

The Good News App

ein Projekt von

Thore August
Matrikelnummer 2332999

und

Madeline Jungnitsch
Matrikelnummer: 2320440

im Studiengang Media Systems
Wahlpflichtmodul Mobile Systeme
bei Jakob Sudau
Sommersemester 2020

Einleitung

Im Rahmen des Wahlpflichtmoduls „Mobile Systeme“ des Studiengangs Media Systems war es unsere Aufgabe, als Prüfungsleistung eine Crossplatform-App mit dem Framework React Native zu entwickeln. Vorgabe war es, dass User-Input-Daten und Internetdaten dargestellt werden und eine Live-Datenvisualisierung stattfindet. Außerdem sollten wir uns ein Konzept für eine App überlegen, die gesellschaftskritisch oder -relevant in der heutigen Zeit ist. So ist das Projekt „The Good News App“ entstanden. Unsere Motivation war hierbei vor allem, dem Benutzer die Möglichkeit zu geben, ausschließlich positive Nachrichten zu lesen, die gerade zu dieser Zeit oftmals zu kurz kommen.

Projektziel

Ursprüngliches Ziel des Projekts war es, am Rundgang des HAW-Campus Finkenau im Sommer teilzunehmen, dort die App auf einer Leinwand zu präsentieren und mithilfe eines QR-Codes den Besuchern direkten Zugang zum Download der App zu geben, damit sie diese selbst ausprobieren können. Da dies aufgrund der aktuellen Situation leider nicht möglich sein wird, wird es eine kursinterne Präsentation geben.

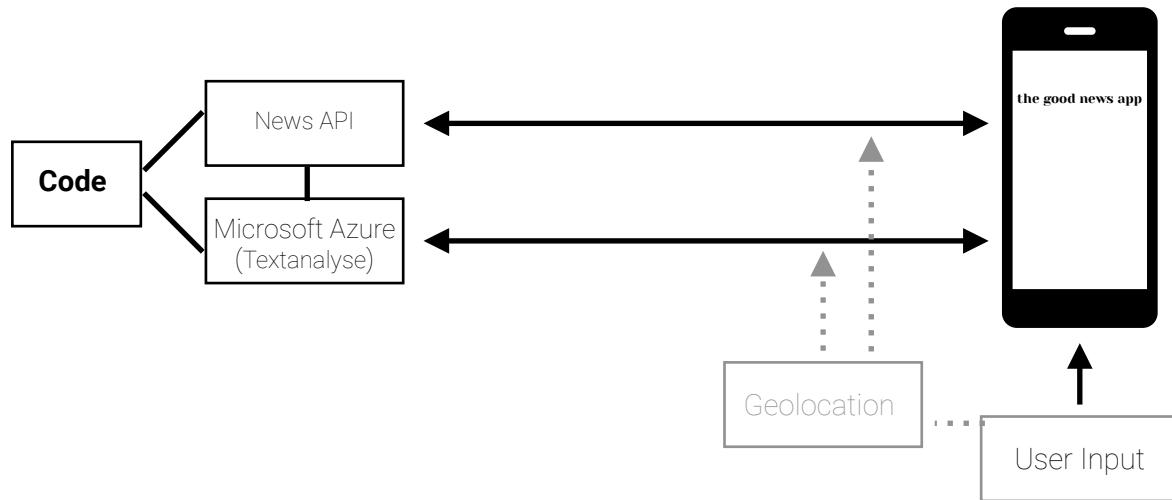
Anforderungsanalyse

Der Benutzer soll durch die Good News App die Möglichkeit bekommen, Zeitungs- und Onlineartikel verschiedenster Webseiten oder Anbieter in einer App lesen zu können. Hierbei geht es vor allem darum, dass die Nachrichten, die vom Benutzer gelesen werden, ausschließlich positiv sein sollen. Der Benutzer soll in der App durch verschiedene Artikel-Kategorien (z.B. Wirtschaft, Gesundheit, Technologie) navigieren können und je nach gewählter Kategorie passende Artikel angezeigt bekommen. Außerdem soll er die Artikel nach Stichwörtern durchsuchen können.

Technische Rahmenbedingungen

Da es sich bei der gegebenen Anforderung des Kurses um die Entwicklung einer Cross-Platform-App handelt, soll die App sowohl auf Android- als auch auf iOS-Smartphones sowie Tablets funktional sein. Programmiert wird sie mit dem React-Native Open-Source Mobile Application Framework von Facebook, mit dem mobile Anwendungen unter anderem für iOS und Android entwickelt werden können. Das Framework basiert auf dem React Framework, welches für die Entwicklung von Webanwendungen entwickelt wurde. Unter React-Native wird vor allem JavaScript als Programmiersprache verwendet. Um die App crossplatform benutzen zu können wird außerdem Expo als Plattform zur Unterstützung beim builden und Deployment der App verwendet.

