Studienarbeit

Web-Anwendungsentwicklung

von Helge Kohl

Inhaltsverzeichnis

Aufbau

Client

Vue.js

Die Clientseite wurde mittels Vue.js als Single-Page-Anwendung implementiert. Zum einen habe ich diese Entscheidung gefällt da ich Vue vorher nicht kannte und ich die Gelegenheit nutzen wollte etwas Neues zu lernen, außerdem kenne ich größere Anwendungen mit JQuery und empfinde diese Lösung eher als weniger elegant.

In einem Vue-Objekt werden alle relevanten Daten gespeichert und über Bindings an die Oberfläche gebunden. Die Daten werden in die Kategorien Stichwörter (Titel, Untertitel, Beschreibung), Sender, Start- und Endzeitpunkt gegliedert. Die Suche in Stichwörtern unterscheiden sich von Sendern dahingehend, dass sie einen Typ und die Art der Verknüpfung (und/oder) enthält, während die Verknüpfung von Sendern mit „und“ keinen Sinn ergibt. Start- und Endzeitpunkt werden als Intervall angegeben, somit ist es möglich eine Sendung, die zwischen zwei Zeitpunkten gestartet bzw. geendet hat, zu suchen. Ein QueryResult Objekt enthält die Suchergebnisse sowie Daten zur Pagination, Facetten und Sortierung. Weiterhin enthält das Vue-Objekt noch weitere Datenfelder zur Navigation zwischen einzelnen Seiten, der Bestimmung des heutigen Datums, der Sendersprache sowie ein Array mit Autocomplete-Vorschlägen.

vue-datetime

Zur Eingabe von Datum und Zeit wird der Datetime-Picker vue-datetime verwendet. Dieser wird lediglich eingebunden und kann dann direkt verwendet werden. Die Werte, die dieser liefert, sind direkt im richtigen Format für Solr, es muss lediglich der Zeitzonenstempel entfernt werden.

Boostrap/Bootstrap-Vue

Zum Erstellen der Oberfläche habe ich Bootstrap bzw. Bootstrap-Vue verwendet. Da Bootstrap selbst mit JQuery arbeitet war durch die Verwendung von Bootstrap-Vue dies auch ohne möglich. Der Vorteil von Bootstrap sind die vielen vordefinierten UI-Elemente und das einfache Konfigurieren des Seitenlayouts.

Font Awesome

Die Icons der Anwendung wurden von der Icon-Bibliothek Font Awesome bereitgestellt. Nach der Einbindung können einfach über CSS-Klassen Icons zugewiesen werden.

Server

Auf der Serverseite der Anwendung ist lediglich ein einfacher Node-Server mit dem Node-Modul Nodemon. Durch dieses Modul wird die Entwicklung stark beschleunigt da der Server nicht nach jeder Änderung neugestartet werden muss.

Zusätzlich liegt hinter dem Node-Server ein Solr-Server für Volltextsuche in einem EPG-Datensatz.

Kommunikation

Zur Kommunikation zwischen den einzelnen Parteien der Promise basierte HTTP Client Axios verwendet. Da ich die Verwendung von Jquery vermeiden wollte und zusätzlich neues ausprobieren wollte bin ich bei meinen Recherchen auf Axios gestoßen, zusätzlich finde ich die die Promise API sehr interessant und ich denke die diese ist in unserem Fall auch sehr nützlich da durchaus größere Datenmengen abgefragt werden können. Ein weiterer Grund für die Verwendung von Axios ist natürlich, dass Requests sehr einfach zu implementieren sind.

Requests von der Client-Seite sind neben den Anfragen für den Sendungen auf die Dateien im Static-Ordner beschränkt.

Zusätzlich habe ich für den Node-Server ein kleines Modul implementiert, um den Zugriff auf den Solr-Server zu vereinfachen. Dieses enthält alle relevanten Daten für Requests an den Solr-Server.

