Beuth Hochschule für Technik Berlin

Fachbereich VI – Informatik und Medien

Entwurf und Entwicklung einer generischen Single-Page Application für die Realisierung von Open Data Plattformen

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Science

Studiengang: Medieninformatik Bachelor

Autor: Dennis Ritter
Matrikelnummer: 821624

Version vom: 14. Oktober 2017

Erstkorrektor: Fabian Kirstein
Zweitkorrektor: Prof. Dr. Thomas Off

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
CSS	Cascading Style Sheets
DCAT	Data Catalog Vocabulary
DI	Dependency Injection
DOM	Document Object Model
HTML	Hyper Text Markup Language
JS	JavaScript
MVC	Model View Controller Entwurfsmuster
NPM	Node Package Manager
SPA	Single-Page Application
URL	Uniform resource locator

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

1. Einleitung und Zielsetzung

1.1. Motivation

Der Geschäftsbereich Digital Public Services (DPS) des Fraunhofer-Instituts für Offene Kommunikationssysteme (FOKUS) beschäftigt sich unter anderem mit dem Entwurf und der Entwicklung von Open Data Plattformen auf nationaler und europäischer Ebene. Das Institut war für die Umsetzung des Prototypen des Datenportals für Deutschland[1] verantwortlich, hat sich an der technischen Realisierung des Transparenzportals Hamburg[2] beteiligt und ist aktuell technischer Partner des Europäischen Datenportals[3]. Auch in Zukunft wird FOKUS an der Verwirklichung solcher Webplattformen beteiligt sein.

Grundsätzlich gilt, dass die öffentlich zur Verfügung gestellten Daten, über jeweils individuell angepasste Frontend Webapplikationen für den Nutzer abrufbar sein sollen. Die Anforderungen an die genauen Funktionalitäten und das Design ändern sich entsprechend des Auftraggebers.

Aufgrund der konstanten Nachfrage an Open Data Plattformen ist es nun das Ziel des FOKUS eine Basisanwendung zu entwickeln, welche als Grundlage für aktuelle und zukünftige Oberflächen von Open Data Webportalen dient. Der Anspruch ist es mithilfe moderner Frontend-Technologien ein Framework zu entwickeln, das es internen und externen Entwicklern ermöglicht Funktionalitäten und Darstellung der Applikation zu verändern und zu erweitern ohne die Kern-Anwendung dabei unmittelbar modifizieren zu müssen.

1.2. Zielsetzung

Ziel dieser Arbeit ist es eine Kern-Anwendung für die Realisierung von Oberflächen existierender und zukünftiger Open Data Plattformen in Form einer Single-Page Application zu entwerfen und zu entwickeln. Um flexible Einsatzmöglichkeiten zu gewährleisten, sollen Oberflächen, die auf dieser Applikation basieren, individualisierbar sein ohne den Quellcode der Kern-Anwendung dabei selbst modifizieren zu müssen. Dies beinhaltet sowohl Änderungen am Design der Benutzeroberfläche, als auch inhaltliche Anpassungen ausgewählter Komponenten und individuell definierbare Schnittstellen zur Beschaffung der darzustellenden Daten. Beim Entwurf der Kern-Anwendung sollen dafür moderne Methoden und Technologien evaluiert und bei entsprechender Eignung während der Umsetzung eingesetzt werden. Als Grundlage für die Auswahl der einzusetzenden Hilfsmittel und Werkzeuge sollen dabei sowohl die Anforderungen der Endnutzer von Open Data Plattform-Oberflächen, als auch die Anforderungen von Entwicklern dieser Webanwendungen dienen.

[1] https://www.govdata.de/
[2] http://transparenz.hamburg.de/
[3] https://www.europeandataportal.eu/

2. Grundlagen

In den folgenden Kapiteln werden die Grundlagen dargestellt, welche zum Verständnis der Arbeit nötig sind. Im Zuge dessen werden in einem ersten Schritt die Begriffe Open Data und Single-Page Application näher erläutert, um anschließend einen Überblick über verschiedene Frontend-Technologien geben zu können.

2.1. Open Data

Als Open Data werden nach Jörn von Lucke und Christian Geiger Daten bezeichnet, die dem Interesse der Gesellschaft dienen und die durch Verwendung offener Nutzungsrechte von jedermann ohne Einschränkungen frei verwendet, weiterverarbeitet und verbreitet werden können. Entscheident ist dabei, dass die entsprechenden Daten keine personenbezogenen oder dem Datenschutz unterliegenden Daten enthalten dürfen. Weiterhin darf die Lizenz den Einsatzzweck des Werkes nicht einschränken und die Daten müssen in einer Form zugänglich gemacht werden, durch die weder finanzielle noch andere Hindernisse bei der Nutzung entstehen. Dies kann erreicht werden indem offene Datenformate verwendet werden, deren Spezifikationen öffentlich dokumentiert sind und die keine finanziellen oder andere Hindernisse bezüglich der Nutzung aufweisen. [Luc10, S. 2-3]

Von der öffentlichen Verwaltung erhobene Daten, die den oben genannten allgemeinen Open Data Richtlinien entsprechen, werden als Open Government Data bezeichnet. Dabei handelt es sich zum Beispiel um Geodaten, Landkarten, Wetterdaten, Satellitenbilder oder statistischen Daten aus den Bereichen Wirtschaft, Verkehr oder Tourismus. [fWuE] Die erhobenen Daten werden zeitnah auf Online-Plattformen bereitgestellt. So sind beispielsweise erhobene Verwaltungsdaten des Bundes, der Länder und Kommunen auf govdata.de[4] abrufbar oder auf dem Europäischen Datenportal[5] die Daten, die von den Mitgliedsländern der europäischen Union erhoben wurden. Die Open Data Plattformen beherbergen dabei nicht die tatsächlich erhobenen Daten selbst, sondern lediglich Metadaten, welche diese beschreiben. Die Metadaten enthalten neben Informationen, wie einem Titel, einer allgemeinen Beschreibung des Datensatzes oder dem Erstellungsdatum, auch eine Adresse in Form einer URL, die den Zugriff auf eine bestimmte Ressource ermöglicht. Meist kann eine Datei direkt über eine solche Adresse heruntergeladen werden. Alternativ oder ergänzend kann der Zugriff auf die Ressource aber auch auf andere Weise ermöglicht werden, beispielsweise durch eine API.

2.1.1. CKAN

CKAN ist eine Software die speziell zur Verwaltung von offenen Daten entwickelt wurde und häufig bei der Realisierung von Open Data Portalen verschiedener Behörden, wie dem europäischen Datenportal, dem Datenportal der Vereinigten Staaten von Amerika oder dem Datenportal der Bundesrepublik Deutschland, Anwendung findet. [Groa] Die Kernfunktionen von CKAN sind über eine Web-API einsetzbar und ermöglichen sowohl das Erstellen, Bearbeiten und Löschen von Datensätzen als auch das Abrufen der Daten mittels Facetten- oder Stichwortsuche. Die Software erlaubt außerdem die Entwickelung und Einbindung eigener

[4] https://www.govdata.de
[5] https://www.europeandataportal.eu

Erweiterungen. [Grob] Dadurch lässt sie sich gut erweitern und an individuelle Anforderungen anpassen.

2.1.2. DCAT-AP

DCAT-AP ist ein auf DCAT[6] basierendes, Format unabhängiges, standardisiertes Vokabular zur Beschreibung von Datensätzen des öffentlichen Sektors in Europa. Ziel ist es, die Kompatibilität zwischen verschiedenen Datenkatalogen zu verbessern und es Anwendungen zu ermöglichen die Daten verschiedener Plattformen zu nutzen. [Pro17] Durch die Einhaltung des standardisierten Vokabulars soll die Maschinenlesbarkeit der vorhandenen Metadaten der Datensätze verbessert und die Wiederverwendung begünstigt werden. [Uni15, S. 3] So besteht konkret die Möglichkeit, dass zum Beispiel Kommunen und Städte eigene Open Data Plattformen betreiben, dessen eingespeiste Daten zudem auf weiteren Open Data Plattformen, wie govdata.de verwendet und dargestellt werden können. Die auf govdata.de gesammelten Daten der einzelnen Regionen der Bundesrepublik Deutschland könnten dann ebenfalls weiterverwendet werden und beispielsweise über das europäische Datenportal abrufbar sein, so wie alle weiteren Datensätze, die durch das DCAT-AP-Vokabular beschrieben werden. Um die Kompatibilität zu gewährleisten, werden im DCAT-AP-Standard zunächst Klassen definiert, die mögliche Entitäten zur Beschreibung eines Datensatzes darstellen. Dabei handelt es sich entweder um Pflicht-Klassen, empfohlene Klassen oder optionale Klassen. Diese Klassen besitzen wiederum Eigenschaften, welche ebenfalls einer der drei Kategorien zugeordnet werden. [Uni15, S. 7-16] Abbildung 1 zeigt exemplarisch die Beschreibung von Pflicht-Eigenschaften und empfohlenen Eigenschaften der Klasse Datasets des DCAT-AP Standards.

4.3 Dataset

4.3.1 Mandatory properties for Dataset

Property	URI	Range	Usage note	Card
description	dct:description	rdfs:Literal	This property contains a free-text account of the Dataset. This property can be repeated for parallel language versions of the description.	1..n
title	dct:title	rdfs:Literal	This property contains a name given to the Dataset. This property can be repeated for parallel language versions of the name.	1..n

4.3.2 Recommended properties for Dataset

Property	URI	Range	Usage note	Card
contact point	dcat:contactPoint	vcard:Kind	This property contains contact information that can be used for sending comments about the Dataset.	0..n
dataset distribution	dcat:distribution	dcat:Distribution	This property links the Dataset to an available Distribution.	0..n
keyword/ tag	dcat:keyword	rdfs:Literal	This property contains a keyword or tag describing the Dataset.	0..n
publisher	dct:publisher	foaf:Agent	This property refers to an entity (organisation) responsible for making the Dataset available.	0..1
theme/ category	dcat:theme, subproperty of dct:subject	skos:Concept	This property refers to a category of the Dataset. A Dataset may be associated with multiple themes.	0..n

Abbildung 1: Eigenschaften eines Datensatzes nach DCAT-AP [Uni15, S. 20]

[6] DCAT - Data Catalog Vocabulary

2.2. Single-Page Applications

Eine Single-Page Application (SPA) ist eine Webanwendung, die aus nur einem HTML-Dokument besteht und alle benötigten HTML, JavaScript und CSS Quelltexte beim initialen Laden der Webanwendung abruft oder diese bei Bedarf dynamisch nachlädt. [Fla06, S. 497]

Im Folgenden werden die Vor- und Nachteile einer Single-Page Application diskutiert und daraus resultierende Einsatzmöglichkeiten erläutert.

2.2.1. Vorteile gegenüber einer klassischen Webanwendung

Eine klassische Webanwendung besteht aus mehreren, untereinander verlinkten HTML-Dokumenten und erfordert ein Neuladen der Seite sobald der Benutzer in der Anwendung navigiert oder sich der Anwendungszustand verändert. Dabei werden auch die Teile wiederholt vom Server angefragt, die sich nicht verändert haben. Daraus resultierend kommt es zu überflüssigem Datenverkehr und einem gestörten Präsentationsfluss für den Benutzer. [Zei15]

Bei einer Single-Page Application werden dagegen alle statischen Inhalte und Code-Bausteine der Webanwendung beim ersten Aufruf der Seite vom Server geladen. Navigiert der Benutzer innerhalb der SPA, rendert der Browser ausschließlich die Teile der Oberfläche neu, die einer Veränderung unterliegen. Dies geschieht durch einen clientseitigen Router, der Veränderungen der URL registriert und den Routen jeweils einen JavaScript Einstiegspunkt sowie ein HTML-Template zuordnet. [Vie17] Ferner werden nur zusätzlich benötigte Daten vom Server mittels Ajax[7]-Requests dynamisch nachgeladen.

Ein Vorteil dieser Eigenschaft ist eine schnellere Reaktionszeit und ein daraus resultierend besseres Nutzererlebnis. Die Anwendung verhält sich im Gegensatz zur klassischen Webanwendung eher wie eine Desktop-Anwendung und lässt einen Präsentationsfluss der Oberfläche ohne vermeidbare Wartezeiten zu. Darüber hinaus wird die Serverlast durch einen geringeren Datenverkehr reduziert. Dieser Effekt wird verstärkt, wenn zusätzlich zur Präsentationslogik, Teile der Geschäftslogik auf den Client verlagert werden und der Anwendungszustand ebenfalls vollständig vom Client verwaltet wird. Außerdem entsteht durch die Entkopplung vom Server die Möglichkeit einen Offline-Betrieb der Anwendung zu realisieren. [Zei15]

2.2.2. Herausforderungen einer SPA

Im Folgenden werden bekannte Problemstellungen, die bei der Entwicklung und dem Betrieb von Single-Page Applications aufkommen, erläutert und Lösungsansätze dargelegt.

Browserverlauf Da eine SPA per Definition nur aus einem HTML-Dokument besteht, werden beim Navigieren durch die Anwendung keine weiteren URLs zum Browserverlauf hinzugefügt, so wie es bei einer klassischen Webanwendung der Fall wäre. Dies verhindert eine Navigation mittels der Vor- und Zurück-Buttons des Browsers.

Mit der Veröffentlichung von HTML5 ist es jedoch möglich geworden den Browserverlauf durch die Verwendung von JavaScript zu manipulieren. So kann seitdem mittels der pushState() Methode manuell ein Eintrag im Browserverlauf hinzugefügt werden, was eine Navigation mit den Vor- und Zurück-Buttons des Browsers erlaubt. [Moz17]

[7] Ajax - Asynchronous JavaScript and XML

Suchmaschinenoptimierung Website Crawler die von Suchmaschinen-Anbietern eingesetzt werden, verwenden beim Durchsuchen der Website unter Umständen kein JavaScript, sondern analysieren lediglich vorhandene HTML-Seiten auf Struktur, Metadaten und Inhalte.

Eine Möglichkeit dem entgegenzuwirken stellt das Prerendering dar. Ein Prerender-Service ermittelt zunächst ob es sich bei dem Benutzer einer Seite um einen Bot handelt. Sollte dies der Fall sein, wird eine bereits vorgerenderte, statische Seite ausgeliefert, die ohne den Einsatz von JavaScript problemlos vom Crawler gelesen und analysiert werden kann. [Dou16]

Allerdings setzt der Crawler von Google Search seit 2015 auch JavaScript bei der Untersuchung von Webanwendungen ein, was ihm eine Indexierung der logischen Seiten innerhalb einer Single-Page Application ermöglicht. Das Prerendering für den Google-Crawler ist demnach nicht mehr nötig, sodass Google explizit davon abrät, die Technologie zu anderen Zwecken als der Geschwindigkeitsoptimierung zu verwenden. [Nag15] Da Google im Jahr 2017 mit rund 92 Prozent weltweitem Marktanteil der Suchmaschinen-Anbieter deutlicher Marktführer ist [Sta17b], kann davon ausgegangen werden, dass andere Anbieter dem Trend folgen werden und Prerendering zu SEO[8]-Zwecken auch dort nicht mehr nötig ist.

Initialer Download Wie bereits weiter oben festgestellt wurde, laden Single-Page Applications beim ersten Aufruf den gesamten Quelltext, der zum clientseitigen Betrieb der Anwendung notwendig ist. Dies führt dazu, dass das Datenpaket, welches der Client beim Initialen Ladevorgang herunterladen muss, größer ist, als das einer klassischen Webanwendung, was zu einer längeren Wartezeit führt. Vor allem beim Betrieb auf mobilen Endgeräten, die häufig über keine stabile Internetverbindung verfügen, kann es dadurch zu langen Ladezeiten kommen, die den Nutzer unter Umständen vom Gebrauch der Webanwendung abhalten. [Sil] Um die Größe des initial zu ladenden Pakets so klein wie möglich zu halten, kann auf Werkzeuge zur Optimierung des entwickelten Codes zurückgegriffen werden. Mithilfe sogenannter Task Runner oder Bundler und dazugehörigen Erweiterungen können beispielsweise Bildgrößen optimiert, CSS-Duplikate verworfen oder der JavaScript Quellcode stark verkürzt werden. [Sac15]

Anforderungen an Entwickler Die Client-Architektur einer Single-Page Application ist komplexer als die einer klassischen Webanwendung, wenn UI-Logik und Geschäftslogik vom Server auf den Client übertragen werden. Zur Lösung dadurch entstehender Herausforderungen an den Frontend-Code, können zahlreiche JavaScript-und CSS-Frameworks oder Bibliotheken eingesetzt werden. Allerdings führt der Einsatz verschiedener Hilfsmittel und eine komplexere Architektur auch dazu, dass die Anforderungen an die Entwickler steigen. Insbesondere wenn mehrere Tools kombiniert werden kann es bei der Suche nach geeigneten Fachkräften zu Schwierigkeiten kommen, da das Feld der Frontend-Tools sehr umfangreich und in stetiger Bewegung ist. [Lie]

[8] SEO - search engine optimization, bzw. Suchmaschinenoptimierung

2.3. Werkzeuge der Frontend-Entwicklung

In den folgenden Kapiteln werden Technologien, die für im Entwicklungsprozess von clientseitigen Webanwendungen eingesetzt werden, beschrieben und anhand ausgewählter Beispiele konkretisiert.

