

Dokumentation

„Dashboard“

Milen Bonev, Marius Großkopf, Henning Löwe, Nils Mahler,
Michelle Schmidt

ON17-4, 12.April 2019

1. Thema des Projekts	2
2. Ausgangssituation	3
3. Vorgehen	3
4. Anforderungsliste	3
5. Konzeption	3
5.1. Technologie-/Werkzeugauswahl	3
5.2. Entwurf	3
6. Ergebnis des Projekts	4
7. Gruppenreflexion	4
7.1 Herausforderungen	4
7.2. Unterstützung	4
7.3. Lernerfolge / Fazit	4
A. Installationsanleitung	4
B. Benutzerdokumentation	5
C. Individuelle Beiträge	5

1. Thema des Projekts

Im Rahmen des Webanwendungs-Projekt, welches im vierten Semester für das WT3-Modul zur Bearbeitung ansteht, fiel die Wahl des Themas auf ein Dashboard.

Ziel des Projekts ist es, den Umgang mit Git zu verbessern und anhand des Themas die Implementierung von API-Schnittstellen zu erlernen. Dieses Thema war für alle Mitglieder sehr interessant und in Kombination mit einem Dashboard eine gute Motivation. Zusätzlich soll die Arbeit innerhalb eines größeren Teams (5 Mitglieder) erlernt werden und die Unterteilung des Teams in kleinere Arbeitsgruppen aus Experten und Nicht-Experten, um den Wissenstransfer in den Bereichen Design, UX, Programmierung und Projektmanagement zu gewährleisten.

Ziel unserer Anwendung ist es, eine übersichtliche Darstellung von verschiedenen Apps auf dem Laptop zu realisieren. Hierfür hat unser Team, welches aus Marius Großkopf, Henning Löwe, Nils Mahler, Milen Bonev und Michelle Schmidt besteht, die Idee eines Dashboards konzipiert, welches Widgets verschiedener Applikationen wie Wetterinformationen, To-Do-Liste, News oder auch YouTube und Spotify darstellen kann.

Hinter dieser Idee steckt der Gedanke "work smart not hard". Der Nutzer soll schnell eine Übersicht über alle wichtigen Informationen haben und seinen Arbeitsalltag anhand des Dashboards individualisiert und auf seine Bedürfnisse angepasst effizienter gestalten. Wir möchten in der heutigen Zeit eine Möglichkeit bieten, den Fokus des Nutzers auf seine Arbeit zu konzentrieren und z. B. eingehende Änderungen innerhalb eines Projekts umgehend zu erfassen, um zeitnah reagieren zu können.

Das Dashboard kann innerhalb eines Rasters vom Nutzer angepasst werden. So kann er die Größe und Position der unterschiedlichen Kacheln je nach Bedarf anpassen und genauso jedem Widget eine Farbe zuordnen.

Vision

Es sollte im späteren Verlauf die Möglichkeit geben, ein "produktives" Dashboard anzulegen, wo man alle wichtigen Bereiche auf einem Blick hat und als Pendant hierzu soll es möglich sein, ein weiteres Dashboard anzulegen für "Unterhaltung". Hier können Widgets für Soziale Netzwerke angezeigt werden. Es kam außerdem

die Idee auf, ein Flyout-Menü zu erstellen, das eine schnelle Übersicht über alle verfügbaren Widgets bietet. Hier soll der Nutzer die benötigten Widgets per Drag and Drop vom Flyout-Menü in den View-Bereich ziehen können. Dieser Teil der App konnte aus zeitlichen Gründen nicht umgesetzt werden, wird aber weiterhin verfolgt.

Weitere Features für die Zukunft:

- Youtube Videos direkt im Dashboard abspielen können, statt per Klick auf die Website weitergeleitet zu werden
- Channels individuell zum RSS-Feed hinzufügen können (momentan kann ein Widget aus 5 dargestellt werden)
- Im Kalender werden alle Tage des Monats angezeigt und sobald man auf einen Tag klickt, werden die zugehörigen Events des Tages angezeigt

Wir sehen unsere Idee als zukunftsweisend, da dies die heute Anforderung an Multitasking unterstützt und eine optimale zeitliche Nutzung bietet, statt das umständliche Öffnen jedes einzelnen Tools. So kann der Nutzer 20% seiner Zeit nehmen, um optimalerweise mehr als 80% seiner Aufgaben zu erledigen und mehr Zeit für sich selbst hat.

2. Ausgangssituation

Im Rahmen des Moduls WT3 des vierten Semesters und die damit verbundene Entwicklung einer Webanwendung hatten alle Teammitglieder bereits Erfahrung mit dem Framework, welches für die Gruppenanwendung gewählt wurde. Unsere Gruppenkonstellation entstand somit auf Basis der Vorkenntnisse mit Angular, die im vierten Semester erlangt wurden. Folgende Matrix beschreibt die Aufgabenverteilung im Rahmen der App-Entwicklung, wobei anzumerken ist, dass jedes Mitglied einen Programmatischen-Beitrag geleistet hat, der dokumentiert wurde und im Git-Repository ersichtlich ist.

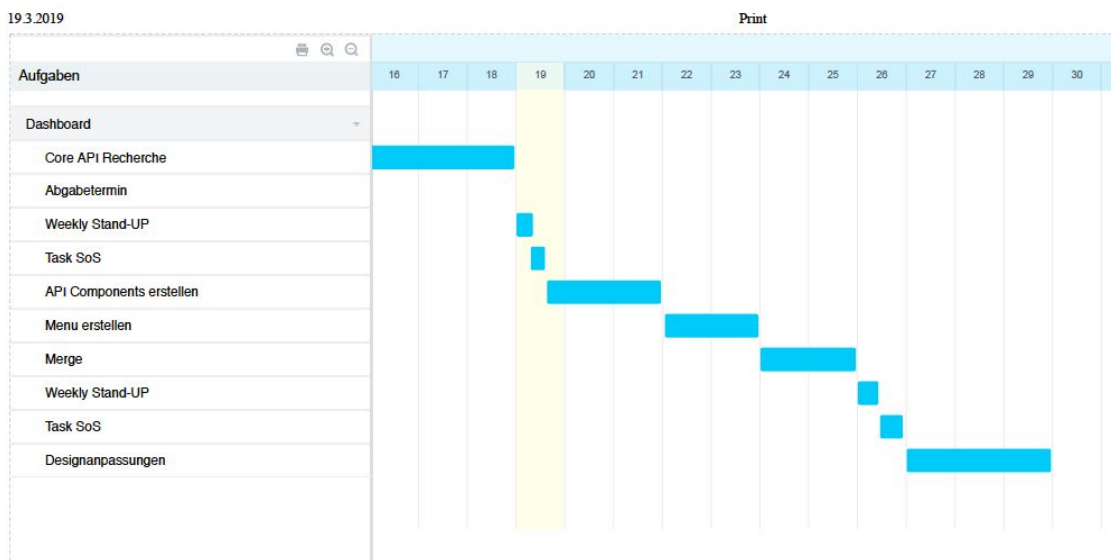
Skillübersicht

Milen	Projektmanager, UX
Henning	Design, Front End
Marius	Backend & Frontend Entwicklung
Nils	Projektmanager, UX
Michelle	UX, Design, Front End

Dadurch, dass jedes Teammitglied einen subjektiven Kenntnisstand vorwies, wurden neben der zugewiesenen Aufgaben Methoden verwendet, um den Austausch von Informationen innerhalb der Gruppe zu gewährleisten und kollektiv den Informationsfluss transparent und Produktiv zu halten (siehe 3. Vorgehen).

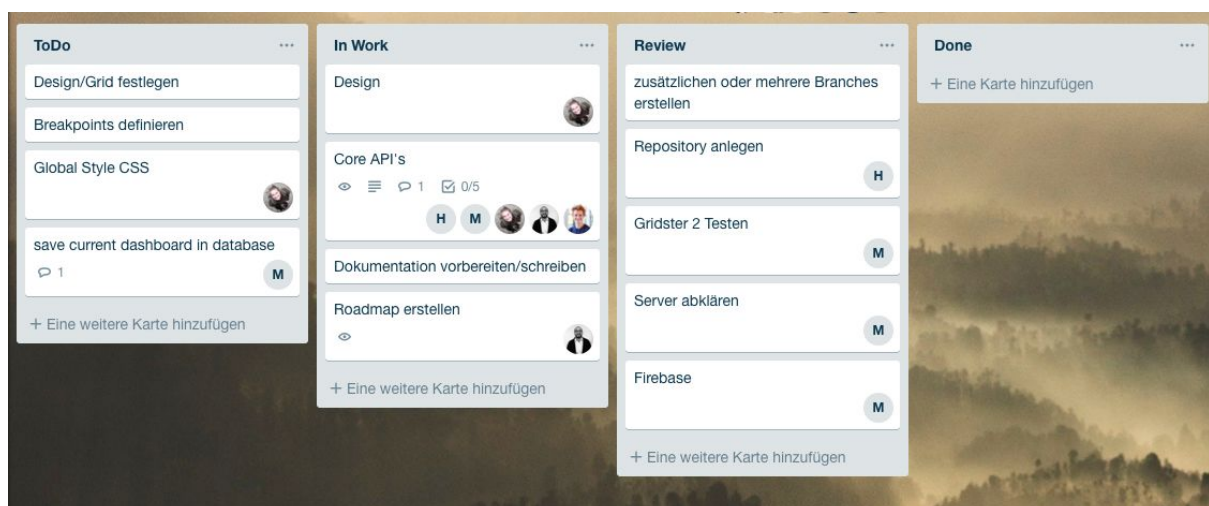
3. Vorgehen

Um effizient entwickeln zu können, haben wir versucht mit agilen Methoden der Softwareentwicklung und Scrum sprintartig zu entwickeln und so wöchentlich Fortschritte zu machen. Dazu zählten Aufgaben wie der Aufbau der Software-Infrastruktur, ein Git-Repository, diverse Branches, die Autorisierung des Users, der Aufbau der Datenbank und sowohl die Recherche, als auch die Implementierung der Kern API-Features des geplanten Dashboards.



Sprintphasen

Die primären Aufgaben wurden auf einem Kanbanboard mit dem Tool Trello erstellt und zeitlich im Rahmen der Projektplanung bearbeitet.



Kanbanboard

Wissenserwerb und Weitergabe stellten wichtige Faktoren unserer Zusammenarbeit dar. Da jedes Mitglied spezifische Aufgaben zu erledigen hatte, fand der subjektive Wissenserwerb teilweise durch direkte Suchanfragen auf Google statt. Diese führten oftmals zu Stackoverflow, Github, Youtube, Foren oder der Angular Homepage oder wurden direkt besucht. Weiterhin wurden hinsichtlich der API-Integration unserer Angular Komponenten alle zugehörigen Dokumentationen hinzugezogen wie beispielsweise der Google Calendar, Spotify, OpenWeatherMap oder YouTube.

Neben der regulären Kommunikation wurden wöchentlich Standup-Sitzungen abgehalten, die der Weitergabe von Wissen dienten und Probleme offen legten, damit die Gruppe in einer anschließender Sitzung Input und Hilfe leisten konnte.

4. Anforderungsliste

Nachfolgend werden alle Anforderungen an die Anwendung aufgelistet mit detaillierter Beschreibung und der Beschreibung, ob das die Anforderung umgesetzt wurde oder nicht.