Oberfläche

Die Oberfläche ist in mehrere kleine Unterseiten gegliedert. So gibt es eine Seite für die Eingabe der Suchdaten, eine für die Suchergebnisse, eine für die aufgenommenen Sendungen und eine Seite für Einstellungen. Seiten mit tendenziell großem Inhalt besitzen einklappbare Elemente, um die Navigation komfortabel zu halten. Die Navigation zwischen den Seiten geschieht über eine Navigationsleiste am oberen Ende der Seite. Zusätzlich wurde die Oberfläche Full-Responsive entworfen, um die Anwendung auch mit mobilen Geräten nutzen zu können. Die Implementierung folgte dem Ansatz „Mobile-First“, da ich bereits öfters festgestellt habe, dass es wesentlich einfacher ist eine Seite für mobile Geräte zu implementieren und diese dann für Desktop-Geräte erweitern als umgekehrt. Die Umsetzung des responsiven Designs wurde größtenteils mit Bootstrap umgesetzt, lediglich kleinere Details mussten einem eigenen CSS-File angepasst werden.

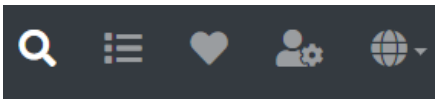

Abbildung 1: Menü

Dank der Verwendung von Vue war es auch sehr einfach bereits verwendete Interface-Elemente an anderen Stellen wieder zu verwenden. So ist zum Beispiel die Seite der aufgenommenen Sendungen dieselbe wie die Seite mit den Suchergebnissen, lediglich einige UI-Elemente werden weggelassen.

Suche

Die Eingabe der Suchparameter erfolgt in den drei Kategorien „Titel, Untertitel und Beschreibung“, „Sender“ und „Datum“. Jede der Kategorien ist einklappbar implementiert, um die Navigation komfortabler zu gestalten.

Zur Stichwortsuche in Titel, Untertitel und Beschreibung können dynamisch Suchfelder hinzugefügt werden. Für jedes dieser Suchfelder ist ein Textfeld vorgesehen, um Freitexteingaben entgegenzunehmen. Zusätzlich gibt es Auswahlfelder, um zu bestimmen ob das Suchfeld enthalten oder nicht enthalten sein soll sowie um die Art der Verknüpfung und den Typ der Suchfelder festzulegen. Dies ist so gestaltet das die Eingabe nahezu fliesend gelesen werden kann, somit lassen sich komplexere Suchanfragen einfach zusammenbauen.

Die Sendersuche ist eine abgespeckte Version der Stichwortsuche. Hier wird auf die Verknüpfung der Eingabe verzichtet, da jede Sendung exakt einem Sender zugeordnet ist und somit „und“-Kombinationen der Eingabe zu keinem Ergebnis führen würden.

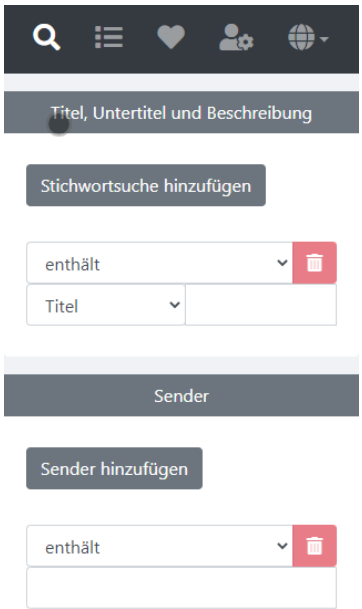

Abbildung 2: Suche in der Mobile-Ansicht

Die Suche nach Start- und Endzeitpunkt einer Sendung erfolgt über vier Date-Picker, jeweils zwei für Start und Ende. Somit kann für jedes der beiden Felder ein Zeitraum angegeben werden. Die Eingabe erfolgt in zwei Schritten, erst wird ein Datum abgefragt danach die Zeit. Bei der Auswahl des Date-Pickers wurde wieder darauf geachtet, dass dieser wieder Mobile-Friendly ist.

Zusätzlich zu den Eingabefeldern gibt des Buttons mit denen man die Eingabe auf die vordefinierten Zeiträume „Vormittag“, „Nachmittag“, „Abend“ und „Nacht“, welche jeweils sechsstündige Zeiträume festlegen. Dieser Zeitraum wird dann auf den vorher gewählten Tag angewendet, sollte jedoch keine Eingabe vorhanden sein wird das als „Heute“ festgelegte Datum verwendet.