2.3.1. Node.js und der Node Package Manager

Node.js[9] ist eine Event-basierte, asynchrone Laufzeitumgebung zur Entwicklung von Netzwerkanwendungen. [Fou] Den Kern bildet die von Google entwickelte V8-Engine, welche auch im Browser Chrome als JavaScript-Engine zum Einsatz kommt. Node.js ermöglicht damit den serverseitigen Einsatz von JavaScript. [MW17]

Die Installation von Node.js beinhaltet außerdem den Node Package Manager (NPM), ein Programm zum Herunterladen von Node.js Paketen, die bei der Entwicklung von Software verwendet werden können. [Dat] Der NPM-Katalog umfasst derzeit etwa 475.000 frei nutzbare Pakete und ist damit der umfangreichste Software-Katalog der Welt. [nIa] Ein Paket oder Modul besteht aus einem Ordner, in dem sich eine oder mehrere Dateien befinden. Die Metadaten zum jeweiligen Paket befinden sich immer in einer Datei namens package.json, die in Listing 1 beispielhaft dargestellt wird. Die Idee hinter dem Node Package Manager ist es, Entwicklern eine Plattform zu geben, um ihren Code zu veröffentlichen und somit anderen Entwicklern zur frei zur Verfügung stellen zu können. Ein Modul sollte dabei bestenfalls ein bestimmtes Problem besonders gut lösen. Auf Basis dieser Idee können komplexere Anwendungen entworfen werden, welche unter Umständen eine Vielzahl von NPM-Paketen einsetzen. [nIb]

```
1  {
2    "name": "module-name",
3    "version": "10.3.1",
4    "description": "An example module to illustrate the usage of a
       package.json",
5    "author": "Your Name <you.name@example.org>",
6    "scripts": {
7      "test": "vows --spec --isolate",
8      "start": "node index.js",
9      "predeploy": "echo im about to deploy",
10   },
11   "main": "lib/foo.js",
12   "repository": {
13         "type": "git",
14         "url": "https://github.com/nodejitsu/browsenpm.org"
15   },
16   "keywords": ["nodejitsu", "example", "browsenpm"],
17   "dependencies": {
18     "async": "~0.8.0",
19     "express": "4.2.x",
20   },
21   "devDependencies": {
```

[9] https://nodejs.org/en/

```
22      "vows": "^0.7.0",
23    },
24  },
25    "license": "MIT"
26  }
```

Listing 1: NPM package.json Beispiel

2.3.2. JavaScript Versionierung

Die Programmiersprache JavaScript ist eine Implementierung der vom ECMA International entworfenen, standardisierten Sprache ECMAScript. Im Jahr 2009 wurde ECMAScript Version 5 veröffentlicht, die aktuell in allen gängigen Browsern implementiert ist. Auf dem ECMAScript 5 Standard basierender JavaScript-Code kann demnach von allen relevanten Browsern interpretiert werden. Im Jahr 2015 wurde die sechste Edition des ECMAScript-Standards veröffentlicht [Int15, S. XVIII], welche weitreichende neue Möglichkeiten und Änderungen für die Entwicklung mit JavaScript mit sich brachte. Es wurde außerdem beschlossen, von nun an jedes Jahr einen neuen ECMAScript-Stand zu veröffentlichen. Einhergehend mit dieser Entscheidung wurde die zunächst als ECMAScript 6 bezeichnete sechste Version des Standards, in ECMAScript 2015 umbenannt. Die folgenden Veröffentlichungen sollen zukünftig dieser Benennung folgen und jeweils anstelle der Versionsnummer das Jahr der Veröffentlichung in ihrer Bezeichnung tragen. [McC15]

JavaScript-Code der auf neueren Standards als ECMAScript 5 basiert, kann derzeit allerdings noch nicht von allen Browsern interpretiert werden. Um aktuelle JavaScript Versionen dennoch in der Entwicklung einzusetzen, können JavaScript Transpiler eingesetzt werden, die den Quellcode in eine von Browsern ausführbare Version zu übersetzen. [Sen16]

2.3.3. JavaScript Transpilers

Ein Transpiler liest Quellcode und übersetzt diesen zu einer anderen Programmiersprache. Für die Entwicklung JavaScript-basierter Anwendungen wurden verschiedene Abstraktionen von JavaScript, wie zum Beispiel Typescript[10] oder Coffeescript[11] entworfen, welche syntaktische Änderungen beinhalten oder bestimmte Eigenschaften der Programmiersprache verändern. Wird eine JavaScript-Abstraktion oder eine ECMAScript-Spezifikation bei der Entwicklung eingesetzt, die von der Laufzeitumgebung der Ziel-Browser nicht interpretiert werden können, muss der Quellcode so kompiliert werden, dass die jeweils unterstützen Browser in der Lage sind diesen zu verarbeiten. Es ist die Aufgabe des Transpilers, den Code der Anwendung nach der Entwicklung so zu übersetzten, dass die Umgebung, in der er ausgeführt werden soll, ihn interpretieren kann. [Sen16]

Als Beispiel für einen solchen Transpiler lässt sich das Werkzeug Babel anführen, das insbesondere auf die Kompilierung von ECMAScript 2015 JavaScript zu ECMAScript Version 5 JavaScript spezialisiert wurde und inzwischen als Standardwerkzeug gilt. [bab]

[10] https://www.typescriptlang.org/
[11] http://coffeescript.org/

2.3.4. Linter

Als Linter wird ein Programm bezeichnet, dass der statischen Quellcode-Analyse dient. Für die Analyse können projektspezifische Regeln definiert werden, die Syntax und Struktur einzelner Code-Bausteine vorgeben. So kann mithilfe eines Linters auch bei großen Entwicklerteams ein einheitlicher Code-Stil erreicht werden, um die Lesbarkeit zu erhöhen und so die Überprüfung des Quellcodes durch andere Teammitglieder zu vereinfachen. Des Weiteren dienen Linter bereits als Vorüberprüfung des Codes, indem Entwickler zum Beispiel auf Syntaxfehler, falsche Namensgebung oder ungenutzte Variablen hingewiesen werden. Somit können viele Fehler bereits vor der Ausführung der Anwendung festgestellt und beseitigt werden.

Ein Beispiel für einen JavaScript-Linter ist das Open Source Projekt ESLint[12], welches ursprünglich von Nicholas C. Zakas entwickelt wurde. ESLint bringt bereits definierte Regeln mit, die nach eigenen Wünschen erweitert, verändert oder entfernt werden können. [ESL] Ein weit verbreitet eingesetzter Basis-Regelsatz für ESLint ist beispielsweise der Open Source Airbnb JavaScript Style Guide[13], welcher je nach den Bedürfnissen der Entwicklerteams und projektspezifischen Anforderungen angepasst werden kann. [T.17]

2.3.5. CSS-Präprozessoren

CSS-Präprozessoren erweitern die Stilsprache CSS[14] um Eigenschaften, die ihr sonst nicht zur Verfügung stehen und die Produktivität von Entwicklerteams erhöhen können, da der Code einfacher zu warten und einfacher wieder zu verwenden wird. Mit einem Präprozessor definierte Stil-Definitionen müssen nach der Entwicklung in natives CSS übersetzt werden, um von Webbrowsern gelesen und angewendet werden zu können.

Ein Beispiel für einen CSS-Präprozessor ist Sass[15]. Sass unterstützt Features wie das definieren von wiederverwendbaren Variablen, Loops, If-Else-Statements, Mathematische Operationen, Funktionen zur Farbmanipulation oder Verschachteln von Stil-Definitionen. Außerdem ist es möglich beliebig viele Sass-Dateien in andere Sass-Dateien zu importieren. Alternativen zu Sass, wie Less oder Stylus haben viele Überschneidungen untereinander und bieten die selben oder ähnliche Features an. Mit der Verwendung eines CSS-Präprozessors lässt sich, die Qualität des Codes in Hinblick auf Wartbarkeit und Wiederverwendbarkeit also deutlich erhöhen. [Cin14] Abbildung 2 zeigt exemplarisch die Syntax der CSS-Erweiterungen Sass, Less und Stylus vor der Umwandlung sowie die entsprechende CSS-Datei nach der Kompilation.

2.3.6. Task Runner und Bundler

Task Runner und Bundler haben die Aufgabe Prozesse auszuführen, die den in der Entwicklung entstandenen Quellcode nachträglich bearbeiten. Oftmals sind dies Operationen, die den Code so übersetzen, dass dieser vom Browser interpretiert werden kann. So kann zum Beispiel die Kompilierung durch einen JavaScript Transpiler und die Umwandlung einer Stylesheet-Sprache wie Sass zu nativem CSS im Task Runner angewiesen werden. Der Task Runner führt die

[12] http://eslint.org/
[13] https://github.com/airbnb/javascript
[14] CSS - Cascading Style Sheets
[15] http://sass-lang.com/

Sass

```
$link-color: #999;
$link-hover: #229ed3;

ul {
    margin: 0;

    li {
        float: left;
    }

    a {
        color: $link-color;

        &:hover {
            color: $link-hover;
        }
    }
}
```

Less

```
@link-color: #999;
@link-hover: #229ed3;

ul {
    margin: 0;

    li {
        float: left;
    }

    a {
        color: @link-color;

        &:hover {
            color: @link-hover;
        }
    }
}
```

Stylus

```
link-color = #999
link-hover = #229ed3

ul
    margin 0
    li
        float left
    a
        color link-color
        &:hover
            color link-hover
```

Css

```
ul { margin: 0; }
ul li { float: left; }
ul a { color: #999; }
ul a:hover { color: #229ed3; }
```

Abbildung 2: Syntax-Vergleich Sass, Less, Stylus und CSS [Cin14]

Operationen dann in angegebener Reihenfolge aus. Hinzukommend besteht die Möglichkeit den Quellcode zusätzlich zu optimieren. Ein Prozess kann zum Beispiel in der ersten Operation alle ECMAScript 2015 JavaScript-Dateien mithilfe eines Babel-Plugins zu ECMAScript 5 Dateien kompilieren und das Ergebnis anschließend in einer weiteren Operation minifizieren, d.h. Leerstellen entfernen und Namensdefinitionen auf ein Minimum verkürzen. [Sac15] In Listing 3 ist solch eine minifizierte Version des in Listing 2 abgebildeten Codes zu sehen.

Als Alternative oder ergänzend zu einem Task Runner kann ein Bundler eingesetzt werden. Bundler dienen in erster Linie dazu den in der Entwicklung entstandenen Code mitsamt aller Abhängigkeiten in einer Datei zu bündeln. [Gim16] Allerdings können Bundler wie Webpack[16] mittlerweile weit mehr Aufgaben übernehmen, als das Zusammenführen von Abhängigkeiten. Webpack ist hochgradig konfigurierbar und unterstützt das Einbinden externer Plugins, um auch die Aufgaben eines Task Runners übernehmen zu können. Viele Funktionen werden bereits von Webpack mitgeliefert und müssen lediglich in die Konfiguration eingebunden werden. [web] Der Umfang und die Konfigurierbarkeit bringen hingegen auch eine hohe Komplexität mit sich und im Vergleich zu Task Runnern wie Gulp[17] oder Grunt[18] ist die Konfiguration von Webpack aufwändiger und kann unter Umständen viel Zeit in Anspruch nehmen. [da117]

```
1 var a = [];
2 for (var i = 0; i < 20; i++) {
3   a[i] = i;
4 }
```

Listing 2: Standard JavaScript-Beispiel

```
1 for(var a=[],i=0;i<20;i++)a[i]=i;
```

Listing 3: Minifiziertes JavaScript-Beispiel

2.4. Frameworks und Bibliotheken zur Erstellung von Single-Page Applications

SPA-Frameworks stellen einen Entwicklungsrahmen für Softwareentwickler bei der Erstellung von Webanwendungen zur Verfügung. Je nach eingesetztem Framework werden bestimmte Entwicklungsstrukturen vorgegeben, die bei der Implementierung der Anwendung einzuhalten sind. In den folgenden Abschnitten werden zunächst Kriterien festgelegt, anhand derer im Anschluss zwei Frameworks, nämlich Angular[19] und Vue[20], sowie die Bibliothek React[21] untersucht werden.

2.4.1. Kriterien

Im Folgenden werden die eingesetzten Metriken, welche zum Teil von den Metriken die Craig McKeachie in einem Blogbeitrag zur Evaluation von SPA-Frameworks einsetzte abgeleitet

[16] https://webpack.js.org/
[17] https://gulpjs.com/
[18] https://gruntjs.com/
[19] https://angular.io/
[20] https://vuejs.org/
[21] https://facebook.github.io/react/

wurden, [McK13] erläutert. Einzelne Kriterien werden zur Bewertung in eine von fünf möglichen verbalisierten Optionen eingeordnet, die die entsprechende Metrik beschreiben. Eine Gesamtbewertung eines Frameworks unter Berücksichtigung aller Metriken kann nur stattfinden, wenn anhand der projektspezifischen Anforderungen eine Gewichtung der einzelnen Bewertungen stattgefunden hat.

Dokumentation Gibt an wie umfangreich die offizielle Dokumentation eines Frameworks und dessen Funktionen ist. Die fünf Optionen für die Bewertung gehen von sehr schlecht dokumentiert bis sehr gut dokumentiert. Eine umfangreiche Dokumentation zeichnet sich insbesondere durch Vollständigkeit und Verständlichkeit aus. Diesbezügliche Mängel wirken sich negativ auf die Bewertung aus

Performance Beschreibt die Geschwindigkeit bestimmter Operationen einer Anwendung. Als Basis wird dafür die von Stefan Krause erstellte Ergebnistabelle [Kra17], die dortige Angabe von Ausführungszeiten sowie die Bewertung in Form des Verlangsamungsfaktors im Vergleich zu nativem JavaScript dienen, den das js web frameworks benchmark[22] angibt. Das Benchmark misst die Geschwindigkeit in der Operationen ausgeführt werden, die mithilfe des jeweils zu testenden Frameworks implementiert wurden. [Kra] Kurze Reaktions- und Ausführungszeiten werden hierbei mit sehr schnell bewertet, langsame Reaktions- und Ausführungszeiten dagegen mit langsam oder sehr langsam.

Umfang Der Umfang eines Frameworks wird anhand der bereits mitgelieferten und offiziell unterstützten, erweiternden Technologien gemessen. Die Bewertung erstreckt sich hierbei von sehr umfangreich bis sehr geringer Umfang Viele mitgelieferte Lösungen zu wiederkehrenden Problemen in der Frontend-Entwicklung werden als Umfangreich angesehen.

Einstieg Gibt an wie gut der Einstieg bei der Entwicklung mit einem Framework tendenziell gelingt. Ein einfacher, schneller Einstieg und eine gute Abdeckung durch Tutorials und Lernmaterialien wirken werden hier mit sehr einfacher Einstieg bewertet, wogegen ein schwieriger, langwieriger Einstieg und wenige vorhandene Lernhilfen mit sehr schwieriger oder schwieriger Einstieg bewertet werden.

Weiterentwicklung Gibt an wie gut die Voraussetzungen für eine erfolgreiche Weiterentwicklung eines Frameworks durch die jeweilige Community sind. Bei großen Teams und vielen Unterstützern wird die Bewertung sehr gute oder gute Voraussetzungen vergeben. Ein kleines Team und eine kleine Community erhalten im Gegensatz dazu die Bewertung schlechte oder sehr schlechte Voraussetzungen.

Entwicklung nativer mobiler Applikationen Beschreibt ob und gegebenenfalls wie sich mithilfe des Frameworks native mobile Applikationen entwickeln lassen. Die Bewertung erstreckt sich dabei von sehr geeignet bis ungeeignet.

[22] http://www.stefankrause.net/js-frameworks-benchmark6/webdriver-ts-results/table.html

2.4.2. Angular

Angular[23] ist ein im September 2016 erschienenes Open-source-Framework, welches von Google als direkter Nachfolger zu AngularJS[24] entwickelt wurde. Eine Angular-Anwendung besteht aus einer Komposition aus in sich abgeschlossenen Komponenten, die jeweils das MVC-Pattern[25] implementieren. Das Framework ist in der Sprache Typescript[26] entwickelt worden, einem von Microsoft entwickelten, typisierten Superset für JavaScript, welche im Anschluss mittels Compiler wieder zu JavaScript umgewandelt werden muss. [Ran]

Dokumentation Die Dokumentation beinhaltet neben einer ausführlichen Beschreibung der API ein ausführliches Tutorial, fundamentale Konzepte und angewendete Techniken. Alle relevanten Aspekte die bei der Entwicklung mit Angular bedeutsam sind können in der offiziellen Dokumentation nachgeschlagen werden und sind mit angemessenen Beschreibungen und Beispielen aufgeführt. [Goob] Es handelt sich um eine sehr gute Dokumentation des Frameworks die keine offensichtlichen Mängel aufweist.

Performance Die Angular-Version 4.1.2 schneidet im js web frameworks benchmark mit dem Gesamt-Verlangsamungsfaktor 1,31 ab. Angular schneidet im Vergleich zu den anderen untersuchten Frameworks, Vue.js und React, am schlechtesten ab. Dabei wird das Ergebnis vor allem durch das vergleichsweise langsame Starten der Testapplikation negativ beeinflusst. [Kra17] Um die Paketgröße einer Angular-Anwendung und damit einhergehend die Ladezeit zum Starten der Anwendung zu verringern, bietet Angular Entwicklern die Möglichkeit zur AOT[27]-Kompilation. Bei der AOT-Kompilation wird die gesamte Anwendung bereits im Build-Prozess von JavaScript-Abstraktionen wie Typescript in natives, vom Browser verständliches JavaScript umgewandelt. Bei der JIT-[28]-Kompilation dagegen, welche standardmäßig verwendet wird, findet diese Kompilation erst zur Laufzeit der Anwendung statt, direkt bevor diese gerendert wird. Durch den Einsatz der AOT-Kompilation, entfällt für den Client das Laden des Angular-Compilers, der die Kompilation vornimmt und die Paketgröße reduziert sich. [Gec] Zusätzlich kann die Paketgröße durch tree shaking verkleinert werden. Beim tree shaking werden von der Anwendung nicht benötige Module und Abhängigkeiten während des Build-Prozesses nicht in die Anwendung integriert, sondern verworfen. [Nih17] Auf Basis der Testergebnisse aus dem js web frameworks benchmark und unter Berücksichtigung der Möglichkeiten zur Optimierung der initialen Ladezeit wird die Performance von Angular mit gut bewertet.

Umfang Angular ist in Module unterteilt, die im Regelfall alle Bereiche der Entwicklung einer Single-Page Application abdecken. Dazu gehören unter anderem offizielle Angular-Lösungen für das Routing, Formulare und HTTP-Anfragen. Angular benutzt außerdem ein eigenes Dependency Injection-Framework, welches beim Gebrauch von Angular benutzt werden muss. [Gooa] Die Architektur einer Angular-Anwendung, also der Aufbau von eigens entwickelten

[23] Oft auch als Angular 2 oder Angular 2+ bezeichnet
[24] Oft auch als Angular 1 oder Angular 1.x bezeichnet
[25] Model View Controller Entwurfsmuster
[26] https://www.typescriptlang.org/
[27] Ahead of Time
[28] Just In Time

Komponenten und die Integration verschiedener Module, werden auch im Allgemeinen streng von Angular vorgegeben. [Ran] Auch für das Design der Anwendung existieren mit dem Material Design[29] von Google bereits offizielle Design-Lösungen, welche für den Einsatz innerhalb von Angular-Anwendungen optimiert wurden. [Gooe] Insgesamt bietet das Angular-Framework ein sehr umfangreiches Gesamtpaket an Werkzeugen, um die Herausforderungen bei der Entwicklung moderner Single-Page Applications erfolgreich meistern zu können ohne auf zusätzliche Module und Technologien angewiesen zu sein.

Einstieg Aufgrund des sehr hohen Umfangs und einer Vielzahl an neuen Konzepten und Technologien, die beim Angular-Framework zum Einsatz kommen, gestaltet sich der Einstieg ohne entsprechendes Vorwissen als schwierig. [Ran] Als Hilfestellung existiert einerseits ein offizielles Tutorial [Gooc] und ein sehr ausführlicher Quickstart-Guide mit Beispielen und Erklärungen zu den in Angular verwendeten Konzepten [Goof] von Google selbst, andererseits aber auch diverse Tutorials und Anleitungen zum Erstellen einer Angular-Anwendung oder zum Erlernen einzelner Konzepte, die von der Community verfasst wurden. [Mar16] [Gooh] Als weitere Hilfestellung sowohl für Einsteiger als auch erfahrene Nutzer, kann das Angular CLI eingesetzt werden, um einen neuen, bereits funktionierenden Angular-Anwendungsrahmen zu generieren. Zudem können einzelne Teile wie Komponenten und Services mithilfe des Tools erstellt werden, dessen Funktionen dann im Anschluss konkret definiert werden. [Good] Zusammengefasst bietet das Angular-Framework trotz ausführlicher Anleitungen, einer Vielzahl an Tutorials und Hilfestellungen wie dem Angular CLI Tool, sowie aufgrund des sehr großen Umfangs und der Menge an zu erlernenden Technologien, einen zeitaufwändigen und schwierigen Einstieg.