Technisches Konzept



Für unsere App war von Beginn an klar, dass API's genutzt werden sollen. In unserem Falle brauchten wir diese, um auf Zeitungs- und Onlineartikel zugreifen zu können. Außerdem mussten wir eine Art „Good News Filter“ implementieren, damit ausschließlich gute Nachrichten angezeigt werden. Hierzu verwenden wir Microsoft Azure zur Textanalyse, um zu prüfen, ob der Artikel einen „positive tone“ hat. Außerdem interagiert der User mit der App durch Keyword-Suche und hat die Möglichkeit, seine Geolocation zu teilen. Genaueres dazu folgt in der technischen Dokumentation.

Bedienkonzept

Die Bedienung der App soll für den Benutzer möglichst einfach und unkompliziert sein und orientiert sich leicht an anderen, bereits etablierten Anwendungen. Dabei liegt der Fokus auf dem Artikel-Screen. So sieht der Benutzer beim Öffnen der App zunächst den Homescreen. Hier befindet sich der Nutzer zuerst automatisch in der Kategorie „Start“, wo Artikel zu allen Kategorien nach Datum und Aktualität sortiert angezeigt werden. Der Artikel wird mit einem Titel, dem Einleitungstext und (wenn vorhanden) einem Bild dargestellt. Nun hat der User die Möglichkeit, durch das Tippen auf den Menu-Button die verschiedenen Kategorien angezeigt zu bekommen. Wählt er eine bestimmte Kategorie aus, wechselt der Screen zur ausgewählten Kategorie und zeigt dazu passende Artikel an. Möchte der Nutzer einen Artikel lesen, wählt er diesen einfach an und wird dann auf die Webseite des Artikels weitergeleitet und kann diesen dort lesen. Durch das Tippen auf den Zurück-Button gelangt er wieder zur vorigen Ansicht zurück. Außerdem hat er die Möglichkeit mithilfe der Suchfunktion nach Stichwörtern, Themen oder Quellen zu suchen. Als kleines Gimmick hat er außerdem die Wahl zwischen drei verschiedenen Designs in denen die App dargestellt werden kann, dies kann er im Menü einstellen.

An dieser Stelle werden exemplarisch einige Screens gezeigt, die wir vor der Entwicklung mithilfe von Adobe XD erstellt haben, sowie Screenshots des Ergebnisses.



Abbildung 1: Designkonzept 09.06.



Abbildung 2: fertiggestellte App

Zeitplan und Teamplanung

Meilensteine bis	12.05.	26.05.	09.06.	23.06.	07.07.
Thore		Struktur	Prototyp mit Dummy Data	Good News Filter + Designimplementierung	Fertigstellung Nice To Haves
Madeline	Projektkonzept	Design	Design Fertigstellung		Design Fixes, Poster, Fertigstellung Bericht

Für unseren Zeitplan war der erste Meilenstein die Präsentation des Projektkonzepts am 12.05. entscheidend. Bis dahin stand die Idee für die App bereits fest sowie eine erste Idee für das Screendesign. Bis zum 26.06. haben wir uns vorgenommen, dass die Ordnerstruktur des Projekts fertiggestellt wird und erste Designentwürfe feststehen. Zur Vorstellung des Prototypen sollte ein Prototyp mit Dummy Data fertiggestellt werden und eine endgültige Designauswahl getroffen werden. Nächster Schritt war es, den „Filter“ für die guten Nachrichten zu implementieren, sodass dem Benutzer nur positive Nachrichten angezeigt werden. Außerdem sollte das Design bis dahin ebenfalls in der App implementiert sein. Bis zur Abgabe am 07.07. war es unser Ziel, zum einen das Projektkonzept und die technische Dokumentation fertigzustellen und zum anderen „Nice To Haves“ in die App einzubauen, wie die Keywordsuche sowie Design Fixes.

Methodendokumentation

Allgemein wurden für das Projekt das Framework React Native mit seinen gegebenen Funktionen wie Core Components sowie Third-Party-Libraries (z.B. Expo Ionicons) verwendet. Für das App Development und Deployment kam Expo sowie der Expo CLI zur Anwendung. Zusätzlich wurden zwei APIs verwendet um auf Onlineartikel zugreifen zu können sowie einen „Good-News-Filter“ einzubauen, der nur positive Nachrichten herausfiltert.

Programmierdokumentation

Im folgenden Abschnitt wird kurz auf die einzelnen Dateien der App eingegangen, sowie bei der wichtigsten weiter ins Detail gegangen.

