozina |
| X0.5 | 20.12.2018 | Datenmodell | Ohran Mujkic |
| V1.0 | 23.12.2018 | Abschlussarbeiten am ganzen Dokument und Freigabe | O. Mujkic und I. Kozina |