Weiterentwicklung Für die Weiterentwicklung von Angular bestehen sehr gute Voraussetzungen. Mehr als 27.000 Github Stars lassen auf eine große Community und hohe Verbreitung unter Entwicklern schließen, die das Framework mit weiterentwickeln und zur Webentwicklung einsetzen. [Ang] Ferner steht Google hinter dem Angular-Framework und entwickelt es kontinuierlich auch für die eigenen Zwecke weiter, sodass etwa alle sechs Monate neue Hauptversionen veröffentlicht werden. [Goog] Aufgrund des Supports von Google und einer großen Community bestehen sehr gute Voraussetzungen für eine langanhaltende Weiterentwicklung und die Einstellung des Projekts ist derzeit sehr unwahrscheinlich.

Entwicklung nativer mobiler Applikationen Mit Ionic[30] existiert ein Open-Source-Framework zur Entwicklung nativer mobiler Applikationen [Ran] für iOS, Android und Windows Phone. Ionic wurde 2013 entwickelt und basiert auf Apache Cordova[31] und Angular. [Neu17] Aufgrund der engen Verzahnung zwischen Angular und Ionic sind bereits vorhandene Angular Kenntnisse bei der Entwicklung einer Ionic-Applikation sehr vorteilhaft. [Mor17] Mit Ionic wurden bereits etwa vier Millionen Applikationen von fünf Millionen Entwicklern entwickelt. Das Framework wird nicht direkt von Google weiterentwickelt, passt sich bisher den großen Releases von

[29] https://material.io/
[30] https://ionicframework.com/
[31] https://cordova.apache.org/

Angular an und stellt eine gute Grundlage zur Entwicklung von hybriden Anwendungen dar. [Dri]

2.4.3. React

React ist eine JavaScript-Bibliothek zur Erstellung von Benutzeroberflächen und wurde 2013 von Facebook veröffentlicht. Kern der Bibliothek ist die Möglichkeit Komponenten zu entwickeln, die einen Zustand besitzen und eine Render-Funktion implementieren. Die Render-Funktion definiert die DOM[32]-Repräsentation der Komponente durch JSX[33], eine Syntaxerweiterung für JavaScript, die ähnlich zu HTML ist. Ändert sich der Zustand der Komponente, wird im Anschluss die Render-Funktion ausgeführt und die Oberfläche aktualisiert sich entsprechend der geänderten Daten. [Kö15]

Dokumentation Die Dokumentation von React vermittelt detailliertes Wissen sowohl über Grundkonzepte, als auch über fortgeschrittene Konzepte, die zum tieferen Verständnis der Bibliothek hilfreich sind. Dabei wird auch auf die Verwendung anderer Frontend-Technologien in Verbindung mit React eingegangen. Des Weiteren werden alle benutzbaren Funktionen der Bibliothek beschrieben und mit Beispielen abgebildet. Es handelt sich somit um eine sehr gute Dokumentation, die alle Aspekte von React berücksichtigt und angemessen detailliert vermittelt. [Facd]

Performance React Version 15.5.4 schneidet im js web frameworks benchmark mit dem Gesamt-Verlangsamungsfaktor 1,30 ab und ist damit minimal schneller als Angular. Auffällig ist, dass das Löschen von Tabellenzeilen mit einem Verlangsamungsfaktor von 2,3 wesentlich langsamer ist als alle anderen durchgeführten Operationen. [Kra17] Obwohl Vue.js (siehe Kapitel 2.4.4) durch seine etwas bessere Virtual DOM Implementierung besser im Test abschneidet [Youc], ist React im Vergleich zum gesamten Feld der SPA-Frameworks und SPA-Bibliotheken performant [Kra17] und wird auf Basis der Ergebnisse aus dem js web frameworks benchmark mit gut bewertet.

Umfang React verfolgt einen minimalistischen Ansatz und trifft keine Annahmen darüber, welche Technologien und Werkzeuge Entwickler für ihre Anwendungen einsetzen. [Facc] Außerdem gibt es keine Struktur der Anwendung vor, wie es Frameworks in der Regel tun. Die React-Bibliothek ermöglicht es Templates mithilfe von JSX zu definieren, welche automatisch aktualisiert werden sobald sich der Zustand einer Komponente ändert. React bezieht sich dabei nur auf einen Oberflächen-Teilbereich der Anwendungsentwicklung. Deshalb hat Facebook das Flux-Architektur-Muster entworfen, welches einen unidirektionalen Datenfluss innerhalb einer Applikation vorgibt. Es existieren verschiedene Implementierungen des Flux-Pattern, worunter sich Redux[34] durchgesetzt hat, das häufig in Verbindung mit React verwendet wird. Beim Arbeiten mit React ist es meist notwendig weitere Pakete von Drittanbietern in die Anwendung zu integrieren, die spezielle Aufgaben wie zum Beispiel die asynchrone Kommunikation mit einer API übernehmen. Einerseits kann es durch den Minimalismus von React bei

[32] Document Object Model
[33] JavaScript Syntax eXtension
[34] http://redux.js.org/

umfangreichen Projekten zu Inkompatibilitäten zwischen zahlreichen integrierten Paketen kommen, andererseits lässt sich React selbst dadurch leicht in bereits bestehende Projekte einbinden. React besitzt einen sehr geringen Funktionsumfang und ist so konzeptioniert, dass es keine Annahmen über die Anforderungen eines Projektes trifft. Je nach Projektanforderungen können so die zu verwendenden Werkzeuge neu evaluiert und die sinnvollsten Lösungen für spezielle Problemstellungen eingesetzt werden. [Ran]

Einstieg Begründet durch den geringen Umfang der React-Bibliothek, können Grundkonzepte schnell erlernt und angewendet werden. Da auch kein strikter Anwendungsrahmen vorgegeben wird, ist es nicht notwendig alle Funktionen von React auf Anhieb zu verstehen, um kleinere Anwendungen mit React zu realisieren. [Ran] Facebook bietet zudem ein umfangreiches Tutorial auf der offiziellen React Homepage an, in dem wichtige Konzepte anhand einer Beispielanwendung erläutert werden. [Fach] Beim Programmieren komplexer Anwendungen mit React unter Umständen Probleme, die nicht durch React selbst, sondern nur durch Pakete von Drittanbietern gelöst werden können. Die Verwaltung zahlreicher Abhängigkeiten und deren Kompatibilität untereinander, sowie das Erlernen eingesetzter Module stellt eine wesentliche Herausforderung bei der Webentwicklung mit React dar. [Ran] Insgesamt fällt der Einstieg in React selbst zwar leicht, allerdings ist für Entwickler notwendig sich ein breites Wissen über Pakete aus dem JavaScript- und React-Ökosystem anzueignen.

Weiterentwicklung Da mit Facebook eine große Software-Firma React weiterentwickelt und neben zahlreichen Firmen auch für eigene Zwecke verwendet, bestehen sehr gute Voraussetzungen für die Weiterentwicklung. [Facg] Ein weiteres Indiz für die hohe Beliebtheit und Verbreitung der Bibliothek sind über 74.000 Github-Stars und mehr als 1000 Unterstützer, die sich an der Weiterentwicklung beteiligen. [Face]

Entwicklung nativer mobiler Applikationen Mit dem ebenfalls von Facebook entwickelten React Native[35] besteht die Möglichkeit native mobile Applikationen zu realisieren. Die Entwicklung mit React Native funktioniert konzeptionell so wie es bei React selbst auch der Fall ist und sofern entsprechendes Vorwissen besteht müssen nur wenige spezifische Unterschiede erlernt werden. [Facb] React Native wurde bereits zur Entwicklung zahlreicher mobiler Anwendungen von einer Vielzahl von Unternehmen eingesetzt, unter anderem von Facebook selbst. [Faci]Ferner lassen über 52.000 Github-Stars auf eine große Beliebtheit unter Entwicklern schließen. [Facf] Somit eignet sich das an React angelehnte React Native bereits jetzt sehr gut zur Entwicklung mobiler Anwendungen und eine Weiterentwicklung des Projektes ist durch die große Beliebtheit und Verbreitung sehr wahrscheinlich.

2.4.4. Vue.js

Vue.js ist ein Open-Source-Framework, das im Jahr 2013 von Evan You mit dem Ziel veröffentlicht wurde, die besten Eigenschaften von AngularJS[36] und React zu vereinen. [Ran] Der Kern des Frameworks fokussiert sich vorerst ausschließlich auf die Komposition von ineinander verschachtelten Komponenten für die View-Ebene einer Anwendung. Durch die

[35] https://facebook.github.io/react-native/
[36] https://angularjs.org/

Erweiterung um offiziell unterstützter Pakete und Werkzeugen von Drittanbietern, eignet sich das Framework für die moderne Entwicklung von Single-Page Applications. [Youg]

Dokumentation Die gesamte API des Framework-Kerns ist mit ausführlichen Erklärungen und Beispielen dokumentiert. [Youb] Darüber hinaus werden Grundlagen und fortgeschrittene Konzepte zum Verständnis näher erläutert und mit Beispielen veranschaulicht. [Youd] Auch die offiziellen Pakete vue-router[37] und vuex[38] verfügen über eine umfangreiche Dokumentation, in der alle Funktionen der Werkzeuge erläutert werden. [Youh] [Youa] Insgesamt handelt es sich aufgrund der Vollständigkeit und Ausführlichkeit, sowie vorhandener Übersetzungen um eine sehr gute Dokumentation.

Performance Vue.js schneidet in seiner aktuellen Version 2.3.3 im js web frameworks benchmark mit dem Gesamt-Verlangsamungsfaktor 1,22 ab. Im Vergleich zu den anderen untersuchten Frameworks handelt es sich hierbei um das beste Gesamtergebnis. Dabei fällt auf, dass Vue.js vor allem beim Löschen von Tabellenzeilen und der Startzeit deutlich besser abschneidet als React und Angular. [Kra17] Insgesamt lässt sich die sehr gute Performance einerseits auf die Virtual DOM Implementierung zurückführen, welche leichtgewichtiger ist als die Umsetzung von Facebooks React Bibliothek und andererseits auf die sehr kleine Paketgröße von 20 Kilobyte. [Youc]

Umfang Der Kern des Frameworks ist zur Entwicklung von Benutzeroberflächen geeignet und bezieht sich hauptsächlich auf die View-Ebene der Anwendung. Allerdings werden von offizieller Seite die Pakete vue-router und vuex angeboten, welche sich problemlos in bestehende Vue-Projekte eingliedern lassen. Beide Module bieten sich zur Entwicklung von Single-Page Applications an. Vue-Router regelt dabei die Navigation, also das Routing, innerhalb der Anwendung. Vuex dagegen stellt eine Implementierung des Flux[39]-Pattern dar, welches den Datenfluss und Zustandsoperationen einer Anwendung vorgibt. Mithilfe der beiden offiziell gewarteten und unterstützten Pakete kann das Framework auch die Logik innerhalb der Anwendung übernehmen, was es Entwicklern ermöglicht, vollwertige und komplexe SPAs zu entwickeln. [Youd] Weiterhin existiert eine Vielzahl von unterstützten Projekten, die im offiziellen Repository zwar gelistet sind, für die allerdings keine Verantwortung in Bezug auf Kompatibilität, Entwicklungsstand oder Qualität übernommen wird. [vueb] Insgesamt kann ein mäßiger Umfang von Vue.js festgestellt werden. Komplexe SPAs lassen sich nur mit offiziellen Paketen realisieren, jedoch werden nicht für alle gängigen Herausforderungen bei der Entwicklung einer SPA Lösungen angeboten, sondern müssen anderweitig beschafft oder selbst entwickelt werden.

Einstieg Um effektiv mit Vue arbeiten zu können, reicht Vorwissen in HTML, CSS und Javascript aus, da andere Technologien oder Werkzeuge nicht notwendig erlernt werden müssen. Ohne die Einbindung weiterer Module ist der Umfang außerdem angemessen beschränkt und überfordert nicht mit Konzepten die für lernende Entwickler gegebenenfalls noch nicht

[37] https://github.com/vuejs/vue-router
[38] https://github.com/vuejs/vuex
[39] https://github.com/facebook/flux/tree/master/examples

nachvollziehbar wären. Die Dokumentation beinhaltet alle Konzepte und Funktionen die vorerst zur Benutzung des Frameworks nötig sind. [Youf] Zudem existieren einige Live-Beispiele, die den Einsatz von Vue.js verdeutlichen und ein Gefühl für das Framework vermitteln. [Youe] Ferner existiert eine vom Vue.js-Team gewartete Liste mit einer Vielzahl an Tutorials und Beispielen, die von der Vue-Community erstellt wurden. [vueb] Des Weiteren kann das vue-cli[40] Tool zum Generieren von bereits lauffähigen Projekten benutzt werden. Die Projektstrukturen und der Umfang von mitgelieferten Entwicklerwerkzeugen richtet sich dabei nach dem angegebenen Entwicklungsrahmen, der vom Benutzer bestimmt wird. [vuec] Insgesamt gestaltet sich der Einstieg als sehr einfach, da das Framework sich auf wenige zentrale Aspekte der Entwicklung beschränkt und darüber hinaus mit einer großen Zahl an Tutorials und Beispielen versehen ist.

Weiterentwicklung Die Voraussetzungen für die Weiterentwicklung von Vue.js sind sehr gut. Entwicklerteams großer Firmen wie zum Beispiel Alibaba[41], Tencent[42] oder Baidu[43] setzen das Framework bereits für die Umsetzung eigener Projekte ein. [Cro] Alibaba entwickelt zudem bereits mit Weex[44] ein Framework zur Entwicklung nativer mobiler Applikationen, das vollständig auf Vue.js basiert. Darüber hinaus existiert eine Patreon[45] Kampagne zur Finanzierung des Open-Source-Projektes durch freiwillige Spenden. [Pat] Ein weiteres Indiz für sehr gute Weiterentwicklungsmöglichkeiten sind die derzeit mehr als 64.000 Github-Stars. Eine Einstellung des Projekts ist sehr unwahrscheinlich, da nicht nur eine große Community besteht, die an der Weiterentwicklung beteiligt ist, sondern teilweise auch Unternehmen, die das Framework selbst einsetzen, für finanzielle Unterstützung sorgen.

Entwicklung nativer mobiler Applikationen Die chinesische Alibaba Gruppe und Apache entwickeln mit Weex derzeit ein auf Vue.js basierendes Framework zur Erstellung nativer mobiler Applikationen. Bis zu diesem Zeitpunkt wurde es jedoch noch nicht veröffentlicht und eignet sich derzeit noch nicht für den Einsatz in komplexen Projekten. Die Projektstruktur und Syntax innerhalb von Weex basieren sehr stark auf der von Vue.js und erlauben es Webentwicklern mit entsprechendem Vorwissen sich schnell zurecht zu finden und mit JavaScript und Vue.js native mobile Applikationen zu entwickeln. [Tur17] Aufgrund des momentanen Entwicklungsstandes von Weex kann dem Framework zu diesem Zeitpunkt nur eine mäßige Eignung zur Entwicklung von mobilen Anwendungen zugeschrieben werden.

2.4.5. Fazit

In Tabelle 1 werden die Bewertungen der einzelnen Kriterien für Vue, Angular und React zusammengefasst dargestellt. Es ist ersichtlich, dass sowohl die Dokumentation, als auch Voraussetzungen zur Weiterentwicklung bei allen Frameworks mit sehr gut bewertet wurde. Auch die Performance der Frameworks liegen nahe beieinander. Allerdings hat nur Vue durch eine bessere Virtual-Dom Implementierung als React und eine kleinere Paketgröße als Angular

[40] https://github.com/vuejs/vue-cli
[41] http://www.alibabagroup.com/en/global/home
[42] https://www.tencent.com/en-us/
[43] http://ir.baidu.com
[44] https://github.com/alibaba/weex
[45] https://www.patreon.com/

ein letztlich sehr gutes Ergebnis erzielt. Sehr große Unterschiede gibt es hingegen beim Umfang und Einstieg. Während Angular für die meisten Probleme und Herausforderungen bei der Entwicklung von Single-Page Applications bereits integrierte Lösungen bereitstellt und einen festen Entwicklungsrahmen vorgibt, setzt React auf ein minimalistisches Konzept und lässt Entwicklern alle Freiheiten bei der Wahl von Werkzeugen, Technologien und Architektur. Vue ist dazwischen einzuordnen. Es bietet mit vue-router und vuex offiziell unterstützte und gewartete Werkzeuge,die für die Entwicklung von Single-Page Applications verwendet werden können, trifft ansonsten allerdings keine Annahmen über die zu verwendenden Tools und ermöglicht es Module von Drittanbietern in das Projekt zu integrieren. Außerdem gibt es einen Entwicklungsrahmen vor, der die Struktur der Anwendung mitbestimmt. Durch den ausgewogenen Umfang den Vue seinen Benutzern anbietet und die Möglichkeit eine SPA nur mit der Hilfe vom Framework selbst und offiziellen Werkzeugen in einem vorgegebenem Rahmen zu entwickeln, ist der Einstieg im Vergleich zu Angular und React sehr einfach. Durch die Möglichkeit Module von Drittanbietern in Vue zu importieren, werden Entwickler allerdings deshalb nicht in ihren Möglichkeiten eingeschränkt, sondern können durch die Kombination aus Vue und anderen Werkzeugen komplexe Applikationen entwickeln. Angular ist sehr umfangreich und es müssen zunächst die vielen mitgelieferten Module und Technologien wie Typescript und die damit verbundene komplexe Struktur einer Angular-Anwendung verstanden werden, bevor das Framework effektiv eingesetzt werden kann. Auf lange Sicht gesehen hat man anschließend mit Angular jedoch ein Werkzeug, mit dem man ohne die Zuhilfenahme von Drittanbieter-Paketen komplexe Anwendungen entwickeln kann. Bei der minimalistischen Bibliothek React fällt der Einstieg zwar deutlich leichter als bei Angular, doch durch die Notwendigkeit der Integration von Drittanbieter-Werkzeugen und einem fehlenden Entwicklungsrahmen, ist der Einstieg in React immer noch wesentlich schwerer als es bei Vue der Fall ist. Der fehlende Entwicklungsrahmen erfordert unter Umständen zudem ein breiteres Wissen über Software-Architektur und natives JavaScript. Die Entwicklung von nativen mobilen Applikationen ist zwar mit jedem der drei Frameworks möglich, dennoch sticht React hier positiv heraus und bietet mit React Native die derzeit beste Lösung. React Native ist bereits weit verbreitet, ausgereift und wird vom React-Entwickler Facebook selbst weiterentwickelt. Für Angular-Entwickler existiert mit Ionic zwar eine geeignete Lösung, allerdings setzt Ionic nur auf Angular auf und wird nicht von Google selbst weiterentwickelt. Basierend auf Vue wird derzeit das Weex Framework entwickelt, welches großes Potenzial hat, im derzeitigen Entwicklungsstadium allerdings noch schlechter zur Entwicklung von mobilen Applikationen geeignet ist, als die bereits erprobten Alternativen React native oder Ionic.