Um die Datumseingaben zu entfernen wurde ein „Zurücksetzen“-Button bereitgestellt, der die Eingabe löscht.

Ergebnisse

Auf der Ergebnisseite werden die gefundenen Sendungen dargestellt. Diese werden standardmäßig im zugeklappten Zustand hinzugefügt und zeigen lediglich das Senderlogo, Zeitraum und Titel der Sendung an. Um die Sendung als Aufnahme zu markieren ist am rechten Ende ein Button, welcher je nach Zustand anders dargestellt wird. Zusätzlich ist ein weiterer Button mit einem Link zur Mediathek-Suche für den Titel im Kopf des Containers vorhanden.

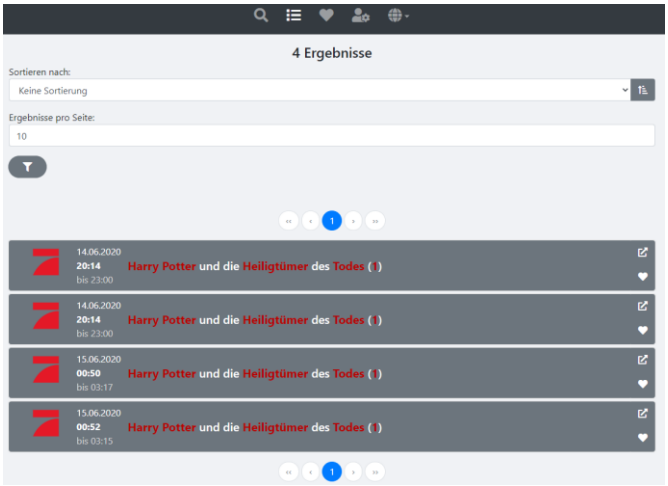

Abbildung 3: Ergebnisseite Desktop-Ansicht

Im ausgeklappten Zustand werden dann zusätzlich Untertitel und Beschreibung der jeweiligen Sendung ausgegeben. Zusätzlich wird am rechts oben ein weiterer Link zur Mediathek-Suche ausgegeben, welcher nach dem Untertitel der Sendung sucht. Sendungen mit einer Aufzählung der Produktions-Crew werden enthalten im Namen der Person einen Link zur Suche in der IMDB-Datenbank.

Am oberen Ende der Seite wird die Anzahl der gefundenen Ergebnisse ausgegeben und darunter einige Möglichkeiten die gefundenen Ergebnisse zu filtern. Eine ausklappbare Sidebar gibt zusätzlich Möglichkeiten zum Facettieren der Suchergebnisse an. Dafür sind Facetten zum Einschränken des Zeitraumes sowie Facetten zu Sendern möglich. Die Auswahl ist über Checkboxen möglich und ausgewählte Facetten werden zusätzlich außerhalb der Sidebar aufgelistet. In dieser Auflistung können die Facetten auch wieder entfernt werden.

Um die Anfragen an den Server, bzw. dessen Antwort, klein zu halten wird eine Pagination angeboten. Diese gibt standardmäßig zehn Sendungen pro Seite aus, es kann aber ausgewählt werden ob nicht bis zu 100 Sendungen auf einer Seite dargestellt werden sollen.

Aufnahme

Die Seite der aufgenommenen Sendungen ist grundsätzlich wie die der normalen Suchergebnisse aufgebaut. Nur die Facetten und Sortierung fallen hier weg.

Einstellungen

Im Einstellungsmenü gibt es die Möglichkeit das heutige Datum festzulegen und einige Facetten zu konfigurieren. Um die anderen Seiten nicht zu überladen wurden diese Optionen in eine eigene Seite ausgelagert. Die Eingabe des Datums erfolgt wieder über einen Date-Picker.

Sprachwahl

Um die Auswahl der Sender auf eine Sprache zu begrenzen, gibt es in der Hauptnavigation ein Dropdownfeld, welches die Sprache festlegt. Diese Einstellung wurde bewusst nicht auf die Einstellungsseite gelegt, um dem Nutzer von jeder Seite aus die Wahl der Sendersprache zu ermöglichen.