Nr.	Anforderung	Details	Fertigstellungsgrad
1	Grid-System	Kacheln in ihrer Größe und Position verändern können	Fertig
2	Kacheln verwalten	Create Read Update (Farbe, Position, Größe) Delete	Fertig
3	Einstellungen	Individualisierte Einstellungen <ul style="list-style-type: none">- Farbe der Kacheln individuell anpassen können- Weitere Dashboards anlegen	Nicht umgesetzt <ul style="list-style-type: none">- Nicht umgesetzt- Nicht umgesetzt
4	Einstellungen	Einstellungs-Bereich <ul style="list-style-type: none">- bei Klick auf "Einstellungen" drehen sich alle Kacheln um und zeigen deren Einstellungsmöglichkeiten dar- Für jedes Widgets angepasste Einstellungen	Fertig <ul style="list-style-type: none">- Fertig- Fertig
5	Wetter Widget	Wetter-Widget über X-API responsiv darstellen <ul style="list-style-type: none">- Temperatur, Wetter-Icon und Standort einsehen- Über IP lokalisiert werden- Standort angeben über PLZ	Fertig <ul style="list-style-type: none">- Fertig- Fertig- Fertig
6	Spotify Widget	Spotify-Widget über API responsiv darstellen <ul style="list-style-type: none">- Aktuellen Titel anzeigen- Musik starten/pausieren- Nächsten Titel abspielen- Vorherigen Titel abspielen	Fertig <ul style="list-style-type: none">- Fertig- Fertig- Fertig- Fertig
7	News Widget	News-Widget über API responsiv darstellen <ul style="list-style-type: none">- Auswahl verschiedener News-Feeds (T3N, Kicker,	Fertig <ul style="list-style-type: none">- Fertig

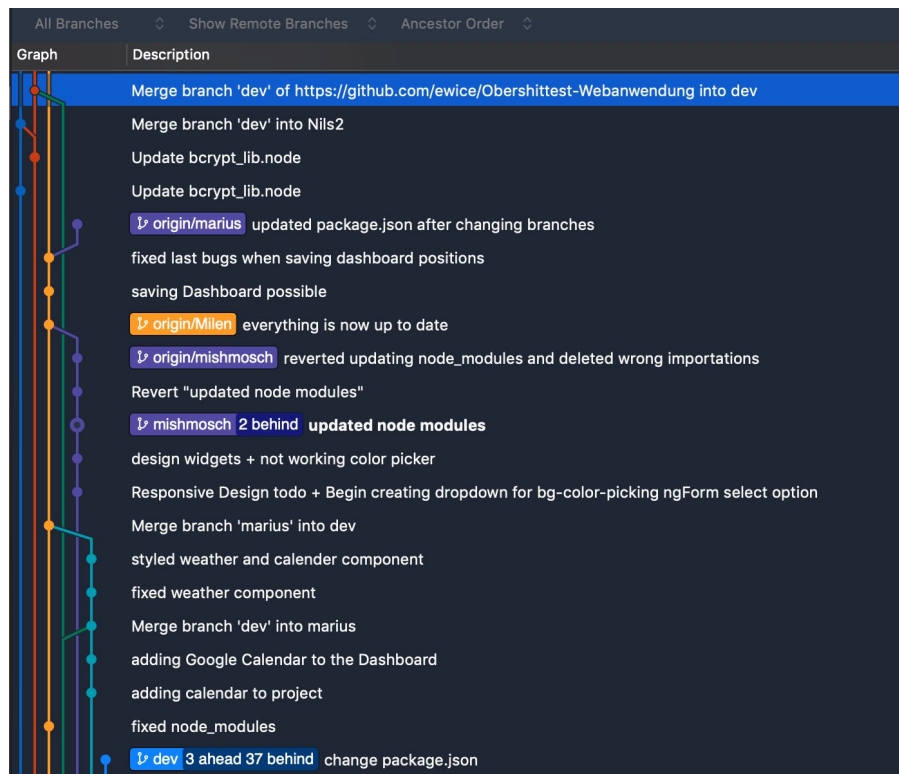
		Spiegel, Handelsblatt, Wirtschaftswoche)	
8	Kalender Widget	Wetter-Widget über Google-Calendar-API responsiv darstellen <ul style="list-style-type: none"> - Termine einsehen - Neuen Eintrag hinzufügen - Eintrag bearbeiten/löschen - Authentifizierung über Google-Account 	Fast fertig <ul style="list-style-type: none"> - Fertig - Fertig - Nicht umgesetzt - Fertig
9	YouTube Widget	YouTube-Widget über API responsiv darstellen <ul style="list-style-type: none"> - Suchfeld - Ansicht Top 10 Videos - Ansicht Thumbnail, Description, Link zum Video - Weiterleitung auf YT 	Fertig <ul style="list-style-type: none"> - Fertig - Fertig - Fertig - Fertig
10	To-Do-Liste	To-Do-Liste programmieren nach dem CRUD-Zyklus <ul style="list-style-type: none"> - To-Dos und erledigte Aufgaben einsehen - Bereiche ausblenden - Neues ToDo Erstellen - Bearbeiten - Löschen - Abhaken 	Fertig <ul style="list-style-type: none"> - Fertig - Fertig - Fertig - Fertig - Fertig - Fertig
11	To-Do-Liste Widget	To-Do-Liste responsiv darstellen	Fertig
12	Login-Funktionalität	Zur Autorisierung / Schutz vor Brute-Force-Attacken	Fertig
13	Logout-Funktionalität	Session wird per Löschen des JWT-Token nach bestimmter Zeit beendet	Fertig
14	Registrierung	<ul style="list-style-type: none"> - Mindestlänge Passwort: 5 Zeichen - Ausbau der Passwortanforderung für mehr Sicherheit 	Fast fertig <ul style="list-style-type: none"> - Fertig - Nicht umgesetzt

5. Konzeption

5.1. Technologie-/Werkzeugauswahl

Das Dashboard basiert auf Angular, ein TypeScript-basiertes Frontend-Webapplikationsframework. Weil wir noch nie in Gruppen an einem größeren Projekt parallel gearbeitet haben, half Angulars Workflow sehr, da es auf Komponenten aufgebaut ist, welche die Funktionalitäten der App trennen und die Kohäsion gewährleisten. Somit konnte jedes Mitglied selbstständig arbeiten, ohne funktional von anderen abhängig zu sein. Weitere Vorteile sind das reaktive Verhalten, wodurch Änderungen automatisch zum neuen Rendern des DOM's führen oder auch die große Anzahl an Bibliotheken. Des Weiteren bietet TypeScript eine Typ Sicherheit, was bedeutet, dass man einer Variable einen festen Datentyp zuweisen kann. Außerdem können Interfaces benutzt werden. Diese beiden Funktionen halfen uns besonders beim Arbeiten mit APIs, da wir so einen Bauplan für die Response anlegen und diese besser verarbeiten konnten. Außerdem konnte so sichergestellt werden, dass keine falschen Daten ins Backend und unsere Datenbank gelangen.

Zusätzlich wurde ein GIT-Repository erstellt mit einem Master- und Dev-Branch (Testing) für das Mergen fertiggestellter Features und Einzelbranches für jedes Teammitglied. Innerhalb der einzelnen Branches wurden entsprechende Komponenten programmiert, die jeweils einer Kachel des Dashboards entsprechen und eine API implementieren.

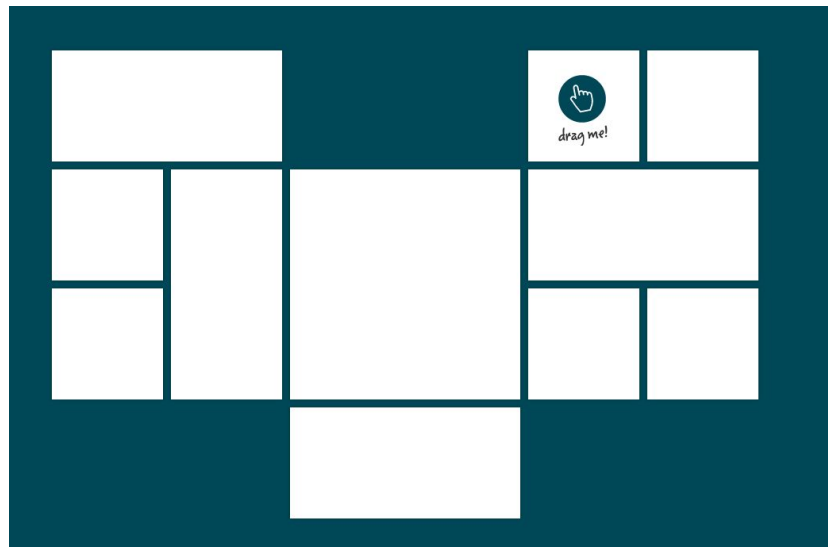


Snippet Git Repo

Für das Backend wurde eine Rest-API mit Express programmiert, die sämtliche Daten in eine MongoDB integriert. Die Wahl der Datenbank fiel auf MongoDB, weil Speichervorgänge nicht wie bei SQL eine bereits erstellte Datentabelle voraussetzen. Vielmehr können Tabellen erstellt werden, wenn beispielsweise ein Event bzw. ein Prozess Variablen speichern soll und durch den Trigger initial eine Tabelle generiert wird. Weiterhin existieren sehr viele Tutorials und eine sehr umfangreiche Dokumentation für den Gebrauch mit NodeJs und Mongoose bzw. MongoDB.

Für die Authentifizierung wurde JWT – Json Web Token verwendet, um einen standardisierten Datenschutz für die Kommunikation zwischen zwei Parteien zu gewährleisten. Dadurch war es möglich Jason Tokens zu decodieren, verifizieren und generieren. Unterstützend wurde die kryptologische Hashfunktion Bcrypt implementiert, die speziell für das Hashen und Speichern von Passwörtern entwickelt wurde.

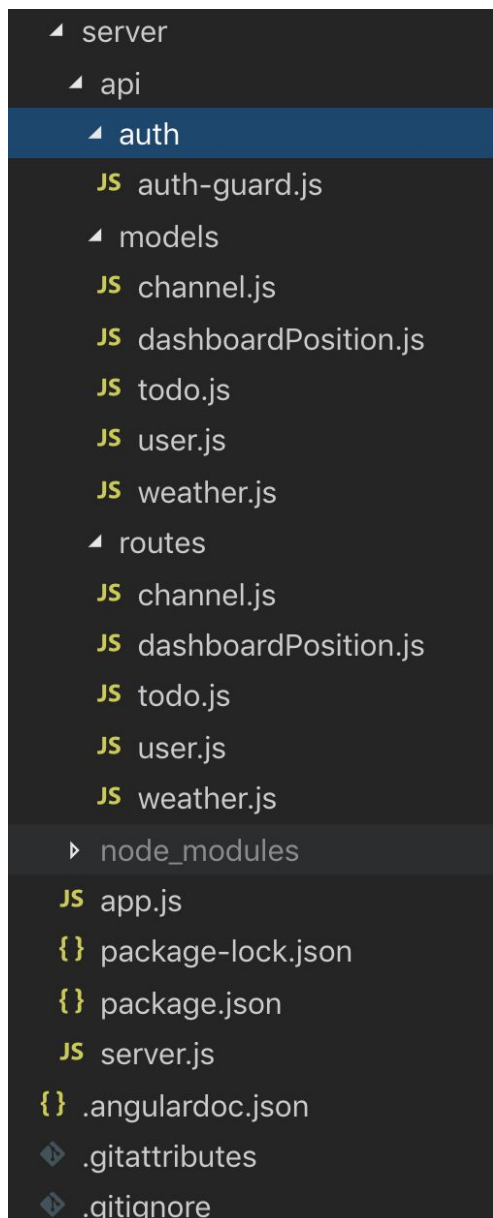
Gridster ist eine Library für intuitive und ziehbare Layout-Elemente, die sich in einem definierbaren Gitter ausbreiten können. Wir haben diese Technologie gewählt, um daraus dynamische Widgets zu entwickeln, die nicht nur einen API-Services auswerten und wiedergeben, sondern die Nutzeroberfläche spezifisch vom User konfiguriert werden kann. Zusätzlich wurde die Bibliothek ngx-flip im Frontend eingebunden, um die Animation der Kacheln darzustellen.



Demo Gridster

Als Code-Editoren wurden Visual Studio Code, PhpStorm oder WebStorm verwendet, weil sie eine Integration zu Git besitzen und Commits somit sofort aus der Entwicklungsumgebung erfolgen können. Teilweise wurden aber auch GitDesktop und Source Tree aus Übersichtsgründen verwendet.

Aus Übersichtsgründen referenziert die Nennung der verwendeten API's auf die Anforderungsliste. Die besagten Schnittstellen wurden Innerhalb der Gridster Komponenten implementiert und laden dort dynamisch den aktuellsten Inhalt oder rufen ihn eventbasiert auf Wunsch des Nutzers ab.