2.5. Entwurfsmuster

2.5.1. Dependency Injection

Als Dependency Injection (DI) wird ein Entwurfsmuster bezeichnet, welches Abhängigkeiten von Klassen und Komponenten zur Laufzeit reglementiert. [Mei14] Das Muster ist auch unter dem Namen Inversion of controll bekannt, da es die Kontrolle der Instanziierung von Objekten an eine zentrale Stelle, den Injector, abgibt. In Listing 4 instanziiert die Funktion findUser selbst eine Datenbank, um anschließend einen Benutzer darin zu finden. Das Datenbank-Object ist dabei nicht austauschbar ohne den Code der Funktion direkt zu manipulieren. Listing 5 zeigt

	Vue	**Angular**	**React**
Dokumentation	sehr gut	sehr gut	sehr gut
Performance	sehr schnell	schnell	schnell
Umfang	ausgewogen	sehr umfangreich	sehr gering
Einstieg	sehr einfach	schwierig	mittelschwer
Weiterentwicklung	sehr gut	sehr gut	sehr gut
Native Mobile-Apps	mäßig geeignet	geeignet	sehr geeignet

Tabelle 1: SPA-Frameworks Vergleich

eine Funktion, der die Kontrolle über die Instanziierung des Datenbank-Objektes entzogen wurde. Diese Aufgabe übernimmt der Injector. Voraussetzung für die findUser Funktion in Listing 5 ist nur, dass das Datenbank-Objekt eine Methode find implementiert hat, was zum Beispiel mithilfe eines Interfaces das von Datenbank-Objekt implementiert wird erreicht werden kann. [Gmb]

Abbildung 3 zeigt ein abstraktes UML-Diagramm, welches das Zusammenspiel von Klassen und Interfaces beim Dependency Injuection Muster beschreibt. Die Klasse Client kennt ausschließlich die Interfaces ServiceA und ServiceB, welche jeweils von den Klassen ServiceA1 und ServiceB1 implementiert werden. Die Instanziierung der konkreten Services wird vom Injector übernommen, der die Objektinstanzen anschließend an den Client übergibt.

Durch den Einsatz des DI-Musters wird eine Entkopplung der einzelnen Code-Teile untereinander erreicht, was die Modularität, Wiederverwendbarkeit, Wartbarkeit und Testbarkeit des Codes erhöht. [Gmb]

```
1 function findUser(user){
2   var db = new Database(...);
3   db.find('users', user);
4 }
```

Listing 4: Klassische Funktion ohne DI [Gmb]

```
1 function findUser(db){
2   db.find('users', user);
3 }
```

Listing 5: Funktion mit DI [Gmb]

2.5.2. Flux

Flux ist ein von Facebook entwickeltes Architektur-Muster zum Entwickeln von clientseitigen Anwendungen. Es gibt einen einseitigen Datenfluss innerhalb einer Anwendung vor und steuert so dessen Zustand. Flux besteht aus den vier in Abbildung 4 dargestellten Grundkomponenten ActionCreator, Dispatcher, Store und View, die in einer festgelegten Reihenfolge miteinander kommunizieren. [Faca] Durch eine Benutzerinteraktion mit der View wird ein ActionCreator

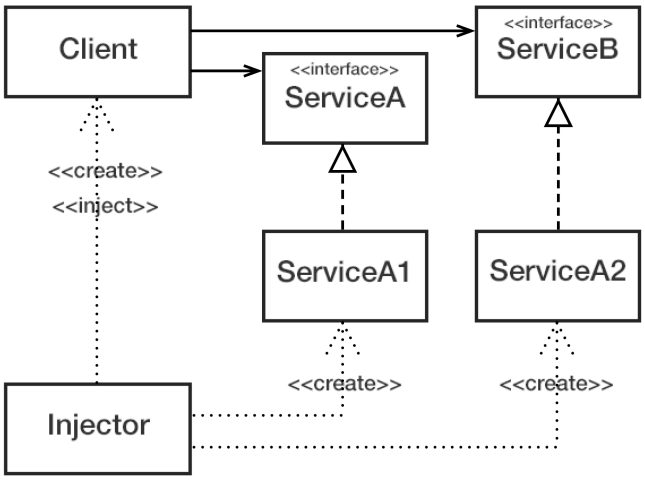

Abbildung 3: Dependency Injection UML-Diagramm [w3s]

aufgerufen. Ein ActionCreator ist eine Funktion, die die geänderten Daten aus der View gegebenenfalls bearbeitet und anschließend an den Dispatcher weiterleitet. Der Dispatcher veranlasst dann eine entsprechende Änderung der Daten im Store, die den Anwendungszustand darstellen. Zuletzt wird die View vom Store darüber benachrichtigt, dass eine Zustandsänderung stattgefunden hat und die View rendert sich auf Basis der aktualisierten Daten neu, um diese darzustellen. [Deu]

Flux wurde entwickelt, um Probleme von bidirektionaler Kommunikation beim Datenaustausch zu vermeiden, die vor allem bei großen, komplexen Anwendungen entstehen können. Bidirektionale Kommunikation bedeutet in diesem Kontext, dass eine Oberfläche auf Änderungen des Zustandes reagiert, sowie der Zustand auf Änderungen der Oberfläche reagiert. Geprüft wird dabei lediglich in welcher der Komponenten Daten manipuliert wurden, worauf die andere Komponente darüber informiert wird und sich synchronisiert. Beim MVC-Muster, wie es in der Webentwicklung eingesetzt wird, können zum Beispiel Model und View über den Controller gegenseitig miteinander kommunizieren und Daten austauschen. Anders als beim Flux-Muster findet eine beidseitige Kommunikation statt. Bei komplexen Applikationen kann dieses Verhalten dazu führen, dass Datenmanipulationen der Anwendung schwer verfolgbar sind. Beispielsweise ruft der Benutzer durch eine Aktion auf der View Änderungen am Model, beziehungsweise des Zustands der Anwendung hervor. Aufgrund der Zustandsänderung, wird eine andere View darüber benachrichtigt seine Daten demnach anzupassen. Darauf benachrichtigt wiederum diese View ein anderes Model darüber, dass sich Daten geändert haben und der Zustand wird dementsprechend aktualisiert. Mit steigender Länge einer solchen Kettenreaktion, können die Ursachen und Auslöser von Änderungen des Models und der View schwerer nachvollziehbar sein. Fehler in beteiligten Komponenten sind somit auch schwerer zu identifizieren und folglich auch zu lösen. Tendenziell sinkt die Wartbarkeit der Anwendung mit steigender Komplexität, sodass eine solche Anwendung letztlich also schlecht skalierbar ist. Bei der Flux-Architektur dagegen, werden alle direkten Zustandsmanipulationen von der Dispatcher-Komponente gesteuert. Ausschließlich der Dispatcher kann Änderungen am Zustand der Anwendung veranlassen. Sollten auf Basis der vorigen Veränderung nun weitere

Zustands-Updates nötig sein, so beginnt der Kreislauf von Neuem und auch darauf aufbauende direkte Zustandsmanipulationen werden nur vom Dispatcher veranlasst. Hierdurch besteht für Entwickler die Möglichkeit den Datenfluss besser zu verfolgen und Fehler darin leichter zu identifizieren. Abbildung 5 stellt ein komplexeres Modell der Flux-Architektur dar, um den tatsächlichen Einsatz in größeren Anwendungen zu veranschaulichen. Folgend werden die Komponenten und deren Rolle innerhalb der Flux-Architektur im Einzelnen erläutert. [Sal] [Sha]

View Die View beinhaltet allgemein zunächst ein Template, in dem festgelegt ist, welche Daten wann und wie dargestellt werden sollen. Die darzustellenden Daten erhält die View vom Anwendungszustand, der sich im Store befindet. Eine View kann sich bei einem Store anmelden, um über Zustandsänderungen informiert zu werden, worauf sie sich zur Darstellung der aktualisierten Daten gegebenenfalls neu rendert.

Eine View hat außerdem zugriff auf ActionCreators, die sie aufrufen kann, wenn Benutzer Aktionen auf der Anwendungs-Oberfläche ausführen, die Auswirkungen auf den Anwendungszustand haben könnten. [Faca]

ActionCreator und Action Der ActionCreator ist eine Funktion die eine oder mehrere Actions erzeugt und diese anschließend an den Dispatcher weiterleitet. Eine Action ist ein JavaScript-Objekt, dass mindestens ein Attribut besitzt, mit dem sich die Action später eindeutig identifizieren lässt. Der Name dieses Attributs ist bei allen Actions gleich und wird oft als type oder actionType bezeichnet. Alle weiteren Attribute einer Action enthalten die weiterzuleitenden Daten. Listing 6 stellt ein Beispiel eines einfachen ActionCreators dar. Der ActionCreator setCounter erhält einen zu setzenden Zählerstand als Parameter und leitet eine Action an den Dispatcher weiter, die mithilfe des type-Attributs identifizierbar ist und zusätzlich die übergebenen Daten enthält. [Deu]

Vor der Erzeugung und Weiterleitung von Actions, kann ein ActionCreator weitere Operationen ausführen. So können relevante Daten zum Beispiel zunächst von einer Schnittstelle angefragt und anschließend manipuliert werden, bevor sie in einer Action abgelegt und darüber an den Dispatcher weitergeleitet werden.

```
1 function setCounter(count) {
2   return {
3     type: ActionTypes.SET_COUNTER,
4     count: count,
5   };
6 }
```

Listing 6: Beispiel eines ActionCreators [Deu]

Dispatcher Der Dispatcher nimmt eine vom ActionCreator erzeugte Action entgegen und leitet diese an die entsprechenden Stores, die sich bei ihm registriert haben, weiter. Dabei behandelt der Dispatcher immer nur eine Action zu einem Zeitpunkt, sodass es Entwicklern jederzeit möglich ist, den Dispatcher anzuhalten und den derzeitigen Event-Fluss einzusehen.

Die Identifizierung vom Problemen und das anschließende Debugging einer Anwendung werden somit erleichtert. [Deu]

Store Der Store verwaltet die Applikationslogik und beinhaltet den Zustand einer Anwendung. Ein Store kann sich bei einem Dispatcher registrieren, um Actions vom ihm zu empfangen. Empfängt ein Store eine Action, wird festgestellt, ob die empfangene Action für diesen Store relevant ist. Dafür wird das Attribut zur Identifizierung der Action untersucht (siehe zum Beispiel Listing 6 type: ActionTypes.SET_COUNTER). Enthält der Store Anweisungen für den Erhalt einer solchen Action, werden diese anschließend ausgeführt und die jeweiligen Daten im Zustand verändert. Zuletzt werden beim Store angemeldete Views darüber benachrichtigt, dass der Anwendungszustand sich verändert hat, worauf diese neu gerendert werden kann, um die Veränderungen auf der Oberfläche darzustellen. [Deu]

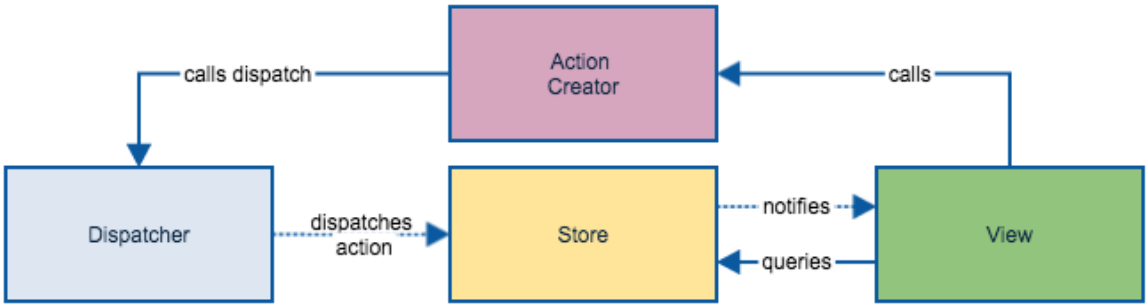

Abbildung 4: Einfache Darstellung des Flux-Konzeptes [Til]

2.5.3. Adapter Pattern

Das Adapter-Pattern ermöglicht es, dass zwei inkompatible Interfaces miteinander arbeiten können. Abbildung 6 bildet das abstrakte UML-Diagramm des Programmiermusters ab. In diesem Beispiel sind die Klassen Client und Adaptee nicht miteinander kompatibel. Um eine Kompatibilität herzustellen, wird deshalb ein Interface vom Client vorgegeben, welches vom Adapter implementiert wird. Der Adapter übersetzt das implementierte Interface in das Interface des Adaptee und ermöglicht dadurch, dass die Klassen miteinander arbeiten können.

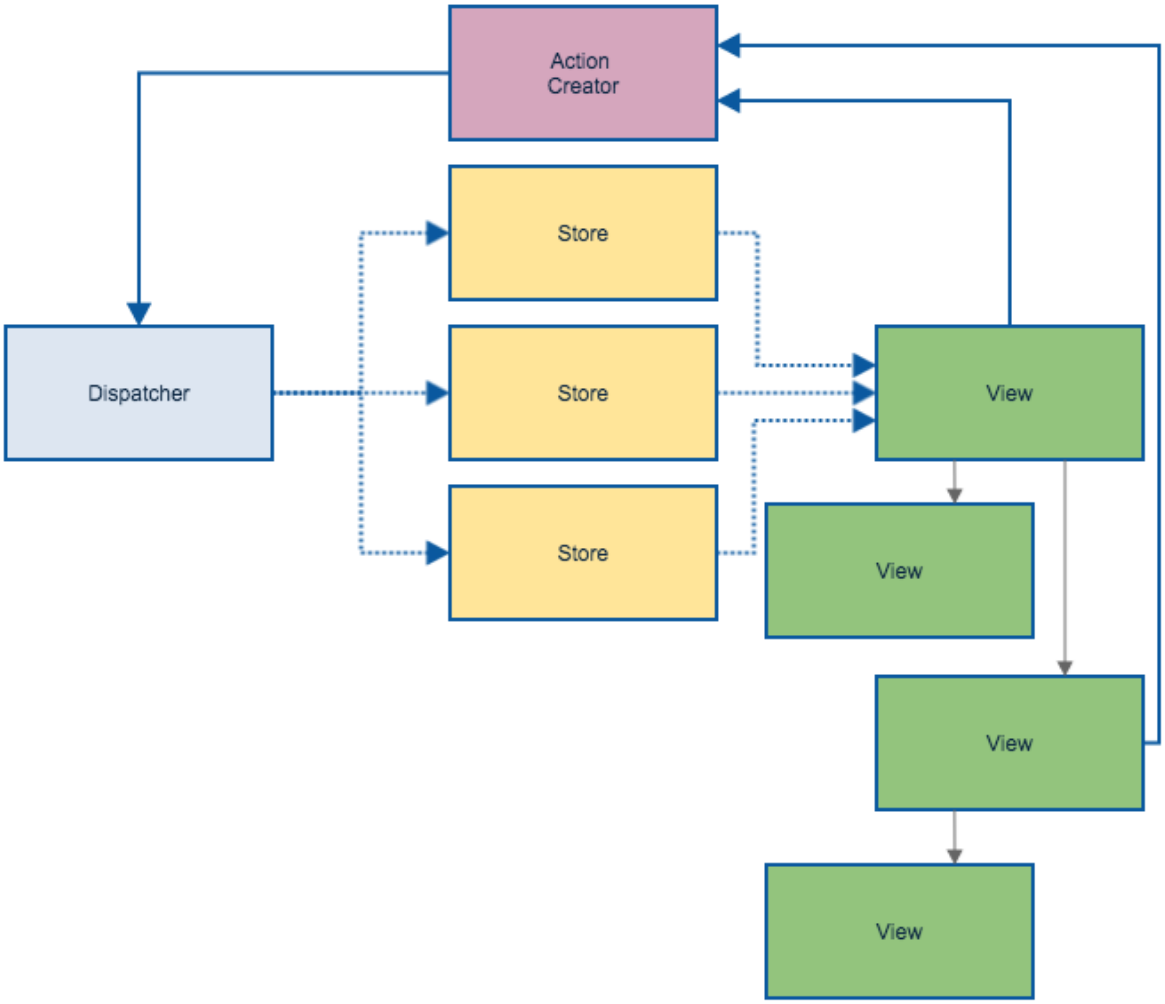

Abbildung 5: Komplexe Darstellung des Flux-Konzeptes [Til]

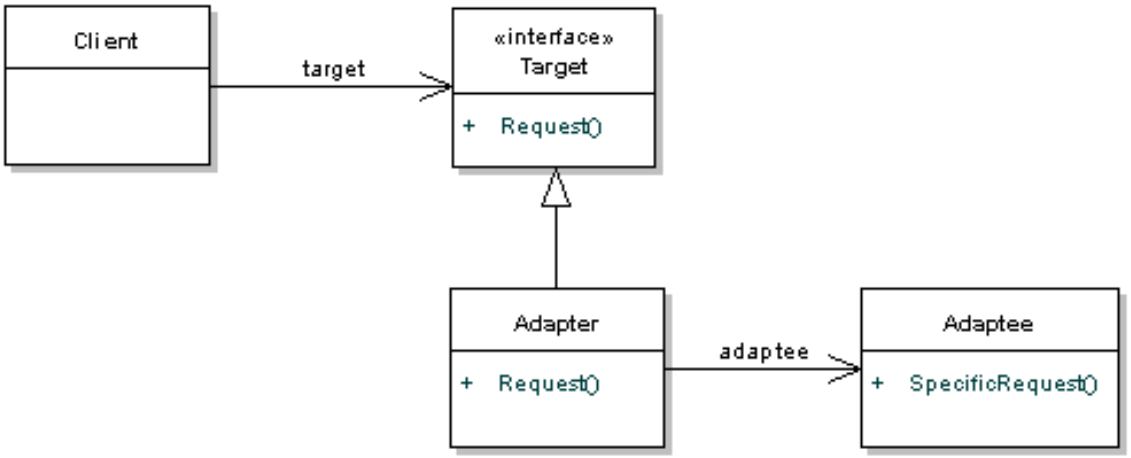

Abbildung 6: UML-Diagramm für das Adapter-Pattern [Sug]

3. Analyse

Im folgenden Kapitel werden die aus der Zielsetzung abgeleiteten Anforderungen an die zu entwickelnde Anwendung formuliert. Zunächst werden die Anforderungen der Nutzer durch Use Case-Diagramme illustriert, bevor in einem weiteren Schritt anhand dieser Diagramme die konkreten funktionalen Anforderungen formuliert werden.

3.1. Use Cases

Der Nutzungskontext der Anwendung unterscheidet sich je nach Art der Nutzer. Deshalb wurden zwei Use Case Diagramme angefertigt, die die nutzerspezifischen Anforderungen an die zu entwickelnde Anwendung verdeutlichen. Zum einen sind dies die Entwickler von Open Data Plattform-Oberflächen, die auf dieser Kern-Anwendung basieren (siehe Diagramm 8), zum anderen sind es die Online-Nutzer, dieser Oberflächen (siehe Diagramm 7). Die Use Cases der Entwickler wurden aus der Zielsetzung des Projekts abgeleitet. Die Use Cases der Online-Nutzer stellen eine Schnittmenge von Anwendungsszenarien bereits bestehender Oberflächen von Open Data Plattformen dar, die vom Fraunhofer FOKUS entwickelt wurden. Im folgenden Unterkapitel werden die Use Case-Diagramme in konkrete funktionale Anforderungen überführt.