App.js

In der App.js-Datei werden die von uns ausgewählten Fonts geladen und der Feedscreen zurückgegeben.

constants / Themes.js

In der Themes.js-Datei sind die drei von uns designten Themes enthalten (*lightTheme*, *darkTheme*, *colorfullTheme*). Durch die constants *getFeedTheme*, *getHeaderTheme* und *getSidebarTheme* werden Header, Feed und Sidebar sowie Schriften die zugehörigen Farbschemata zugewiesen.

components / Menu.js

In der Menu-Funktion befindet sich ein Array, das alle verfügbaren Kategorien enthält. Dies wird in einer *Flatlist* ausgegeben. Dabei ist jede Kategorie eine *Touchable Opacity* mit Text. Außerdem wird hier die Methode *newsHandler* getriggert. In einer View sind drei Buttons definiert, als klickbare *Ionicons* für die verschiedenen Themes.

components / FeedTile.js

Die FeedTile.js-Datei definiert den Aufbau eines Artikels im Feed. Er enthält jeweils einen Titel, eine Beschreibung, die Quelle und das Datum. Wenn ein Bild vorhanden ist, wird dieses dargestellt, wenn nicht wird eine leere View angezeigt. Hierbei ist der gesamte Artikel als View eine *TouchableOpacity*, die beim Drücken die *toArticle()*-Methode aufruft, die zu dem Artikel führt.

screens / ArticleScreen.js

Hier wird der Aufbau der Artikelseite durch die *Modal*-Komponente definiert, die einen Header mit einem Zurück-Button sowie eine *WebView*, in der die Webseite des Artikels zurückgegeben wird, enthält.

screens / FeedScreen.js

In der FeedScreen.js-Datei wird der Feed-Screen und sein Aufbau definiert, auf die NewsAPI zugegriffen sowie auf die Textanalyse von Microsoft Azure. Außerdem wird hier die Methode zum Zugriff auf die Geolocation des Nutzers definiert. Im Folgenden werden diese wichtigsten Bestandteile erläutert.

```
28  //holt die news für die ausgewählte Kategorie
29  const getCategoryNews = async (category) => {
30      let articleId = 0;
31      let response;
32      var url = `http://newsapi.org/v2/top-headlines?category=${category}&country=de&pageSize=100&apiKey=${NEWS_APIKEY}`;
33      if (category === 'local') {
34          response = await getLocalNews();
35          return response;
36      }else{
37          try {
38              response = await fetch(new Request(url));
39              response = await response.json();
40              response.articles.forEach(element => {
41                  if (element !== undefined) {
42                      element.id = 'a' + articleId;
43                      articleId++;
44                  }
45              });
46              return response.articles;
        }
```

Es erfolgt der Zugriff auf die NewsAPI, um auf die verschiedenen Artikel-Kategorien zugreifen zu können. Bei einer if-Abfrage, ob als Kategorie „lokal“ ausgewählt wurde, wird dann, wenn dies zutrifft, die *getLocalNews()*-Methode aufgerufen, um Lokalnachrichten zurückzugeben. Bei der *getLocalNews()*-Methode wird der *locationName* des Benutzers durch *getLocationName()* abgeholt. Diese Methode beinhaltet als Rückgabewert die aktuelle Position des Nutzers. Der Ortsname wird dann in die API gegeben, die dann passende Artikel zur jeweiligen Stadt, in der sich der Benutzer befindet ausgibt.

getNews() fungiert als Wrapper für die *getCategoryNews()*-Methode. Hierbei werden die deutschen Kategorien in die englische Version übernommen. Außerdem werden mit *getGoodNews(allArticles)* alle positiven Artikel herausgefiltert.