Suche

Beim Wechseln der Seite zur Ergebnisseite werden die Nutzereingaben mit Axios als asynchrone Post-Request an den Node-Server gesendet. Dieser baut mit den Angaben dynamisch einen Querylink für den Solr-Server zusammen. Dieser beinhaltet neben den Suchanfragen zusätzlich vordefinierte Facetten und Angaben zur Anzahl der Ergebnisse, welche über die Pagination gesetzt werden. Für das Highlighting wird der Link um benötigte Parameter erweitert. Zum Schluss war zu beachten das der erstellte Link richtig codiert wurde.

Die Antwort des Solr-Servers um Daten zum Senderlogo und Senderart, Links zu Artisten und Mediatheken, Highlighting und ob die Sendung bereits aufgenommen wurde ergänzt. Zuletzt werden noch Linebreaks mit HTML-Tags ersetzt.

Facetten

Zeitraum

Zur Auswahl stehen „Heute“, „Morgen“, „Gestern“, „nächste *n* Tage“ und „vorherige *n* Tage“ Facetten. Diese werden abhängig vom heutigen Datum, welches der Benutzer einstellen kann, erstellt. Diese beziehen sich immer auf den Startzeitpunk einer Sendung und werden jeweils als einzelne Facet-Querys an die Anfrage angehängt. Zur Generierung werden die Funktionalitäten zum Rechnen mit Daten von Solr genutzt. So wird das Datum auf den Tag gerundet und Tage werden dynamisch addiert oder subtrahiert.

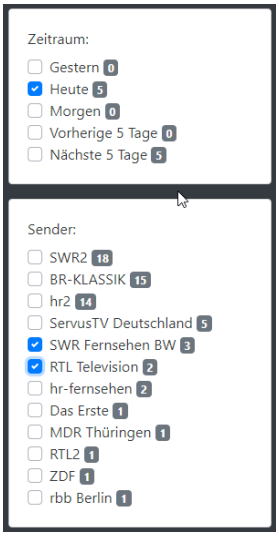

Abbildung 4: Facetten-Filter

Sender

Die Sender-Facetten werden über einen Facet-Field Query implementiert. Dieser gibt zu den Ergebnissen alle Sender in einem Array zurück. In diesem Array werden auf der Client-Seite alle Sender ohne Ergebnis entfernt.

Filter

Datum

Die Filterung nach Datum geschieht, sofern angegeben, über den normalen Query.

Facetten

Zusätzlich gibt es die Option nach den oben genannten Facetten zu filtern. Diese können vom Nutzer in der Oberfläche ausgewählt werden und der Node-Server sendet diese dann als Filter-Query mit. Dadurch können die Suchergebnisse eingeschränkt werden ohne das der eigentliche Query angepasst werden muss. Wird ein Facetten-Filter entfernt wird nur der zugehörige Filter gelöscht.

Dies hat den Vorteil das die Filter zu den beiden Facettenarten dynamisch hinzugefügt werden können und die Anzahl der gefundenen Ergebnisse praktikabel aktualisiert werden kann.

Sortierung

Weiterhin können die Ergebnisse nach den Feldern „Titel“, „Untertitel“, „Sender“, „Start“ und „Ende“ sortiert werden. Über einen Button kann gewählt werden ob dies auf- oder absteigend geschehen soll. Die Sortierung erfolgt bereits auf dem Solr-Server über das „Sort“-Feld.

Pagination

Um den Nutzer direkt mit allen Ergebnissen zu überfluten wurde eine Pagination implementiert. Diese nutzt die „Start“ und „Row“ Felder des Solr um immer eine begrenzte Anzahl an Ergebnissen zurückzuliefern. Über ein Eingabefeld kann der Nutzer wählen, wieviele Einträge zwischen 0 und 100 er auf einmal angezeigt bekommen möchte. Dies ist zusätzlich positiv für die Performance der Anwendung. Welche Einträge angezeigt werden wird über die Seitenzahl und die gewählte Anzahl der Einträge berechnet.