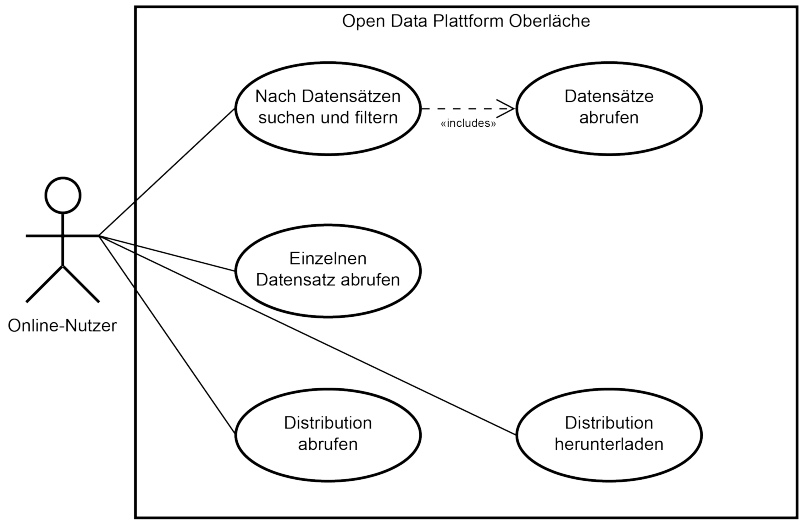

Abbildung 7: Use Case Diagramm für einen Online-Nutzer

3.2. Funktionale Anforderungen

Datensätze abrufen Die Anwendung muss in der Lage sein eine Liste von Datensätzen abzurufen und diese anhand ausgewählter Metadaten sinnvoll darzustellen.

Nach Datensätzen suchen und filtern Der Nutzer soll die Ergebnismenge abgerufener Datensätze durch eine textbasierte Suche und vorgegebene Filtermöglichkeiten nach seinen Bedürfnissen einschränken können.

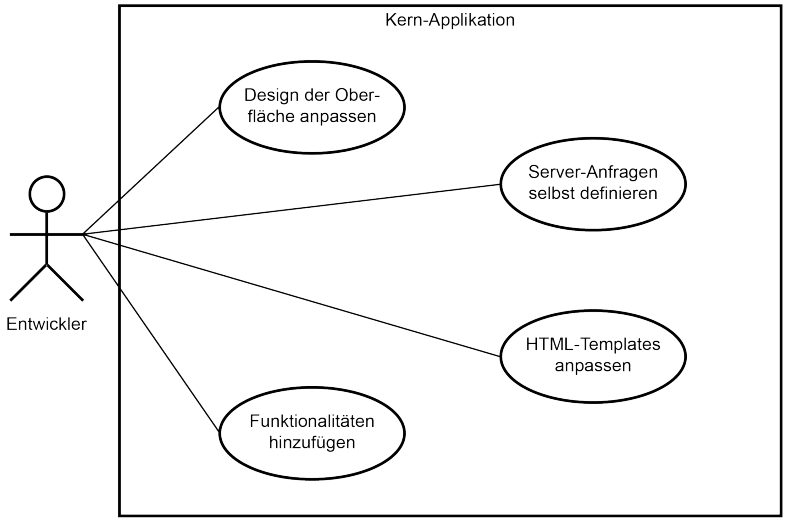

Abbildung 8: Use Case Diagramm für einen Entwickler

Einzelne Datensätze abrufen Die Oberfläche der Plattform ist in der Lage einen Datensatz abzurufen und diesen Anhand seinen Metadaten darzustellen.

Eine Distribution eines Datensatzes abrufen Es ist erforderlich, dass ein Nutzer die tatsächlichen Nutzdaten eines Datensatzes abrufen kann. Dies beinhaltet zum einen das einsehen ausgewählter Metadaten, zum anderen das Herunterladen der Daten im vorliegenden Format.

Aussehen der Oberfläche anpassen Es muss die Möglichkeit geben das Aussehen der Oberfläche teilweise zu verändern, ohne dabei die in der Kern-Anwendung definierten Stile direkt zu bearbeiten.

Unabhängigkeit vom Backend Damit die Kern-Anwendung flexibel einsetzbar ist, soll die Anwendung nicht vom jeweiligen Backend das für die Datenbeschaffung zuständig ist abhängig sein. Es muss die die Möglichkeit bestehen alle darzustellenden Daten von einem beliebigen Server abzurufen ohne die Kern-Anwendung dabei unmittelbar zu modifizieren.

Flexibilität von Elementen Aussehen sowie darzustellende Parameter zentraler Elemente der Oberfläche können individuell angepasst werden. Es besteht außerdem die Möglichkeit entsprechende Bereiche um selbst definierte Funktionen zu erweitern.

3.3. Nicht-funktionale Anforderungen

Single-Page-Webanwendung Die Navigation innerhalb der Webanwendung und das Abrufen von Daten erfordert kein Neuladen der Internetseite. Zu aktualisierende Teile der Oberfläche und Daten werden durch Ajax-Anfragen dynamisch nachgeladen ohne den Präsentationsfluss zu stören.

Modularität Die einzelnen Teile der Anwendung sollen so entwickelt werden, dass sie bei Bedarf entfernt oder ausgetauscht werden können.

Wartbarkeit Durch Unit- und Integrationstests soll sichergestellt werden, dass die Anwendung sich auch nach Anpassungen stets wie erwartet verhält.

Dokumentation Jede Möglichkeit das Verhalten der Anwendung von außen zu verändern muss ausführlich und für den Anwender verständlich dokumentiert werden.

Einfache Einarbeitung für Entwickler Bei der Auswahl der verwendeten Technologien ist stets zu beachten, dass die fachlichen Anforderungen an Entwickler sowie die Einarbeitungszeit so gering wie möglich gehalten werden.

Browser Kompatibilität Die Anwendung ist für Browser Versionen optimiert deren Marktanteil im Juli 2017 weltweit drei Prozent überstieg. [Sta17a] Dies schließt die Browser Chrome 59.0, Firefox 54.0, Internet Explorer 11.0 und Safari 10.1 ein. Zusätzlich wird der Browser Microsoft Edge in seiner aktuell stabilen Version unterstützt.

3.4. Abgrenzungen

Design der Oberfläche Das vorgegebene Design der Basis-Anwendung ist nicht Teil dieser Arbeit. Die Gestaltung der Oberfläche dient dazu die Funktionen der Anwendung beispielhaft zu veranschaulichen.

4. Entwurf

4.1. Wahl des SPA-Frameworks

Zur Auswahl des einzusetzenden Single-Page Application Frameworks werden die Ergebnisse der Framework-Untersuchung aus Kapitel 2.4 herangezogen und mit den in Kapitel 3.2 und 3.3 ermittelten Anforderungen abgeglichen. Als Grundvoraussetzungen werden außerdem eine kontinuierliche Weiterentwicklung, sowie eine hinreichende Dokumentation der Funktionen des Frameworks festgelegt.

Gemeinsamkeiten Die drei untersuchten Frameworks Vue, Angular und React eignen sich im Allgemeinen alle zur Entwicklung von Single-Page Applications, sind kompatibel mit allen von der Anwendung unterstützten Browsers (vgl. Kapitel 3.3) und verfügen sowohl über eine lückenlose Dokumentation, als auch sehr gute Voraussetzungen zur langfristigen Weiterentwicklung. Alle drei Frameworks sind zudem komponentenbasiert und dadurch gut geeignet, um den Anforderungen an die Modularität der zu entwickelnden Anwendung gerecht zu werden.

Umfang Große Unterschiede können hingegen beim Umfang festgestellt werden. Während Angular einen strikten Entwicklungsrahmen vorgibt und bereits eine Vielzahl an Werkzeugen zur Entwicklung von Single-Page Web-Anwendungen mitliefert, trifft React keine Vorannahmen über die einzusetzenden Hilfsmittel oder die konkrete Architektur bei der Anwendungsentwicklung. Beide Konzepte bringen sowohl Vor- als auch Nachteile mit sich. Mit Angular erhält man einen robusten Architekturvorschlag und qualitative, gut dokumentierte und untereinander kompatible Lösungen, sodass zusätzliche externe Abhängigkeiten leichter vermieden werden können. Mit React ist man dagegen flexibler und bekommt die Möglichkeit die Struktur und die eingesetzten Tools des Projektes selbst zu bestimmen. So können, wenn im JavaScript-Ökosystem bereits vorhanden, jeweils optimale Lösung für zu lösende Aufgaben eingebunden und eingesetzt werden. Bewusst eingebundene externe Module können meist leichter entfernt oder ersetzt werden, stellen aber auch keine Kompatibilität untereinander sicher. Außerdem nimmt die Auswahl der Werkzeuge und die spätere Verwaltung der Abhängigkeiten unter Umständen zusätzlichen Zeitaufwand in Anspruch. Das Vue Framework ist auf den Umfang bezogen zwischen Angular und React einzuordnen. Es stellt bereits essentielle Werkzeuge wie einen Router zur Verfügung, um die Entwicklung einer einfachen Single-Page Application ohne die Benutzung externer Module zu ermöglichen. Steigt die Komplexität der Anwendung, können externe Module über den Node Package Manager oder aus anderen Quellen eingebunden werden, mit deren Hilfe es möglich ist speziellere Aufgaben zu lösen. Vue gibt außerdem bereits durch Regeln für den Aufbau und die Verknüpfung seiner Komponenten einen Anwendungsrahmen vor.

Einstieg Einhergehend mit dem Umfang verändern sich auch die Bedingungen für den Einstieg in das jeweilige Framework. Vue erlaubt es Neulingen einfache Applikationen zu entwickeln, ohne zusätzlich weitere Pakete studieren zu müssen. Benutzer können so die vorgegebenen Strukturen und Funktionen des Frameworks während der Benutzung kennenlernen

und bereits einfache Anwendungen realisieren. Später können dann nacheinander externe Module einbezogen und gegebenenfalls erlernt werden, ohne dass Entwickler zunächst überfordert sind. Durch eine Vielzahl vorhandener Tutorials und teils offiziellen Anleitungen und Beispielen wird der Lernprozess weiter beschleunigt. Zwar bieten auch die Entwickler von Angular und React sehr gute Tutorials, Anleitungen und Beispiele an, jedoch führt der Umfang ihrer polarisierten Konzepte dazu, dass der Lernprozess ohne entsprechendes Vorwissen komplizierter ist. Eine nicht-funktionale Anforderung der zu entwickelnden Kern-Anwendung ist eine einfache Einarbeitung für Entwickler (siehe Kapitel 3.3). Angular kann diese Anforderung nicht erfüllen. Bevor Entwickler ohne Vorkenntnisse effizient mit Angular arbeiten können, müssen eine Vielzahl von Konzepten und Werkzeugen verstanden werden. Zusätzlich erschweren Technologien wie das von Angular eingesetzte Typescript den anfänglichen Lernprozess. Erst wenn das Basiswissen von Angular als Ganzes verstanden wurde, kann effektiv mit dem Framework gearbeitet werden. Im Gegensatz dazu lässt sich das Grundprinzip der minimalistischen Bibliothek React schneller verstehen. Der geringe Umfang führt jedoch gleichzeitig dazu, dass schnell weitere Hilfsmittel nötig werden, selbst um einfach Single-Page Applications zu realisieren. Dafür ist es wiederum notwendig sich Wissen über die Funktionsweise der benötigten externen Module anzueignen. Dinge wie Kompatibilität mit dem Rest der Anwendung, Weiterentwicklung und Dokumentation sollten vor dem Einsatz überprüft werden. Bei einer Auswahl von fast einer halben Million Pakete, die allein über den Node Package Manager zur Verfügung stehen (siehe Kapitel 2.3.1), kann der Auswahlprozess einzelner Pakete viel Zeit in Anspruch nehmen und Entwickler ohne weitreichende Vorkenntnisse schnell überfordern. Ferner müssen gewählte Abhängigkeiten selbst verwaltet und sinnvoll in die Projektstruktur eingefügt werden, da React kein Grundgerüst zur Einbindung dieser anbietet. Auch der React-Ansatz ist somit hinsichtlich der geforderten Einsteigerfreundlichkeit dieses Projektes nicht wünschenswert. Vue kann sich folglich durch seinen vergleichsweise einfachen Einstieg deutlich von den beiden anderen Optionen abgrenzen.

Performance Auch bei der Bewertung der Performance besitzt Vue den geringsten Verlangsamungsfaktor und kann mit seiner guten Implementierung des Virtual-DOM und der kleinen Paketgröße React und Angular hinter sich lassen. (siehe Kapitel 2.4) Die Unterschiede sind zwar nicht so groß, wie es beim Umfang und Einstieg der Fall war, zumal die Performance der Frameworks einer ständigen Optimierung unterliegt. Dennoch wird deutlich, dass Vue über eine bessere Performance verfügt als Angular und React und ihnen auch technologisch ebenbürtig ist.

Bezug zur Zielsetzung Damit die Anwendung flexibel eingesetzt werden kann, muss sie eine gute Erweiterbarkeit aufweisen. Im Wesentlichen begünstigt nur Angular die Erweiterbarkeit einer Anwendung dadurch, dass es auf Dependency Injection und Typescript basiert. Dependency Injection fördert die lose Kopplung innerhalb einer Anwendung, weil Abhängigkeiten von Komponenten leicht ausgetauscht werden können (siehe Kapitel 2.5.1), was der Erweiterbarkeit der Anwendung zuträglich ist. Außerdem können mit TypeScript Interfaces definiert werden, die klare Voraussetzungen und Regeln für Erweiterungen schaffen, sodass die Entwicklung von Erweiterungen erleichtert wird.

Fazit Abschließend kann festgestellt werden, dass sich alle drei Frameworks zur Erstellung von Single-Page Applications eignen und in der Lage sind die Basis zur Erfüllung der Projekt-Anforderungen zu bilden. Allein Vue kann sich mit einem sehr einfachen Einstieg deutlich von der Konkurrenz absetzen und entspricht damit am ehesten den Anforderungen der zu entwickelnden Anwendung. Zwar eignet sich in Angular durch seine begünstigte Erweiterbarkeit mit Dependency Injection und Typescript für das Vorhaben, allerdings muss an dieser Stelle auch angemerkt werden, dass Dependency Injection sich auch in React und Vue umsetzen lassen. Dafür existieren sowohl für Vue als auch für React zahlreiche Implementierungen. Des Weiteren ist der Einsatz von Typescript zwar in diesem Kontext sinnvoll, allerdings steigen dadurch auch die Anforderungen an die Entwickler der Erweiterungen und der Kern-Anwendung selbst. Auch in diesem Fall kann TypeScript mit React und Vue eingesetzt werden, falls explizit erwünscht.

Für den Einsatz von Vue spricht insbesondere die sehr gute Performance, aber auch der ausgeglichene Umfang im Gegensatz zu einem polarisierten Konzept. Aus diesem Grund wird Vue für die zu entwickelnde Kern-Anwendung verwendet werden.

4.2. Entwickler-Tools

Im Folgenden werden die Entwickler-Tools vorgestellt, die bei der Entwicklung dieses Projektes zum Einsatz kommen werden.

NodeJS und NPM NodeJS (siehe Kapitel 2.3.1) wird bei der Entwicklung hauptsächlich eingesetzt, um Pakete von Drittanbietern unkompliziert in das Projekt einbinden zu können. Es besteht außerdem die Möglichkeit, Skripte in der package.json Datei des Projektes zu definieren, die selbst definierte Prozesse durchführen.

Bulma Bulma ist ein auf Flexbox basierendes CSS-Framework. Es bietet Entwicklern ein vordefiniertes Design für zahlreiche HTML-Elemente und daraus zusammengesetzte Komponenten. Ferner stellt es ein Raster zur Verfügung mit dessen Hilfe sich die Positionierung und Anordnung der Elemente mittels CSS-Klassen unkompliziert vornehmen lässt. Es zeichnet sich besonders dadurch aus, dass alle bereitgestellten Komponenten und Elemente allein durch CSS, bzw. Sass definiert werden und keine zusätzlichen Funktionen mittels JavaScript hinzugefügt wurden. Die von Bulma vordefinierten Stile der Komponenten lassen sich dadurch sehr einfach anpassen, ohne oftmals eingebaute JavaScript Funktionalitäten dabei unerwartet zu beeinflussen. Es müssen lediglich die verwendeten Sass-Variablen des Frameworks überschrieben werden, um verwendete Design-Parameter nach individuellen Bedürfnissen anzupassen. Gleichzeitig sind alle bereitgestellten Elemente und Komponenten ausführlich und mit Beispielen dokumentiert.

Vue-Router Wie in Kapitel refsec:spa-vorteile wird zur Umsetzung einer Single-Page Application ein clientseitiger Router benötigt, um Inhalte der Webanwendung beim Aufruf definierter URLs dynamisch auszutauschen. Vue-Router[46] stellt die vom Vue-Team entwickelte, auf das Framework zugeschnittene Lösung eines solchen Routers dar.