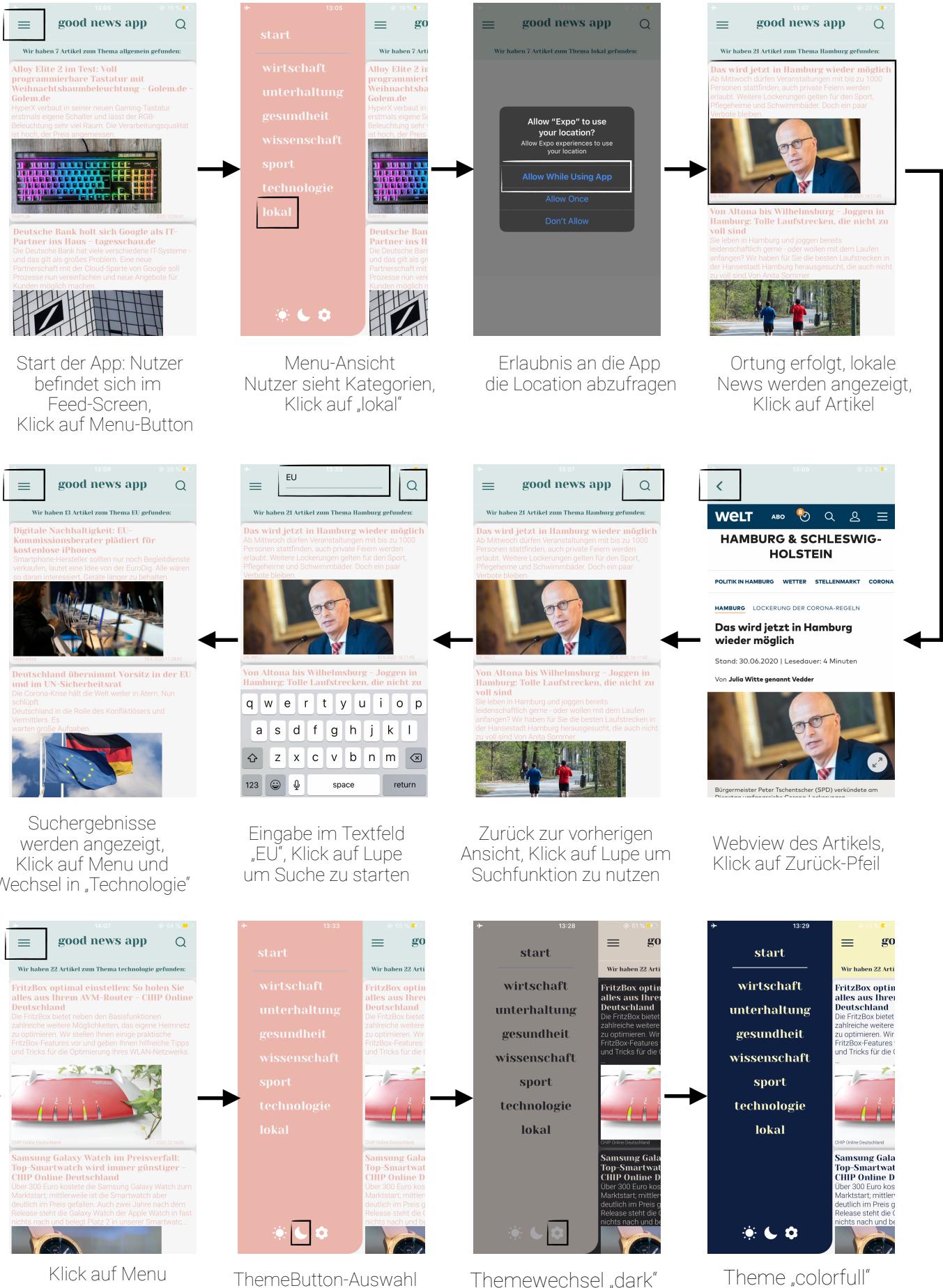
Mit *getKeyWordNews()* wird der User-Input vom Inputfeld als Keyword in die NewsAPI gegeben, um sich dann schließlich wieder mithilfe von *getGoodNews()* die positiven Artikel passend zum Suchbegriff ausgeben zu lassen.

```
158  let goodArticles = [];
159  try {
160      let response = await getSentiment(endpoint+path, requestOptions);
161      response = JSON.parse(response);
162      console.log(response);
163      response.documents.forEach(item => {
164          if (item !== undefined) {
165              if (item.score > 0.65) {
166                  goodArticles.push(allArticles[parseInt(item.id)]);
167              }
168      }
        })
```

getSentiment() ist die Filterfunktion für die zweite verwendete API, Microsoft Azure, die für die Textanalyse der Artikel zuständig ist. Der „Good-News-Filter“ wird mithilfe von *getGoodNews()* erstellt. Hierbei werden ein Header, Body und Text zusammengestellt und in *getSentiment()* gegeben. Wenn der festgelegte Score der Analyse größer als 0,65 ist wird der Artikel als positiv bewertet und in ein Array aus positiven Artikeln gegeben. Anschließend wird die Methode zur Ortung des Nutzers aufgeführt (*getLocationName()*) sowie die Methode, durch die der User seine Erlaubnis geben kann, seinen Standort zu teilen.

Benutzerdokumentation

Anschließend folgt eine exemplarische User Journey um die Benutzung der Good News App deutlich zu machen.



Entwicklungsdocumentation

Die Projektorganisation erfolgte in Bezug auf die Kommunikation aufgrund der aktuellen Situation hauptsächlich über Microsoft Teams. Zu Anfang haben wir bereits schnell die Stärken des jeweils anderen gesehen und uns somit in die Felder Design und Development aufgeteilt.

Den Zeitplan, den wir uns vorher grob aufgestellt haben sowie die gesetzten Meilensteine konnten wir gut einhalten. Einzige Problematiken stellten sich in Bezug auf den Crossplatform-Ansatz heraus, da die App zunächst nur für Android funktional war. Dieses Problem konnten wir allerdings beheben und haben somit ein Ergebnis für Android- und iOS-User entwickelt.