Ordner Auth/ auth-guard.js

- Sichern der Routes in der Rest Api, selbst erstellter Code

Models:

enthält die Baupläne der Daten, die in die Datenbank gespeichert werden, sodass diese validiert werden, selbst erstellter Code

Routes:

enthält die Routes für die jeweiligen Collections

routes/channel.js

- Methoden, um Channels abzurufen/speichern/etc., selbst erstellter Code

routes/dashboardPosition.js

- Methoden um dashboardPositions abzurufen/speichern/etc., selbst erstellter Code

routes/todo.js

- Methoden, um dashboardPositions abzurufen/speichern/etc., selbst erstellter Code

routes/user.js

- Methoden, um User abzurufen/speichern/etc., selbst erstellter Code

routes/weather.js

- Methoden um Wettereinstellungen abzurufen/speichern/etc., selbst erstellter Code

app.js

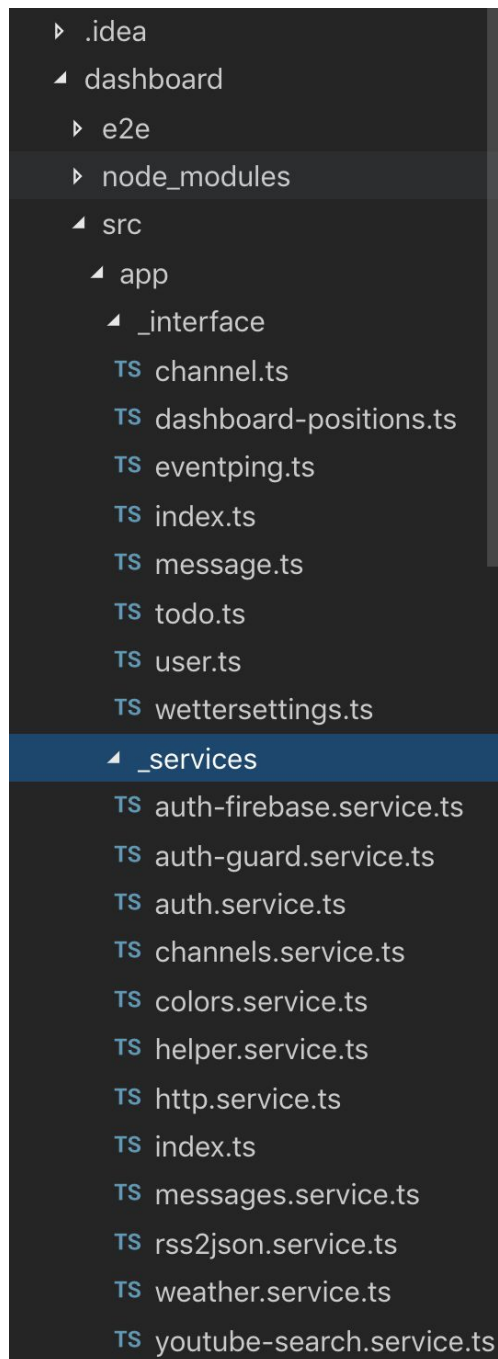
- alle Routes laufen zusammen und alle Libraries werden eingebunden; außerdem wird der Body der Request geparsed und validiert; Header werden gesetzt und CORS umgangen, die Request werden an die richtigen Routes weitergeleitet, selbst erstellter Code

Server.js

- Hier wird der Server an sich erstellt und der Port zugewiesen
- Außerdem wird die Datei app.js importiert
- selbst erstellter Code

package/package-lock.json

- Konfigurationsdatei
- generiert von NPM



Interface:

- Im Ordner `_interfaces` sind alle Baupläne zu unseren Objekten, die in die Datenbank geschickt werden, sodass die Daten richtig ankommen und wir besser auf die Datenfelder zugreifen können
- alle Interfaces wurden selbst erstellt
- die `index.ts` wurde erstellt, um die Interfaces mit einem einzigen Befehl zu laden und dass man nicht immer den ganzen Pfad bei den Imports angeben muss

Services:

`_services/auth-firebase.service.ts`

- die Authentifizierung für Firebase. Dabei wird überprüft, ob der Token von Firebase geliefert wurde, Code stammt aus einem Tutorial (<https://angularfirebase.com/lessons/google-calendar-api-with-firebase/>)

`_services/auth-guard.service.ts`

- Schützt unsere Komponenten und lässt User nur durch, wenn Token aus dem Backend mitgeliefert wird, selbst erstellter Code

`_services/auth.service.ts`

- der Auth-Guard greift auf diesen Service zu, um den Token zu überprüfen, selbst erstellter Code

`_services/channels.service.ts`

- damit kann der Channel geändert werden und der Rss Feed von dem Channel abgerufen werden, selbst erstellter Code

`_services/colors.service.ts`

- enthält unsere Farbpalette, um die Komponenten einzufärben, selbst erstellter Code

`_services/helper.service.ts`

- keine Verwendung mehr, selbst erstellter Code

`_services/http.service.ts`

- Kommunikation mit Server und Apis, Get, Post, Update und Delete Methoden, selbst erstellter Code

`_services/messages.service.ts`

- nimmt Messages entgegen und speichert diese in einem Array, Übergabe an `MessageList`, selbst erstellter Code

`_services/rss2json.service.ts`

- Konvertiert XML in JSON über API-Aufruf, selbst erstellt

`_services/weather.service.ts`

- ruft über Http Service Wetter Api auf und ermittelt Standort über IP-Adresse, selbst erstellt

`_services/youtube-search.service.ts`

- enthält den API-Key und URL und handhabt das Suchen via Queryparameter und den HTTP-Response, angepasster Fremdcode (<https://github.com/assimovt/angular-youtube-search>)

- └─ _template
 - └─ template-clock
 - <> template-clock.compon...
 - 🔗 template-clock.compon...
 - TS template-clock.compon...
 - └─ template-todo
 - <> template-todo.compon...
 - 🔗 template-todo.compon...
 - TS template-todo.compon...
 - └─ template-todo-form
 - <> template-todo-form.co...
 - 🔗 template-todo-form.co...
 - TS template-todo-form.co...
 - └─ template-todo-list
 - <> template-todo-list.com...
 - 🔗 template-todo-list.com...
 - TS template-todo-list.com...
 - TS index.ts
- └─ calendar
 - <> calendar.component.html
 - 🔗 calendar.component.sass
 - TS calendar.component.ts
- └─ channels
 - └─ channel
 - └─ messages-list
 - <> messages-list.compon...
 - 🔗 messages-list.compon...
 - TS messages-list.compon...
 - <> channel.component.html
 - 🔗 channel.component.sass
 - TS channel.component.ts

Templates:

`_template/template-clock`

- Eine Uhr, die nicht implementiert wurde, da sie nicht 100% funktionierte, Erweiterung für die Zukunft, selbst erstellter Code

`_template/template-todo`

- Beschreibt den Aufbau eines To-Do-Punkts mit Checkbox, Label und Delete-Button, selbst erstellter Code

`_template/template-todo-form`

- Feld, um neuen To-Do-Punkt zu erstellen, selbst erstellter Code

`_template/template-todo-list`

- Ausgabe der To-Dos. Einteilung in offene und erledigte Todos, selbst erstellter Code

Calendar:

- Ruft Kalender Events von Google ab, und gibt diese im HTML aus, selbst erstellter Code

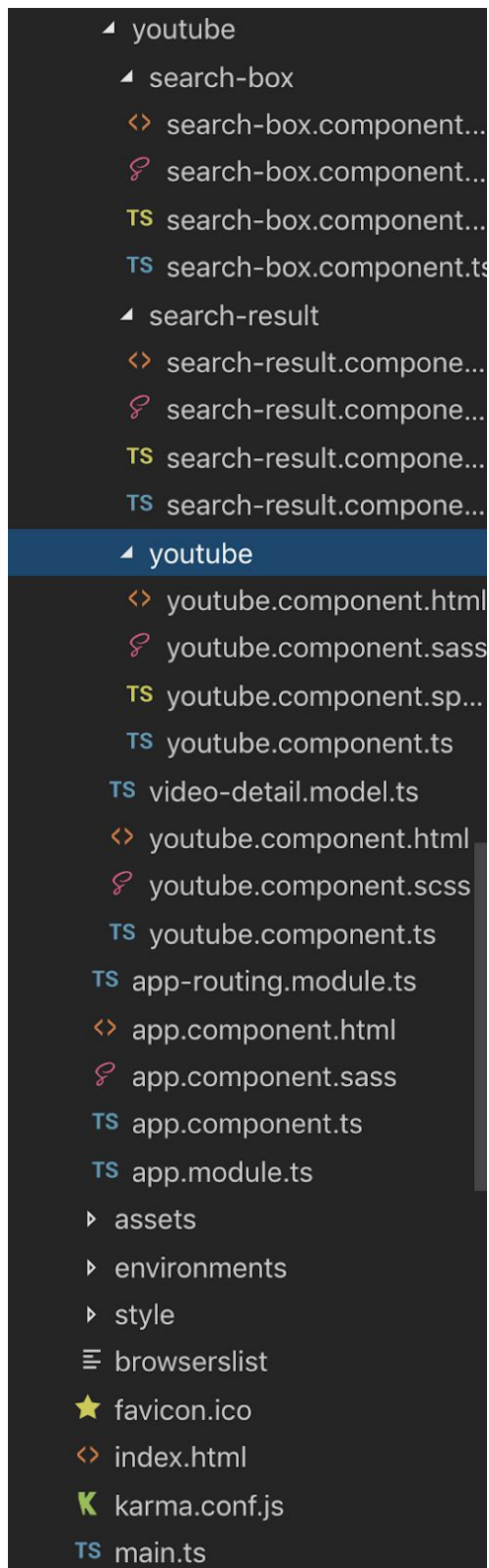
Channels:

`channels/channel/messages-list`

- Ausgabe von den Nachrichten durch Iteration durch Array und Interpolation, selbst erstellter Code

`channels/channel.component`

- enthält die Komponente Messages List und gibt Namen und Logo des Rss Feeds wieder



YouTube:

Search-Box

- Konvertiert den Datenstream in ein Observable

Search-Result

- Implementiert den Input der VideoDetail Ergebnisse

Video-Detail

- Konstruktor für die Metadaten id, title, description, thumbnail und videoUrl

```
▸ assets
▸ environments
▸ style
≡ browserslist
★ favicon.ico
<> index.html
K karma.conf.js
TS main.ts
TS polyfills.ts
🔗 styles.sass
TS test.ts
{} tsconfig.app.json
{} tsconfig.spec.json
{} tslint.json
{} .angulardoc.json
⚙ .editorconfig
{} angular.json
{} package-lock.json
{} package.json
{} tsconfig.json
{} tslint.json
```