[46] https://router.vuejs.org/de/

Vuex Vuex[47] ist eine vom Vue-Team entwickelte Implementierung des Flux-Patterns (siehe Erläuterung Kapitel 2.5.2) speziell für Vue-Applikationen. Abbildung 9 veranschaulicht den einseitigen Event- und Datenfluss innerhalb einer Vue-Applikation, die Vuex einsetzt. Von der View, hier Vue-Komponenten, aus können Actions aufgerufen werden. In Actions können zum Beispiel Daten von einer API angefragt und anschließend Operationen darauf ausgeführt werden. Anschließend werden die Daten an Mutationen übergeben, die den Zustand der Anwendung darauf verändern. Nach einer Zustandsänderung wird die entsprechende Vue-Komponente benachrichtigt, die sich gegebenenfalls neu rendert, um die Änderung nach außen darzustellen. [vueg]

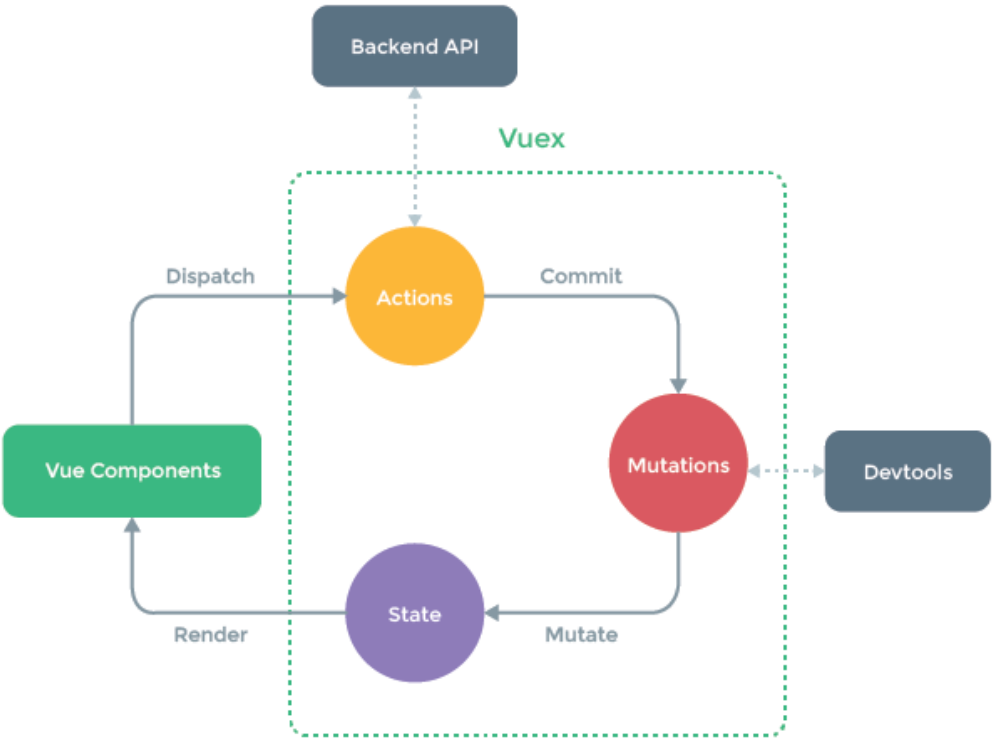

Abbildung 9: Eventfluss einer Vue-Applikation mit Vuex [vueg]

Vue-Devtools Vue-Devtools[48] ist ein Browser-Addon, dass das Debugging von Vue-Applikationen zur Laufzeit vereinfacht. Das Programm wird für die Browser Chrome, Firefox und Safari angeboten und ermöglicht die komfortable Einsicht und Manipulation einzelner Bereiche der zu testenden Web-Anwendung. So können zum Beispiel Momentaufnahmen von Komponenten und deren Unterkomponenten eingesehen und ihre Daten verändert werden. In Verbindung mit Vuex besteht außerdem die Möglichkeit, den gesamten Anwendungszustand einzusehen und anzupassen, sowie Events rückwirkend zu betrachten und Veränderungen dadurch unkompliziert nachzuvollziehen. [Bemb]

[47] https://vuex.vuejs.org/en/intro.html
[48] https://github.com/vuejs/vue-devtools

4.2.1. Vue-Cli und Vue-Webpack Bundle

Mithilfe des Vue-Cli[49]-Pakets können vordefinierte Projektstrukturen mitsamt eingebetteter und vorkonfigurierter Werkzeuge über die Kommandozeile erzeugt werden. [vued] Eine der zur Verfügung stehenden Projektkonfigurationen stellt die Vue-Webpack-Boilerplate dar. Das Bundle stellt ein Projekt-Template dar, das moderne Entwicklerwerkzeuge zur Entwicklung einer Vue-Applikation bietet. Folgend werden die enthaltenen Werkzeuge und deren Funktion beschrieben.

Webpack Das Kernstück des Projekt-Templates stellt Webpack (siehe Kapitel 2.3.6) dar. Mit Webpack werden die eingebundenen Tools konfiguriert und Build-Prozesse definiert, in denen diese Anwendung finden. Dabei ist die Konfiguration jederzeit vom Entwickler anpassbar und kann beliebig manipuliert werden. [vuef]

Vue-Loader Vue-Loader ermöglicht es Vue-Komponenten in einer Vue-Datei zu definieren. Eine Vue-Datei besteht aus drei semantisch voneinander getrennten Bereichen, die das HTML-artige Template, das Design und die Logik der Komponente beinhalten. Vue-Loader hat die Aufgabe diese Vue-Dateien zu spalten und entsprechend natives HTML, JavaScript und CSS daraus zu generieren. [vuee]

Babel Babel ist ein JavaScript-Transpiler und übersetzt den Quellcode JavaScript-basierter Programmiersprachen in natives JavaScript. Dazu gehört auch das Umwandeln von bereits nativem JavaScript, das auf neuen ECMAScript-Spezifikationen wie ECMAScript 2015 basiert, zu früheren Versionen wie ECMAScript 5. (JavaScript-Versionierung wird in Kapitel 2.3.2 näher erläutert) So bietet das Vue-Webpack Bundle mit Webpack und Babel die nötigen Tools, um modernes ES2015-JavaScript bei der Entwicklung einzusetzen und durch die spätere Übersetzung zu ES5-JavaScript dennoch keine Inkompatibilitäten mit älteren Browsern aufzuweisen, die den neueren JavaScript-Standard noch nicht implementieren können. (siehe Kapitel 2.3.3)

ESLint ESLint ist ein Programm zur statischen Quellcode-Analyse. Für die Analyse können eigene Regeln definiert werden, nach denen der Quellcode überprüft werden soll. Das Vue-Webpack Bundle bindet dafür den Regelsatz des Airbnb JavaScript Style Guides ein. Durch den Einsatz von ESlint können viele syntaktische Fehler der Anwendung bereits vor dem Start der Anwendung erkannt und behoben werden. Zudem kann mithilfe des Tools ein einheitlicher Code-Stil innerhalb des Projektes umgesetzt werden, auch wenn mehrere Entwickler an der Realisierung beteiligt sind. (siehe Kapitel 2.3.4)

UglifyJS, html-minifier und cssnano UglifyJS[50], html-minifier[51] und cssnano[52] sind Programme, die den bei der Entwicklung entstandenen Quellcode nachträglich minifizieren. Sie werden in Verbindung mit Task-Runnern oder Bundlern eingesetzt und in den darin definierten

[49] Vue-Command line interface
[50] https://github.com/mishoo/UglifyJS2
[51] https://github.com/kangax/html-minifier
[52] https://github.com/ben-eb/cssnano

Operationsketten eingebunden. Das Vue-Webpack Bundle optimiert in den vordefinierten Build-Prozessen JavaScript durch den Einsatz von UglifyJS, html mithilfe von html-minifier und CSS durch das Einbinden von cssnano. Alle drei Programme haben die Aufgabe den Quellcode zu optimieren und die Größe des entstehenden Pakets zu minimieren, wodurch die Reaktionszeit der Anwendung deutlich verbessert werden kann. (siehe Kapitel 2.3.6)

Karma und Mocha Karma und Mocha sind Programme, die zum Testen einer Anwendung durch Unit-Tests eingesetzt werden. Karma ist ein Test-Runner, der durch Entwickler definierte Unit-Tests ausführt und anschließend das Ergebnis jedes durchgeführten Tests liefert. Dafür startet Karma einen Webserver und testet die Anwendung mithilfe der Unit-Tests unter den Bedingungen der in der Konfiguration eingebundenen Browser. [Zie]

Zur Programmierung der eigentlichen Tests wird das JavaScript-Unit-Testing-Framework Mocha verwendet. Es bietet viele Funktionen, um den eigentlichen Quellcode der Anwendung durch Unit-Tests abzudecken und so auch nach weitreichenden Änderungen die Funktionstauglichkeit der Applikation zu gewährleisten. Die beiden Werkzeuge wurden im Vue-Webpack Bundle bereits konfiguriert und in das Projekt eingebunden, lassen sich jedoch zu jeder Zeit nachträglich anpassen. [Moc]

Nightwatch Nightwatch ist ein Framework zum Definieren und Durchführen von Integrations-, beziehungsweise End-to-End-Tests. Durch End-to-End-Tests soll das Verhalten eines Anwenders während der Benutzung der Anwendung simuliert werden, um festzustellen ob die Anwendung nicht vorhersehbare Reaktionen für ein bestimmtes Nutzerverhalten aufweist, die die Applikation zum Absturz bringen. Außerdem kann durch das ausführen der Tests in verschiedenen Browsern die Kompatibilität unterschiedlicher Browser-Umgebungen im Kontext eines konkreten Benutzter-Szenarios getestet werden. Das simulierte Nutzerverhalten eines Tests sowie die erwartete Reaktion der Anwendung muss innerhalb des Test-Quellcodes genau definiert werden. Weicht das tatsächliche Verhalten vom erwarteten Verhalten ab, wird eine Fehlerbeschreibung ausgegeben und die Testreihe gestoppt, damit der Fehler behoben werden kann. [Nem17] Nightwatch bietet hilfreiche Funktionen zum Definieren des Verhaltens und der erwarteten Ergebnisse des eines Tests. Es stellt zugleich den Test-Runner dar, der für die Durchführung der Tests zuständig ist. [Nig]

4.3. Anwendungsstruktur

Die zu entwickelnde Anwendung stellt den Rahmen einer Frontend-Anwendung dar, die Daten über eine Backend-Anwendung mittels einer bereitgestellten API abruft, verarbeitet und anschließend darstellt.

Die Struktur der Anwendung wird maßgeblich durch seine funktionalen Anforderungen aus Kapitel 3.2 bestimmt. Die aus dem Use Case Diagramm (siehe Abbildung 7) abgeleiteten Anforderungen eines Online-Nutzers werden in View-Komponenten übersetzt werden. Dabei basiert der Aufbau der einzelnen Oberflächen auf vergleichbaren Views bereits existierender Open Data Plattformen wie zum Beispiel dem Datenportal der Bundesrepublik Deutschland[53] oder dem europäischen Datenportal[54]. Zur Erfüllung der aus Abbildung 8 funktionalen

[53] https://www.govdata.de/web/guest/suchen
[54] https://www.europeandataportal.eu/data/en/dataset

Anforderungen eines Entwicklers hingegen werden Architektur-Konzepte dargelegt, die es ermöglichen das Verhalten und Aussehen dieser View-Komponenten zu verändern. Bei der Konzeption von Lösungen der funktionalen Anforderungen, wird auf die nicht-funktionalen Anforderungen aus Kapitel 3.3 der Anwendung zurückgegriffen.

Im Folgenden wird zunächst das Konzept zum Grundaufbau der Oberfläche dargestellt und anschließend die funktionalen Anforderungen an die Applikation und das jeweilige Konzept zur Lösung der Anforderung erläutert.

4.3.1. Grundaufbau

Das Grundgerüst der Oberfläche besteht aus einer Titelleiste, einer Seiten-Navigation und einem Inhaltsbereich (vgl. Abbildung 10). Titelleiste und Navigation verändern sich während der Benutzung der Webanwendung nicht. Im Inhaltsbereich hingegen werden die einzelnen logischen Seiten[55] der Anwendung angezeigt und die dargestellten Unterbereiche und Inhalte ausgetauscht. Für die Navigation innerhalb der unterschiedlichen logischen Seiten der Applikation wird, wie in Kapitel 2.2.1 erläutert, ein clientseitiger Router eingesetzt. Innerhalb des Routers wird dafür jeder logischen Seite eine URL und eine Vue-Komponente zugewiesen. Navigiert ein Benutzer zu einer entsprechenden URL, wird die dazugehörige Komponente dynamisch in den Inhaltsbereich geladen, ohne dass dabei ein Neuladen der Seite stattfindet.

Identifizierung der Vue-Komponenten Anhand des skizzierten Seitenaufbaus in Abbildung 10 lassen sich zwei konkrete Komponenten identifizieren. Die eine Komponente repräsentiert die Titelleiste, die andere die Hauptnavigation der Webanwendung. Dem Inhaltsbereich kann keine konkrete Komponente zugeordnet werden, allerdings wird jede logische Seite der Anwendung aus den beiden oben genannten Komponenten, sowie einer weiteren Vue-Komponente bestehen. Unter Umständen enthalten die jeweiligen Inhaltskomponenten dabei weitere Unterkomponenten.

4.3.2. Datensätze abrufen

Damit ein Benutzer sich eine Liste von Datensätzen anzeigen lassen kann, müssen zuerst die erforderlichen Daten zur Beschreibung der Datensätze mittels Ajax-Request (siehe Kapitel 2.2.1) vom Speicherort abgerufen werden. Anschließend werden die abgerufenen Metadaten gegebenenfalls aufbereitet und letztlich in Form einer Liste auf einer dafür bereitgestellten logischen Seite dargestellt. Die einzelnen Elemente der Liste stellen dabei bereits ausgewählte Informationen über den jeweiligen Datensatz zur Verfügung.

Identifizierung der Vue-Komponenten Mithilfe von Abbildung 11 lässt sich eine Vue-Komponente identifizieren, die für die Auflistung von Datensätzen, beziehungsweise ihren Metadaten zuständig ist. Die Komponente entspricht dabei einer logischen Seite der Anwendung und ist über einen Hauptnavigationspunkt (in Abbildung 11 mit Data benannt) erreichbar.

[55] Da man sich innerhalb einer SPA stets nur auf einer physischen Seite befindet, ist hier von logischen Seiten die Rede. Eine logische Seite in einer SPA entspricht einer HTML-Seite in einer klassischen Webanwendung

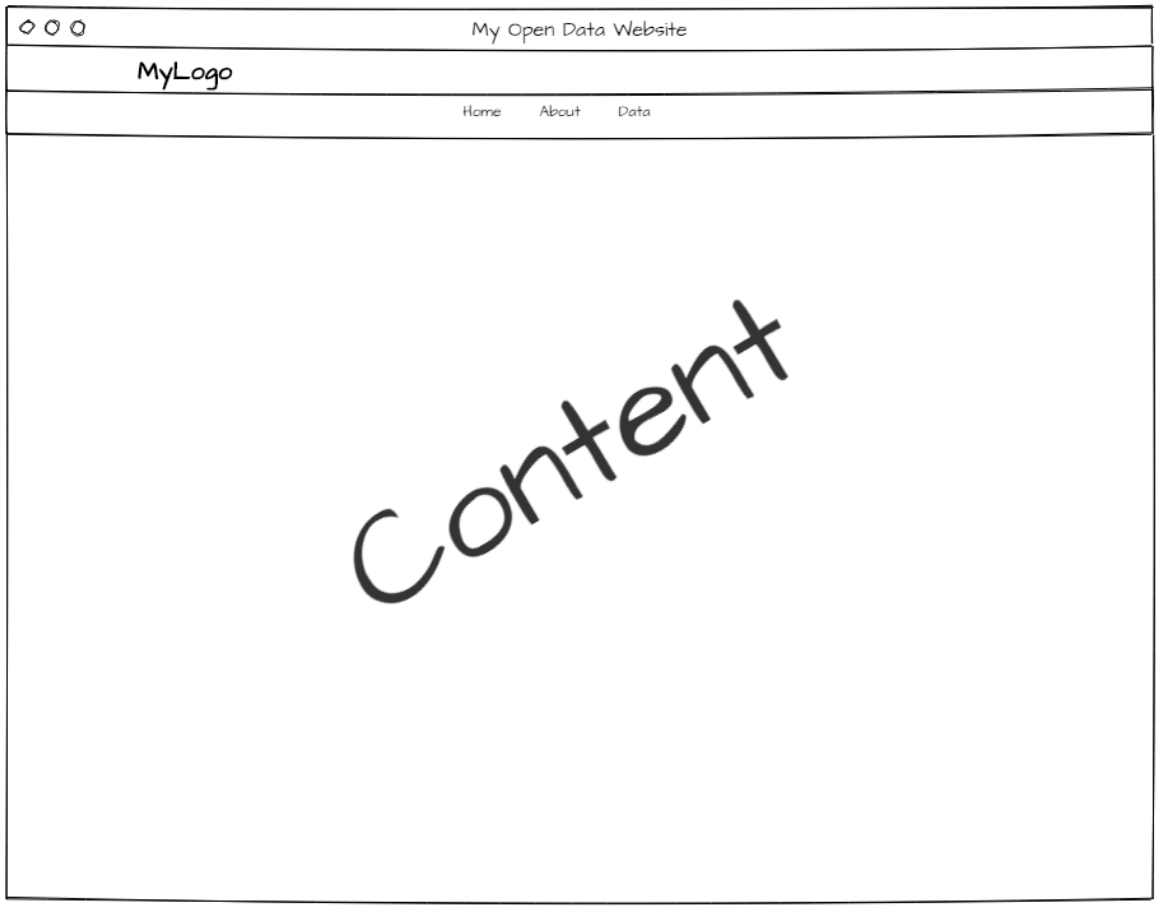

Abbildung 10: Grundaufbau der Oberfläche

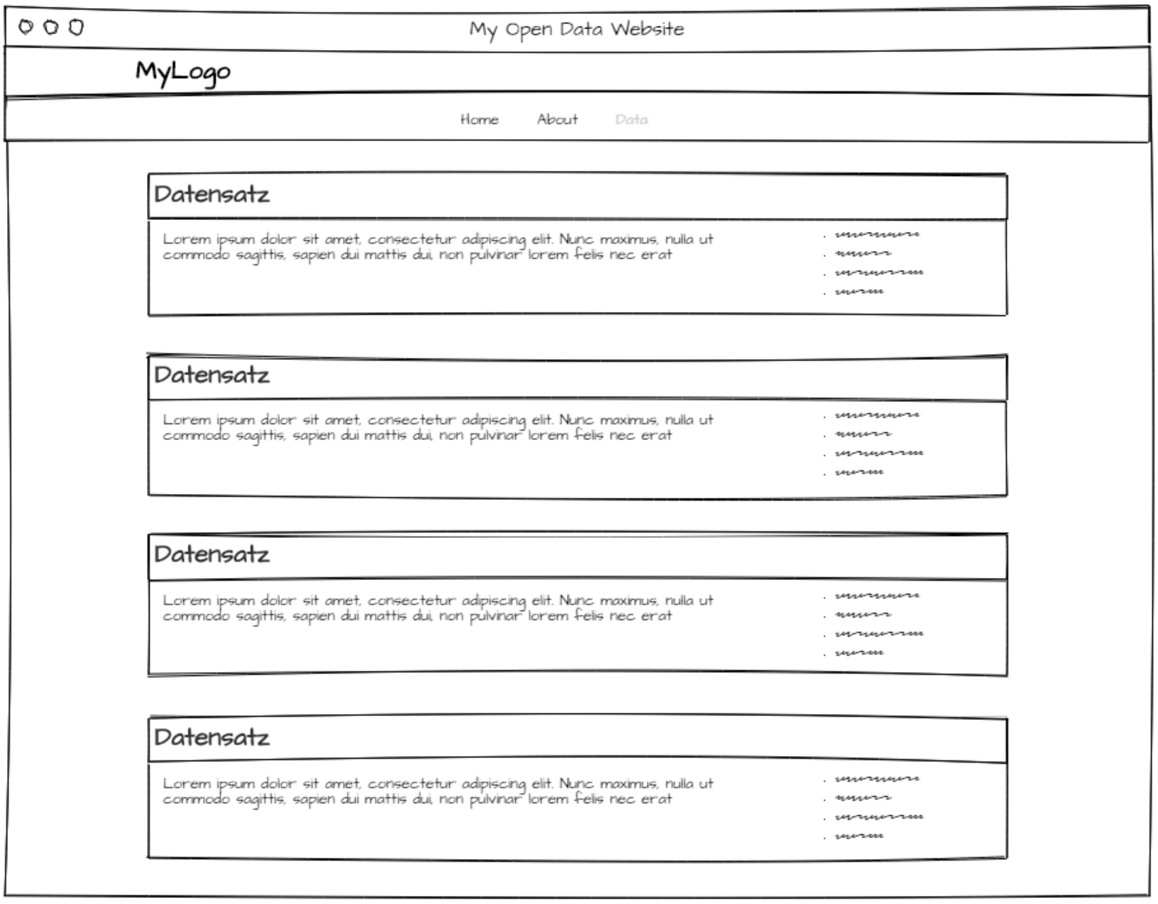

Abbildung 11: Darstellung einer Liste von Datensätzen

4.3.3. Nach Datensätzen suchen und Filtern

Um die Suchergebnisse beim Abrufen von Datensätzen gezielt einschränken zu können, soll ein Benutzer die Möglichkeit haben, vorgegebene Facetten mit einer textbasierten Suche zu kombinieren. Das in Abbildung 7 zu sehende Use Case Diagramm zeigt, dass das Filtern und Suchen nach Datensätzen, das Abrufen von Datensätzen mit einschließt. Aufgrund dieses Zusammenhangs, wird die Funktionalität für das Suchen und Filtern nach bestimmten Datensätzen auf der selben logischen Seite untergebracht (siehe Abbildung 11). Wählt ein Benutzer eine Facette aus oder gibt einen Text in das Suchfeld ein, werden mittels Ajax-Request Datensätze abgerufen, die den neuen Suchkriterien entsprechen. Außerdem werden die angewendeten Facetten und Suchbegriffe in Form von URL-Parametern an die URL angehängt. Beim Aufruf der URL mit spezifischen Suchparametern, wird die Seite bereits mit den vorkonfigurierten Suchoptionen und der dazugehörigen Ergebnisliste geladen. Dies ermöglicht zum einen das Speichern von Suchanfragen in Form von Lesezeichen und zum anderen eine unkomplizierte Weitergabe der Suchanfragen an andere Nutzer. Außerdem ist es dadurch möglich über die Vor- und Zurück-Buttons des Browsers auf der Seite zu navigieren.