Highlighting

Beim Highlighting wird über alle gefundenen Highlighting-Objekte iteriert. Mit einem regulären Ausdruck wird aus dem Highlighting-Objekt der zu kennzeichnende String ermittelt. Mit einer weiteren Regex-Suche werden alle stellen die gekennzeichnet werden sollen ermittelt und um ein HTML-Tag ergänzt. Dabei ist zu beachten, dass in den generierten Links kein Tag gesetzt wird. Alternativ könnte man vor der Linkgenerierung highlighten allerdings wäre diese wesentlich schwieriger gewesen.

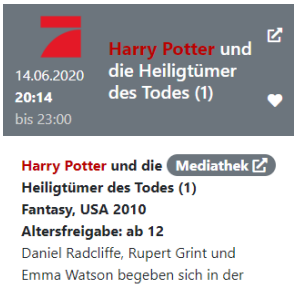

Abbildung 5: Highlighting von Harry Potter

Linkgenerierung

IMDB-Links

Auch hier wird wieder mit regulären Ausdrücken gearbeitet. Hier habe ich jeweils einen Ausdruck für die Mitwirkenden und einen für die Darsteller erstellt.

Mitarbeiter

Der Ausdruck für die Mitarbeiter sucht lediglich nach Namen mit den angegebenen Schlüsselworten vor dem Namen. Zu beachten war das pro Schlüsselwort auch mehrere Personen aufgelistet werden konnten und die Namen teilweiße aus mehr als zwei Worten bestanden und abgekürzt sein können.

```
1. Regie: Griffin Dunne
Drehbuch: Akiva Goldsman, Robin Swicord, Adam Brooks
Komponist: Alan Silvestri
Kamera: Andrew Dunn
Buch/Autor: Axel Bold, Edin Hadzimahovic
Schnitt: Michele Gentile
Musik: Uwe Schenk
Regie: Tapaas Chakravavarti, Siddharth Vasudeva
```

Abbildung 6: Regex-Suche nach Mitarbeitern

Hier wurde mit dem „Positiv Lookbehind“-Operator (?<=…) nach den Schlüsselworten und ggf. nach Auflistungen von Personen gesucht.

Darsteller

Der Ausdruck für Darsteller sucht nach dem Schlüsselwort „Darsteller:“ und unterscheidet dann nach zwei Varianten von Auflistungen. Die erste Variante beinhaltet den Namen gefolgt von einer Klammer mit dem Namen der Rolle. Die zweite Variante überspringt den Namen der Rolle und den darauffolgenden Bindestrich zum Trennen.

```
Darsteller:
Daniel Radcliffe (Harry Potter)
Rupert Grint (Ron Weasley)

Darsteller:
Vater Niccolo - Boris Aljinovic
Marco - Elia Francolino
```

Abbildung 7: Regex-Suche nach Darstellern

Mediathek-Link

Die Mediathek-Links werden ausschließlich aus dem Titel bzw. Untertitel erstellt. Hier war nur zu beachten das Leerzeichen richtig codiert sind.

Aufnahmeprogrammierung

Hier ist zu erwähnen, dass ich die Aufnahmeprogrammierung eher als „Favoriten“ umgesetzt habe. Dies erschien mir angebrachter da wir nicht wirklich Sendungen aufzeichnen.

Aufnahme hinzufügen

Bei der Aufnahmeprogrammierung werden Sendungen in der „aufnahme.json“-Datei auf dem Server abgelegt. Der Client sendet das Sendungsobjekt an den Node-Server und aktualisiert danach seine Suchergebnisse, um die Darstellung der bereits aufgenommenen Sendungen richtig darzustellen.

Der Node-Server zuerst ob die Datei bereits existiert und erstellt diese bei Bedarf. Daraufhin wird geprüft ob die Sendung bereits aufgenommen wurde, um mehrfache Aufzeichnung zu unterbinden. Wenn das zu speichernde Objekt bereits Highlighting besitzt, wird dieses vor dem Speichern entfernt.