index.html

- Grundgerüst der Anwendung, von Angular generiert

style.sass

- Globale Styles, Inhalt von uns, erstellt von Angular

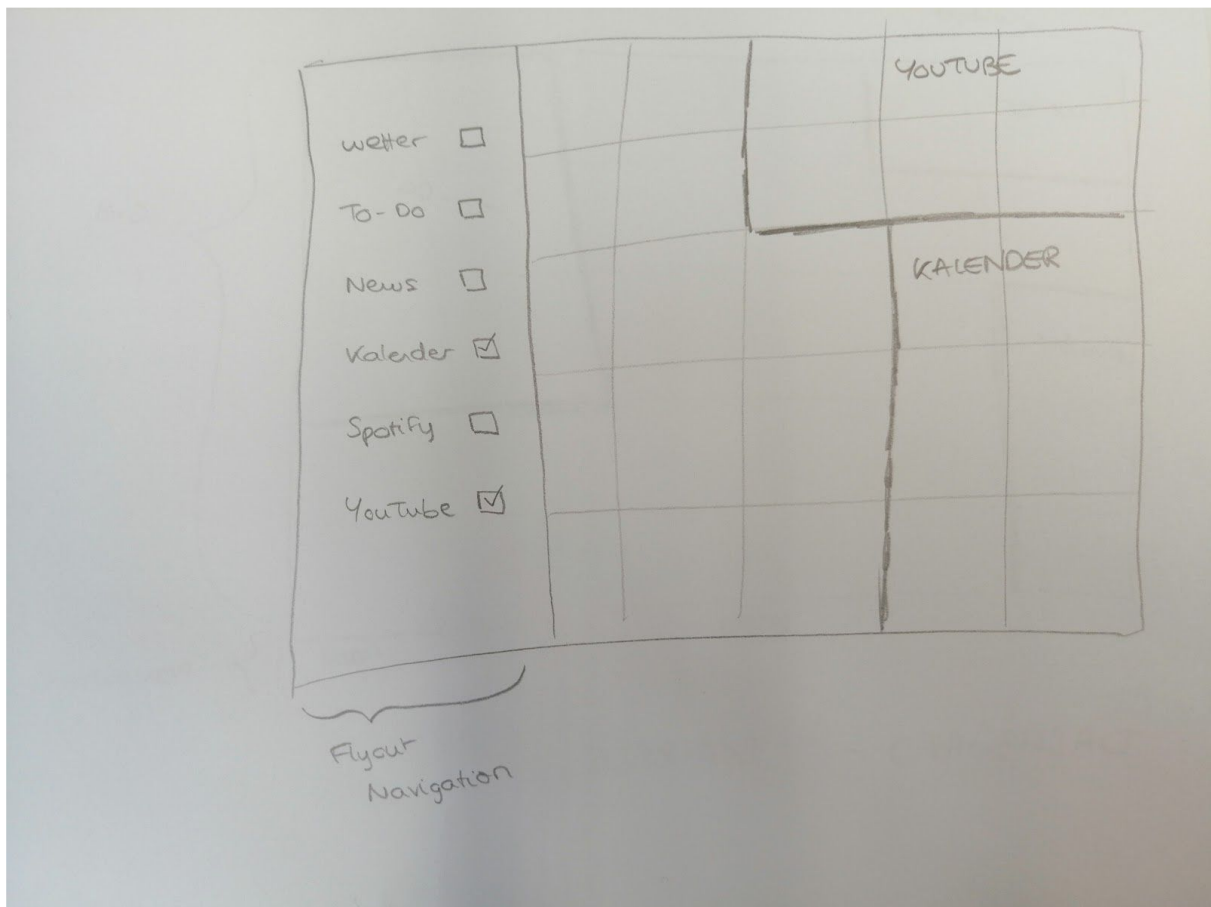
main.ts, karma.conf.js, polyfill.ts, test.ts, tsconfig.app.json, tsconfig.spec.json, tslint.json, angulardoc.json, .editorconfig, angular.json, package.json, package-lock.json, tsconfig.json und tslint.json wurden von Angular erstellt und nicht von uns direkt benutzt oder bearbeitet

5.2. Entwurf

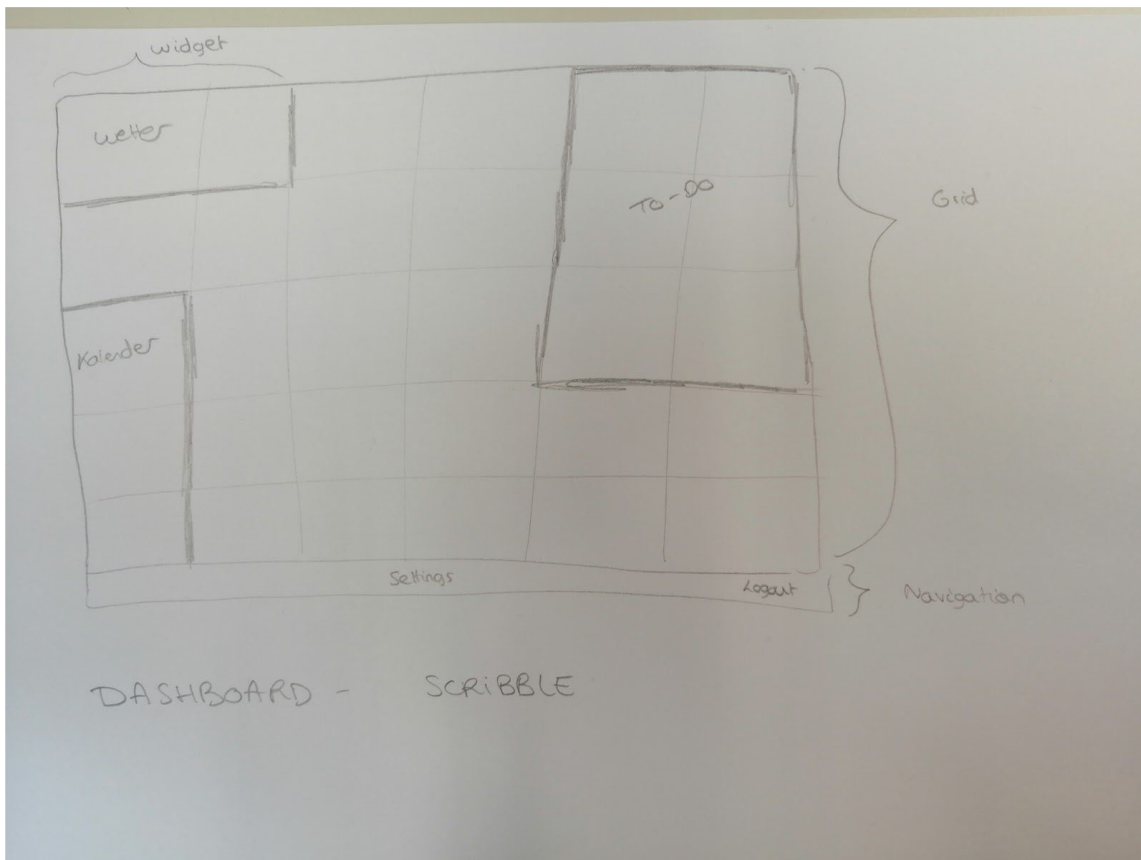
User Interface

Als ersten Schritt wurden Scribbles angefertigt, um die Idee des Dashboards zu visualisieren. Dieser Entwurf diente als Orientierung für die weitere Entwicklung der Anwendung

Scribbles



Flyout-Navigation



Dashboard-Ansicht

Die Widgets bzw. das Grid sollen den gesamten Bildschirm in Anspruch nehmen, um maximalen Platz auszunutzen und eine große Fläche anzubieten. Um von überflüssigen Features abzusehen, könnten weitere Funktionalitäten innerhalb eines Flyout-Menüs dargestellt werden.

Zusätzlich oder alternativ kann im Footer-Bereich eine Leiste mit der Logout-Funktionalität dargestellt werden. Hier können, falls das Flyout-Menü zu viele Kapazitäten in Anspruch nimmt, auch die Einstellungen für die Kacheln hinterlegt werden. So, dass bei Klick auf "Einstellungen" aus dem Dashboard ein Einstellungs-Bereich für alle angezeigten Kacheln angezeigt wird. Zusätzlich wird ein Fenster angezeigt, um dem Dashboard andere Widgets hinzuzufügen.

Styling

Das globale Styling ist in der Datei "_var.sass" wiederzufinden. Wir arbeiten mit Variablen für Farben und Abstände, um das Stylen effizienter und schneller zu gestalten und die App schnell anpassen zu können. Genauso werden Schriftformate festgelegt und Formatierungen, die alle Gridster-Items (Kacheln) betreffen.

Es wird Sass eingesetzt, da es für das Team teils eine Wissenserweiterung teils ein bekanntes Umfeld ist und weniger Schreibaufwand stattfindet. Außerdem können Variablen verwendet werden, sodass bei beispielsweise Farbänderungen nicht jede CSS Anweisung angepasst werden muss, sondern nur die eine Variable.

Das Dashboard besteht aus einer Index.html in der sich der Standard Aufbau einer HTML Seite befindet. Darin ist die App Component enthalten. Dies ist die grundlegendste Komponente des Projektes. Sie enthält das Router Outlet, welches die verschiedenen Komponenten beinhaltet, die durch das Routing ausgewählt werden.

Die Kacheln, die auf dem Dashboard angezeigt werden, wurden in einzelne Komponenten aufgeteilt, sodass weniger bis keine Redundanzen entstehen. Die Komponenten wurden dann je nach Typ des Dashboard Elements angezeigt oder ausgeblendet.

Die Kommunikation mit dem Server und unserer Datenbank, sowie mit anderen Api's erfolgte über einen extra Service, den HTTP-Service. Dieser verwendet den Http Client um Request zu tätigen.

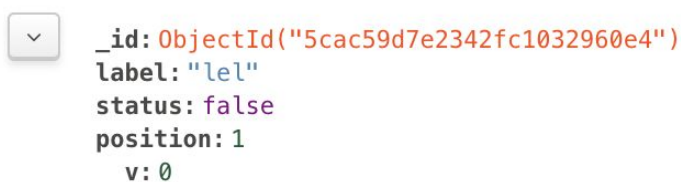
Die Kommunikation zum Google Kalender und die zugehörige Authentifizierung erfolgt über Google Firebase.

In der Datenbank gibt es die Collections Users, Todo, DashboardPositions, Channels und Weathersettings. Um die Todos, Dashboardpositions und Weathersettings dem User zu zuordnen, wird als Fremdschlüssel die UserId mitgespeichert.

Der Aufbau der Dashboardpositions ist von der Library Gridster vordefiniert und wurde so übernommen.

Die Channels sind nicht User gebunden.

Todo Collection

A screenshot of a single item from a 'Todo' collection in a database viewer. The item is displayed as a small card with a dropdown arrow on the left. The data is as follows:

```
_id: ObjectId("5cac59d7e2342fc1032960e4")
label: "lel"
status: false
position: 1
__v: 0
```

User Collection


```
_id: ObjectId("5c8a663cfa309e420c2434e4")
email: "info@dhbw.mosbach.de"
password: "$2b$10$dxM.4ABv/ofC6br6A1a30e0PVsaklWQWiBebsYB.swGVc7bWKNQv2"
name: "info@wit.de"
__v: 0
```

Weather Collection

```
_id: ObjectId("5ca61ac29098d649dc9191cf")
userId: "5c8a663cfa309e420c2434e4"
zip: "74821"
automaticLocation: true
__v: 0
```

Dashboard Positions

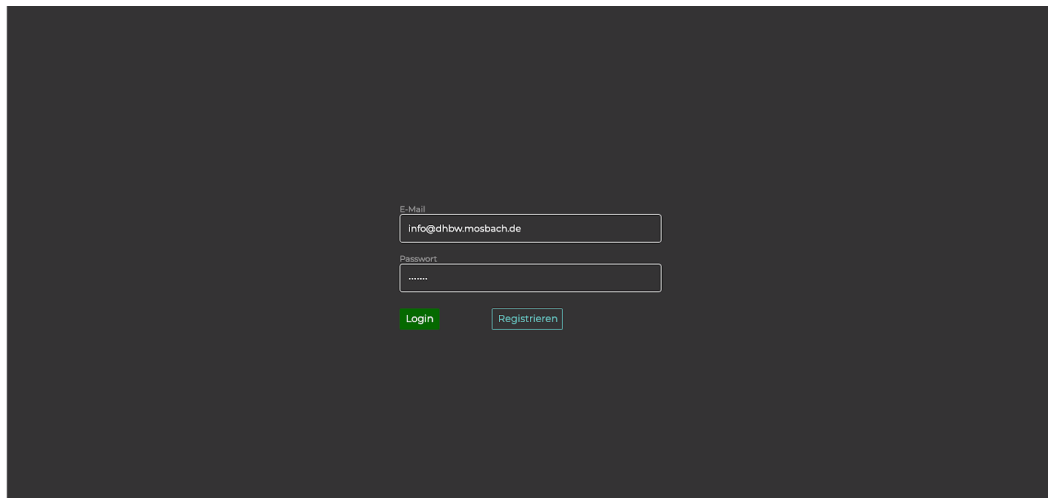
```
_id: ObjectId("5ca92d005d23a04cae76534d")
dashboard: Array
  0: Object
    cols: 2
    rows: 1
    y: 0
    x: 0
    hasContent: true
    type: "weather"
    bg: "$background1"
  1: Object
    cols: 2
    rows: 2
    y: 4
    x: 0
    hasContent: false
    type: "todo"
  2: Object
    cols: 1
    rows: 1
    y: 0
    x: 2
    hasContent: true
    type: "calendar"
  3: Object
    cols: 2
    rows: 2
    y: 1
    x: 1
    hasContent: true
    type: "todo"
    bg: "$background1"
  4: Object
    cols: 3
    rows: 2
    y: 1
    x: 3
    hasContent: true
    type: "youtube"
  5: Object
    cols: "2"
    rows: "2"
    x: "3"
    y: "3"
    hasContent: "true"
    type: "spotify"
    bg: "EAEAEA"
userId: "0"
__v: 0
```