Identifizierung der Vue-Komponenten Der Skizze in Abbildung 12 stellt die gleiche logische Seite wie Abbildung 11 dar und wurde lediglich um ein Suchfeld und eine Liste mit auswählbaren Facetten erweitert. Die in Kapitel 4.3.2 identifizierte Komponente zum Abrufen und Auflisten von Datensätzen, wird auf dieser Basis ebenfalls erweitert. Such- und Filterfunktionalitäten werden mit in die Komponente integriert. Dabei ist es sinnvoll die Facettenliste in Form einer eigenen Komponente umzusetzen und diese als Unterkomponente zu integrieren. So besteht gegebenenfalls die Möglichkeit, gemäß der Anforderungen an die Modularität der Anwendung, eine Facettenliste bei Bedarf unkompliziert zu entfernen oder durch eine andere Komponente zu ersetzen. Die Textsuche dagegen wird zunächst nicht in Form einer eigenen Komponente umgesetzt, sondern aufgrund der niedrigen Komplexität lediglich in die Eltern-Komponente integriert.

4.3.4. Einzelne Datensätze abrufen

Die Darstellung der Metadaten eines vom Benutzer ausgewählten Datensatzes stellt eine eigene logische Seite innerhalb der Webanwendung dar. Diese Detailseite repräsentiert den ausgewählten Datensatz und soll diesen möglichst genau beschreiben sowie die Möglichkeit bieten, Informationen der Distributionen des Datensatzes einzusehen. Benutzer können zum einen durch die Auswahl eines Datensatzes aus der in Kapitel 4.3.2 beschriebenen Liste zu dieser Detailansicht gelangen, zum anderen kann die Seite auch eindeutig über eine URL identifiziert und aufgerufen werden. Neben Informationen wie dem Erstellungsdatum, der Lizenz oder einem beschreibenden Text, sollen außerdem Angaben über die Distributionen des Datensatzes abrufbar sein. Um dies zu realisieren, müssen zunächst die darzustellenden Angaben des entsprechenden Datensatzes mittels Ajax-Request vom Backend-Dienst abgerufen werden. Die erhaltenen Daten werden anschließend gegebenenfalls bereinigt und in eine darstellbare Form gebracht. Abbildung 13 veranschaulicht in welcher Form die Metadaten dargestellt werden. Anhand getrennter Bereiche können Angaben strukturiert werden, wobei der Betreiber der

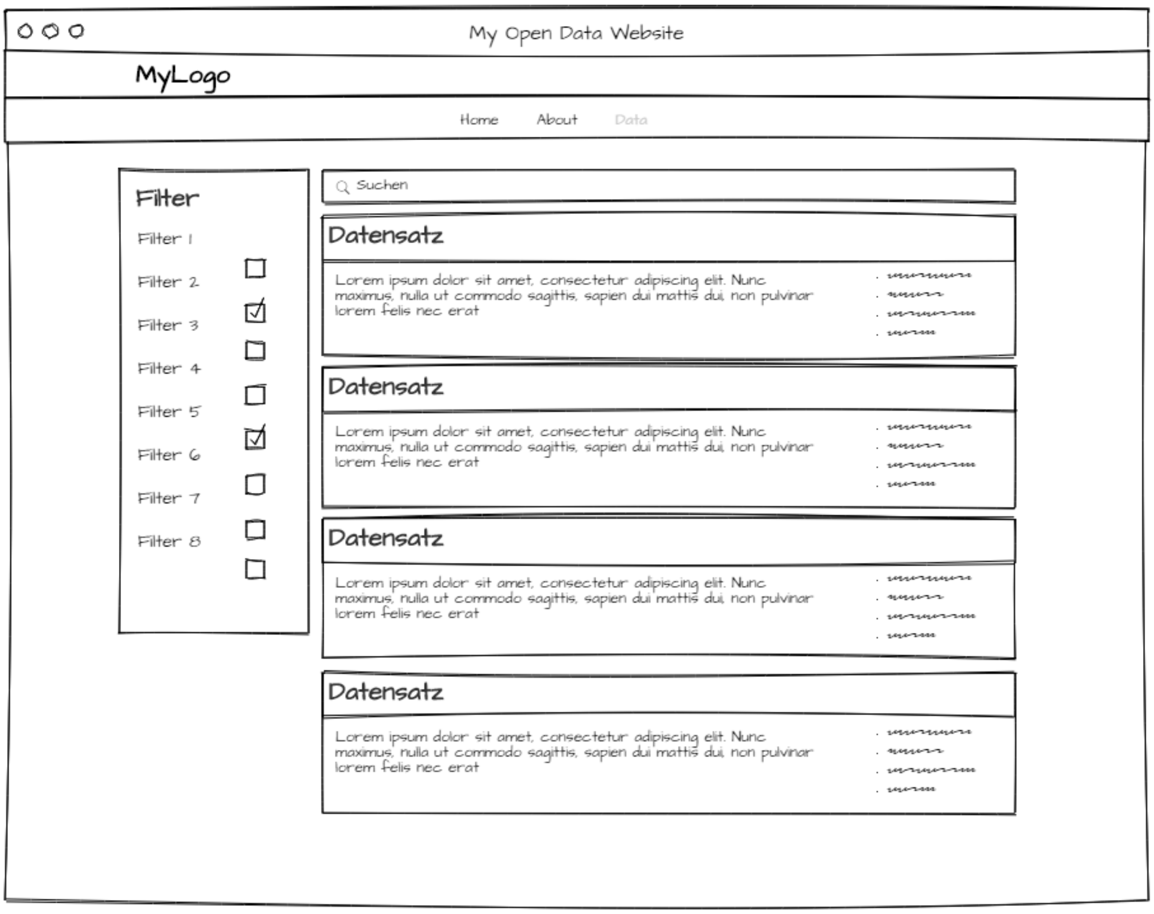

Abbildung 12: Darstellung einer Liste von Datensätzen

jeweiligen Plattform selbst darüber entscheiden kann, ob und an welcher Stelle diese dargestellt werden.

Identifikation der Vue-Komponenten Abbildung 13 zeigt den skizzierten Aufbau der Detailseite eines Datensatzes. Da es sich um eine eigene logische Seite handelt, lässt sich bereits eine Komponente identifizieren, die alle Elemente der Seite enthält. Wie im oberen Abschnitt dieses Kapitels beschrieben, dienen die in der Abbildung dargestellten Bereiche lediglich der Informationsstrukturierung und besitzen darüber hinaus keine komplexen Funktionalitäten. Aufgrund dessen ist eine weitere Unterteilung in Unterkomponenten nicht erforderlich.

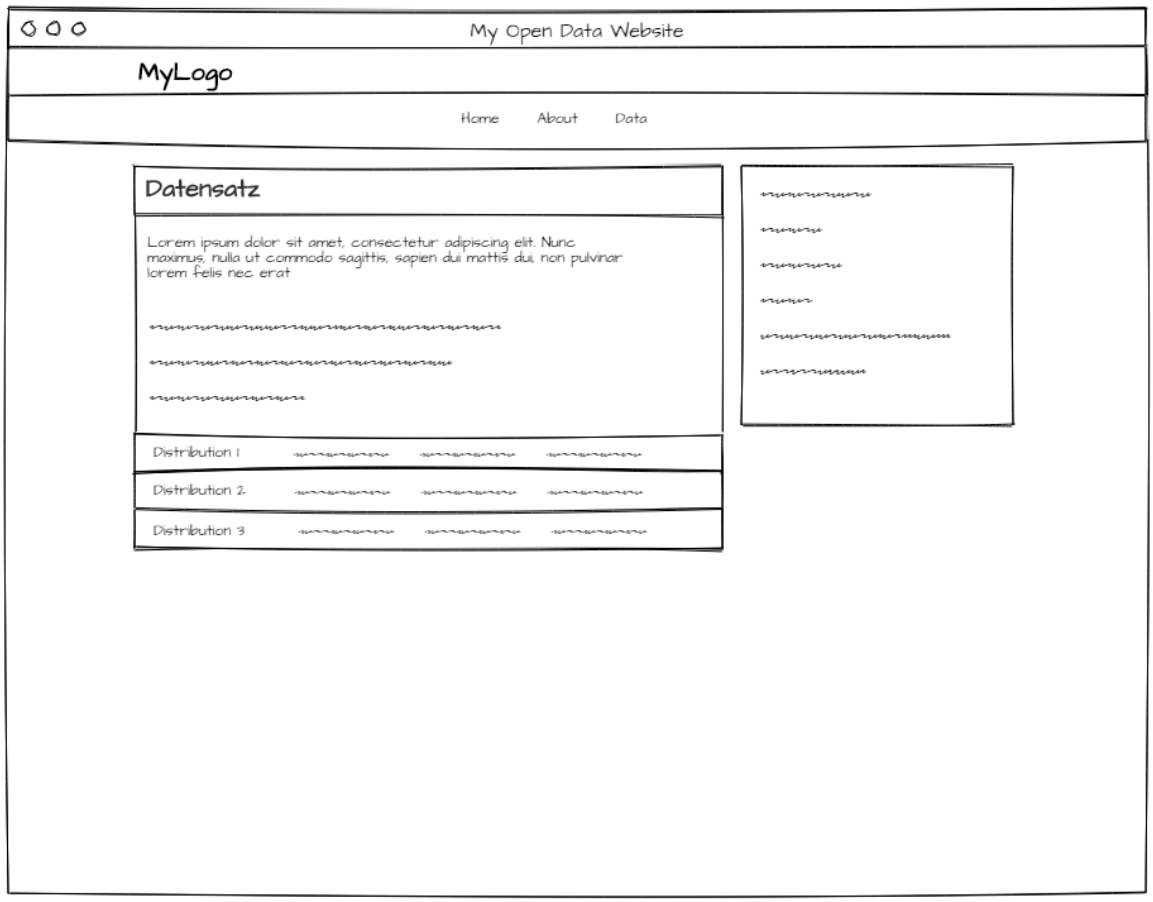

Abbildung 13: Skizze der Detailseite eines Datensatzes

4.3.5. Distribution eines Datensatzes abrufen

Ähnlich wie in Kapitel 4.3.4 beschrieben, wird auch die Anforderung zum Abrufen und darstellen der Metadaten einer Distribution umgesetzt werden. Wählt ein Benutzer in der Datensatz-Detailansicht eine Distribution aus, gelangt er zu einer weiteren logischen Seite, die diese Distribution näher beschreibt. Ferner besteht wiederum die Möglichkeit, die Seite direkt über eine URL zu erreichen. Analog zur Datensatz-Detailseite besteht auch hier die Option die Metadaten der Distribution durch getrennte Bereiche bedarfsgerecht zu strukturieren. Darüber hinaus können die tatsächlichen Daten von Distributionen, die in unterstützten Formaten vorliegen, bereits eingesehen werden.

Identifikation der Vue-Komponenten Abbildung 14 zeigt den skizzierten Aufbau der Detailseite einer Distribution. Es kann bereits eine Komponente identifiziert werden, die den Rahmen dieser eigenen logischen Seite bildet und wie in Kapitel 4.3.5 beschrieben alle Bereiche zur Darstellung der Metadaten definiert. Die Darstellung der tatsächlichen Nutzdaten der Distribution wird dagegen in einer eigenen Unterkomponente gekapselt. Hier bietet es sich an, je nach Format der Daten und Art der Darstellung, eine passende Komponente einzubinden. Wird das Datenformat der Distribution nicht unterstützt, kann diese Unterkomponente lediglich nicht eingebunden werden. Außerdem können verschiedene Darstellungsarten unkompliziert zur Laufzeit ausgetauscht werden, sofern eine Komponente existiert, die sowohl die Darstellungsart als auch das Datenformat unterstützt.

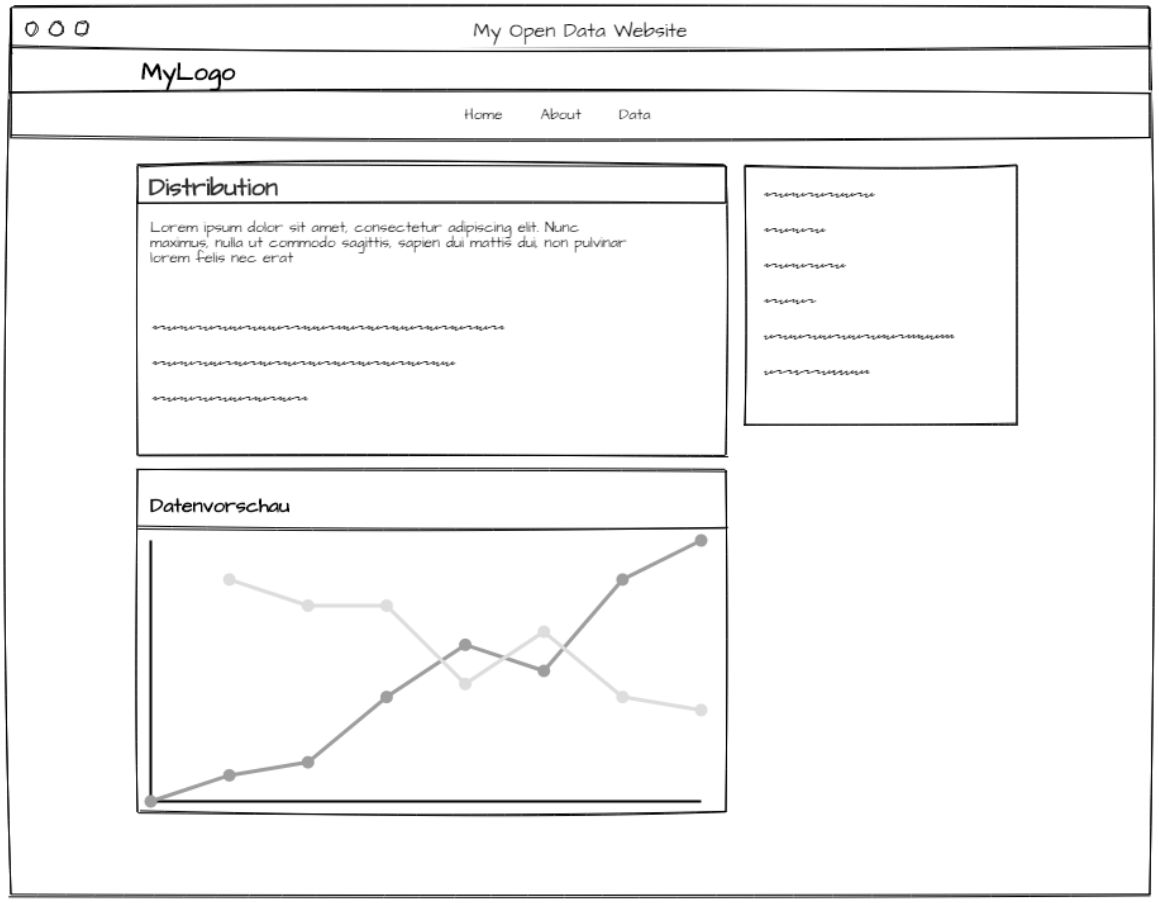

Abbildung 14: Skizze der Detailseite einer Distribution

4.3.6. Aussehen der Oberfläche anpassen

Die Grundlage für das Aussehen der Komponenten bildet wie bereits erwähnt das Design-Framework Bulma[56]. Die in Bulma definierten Standardstile können durch das Überschreiben der innerhalb des Frameworks verwendeten Sass-Variablen verändert werden. Dafür werden zunächst die in Bulma definierten Variablen in einer Sass-Datei importiert. Darauf können die Variablen in dieser Datei verwendet, beziehungsweise auch überschrieben werden. Zuletzt werden die restlichen Sass-Dateien von Bulma importiert, die die vorher zugewiesenen Stile

[56] http://bulma.io/

dann einsetzen. [Tho] Die zu entwickelnde Kern-Anwendung muss wiederum das entstandene Bundle integrieren, damit die vordefinierten und gegebenenfalls angepassten Stile des Design-Frameworks eingesetzt werden können.

Allerdings soll nicht nur die Möglichkeit bestehen die eingesetzten Stile des Bulma-Frameworks anpassen zu können, sondern auch darüber hinaus Anpassungen vorgenommen werden können, um die Gestaltung der Web-Anwendung so flexibel wie möglich zu gestalten. Um dies zu ermöglichen, werden weitere Sass-Variablen definiert, die anschließend innerhalb der Vue-Komponenten zur Zuweisung der konkreten Stile benutzt werden. Diese zusätzlichen Variablen werden in die oben erläuterte Struktur von Bulma integriert, sodass nach dem Import der Bulma-Variablen, alle in der Anwendung verwendeten Variablen mit Standardwerten erzeugt werden. Diese Standardwerte können anschließend überschrieben werden, indem nachträglich eine weitere Sass-Datei eingebunden wird, in der den zuvor definierten Variablen neue Werte zugewiesen werden.