Aufnahme entfernen

Beim Entfernen einer Aufnahme wird auf Client-Seite wieder das zu entfernende Objekt an den Server gesendet und die Suchergebnisse aktualisiert. Dieses Mal wird je nachdem ob aktuell die normalen Suchergebnisse oder die Aufnahmen dargestellt werden, die jeweilige Seite aktualisiert.

Der Node-Server sucht in der Aufnahme-Datei nach dem Sendungsobjekt, dass der Client übermittelt hat und entfernt dieses.

Aufnahme abfragen

Bei der Abfrage wird vom Client lediglich die Anzahl der Sendungen pro Seite und die aktuelle Seite übergeben. Als Antwort empfängt dieser die Sendungen und die Gesamtanzahl der aufgenommenen Sendungen. Diese werden wie bereits erwähnt in der gleichen Form wie die Suchergebnisse dargestellt.

Auf der Server-Seite wird wieder mittels Seitenzahl und Ergebnisse pro Seite ermittelt welche Sendungen zurückgegeben werden müssen. Da die Sendungen in der Datei in einem Array abgelegt sind muss dafür nur über den ermittelten Indexbereich iteriert werden.

Autocompletion

Bei meinen Recherchen zur Autocompletion bin ich auf eine weitere Lösung mit Solr gestoßen, die SuggestComponent. Aufgabe dieser Komponente ist es den Benutzer mit automatischen Vorschlägen für Suchanfragen zu unterstützen. Der Vorteil dieser ist, dass man wählen kann welches Dictionary man für einzelne Suchfelder nutzen möchte. So gibt es unter anderem auch Dictionarys mit Fehlerkorrektur. Den Suggester zu implementieren ist relativ einfach, man muss lediglich eine SearchComponent in der solrconfig.xml definieren und diese mit sogenannten Suggestern bestücken.

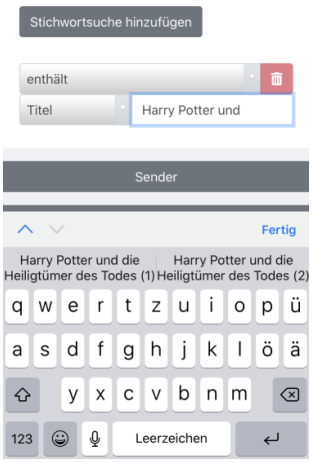

Abbildung 8: Autocompletion mit Safari auf iOS 13.6

In meinem Fall habe ich für Felder Titel, Untertitel und Sender, wobei bei jede davon auch eine französische Variante enthält. So wird bei der Suche nach Vorschlägen auf sprachspezifische Vorschläge geachtet. Alle Suchkomponenten sind mit der AnalyzingInfixLookupFactory implementiert. Diese sucht nach passenden Vorschlägen im gesamten Suchfeld.

Zusätzlich zur SuggestComponent musste der Solr-Server noch um einen Requesthandler erweitert werden. In diesem wird unter anderem angegeben wie viele Vorschläge ermittelt werden sollen. Dies habe ich auf zehn Vorschläge beschränkt.

Da die Vorschläge für jedes gefundene Feld ausgegeben werden und z.B. Titel durchaus öfter vorkommen können mussten Duplikate noch herausgefiltert werden. Es wäre auch möglich gewesen die Eingabe im Vorschlag mit einem Highlighting zu versehen, darauf habe ich aber verzichtet.

Quellen

[1] Apache Solr Reference Guide, URL: https://lucene.apache.org/solr/guide/ (Stand: 25.07.2020)

[2] Vue.js, URL: https://vuejs.org/v2/guide/ (Stand: 25.07.2020)

[3] BooststrapVue, URL: https://bootstrap-vue.org/docs/components/form-input (Stand: 25.07.2020)

[4] Font Awesome, URL: https://fontawesome.com/ (Stand: 25.07.2020)

[5] Faraz Kelhini: How to make HTTP requests like a pro with Axios, URL: https://blog.logrocket.com/how-to-make-http-requests-like-a-pro-with-axios/ (Stand: 25.07.2020)

[6] Alessandro Benedetti: Solr: „You complete me!“: The Apache Solr Suggester, URL: https://sease.io/2015/07/solr-you-complete-me.html (Stand: 25.07.2020)