Channels

```
_id: ObjectId("5cab3f942e779f34f81f69b1")  
title: "SPIEGEL ONLINE"  
url: "http://www.spiegel.de/schlagzeilen/tops/index.rss"  
description: "Deutschlands führende Nachrichtenseite. Alles Wichtige aus Politik, Wi..."  
image: "http://www.spiegel.de/static/sys/logo_120x61.gif"  
__v: 0
```

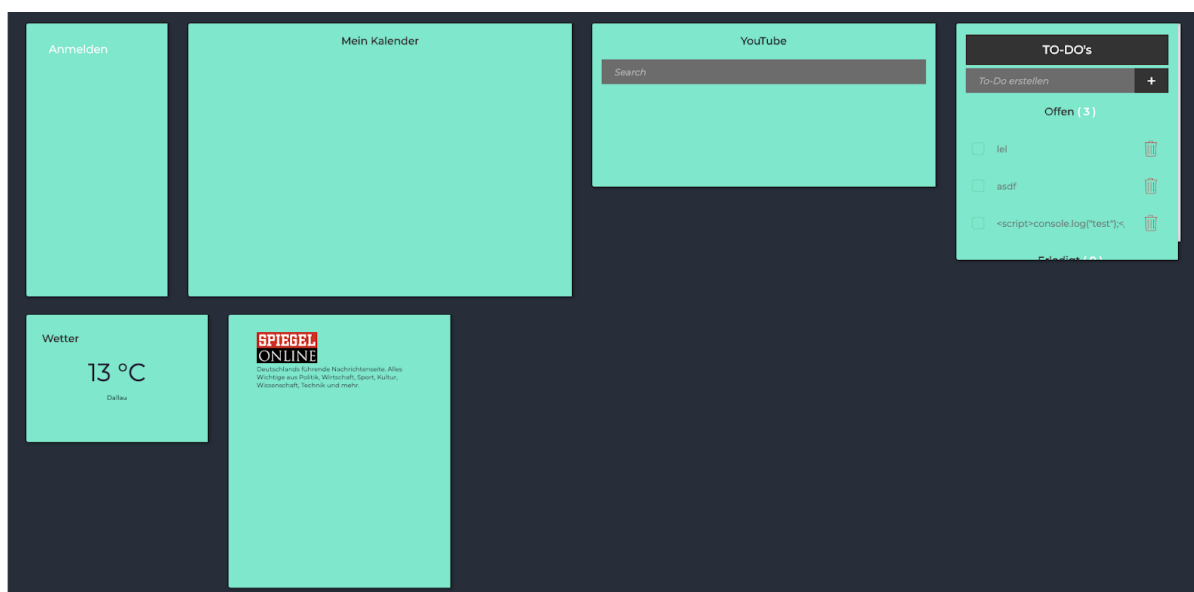
6. Ergebnis des Projekts

Im ersten Schritt ist das Dashboard nur für Authentifizierte User zugänglich, d.h. nur mittels erfolgreichem Login erreichbar. Dazu kann der User im Login-Bereich ein Account anlegen.

A login form on a dark background. It contains two input fields: 'E-Mail' with the value 'info@dhbw-mosbach.de' and 'Passwort' with masked characters '*****'. Below the fields are two buttons: 'Login' (highlighted in green) and 'Registrieren'.

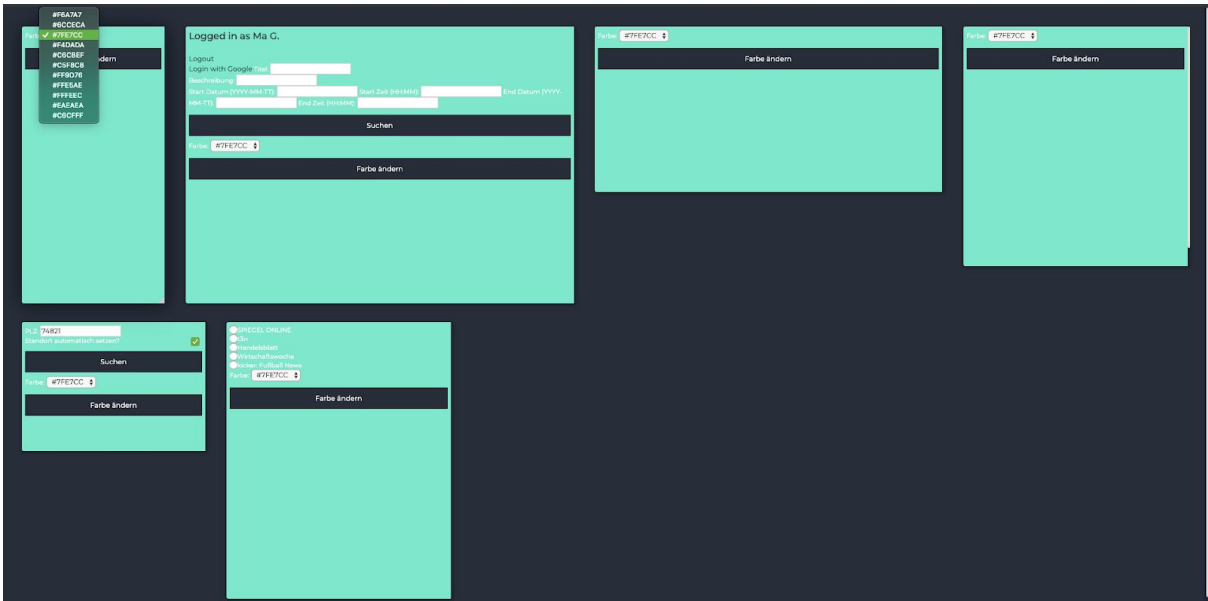
Login des Dashboards

Ist der User eingeloggt, so findet er ein Dashboard mit Widgets vor, die dynamisch in ihrer Größe und Anordnung auf eigenen Wunsch angeordnet werden können.

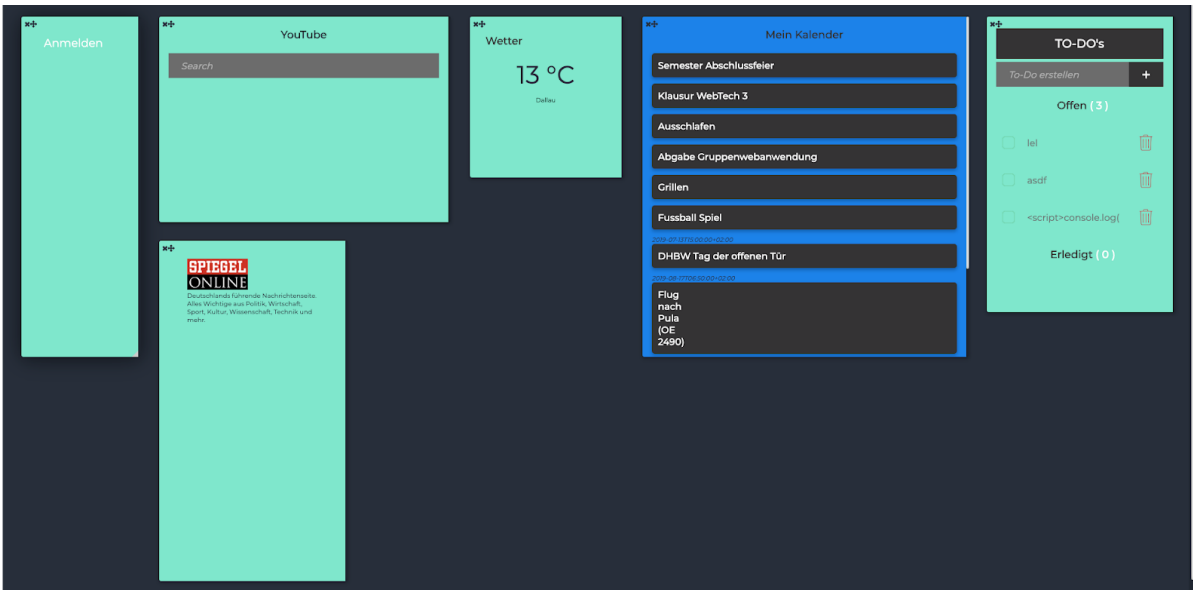


Initialzustand des Dashboards

Desweiteren lässt sich jede Kachel über das Settings-Menu mit individuellen Farben versehen, um dem User eine bessere Zuordnung des Contents zu geben.



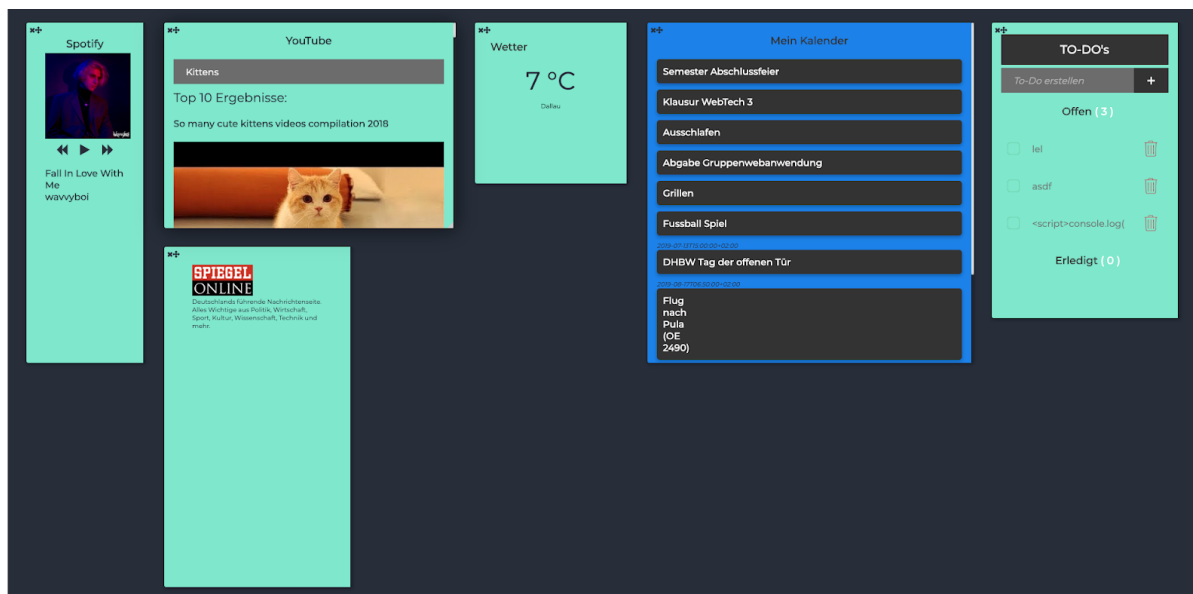
Setting-Übersicht



Widget-Anpassungen

Außerdem bietet jedes Widget eine eigene Funktionalität:

- im RSS Feed können diverse Kanäle selektiv gewählt werden
- über Spotify kann sich der User authentifizieren und dann den Player dann über sein Dashboard steuern
- der user kann sich das Wetter anhand einer Postleitzahl oder seiner IP anzeigen lassen
- für die Verwendung des Kalender Widgets wird der User zu Google weitergeleitet, wo er sich authentifizieren muss. Sobald der Login dort erfolgreich war und er zurück zum Dashboard wechselt, erscheinen dort seine derzeitigen Einträge
- das Suchfeld des Youtube Widgets ist analog zum echten Suchfeld und liefert die 10 besten Suchtreffer
- innerhalb des To-Do's Widgets können Aufgaben erstellt, gelöscht und erledigt werden



Widget Content-Übersicht

Wir beurteilen das Produkt in seinem jetzigen Zustand als MVP-fähig und sehen darin auch ein Potenzial für den Privatgebrauch. Die implementierten Funktionalität liefern einen Vorgeschmack darauf, wie ein Dashboard Informationen konsolidieren könnte. Trotzdem ist anzumerken, dass aktuell noch nicht viel Spielraum für eine persönliche Selektion der zu verwendeten Medien erfolgt, da die Anzahl der verwendeten API's gering ist. Weitere Ausbaustufen wären die die Erhöhung dieser besagten Anzahl und Verbesserungen an den Settings.

7. Gruppenreflexion

7.1 Herausforderungen

Keiner von uns hat bis jetzt in einer Gruppe dieser Größe gearbeitet. Dies hat zur Folge, dass wir uns nicht auf standardisierte Commits geeinigt haben und auch unterschiedliche Editoren verwendet haben. Sollte ein weiteres Projekt dieser Art aufkommen, werden diese zwei Punkte umgehend berücksichtigt. Desweiteren ist das Zeitmanagement manchmal etwas problematisch gewesen, wenn unvorhersehbare Verzögerungen bzw. Fehler eintrafen, die wir nicht in kurzer Zeit beheben konnten.

Wir haben allerdings von Anfang an für jedes Gruppenmitglied einen eigenen Branch angelegt, was gut funktioniert hat bis zum Mergen. Im Laufe des Projekts haben wir die Aufgabenverteilung angepasst und drei Mitglieder haben sich um die Programmierung gekümmert, während zwei Mitglieder Design, UX, Projektmanagement und Dokumentation übernommen haben. Nach dieser Änderung ist das Projekt schneller umgesetzt werden, da gerade im Bereich Projektmanagement eine bessere Zeitplanung entstanden ist.

Genauso wichtig ist das gegenseitige Zuhören und aufeinander eingehen. Die Stärken jedes einzelnen sollten ausgenutzt werden. Für ein erfolgreiches Projekt ist das A und O zusätzliche eine entspannte Arbeitsatmosphäre. Als diese auf unser Team angepasst wurde, war jedes einzelne Mitglied deutlich produktiver.

Eine weitere Herausforderung war das Abwägen zwischen effizientem Arbeiten und der Wissenserweiterung von Nicht-Experten. So hat unser Head of Programming viel Zeit und Nerven investiert, anderen zu helfen und dafür seine Aufgaben vernachlässigt. Auch waren die Aufgaben zum Teil zu anspruchsvoll für die Nicht-Experten, was zu Frustration geführt hat. Nach Umverteilung der Aufgaben war allerdings genug Zeit für produktives Programmieren unseres MVPs und dem Wissenstransfer an alle Mitglieder des Teams.

7.2. Unterstützung

Bei Fehlern oder Sackgassen wurde als Erstes online nach ähnlichen Problemen recherchiert. Hierfür wurden folgende Plattformen eingesetzt:

Plattformen zur Unterstützung

Github

YouTube

StackOverflow

Angular Dokumentation

Spotify Web API Dokumentation

Haben selbst diese Plattformen nicht geholfen, wurden die entsprechenden Experten befragt. Um nicht ständig den gleichen Experten aus dem Rhythmus zu bringen, hat jeder Nicht-Experte einen Ansprechpartner bekommen.

Auch die Experten wussten nicht immer um Rat, konnten sich dann aber dem Problem widmen und dieses schneller lösen, während die Nicht-Experten eine andere Aufgabe übernommen haben.

7.3. Lernerfolge

Da bereits unter dem Punkt **6. Ergebnis** des Projekts eine Beurteilung des Projekts angesprochen wurde, fokussiert sich dieser Abschnitt auf die Zusammenarbeit der Gruppe und daraus resultierende Erkenntnisse. Während der Entwicklungszeit des Dashboards war die Zusammenarbeit zwischen den Mitgliedern außerordentlich. Jedes Mitglied war gewillt, sich seinen Aufgaben zu widmen und sich mit den spezifischen Anforderungen der Features fachlich auseinanderzusetzen.

Gleichzeitig war es auch in Ordnung anderen zu helfen, wenn Barrieren Fortschritt lähmten oder gar vorübergehend zum Stillstand brachten. Obwohl der Wissensstand für jeden unterschiedlicher Natur ist, weiß jeder, dass dieses Projekt lediglich ein erster Schritt in Richtung Software-Entwicklung darstellt. Das erworbene Wissen und die Bereitschaft, agil und kommunikativ mit anderen an einer größeren Aufgabe zu arbeiten hat jeden subjektiv geprägt und inspiriert, sich persönlich weiterzuentwickeln. Das entspricht vor Allem der positiven Atmosphäre und dem Umgang miteinander.

Wenn man über Fortschritt und Evaluation für künftige Projekte und subjektive Empfindungen spricht, ist es immer sehr schwierig diese Erfahrungen für ein Kollektiv repräsentativ zu formulieren. Deshalb möchte ich vielmehr einen weiteren Aspekt ansprechen, der für die persönliche Weiterentwicklung wichtig ist und durch das gemeinsame Unterfangen definiert werden kann.

Die Rollen, die wir verteilt haben entsprachen jenen, die wir im Alltag unternehmerisch verinnerlichen oder auch durch das Projekt nun ausgeübt werden konnten. Somit konnte jeder aus seiner Komfortzone heraus und auf Barrieren stoßen, die den angesprochenen Freiraum der persönlichen Weiterentwicklung begünstigen, woraus jeder sein Potenzial ausschöpfen oder neu kennenlernen kann.

Wie spricht man also am besten über die Zukunft und was besser laufen kann? Ich denke darüber lässt sich sehr gut in den persönlichen Reflexionen im späteren Verlauf der Dokumentation verweisen, denn daraus wird erkenntlich, dass die Summe der Individuen, die am Projekt teilnahmen engagiert waren.

A. Installationsanleitung

Nachdem die VM in Virtual Box geöffnet wurde, melden sie sich im Profil Marius mit dem Passwort "dashboard" an. Öffnen Sie bitte ein Terminal. Dort geben muss der Befehl "sudo mongod --port 27018" ausgeführt werden.

Danach muss ein weiterer Reiter im Terminal geöffnet werden und folgende Befehle ausgeführt werden:

```
cd /home/marius/Obershittest-Webanwendung/dashboard  
ng serve -o
```

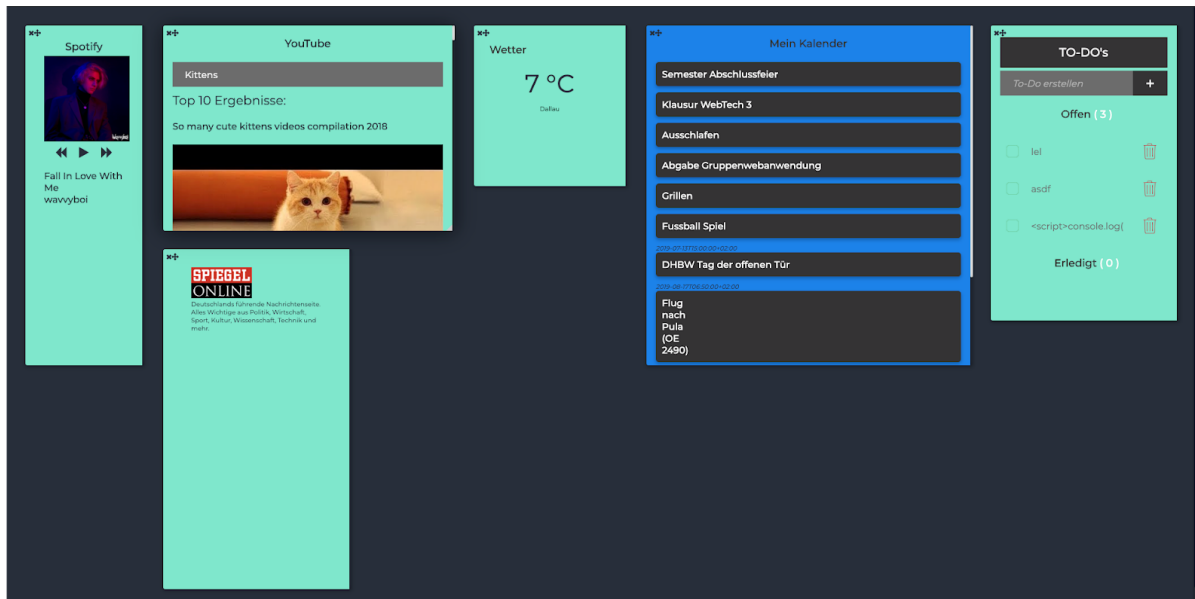
In einem weiteren Reiter müssen dann die Befehle

```
cd /home/marius/Obershittest-Webanwendung/server  
npm start
```

ausgeführt werden.

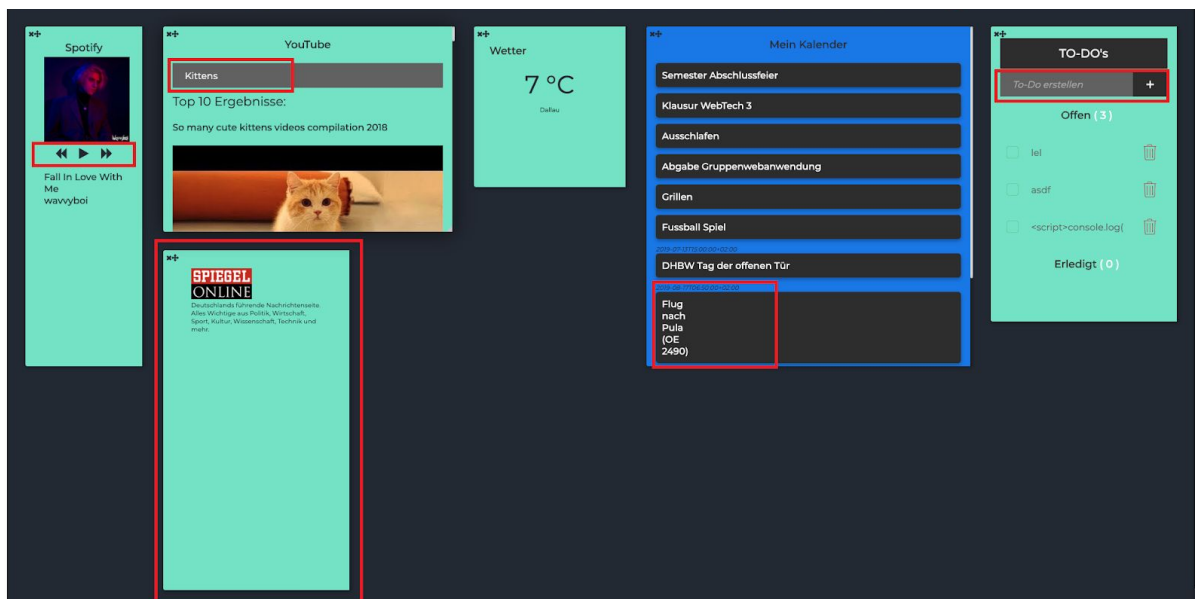
B. Benutzerdokumentation

Da bereits im Ergebnis des Projekts auf das Dashboard eingegangen wurde, fokussiert sich dieser Abschnitt auf den Workflow innerhalb der Anwendung.



Testcase Dashboard

Hat der User, wie es im oberen Screenshot zu entnehmen ist, sein Dashboard konfiguriert, so kann er jedes Widget nach belieben verwenden. Dabei findet die Interaktion unmittelbar innerhalb des Widgets:



Widgetinteraktion

C. Individuelle Beiträge

Eine genauere Übersicht der entstandenen Daten mit Code-Zugehörigkeit ist der Historie des Git-Repositories zu entnehmen und die Referenzierung der Commits. Dieser Anhang befindet sich nach den Reflexionsberichten.

C.1. Milen Bonev

In diesem Projekt konnte mich hinsichtlich der Programmierung weiter verbessern, wobei ich anmerken muss, dass ich nach wie vor kein wirklich Entwickler bin. Beim Verteilen der ersten Aufgaben hätte ich mich um die Wetter-Komponente im Angular Projekt kümmern sollen, doch ich habe es nicht geschafft sie sinnvoll zu implementieren, weshalb Marius die Implementierung übernahm. Weil ich während den Praxisphasen im Produktmanagement unterwegs bin, weiß ich theoretisch, wie Dinge funktionieren, doch ich implementiere sie nicht selbst. Neben der organisatorischen Hilfestellung im Team, habe ich im zweiten Schritt mich mit der Youtube API auseinandergesetzt und diese innerhalb unserer Gridsterkomponente implementiert. Außerdem war es für mich das erste Mal, dass ich Teil einer Entwicklungsgruppe war und selbst am Git-Repository eines Gruppenprojekts teilnahm. Dadurch konnte ich Angulars Infrastruktur besser verstehen und Dependencies wie beispielsweise den `node_modules` des Projekts und Clientseitige Probleme, die dadurch bei Git Pulls entstehen können. Da ich letztes Jahr schon mal mit Javascript gearbeitet habe, viel mir bei der Entwicklung von Angular auf, dass es sich sehr gut für eine Gruppenanwendung eignet, weil die strikte Komponententrennung isoliertes Arbeiten zulässt. Trotzdem denke ich, dass ich meine Stärken nicht im Bereich der Entwicklung liegen, obwohl ich beim zweiten Versuch eine API in unserem Projekt zu implementieren Erfolg hatte.

Aufgabenbereiche:

- Organisatorische Hilfestellung
- Recherche über Wetter Komponenten, jedoch Implementierung gescheitert
- Implementierung der YouTube Komponente

C.2. Marius Großkopf

Durch dieses Projekt habe ich gelernt, während dem Programmieren den anderen Teammitgliedern Hilfestellung zu geben und mich schnell in deren Code hineinzudenken. Dies war zu Beginn noch sehr schwer und ich bin mit meinen Aufgaben nur sehr langsam vorangekommen. Dazu habe ich gelernt, beim Erklären von Code gegenüber unerfahrenen Teammitgliedern, geduldig zu bleiben und die Ruhe zu bewahren.

Des Weiteren habe ich tiefere Kenntnisse im Bereich Git und Versionierung erworben. So kann ich nun mit Merge Konflikten umgehen und mit verschiedenen Branches arbeiten. Allerdings würde ich beim nächsten Projekt das Repository besser vorbereiten, da wir am Anfang zum Beispiel alle Node Modules hochgeladen haben, was zu Fehlern geführt hatte.

Die Qualität meines Codes hat sich im Vergleich zu meiner letzten Webanwendung verbessert. Besonders die Methodennamen wurden gut ausgewählt, sodass jeder in der Gruppe die Aufgabe der Methode verstehen konnte. Außerdem habe ich Interfaces und Services gut eingesetzt, sodass die Anwendung strukturiert und modular aufgebaut ist.

Außerdem habe ich mich mit vielen Angular Libraries beschäftigt, sodass ich diese auch in anderen Projekten verwenden kann. Dazu gehört zum Beispiel die Library Gridster 2.

Zudem habe ich auch tiefere Einblicke in die Verwendung der Lifecycle Hooks von Angular bekommen, sodass ich diese noch effektiver verwenden kann und nun auch weiß, wann und wie diese richtig eingesetzt werden.

Aufgabenbereich:

Wetter Component erstellt inkl Funktionalität

- Abrufen des Standortes über IP Adresse
- Aktivieren der Standortwahl über IP
- Eingabemöglichkeit zum Einstellen einer Postleitzahl
- Abrufen der Temperatur und der zugehörigen Icons über die Api

Wetter Component Schnittstelle zur Datenbank gebaut

- Einstellungen der Wetter Component werden über Schnittstelle an die Rest Api gesendet und in der Datenbank gespeichert

Automatisches Speichern des Dashboards implementiert

- Sobald eine Kachel auf dem Dashboard geändert wird, wird der Stand in der Datenbank gespeichert

Anderen Mitgliedern Hilfestellung geben

Speichern von Farben der einzelnen Kacheln

Implementierung des Flip-Effekt zur Anzeige der Einstellungsseite der Kacheln

Implementierung des Gridster Layouts

C.3. Henning Löwe

In diesem Projekt habe ich vor allem die Arbeit im Team gelernt. Dabei geht es nicht nur um das schnelle Einlesen in fremden/anderen Code, sondern auch das sinnvolle kommentieren des Eigenen zum besseren Verständnis für andere Personen. Durch diesen Lernprozess und den angestandenen Pitch haben wir am Anfang viel Zeit verloren und mussten zum Schluss die Aufgaben neu verteilen, um die Idee rechtzeitig umsetzen zu können. Das größte Problem hatten wir aber mit Git, denn das Mergen unterschiedlicher Branches, haben die verschiedenen Editoren unterschiedlich gut behandelt. In Visual Studio Code wurden die Konflikte fest in den Code geschrieben, wodurch alle Dateien durchsucht und Fehler manuell beseitigt werden mussten. Darüber hinaus hätte zu Beginn eine sinnvolle Kommentierung und Versionierung der Commit-Messages stattfinden soll. Ich habe außerdem im Bereich Angular noch mehr über Observable, BehaviorSubject, ReplaySubject und EventEmitter gelernt, welche ich erfolgreich im Projekt integrieren konnte.

Aufgabenbereiche:

- Entwicklung
 - Login und Registrierung
 - ToDo-Liste
 - News
 - REST API
 - User
 - ToDo
 - Channel
 - Services
- SASS-Styling
- Anderen Hilfestellung geben

C.4. Nils Mahler

Bei der Arbeit an diesem Projekt habe ich vor allem gelernt, ein Projekt im Team aufzusetzen und zu realisieren. Beispielsweise das Arbeiten mit Github in einer großen Gruppe habe mitgenommen. Hier kommt, abgesehen von einem neuen Bewusstsein für Commit Messages (diese waren von meiner Seite aus wenig informativ und nicht gemeinschafts tauglich), das Beheben von Problemen mit Git (zum Beispiel anlegen einer korrekten Gitignore) und Fehlerprävention beim Mergen hinzu.

Wichtig war auch, generell das Suchen von Hilfe beziehungsweise eines anderen Blickwinkels, welches so manche Probleme in wesentlich kürzerer Zeit löste, als ewig selbst herumzuprobieren. Anfangs hatte ich einen Ehrgeiz, alles selbst machen zu wollen, ohne mir Hilfe zu suchen. Doch beim Arbeiten im Team ist unter anderem genau das der Vorteil.

Des Weiteren habe ich mir ein gutes Verständnis der Spotify API erarbeitet. Ich verstehe, wie die API aufgebaut ist, was sie an Informationen liefert und wie man diese abrufen kann. Auch wie sich die bereitgestellten Informationen möglichst nahtlos in die eigentliche Anwendung integrieren lassen, habe ich hier mitgenommen. Zusätzlich habe ich einiges über die Authentifizierung einer API im Generellen, aber auch bei der Spotify API im konkreten gelernt. Speziell, wie man mit Angular im Header ein Token mitgibt, um einen Nutzer zu authentifizieren.

Hierbei habe ich auch mein Verständnis von POST, GET, und PUT Requests verfeinert. Ich konnte hier mein Wissen aus der API Vorlesung in der Praxis anwenden. Beispielsweise das Verwenden des Programms Postman, um bestimmte Requests auszuprobieren ohne diese hart Coden zu müssen.

Alles in allem hat sich mein Verständnis von Angular 2 auf ein neues Level gehoben. Ich habe auch viele kleine Dinge, die man aus mangelnder Übung falsch macht ausgemerzt und bin guter Dinge, mein nächstes Projekt schneller und besser realisieren zu können.

Aufgabenbereiche:

Auseinandersetzen Spotify API

Spotify Component Funktionalität

Spotify Component grundlegende Styles

C.5. Michelle Schmidt

In diesem Projekt habe ich gelernt, dass eine gute Vorbereitung und ein frühzeitiger Beginn extrem wichtig für einen reibungslosen Ablauf ist. Bei einem Team dieser Größe ist eine strukturierte Planung Voraussetzung für eine sauber umgesetzte Anwendung. Dieser strukturierte Vorgang hätte anfangs besser ablaufen können, wodurch wir gegen Ende etwas effizienter arbeiten mussten.

Ich habe anfangs das Design und CSS-Styling übernommen und versucht, einen Color Picker mit Angular zu programmieren. Ich habe meine CSS-Kenntnisse vertieft und den Umgang mit Flexboxen verbessert. Außerdem habe ich mich mit dem Umgang mit Git und Sourcetree vertrauter gemacht, genauso wie mit der Handhabung von Commits und Merges, wenn man auf unterschiedlichen Branches arbeitet.

Ich habe die Vorteile und Nachteile von Gruppenarbeiten kennengelernt: einerseits ist es sehr nützlich, schnell eine Meinung bzw. Hilfe zu einem Problem einzuholen und somit effektiver, als alleine nach der Lösung für ein Problem zu suchen. Andererseits waren bei uns vor allem Merges sehr zeitaufwändig, wenn unterschiedliche Code-Editoren eingesetzt wurden und es zu Fehlern kam. Durch die lange Fehlerbehebung habe ich mitgenommen, dass anfangs gewisse Standards festgesetzt werden müssen, um Zeit zu sparen.

Nachdem wir in zeitliche Not kamen, mussten wir die Aufgaben neu verteilen, sodass jeder seine Stärken einsetzt. So bin ich Ansprechpartner für Design- und UX-Fragen geblieben und habe mich mit der Umsetzung der Dokumentation und der Präsentation beschäftigt. Bei dem Wissenstransfer habe ich viel über die unterschiedlichen Komponenten gelernt und den Einsatz von Promises. Außerdem habe ich das abstrakte Denken des Programmierens besser verstanden, dass ich hoffentlich in meiner kommenden Projektarbeit einsetzen kann.

Aufgabenbereiche:

- Design des Dashboards
- Design Hilfestellung
- Koordination / Projektmanagement
- UX-Recherche für eine besser Usability des Dashboards
- Sass-Styling
- Font-Awesome-Implementierung
- Versuch, Farben der einzelnen Dashboard-Kacheln über einen Angular-Color-Picker auswählen zu können

C.6. Git Historie

Changes 4

History

Select Branch to Compare...

Merge branch 'dev' into Milen

bonevmilen committed 2 days ago

Minor Changes

bonevmilen committed 2 days ago

fixed some bugs

magrosskopf committed 2 days ago

Merge branch 'dev' of https://github....

ewice committed 3 days ago

fixed merge errors

ewice committed 3 days ago

Merge branch 'dev' of https://github....

magrosskopf committed 3 days ago

added sanitizer to app.js and some b...

magrosskopf committed 3 days ago

Merge branch 'henning' into dev

ewice committed 3 days ago

fixed Merge

ewice committed 3 days ago

merged dev

ewice committed 3 days ago

Merge branch 'dev' into henning

ewice committed 3 days ago

Merge branch 'dev' into Nils2

SirLocke committed 3 days ago

added more styles

SirLocke committed 3 days ago

implemented adding components to ...

magrosskopf committed 3 days ago

Merge branch 'dev' into Milen

bonevmilen committed 850514a8 76 changed files

			@@ -1,2 +1,2 @@
.gitignore		1	server/node_modules/
dashboard/.DS_Store	+	2	-dashboard/node_modules/
dashboard/package-lock.json			+dashboard/node_modules/
dashboard/package.json			
dashboard/src/app/_int.../channel.ts	+		
dashboard.../dashboard-positions.ts	+		
dashboard/src/app/_interf.../index.ts			
dashboard/src/app/_in.../message.ts	+		
dash.../auth-firebase.service.spec.ts	-		
dashboar.../auth-firebase.service.ts			
dashboard/src/app.../auth.service.ts			
dashboard/src.../channels.service.ts	+		
dashboard/src/ap.../colors.service.ts	+		
dashboard/src/ap.../helper.service.ts	+		
dashboard/src/app.../http.service.ts			
dashboard/src/app/_servi.../index.ts			
dashboard/sr.../messages.service.ts	+		
dashboard/src/_.../rss2json.service.ts	+		
dashboard/src/_.../weather.service.ts	+		
da.../youtube-search.service.spec.ts	-		
dashboard/src/app/_templ.../index.ts			
dashboard/src/a...r.component.html	-		
dashboard/src/a...r.component.sass	-		
d.../template-calendar.component.ts	-		

Changes 4

History

Select Branch to Compare...

fixed RSS-Component

ewice committed 3 days ago

changed button styles; Styled Weath...

SirLocke committed 3 days ago

Merge branch 'dev' into Nils2

SirLocke committed 3 days ago

Update weather.service.ts

magrosskopf committed 3 days ago

fixed weather component

magrosskopf committed 3 days ago

added some styles

SirLocke committed 3 days ago

Merge branch 'dev' into Nils2

SirLocke committed 3 days ago

fixed play button

SirLocke committed 3 days ago

implemented form to create event in ...

magrosskopf committed 3 days ago

added authguard to imortant routes

magrosskopf committed 3 days ago

Merge branch 'dev' of https://github....

magrosskopf committed 4 days ago

impemented an lframe to dispaly you...

magrosskopf committed 4 days ago

Added Scrollbar

SirLocke committed 4 days ago

removed spotify webplayback test

SirLocke committed 4 days ago

Merge branch 'dev' into Milen

bonevmilen committed 850514a8 76 changed files

			@@ -1,2 +1,2 @@
dashnoo...spec.ts	dashnoo...spec.ts	1	server/node_modules/
dashboard/s.../spotify.component.ts	+	2	-dashboard/node_modules/
dashboa.../weather.component.html			+dashboard/node_modules/
dashboa.../weather.component.sass			
dashboard.../weather.component.ts			
das.../search-result.component.html			
das.../search-result.component.sass			
dashb.../search-result.component.ts			
dashboa.../youtube.component.html	+		
dashboa.../youtube.component.sass	+		
dashbo...spec.ts	dashbo...spec.ts		
dashboard/_.../youtube.component.ts	+		
dashboard/src/index.html			
dashboard/src/style/_var.sass			
dashboard/src/styles.sass			
server/api/models/channel.js	+		
server/api/m.../dashboardPosition.js	+		
server/api/routes/channel.js	+		
server/api/ro.../dashboardPosition.js	+		
server/api/routes/todo.js			
server/api/routes/user.js			
server/api/routes/weather.js			
server/app.js			
server/package-lock.json			
server/package.json			

Changes 4

History

Select Branch to Compare...

Merge branch 'dev' into Nils2

SirLocke committed 4 days ago

updated weather component

magrosskopf committed 4 days ago

added Styles

SirLocke committed 4 days ago

Merge branch 'dev' into marius

magrosskopf committed 4 days ago

added color settings and flip effect t...

magrosskopf committed 4 days ago

Abfrage zum anmelden bei Spotify b...

SirLocke committed 4 days ago

added RSS

ewice committed 4 days ago

package json updated

SirLocke committed 4 days ago

Merge branch 'dev' into Nils2

SirLocke committed 4 days ago

updated package json

SirLocke committed 4 days ago

Merge branch 'dev' of https://github....

SirLocke committed 4 days ago

Added Album Image

SirLocke committed 4 days ago

Added Basic styles; format html

SirLocke committed 4 days ago

updated package.json after changin...

magrosskopf committed 5 days ago

Merge branch 'dev' into Milen

bonevmilen committed 850514a8 76 changed files

oasndoo...spec.ts

oasndoo...spec.ts

+

dashboar.../spotify.component.ts

+

dashboa.../weather.component.html

+

dashboa.../weather.component.sass

+

dashboar.../weather.component.ts

+

das.../search-result.component.html

+

das.../search-result.component.sass

+

dashb.../search-result.component.ts

+

dashboa.../youtube.component.html

+

dashboa.../youtube.component.sass

+

dashbo...spec.ts

dashbo...spec.ts

+

dashboar.../youtube.component.ts

+

dashboard/src/index.html

+

dashboard/src/style/_var.sass

+

dashboard/src/styles.sass

+

server/api/models/channel.js

+

server/api/m.../dashboardPosition.js

+

server/api/routes/channel.js

+

server/api/ro.../dashboardPosition.js

+

server/api/routes/todo.js

+

server/api/routes/user.js

+

server/api/routes/weather.js

+

server/app.js

+

server/package-lock.json

+

server/package.json

+

@@ -1,2 +1,2 @@

1 1 server/node_modules/

2 -dashboard/node_modules/

2 +dashboard/node_modules/

Changes 4

History

Select Branch to Compare...

fixed last bugs when saving dashboa...

magrosskopf committed 5 days ago

saving Dashboard possible

magrosskopf committed 6 days ago

changed stylesheet to sass from scs...

magrosskopf committed 6 days ago

everything is now up to date

magrosskopf committed 6 days ago

Merge branch 'marius' into dev

magrosskopf committed 6 days ago

fixed node_modules

magrosskopf committed 6 days ago

reverted updating node_modules an...

magrosskopf committed 6 days ago

Revert "updated node modules"

magrosskopf committed 6 days ago

styled weather and calender compon...

magrosskopf committed 6 days ago

prettyfy code

SirLocke committed 7 days ago

updated node modules

Michelle committed 7 days ago

treid to update track when forward

SirLocke committed 7 days ago

Delete Node_Moduels

bonevmilen committed Apr 5, 2019

spotify display artist

SirLocke committed Apr 5, 2019

Merge branch 'dev' into Milen

bonevmilen committed 850514a8 76 changed files

oasndoo...spec.ts

oasndoo...spec.ts

+

dashboar.../spotify.component.ts

+

dashboa.../weather.component.html

+

dashboa.../weather.component.sass

+

dashboar.../weather.component.ts

+

das.../search-result.component.html

+

das.../search-result.component.sass

+

dashb.../search-result.component.ts

+

dashboa.../youtube.component.html

+

dashboa.../youtube.component.sass

+

dashbo...spec.ts

dashbo...spec.ts

+

dashboar.../youtube.component.ts

+

dashboard/src/index.html

+

dashboard/src/style/_var.sass

+

dashboard/src/styles.sass

+

server/api/models/channel.js

+

server/api/m.../dashboardPosition.js

+

server/api/routes/channel.js

+

server/api/ro.../dashboardPosition.js

+

server/api/routes/todo.js

+

server/api/routes/user.js

+

server/api/routes/weather.js

+

server/app.js

+

server/package-lock.json

+

server/package.json

+

@@ -1,2 +1,2 @@

1 1 server/node_modules/

2 -dashboard/node_modules/

2 +dashboard/node_modules/

Changes 4History

Select Branch to Compare...

design widgets + not working color p...

Michelle committed Apr 5, 2019

Get Album Infos implemented

SirLocke committed Apr 5, 2019

added Toggle styles

SirLocke committed Apr 5, 2019

Merge branch 'Milen' into dev

bonevmilen committed Apr 5, 2019

implemented Toggle for PLayer/resume

SirLocke committed Apr 5, 2019

Minor Changes

bonevmilen committed Apr 5, 2019

Minor Changes

bonevmilen committed Apr 5, 2019

Implemented play function spotify

SirLocke committed Apr 5, 2019

youtube geht junge

bonevmilen committed Apr 5, 2019

Youtube-Kachel rdy

bonevmilen committed Apr 5, 2019

Youtube funktionsfähig aber noch in ...

bonevmilen committed Apr 5, 2019

Youtube Module CSS Issue

bonevmilen committed Apr 5, 2019

Youtube Home Component Integration

bonevmilen committed Apr 5, 2019

fixed weather component

magrosskopf committed Apr 5, 2019

Merge branch 'dev' into Milen

bonevmilen committed 850514a8 76 changed files

dashbo...spec.ts	dashbo...spec.ts			@@ -1,2 +1,2 @@
dashboard/s.../spotify.component.ts		1	1	server/node_modules/
dashboa.../weather.component.html		2		-dashboard/node_modules/
dashboa.../weather.component.sass			2	+dashboard/node_modules/
dashboard/.../weather.component.ts				
das.../search-result.component.html				
das.../search-result.component.sass				
dashb.../search-result.component.ts				
dashboa.../youtube.component.html				
dashboa.../youtube.component.sass				
dashbo...spec.ts	dashbo...spec.ts			
dashboard/.../youtube.component.ts				
dashboard/src/index.html				
dashboard/src/style/_var.sass				
dashboard/src/styles.sass				
server/api/models/channel.js				
server/api/m.../dashboardPosition.js				
server/api/routes/channel.js				
server/api/ro.../dashboardPosition.js				
server/api/routes/todo.js				
server/api/routes/user.js				
server/api/routes/weather.js				
server/app.js				
server/package-lock.json				
server/package.json				

Changes 4History

Select Branch to Compare...

Responsive Design todo + Begin cre...

Michelle committed Apr 4, 2019

Setup Youtube Struktur

bonevmilen committed Apr 4, 2019

Spotify API Stage1, Authorization iss...

SirLocke committed Apr 4, 2019

Spotify webplayer implemented

SirLocke committed Apr 4, 2019

Spotify Component angelegt

SirLocke committed Apr 4, 2019

Merge branch 'dev' into Nils2

SirLocke committed Apr 4, 2019

Merge branch 'dev' into marius

magrosskopf committed Apr 4, 2019

Merge branch 'dev' of https://github....

SirLocke committed Apr 4, 2019

Merge branch 'mishmosch' into dev

Michelle committed Apr 4, 2019

sie weis nicht was sie getan hat

Michelle committed Apr 4, 2019

adding Google Calendar to the Dash...

magrosskopf committed Mar 20, 2019

adding calendar to project

magrosskopf committed Mar 19, 2019

changed Gridster Layout

magrosskopf committed Mar 19, 2019

Merge branch 'dev' into Nils2

SirLocke committed Mar 19, 2019

Merge branch 'dev' into Milen

bonevmilen committed 850514a8 76 changed files

dashbo...spec.ts	dashbo...spec.ts			@@ -1,2 +1,2 @@
dashboard/s.../spotify.component.ts		1	1	server/node_modules/
dashboa.../weather.component.html		2		-dashboard/node_modules/
dashboa.../weather.component.sass			2	+dashboard/node_modules/
dashboard/.../weather.component.ts				
das.../search-result.component.html				
das.../search-result.component.sass				
dashb.../search-result.component.ts				
dashboa.../youtube.component.html				
dashboa.../youtube.component.sass				
dashbo...spec.ts	dashbo...spec.ts			
dashboard/.../youtube.component.ts				
dashboard/src/index.html				
dashboard/src/style/_var.sass				
dashboard/src/styles.sass				
server/api/models/channel.js				
server/api/m.../dashboardPosition.js				
server/api/routes/channel.js				
server/api/ro.../dashboardPosition.js				
server/api/routes/todo.js				
server/api/routes/user.js				
server/api/routes/weather.js				
server/app.js				
server/package-lock.json				
server/package.json				

Changes 4

History

Select Branch to Compare...

package.json changed

ewice committed Mar 19, 2019

Update bcrypt_lib.node

SirLocke committed Mar 19, 2019

delete node_modules

ewice committed Mar 19, 2019

change gitignore

ewice committed Mar 19, 2019

Update bcrypt_lib.node

SirLocke committed Mar 19, 2019

Update .gitignore

SirLocke committed Mar 19, 2019

Update .gitignore

SirLocke committed Mar 19, 2019

Merge branch 'dev' into Milen

bonevmilen committed Mar 19, 2019

Merge branch 'marius' into dev

magrosskopf committed Mar 19, 2019

Revert "Revert "Merge branch 'mariu..."

magrosskopf committed Mar 19, 2019

Dev Merge

bonevmilen committed Mar 19, 2019

Merge branch 'mishmosch' of https://...

Michelle committed Mar 19, 2019

design dashboard dark-mode

Michelle committed Mar 19, 2019

Revert "Merge branch 'marius' into ...

magrosskopf committed Mar 19, 2019

initial commit

ewice committed Mar 19, 2019

changed gitignore

ewice committed Mar 19, 2019

added gitignore and gridster 2

magrosskopf committed Mar 12, 2019

initial commit

ewice committed Mar 11, 2019

Initial commit

ewice committed Mar 11, 2019

Merge branch 'dev' into Milen

bonevmilen committed 850514a8 76 changed files

dasnoo...spec.ts

dasnoo...spec.ts

@@ -1,2 +1,2 @@

dashboard/s.../spotify.component.ts

1

1

server/node_modules/

dashboa.../weather.component.html

2

-dashboard/node_modules/

dashboa.../weather.component.sass

2

+dashboard/node_modules/

dashboard/.../weather.component.ts

das.../search-result.component.html

das.../search-result.component.sass

dashb.../search-result.component.ts

dashboa.../youtube.component.html

dashboa.../youtube.component.sass

dashbo...spec.ts

dashbo...spec.ts

dashboard/.../youtube.component.ts

dashboard/src/index.html

dashboard/src/style/_var.sass

dashboard/src/styles.sass

server/api/models/channel.js

server/api/m.../dashboardPosition.js

server/api/routes/channel.js

server/api/ro.../dashboardPosition.js

server/api/routes/todo.js

server/api/routes/user.js

server/api/routes/weather.js

server/app.js

server/package-lock.json

server/package.json

server/api/m.../dasnooaraPosition.js

server/api/routes/channel.js

server/api/ro.../dashboardPosition.js

server/api/routes/todo.js

server/api/routes/user.js

server/api/routes/weather.js

server/app.js

server/package-lock.json

server/package.json