Da das Aussehen der Oberfläche anpassbar sein muss, ohne die Kern-Anwendung dabei zu manipulieren (siehe Kapitel 3.2), ist es notwendig, dass die Sass-Dateien, die für das Überschreiben der Standardstile zuständig sind, außerhalb des Projektes abgelegt werden. Der Pfad zu den Dateien muss der Kern-Anwendung dabei vorher bekannt sein, damit die Dateien von dem entsprechenden Ort importiert und ihre Stile innerhalb der Anwendung angewendet werden können. Listing 7 veranschaulicht die erläuterte Reihenfolge der Sass-Imports des in der Anwendung integrierten Bundles.

```
1 // importiere Bulmas vordefinierte Standard-Variablen
2 @import "bulma_variables_default"
3 // importiere weitere vordefinierte Standard-Variablen der Kern-Anwendung
4 @import "custom_variables_default"
5
6 // importiere ueberschriebene Bulma-Variablen au\{ss}erhalb des Projektes
7 @import "path/to/custom_bulma_variables"
8 // importiere ueberschriebene zusaetzliche Variablen au\{ss}erhalb des
      Projektes
9 @import "path/to/custom_vars"
10
11 // importiere Rest von Bulma
12 @import "bulma_styles"
```

Listing 7: Reihenfolge der Sass-Imports

4.3.7. Unabhängigkeit vom Backend

Damit Anforderungen wie beispielsweise das Abrufen von Datensätzen oder Distributionen erfüllt werden können, müssen darzustellende Metadaten von einem oder mehreren Servern angefordert und verarbeitet werden. Die Art wie Anfragen an diese Server gestellt werden müssen und die Struktur der in den Antworten enthaltenen Daten unterscheidet sich dabei je nachdem wie die im Einzelfall benötigte Schnittstelle implementiert wurde.

Da die Basis-Anwendung unabhängig von der jeweils eingesetzten Schnittstelle sein muss, können die konkreten Anfragen an den Server nicht innerhalb dieser Anwendung implementiert

werden. Es ist Aufgabe der Entwickler einer auf der Kern-Anwendung basierenden Web-Anwendung, die jeweilige Form in der die Anfragen gestellt werden müssen zu kennen und diese entsprechend zu implementieren. Um die richtige Interpretation der in der Antwort enthaltenen Daten innerhalb der Kern-Anwendung zu gewährleisten, müssen diese außerdem anschließend in ein vorgegebenes Format umgewandelt werden. Dieser Lösungsansatz und die dazugehörige Problemstellung findet sich auch im Adapter-Pattern wieder (siehe Kapitel 2.5.3). Dabei stellt die Basis-Anwendung den in Abbildung 6 aufgeführten Client dar. Das Interface Target entspricht dem geforderten Datenmodell. Der Adaptee repräsentiert den Server an den Anfragen gestellt werden müssen. Der Adapter muss dementsprechend vom Benutzer, also Entwickler, implementiert werden. Da in JavaScript keine Interfaces definiert werden können, muss die Schnittstelle der Basis-Anwendung, also das geforderte Datenmodell, zunächst konkretisiert und dokumentiert werden, damit potenzielle Entwickler eines Adapters abgerufene Daten entsprechend anpassen können.

Datenmodell Die entworfenen Datenmodelle für Datensätze und Distributionen (siehe Tabelle 2 und Tabelle 3) orientieren sich stark am standardisierten DCAT-AP-Vokabular (siehe Kapitel 2.1.2). Die Orientierung an dem etablierten Standard soll es der Kern-Anwendung ermöglichen einen Großteil der Metadaten von Datensätzen und Distributionen darstellen zu können, um diese bedarfsgerecht darstellen zu können. Zudem fällt es Entwicklern, denen das DCAT-AP-Vokabular bereits bekannt ist, unter Umständen leichter vom Server abgerufene Daten dahingehend zu strukturieren.

4.3.8. Flexibilität von Elementen

Um einen flexiblen Einsatz der Basis-Anwendung zu gewährleisten, müssen die Inhalte, das Verhalten und das Aussehen zentraler Elemente der Oberfläche von Entwicklern angepasst werden können. Dafür eignet sich das Slot-Feature des Vue-Frameworks. Slots sind Bereiche innerhalb des Templates einer Vue-Komponente, die als Platzhalter dienen. Listing 8 und Listing 9 veranschaulichen, wie Slots angewendet werden können. Das Template der Kind-Komponente muss dafür das Slot-Element besitzen, welches den Bereich definiert, der von einer Eltern-Komponente individuell gefüllt werden kann. Das Template der Eltern-Komponente muss dazu die Kind-Komponente als Element beinhalten. Innerhalb dieses Elements können dann beliebige weitere Elemente und Inhalte untergebracht werden, die anschließend an der Stelle eingefügt werden, an der das Slot-Element in der Kind-Komponente definiert wurde. Darüber hinaus besteht die Möglichkeit dem Slot-Element ein name-Attribut anzugeben, wodurch mehrere Slots innerhalb einer Kind-Komponente definiert werden können. Darüber hinaus können weitere, selbst definierte Attribute angegeben werden, auf die innerhalb der Eltern-Komponente zugegriffen werden kann. Weiterhin kann innerhalb des Slot-Elements bereits ein Template angegeben werden, welches dann automatisch als Standard-Template gilt und gerendert wird, falls der Bereich nicht neu definiert wird. [vuea] [Bema] Mit dem Einsatz von Slots in Komponenten der Kern-Applikation kann demnach ein Standard-Template definiert werden, welches zum Beispiel ausgewählte Metadaten eines Datensatzes darstellt. Falls Entwickler die Inhalte des fest definierten Bereiches anpassen oder dessen Verhalten erweitern wollen, kann der gewünschte Inhalt des Slots ähnlich wie in Listing 9 angegeben und

Name	Typ	Beschreibung
categories	Array<JSON>	Gruppen und Kategorien denen der Datensatz angehört
description	String	Eine Beschreibung des Datensatzes
distributions	Array<JSON>	Metadaten der tatsächlichen Daten des Datensatzes
distributionFormats	Array<String>	Dateiformate in denen Distributionen vorhanden sind
id	String	Eindeutiger Bezeichner des Datensatzes
idName	String	Eindeutiger, aus 'title' abgeleiteter Bezeichner des Datensatzes
language	String	Die Sprache die für die Beschreibung des Datensatzes benutzt wird
licence	String	Die Lizenz unter der der Datensatz veröffentlicht wurde
modificationDate	DateTime	Zeitpunkt an dem der Datensatz das letzte mal bearbeitet wurde
releaseDate	DateTime	Zeitpunkt an dem der Datensatz erstellt wurde
publisher	String	Organisation oder Person die für die Veröffentlichnung des Datensatzes verantwortlich ist
tags	Array<String>	Beschreibende Schlagworte zum Datensatz
title	String	Darzustellender Titel des Datensatzes

Tabelle 2: Datenmodell eines Datensatzes

Name	Typ	Beschreibung
accessUrl	String	Url die Zugriff auf die Distribution ermöglicht
description	String	Eine Beschreibung der Distribution
downloadUrl	String	Url die zum direkten Download der Distribution führt
format	String	Datenformat der Distribution
id	String	Eindeutiger Bezeichner der Distribution
licence	String	Die Lizenz unter der die Distribution veröffentlicht wurde
modificationDate	DateTime	Zeitpunkt an dem die Distribution das letzte mal bearbeitet wurde
releaseDate	DateTime	Zeitpunkt an dem die Distribution erstellt wurde
title	String	Darzustellender Titel der Distribution

Tabelle 3: Datenmodell einer Distribution

Verhaltensweisen der Elemente innerhalb des Bereiches durch Funktionen definiert werden. Damit dabei auf die Daten der Kind-Komponente zugegriffen werden kann, um zum Beispiel andere Metadaten eines Datensatzes darzustellen als es beim Standard-Template der Fall ist, müssen alle in Frage kommenden Angaben im Slot-Element als Attribut angegeben werden. Bei einer Komponente zur Beschreibung eines Datensatzes wären dies zum Beispiel alle Felder im Datenmodell eines Datensatzes (siehe Tabelle 2). Welche Felder konkret dargestellt werden, entscheiden dabei die Entwickler der Eltern-Komponente.

```
1 <template>
2   <div>
3     <p>I am the child component!</p>
4     <!-- Content from the parent gets rendered here. -->
5     <slot></slot>
6   </div>
7 </template>
```

Listing 8: Vue-Slots Beispiel - Kind-Komponente [Bema]

```
1 <template>
2   <div>
3     <child-component>
4       <p>I am injected content from the parent!</p>
5       <p>I can still bind to data in the parent's scope, like this! {{
            myVariable}}</p>
6     </child-component>
```

```
7     </div>
8 </template>
9
10 <script>
11 import ChildComponent from './ChildComponent.vue';
12
13 export default {
14   components: {
15     ChildComponent
16   },
17
18   data: () => ({
19     myVariable: `I'm just a lonely old variable.`
20   })
21 }
22 </script>
```

Listing 9: Vue-Slots Beispiel - Eltern-Komponente [Bema]

4.3.9. Gesamtstruktur

Das in Abbildung 15 abgebildete Strukturdiagramm zeigt die wesentlichen Vue-Komponenten und Vuex-Stores der Kern-Anwendung sowie die zu entwickelnden Adapter einer darauf basierenden Web-Anwendung. Das Wurzel-Element der Anwendung stellt die App-Komponente dar, die die View der gesamten Anwendung repräsentiert. Die direkten Kind-Komponenten von App bilden die logischen Seiten der Single-Page Application und werden bei einem Seitenwechsel dynamisch vom Router ausgetauscht. CustomDatasets, CustomDatasetDetails und CustomDistributionDetails werden benötigt, um bei Bedarf die anpassbaren Bereiche ihrer Kind-Komponenten neu definieren zu können. (siehe Kapitel 4.3.8) Die Kind-Komponenten selbst definieren das Template und die View-Logik der einzelnen logischen Seiten der Anwendung. (siehe Kapitel 4.3.2, 4.3.4, 4.3.5) Durch weitere Unterkomponenten wie DatasetFacets lassen sich die Funktionsbereiche der entsprechenden Eltern-Komponente sinnvoll aufteilen. (siehe Kapitel 4.3.3) Die Komponenten, in denen die View-Logik definiert ist, besitzen Zugriff auf die einzelnen Module des Vuex-Stores. Dabei besitzt jede View einer logischen Seite einen Store, der den Zustand dieser Seite abbildet. Er beinhaltet sowohl die darzustellenden Daten, als auch weitere benötigte Informationen, die für diesen Teil der Anwendung von Bedeutung sind. Außerdem besitzen sie Zugriff auf die Methoden der jeweiligen Adapter, in denen die Anfragen an den Server definiert werden, der die darzustellenden Daten bereitstellt. Die abgerufenen Daten werden anschließend entsprechend dem im Datenmodell definierten Format zurückgegeben und wie bereits erwähnt im Store abgelegt. Die konkrete Implementierung des Adapters hängt dabei von der Implementierung der Server-Schnittstelle ab. (siehe Kapitel 4.3.7)

Diese Struktur ermöglicht es die Anwendung problemlos um weitere Views zu erweitern. Zum Hinzufügen weiterer logischer Seiten müssen lediglich eine weitere Kind-Komponente von App sowie gegebenenfalls weitere ihrer Unterkomponenten implementiert werden und der Zugriff auf einen der bereits existierenden oder eines weiteren Store-Moduls gewährleistet werden. Im Falle eines hinzugekommenen Store-Moduls muss außerdem ein weiterer Adapter

implementiert werden, um weiterhin die Entkopplung der Kern-Anwendung vom jeweiligen Server zu gewährleisten. Im Umkehrschluss lassen sich nicht mehr benötigte Teile der Anwendung, bis hin zu ganzen logischen Seiten, unkompliziert austauschen oder komplett entfernen, ohne den Rest der Anwendung dabei zu beeinflussen.

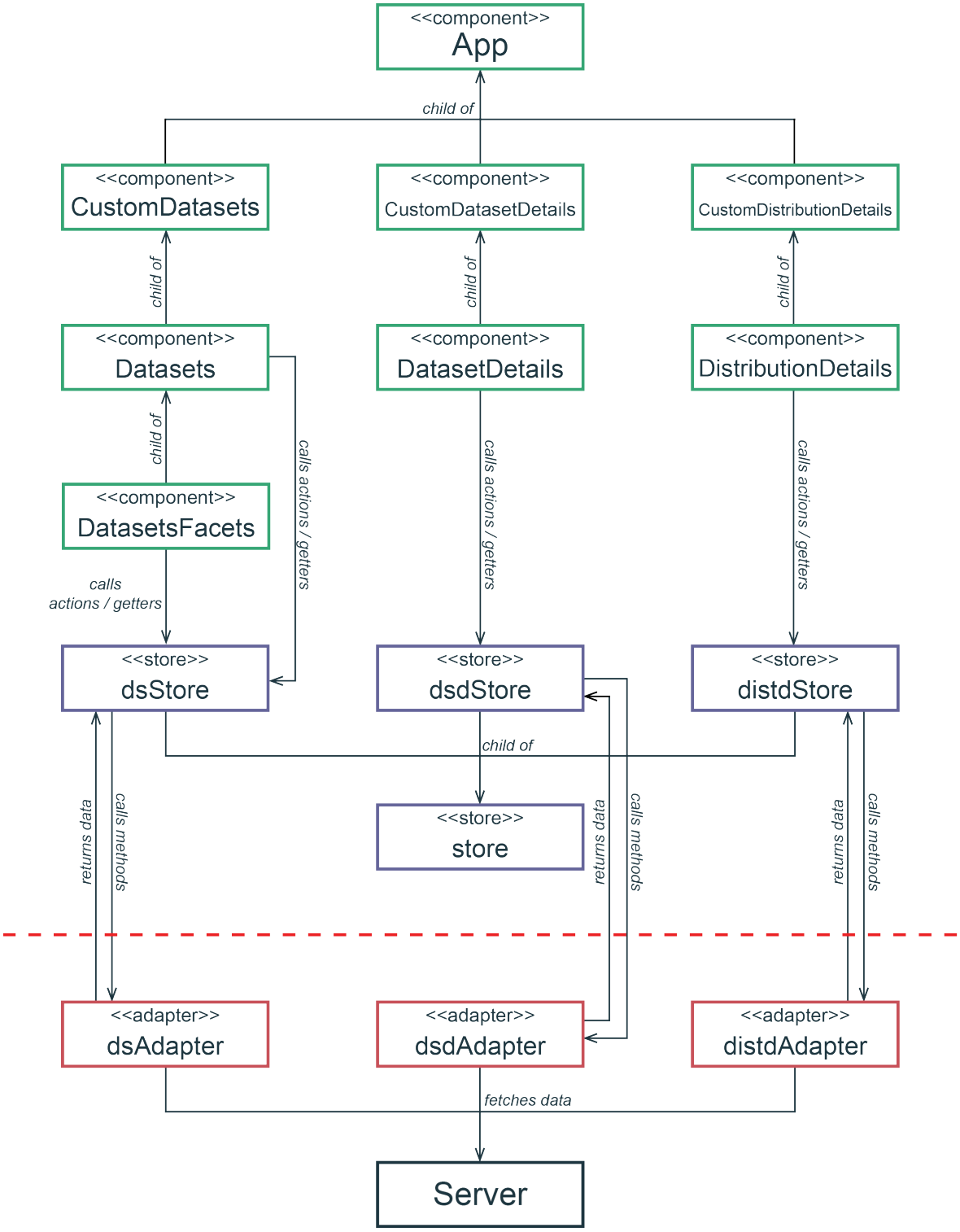

Abbildung 15: Strukturdiagramm der Anwendung

5. Implementierung

Dieses Kapitel beschreibt die prototypische Implementierung der zuvor konzipierten Web-Anwendung anhand ausgewählter Codebeispiele.

5.1. Integration der Adapter zur Datenbeschaffung

Um die Metadaten von Datensätzen von einem Server abrufen und im Anschluss darzustellen zu können, müssen beschrieben Adapter implementiert werden, in denen die konkreten Serveranfragen definiert sind (vgl. Kapitel 4.3.7). Die abgerufenen Daten müssen anschließend dem jeweiligen Datenmodell (siehe Tabelle 2 und Tabelle 3) entsprechend an die Kern-Anwendung übergeben werden.

5.1.1. Beschreibung der Adapter

Ein Adapter entspricht einer ECMAScript 2015 JavaScript Klasse und stellt Methoden bereit, die die Daten einer Entität, wie Datensätzen oder Distributionen, vom Server abrufen und anschließend verarbeiten. Die Methodennamen sind dabei an Namen von HTTP-Methoden angelehnt und vermitteln bereits die Aufgabe einer Methode. So wird in der get-Methode des Adapters, der für die Datenbeschaffung von Datensätzen zuständig ist, eine Liste von Datensätzen abgerufen, in der getSingle-Methode lediglich ein spezieller Datensatz. Jede Adapter-Methode gibt dabei eine Promise zurück, die die zuvor vorbereiteten Daten gemäß des Datenmodells beinhaltet, wodurch der Rest der Anwendung nicht blockiert wird, während die Methode ausgeführt wird. Eine Promise ist ein JavaScript-Objekt, das eine asynchrone Operation kapselt und das Ergebnis dieser Operation repräsentiert. Listing 10 veranschaulicht die Struktur eines solchen Adapters. Eine konkrete Implementierung eines Datensatz-Adapters für eine CKAN-API (siehe Kapitel 2.1.1) kann der beigefügten exemplarischen Implementierung einer CKAN-basierten Web-Anwendung auf Basis der entwickelten Kern-Anwendung entnommen werden.

```
1  export default class DatasetsAdapter {
2    constructor(baseUrl) {
3      this.baseUrl = baseUrl;
4    }
5    /**
6     * @description GET dataset by given id.
7     * @param id - String - Given id of the Dataset to fetch.
8     */
9    getSingle(id) {
10     return new Promise((resolve, reject) => {
11       // ...
12     });
13   }
14   /**
15    * @description GET all datasets matching the given criteria.
16    * @param query - String - The given Query String
17    * @param facets - JSON - Activated facets/filters
18    * @param limit - Int - Maximum count of results to fetch
```

```
19     * @param offset - Int - Number of results to skip
20     * @returns {Promise}
21     */
22    get(query, facets, limit, offset) {
23      return new Promise((resolve, reject) => {
24        // Server request
25        })
26        .then((response) => {
27          // Prepare data according to the defined datamodel
28          resolve(resData);
29        })
30        .catch((error) => {
31          reject(error);
32        });
33      });
34    }
35 }
```

Listing 10: Exemplarische Adapter-Klasse zum Abruf von Datensätzen

5.1.2. Integration und Verwendung der Adapter in die Kern-Anwendung

Im Folgenden wird beschrieben, wie die entwickelten Adapter in die Kern-Anwendung integriert werden, um die bereitgestellten Methoden zu benutzen.

Übergeordnete Verzeichnisstruktur Der folgende Verzeichnisbaum stellt eine abstrakte Verzeichnisstruktur einer auf der Kern-Anwendung basierenden Web-Applikation dar.

```
my-projekt
├── core-app
│   ├── src
│   └── ...
├── my-adapters
│   ├── src
│   └── ...
└── glue-config.js
```

Der oben dargestellte Ordner core-app repräsentiert das Projektverzeichnis der Kern-Applikation. Der Ordner my-adapters beinhaltet die im vorigen Kapitel beschriebenen Adapter zur Datenbeschaffung, wobei der konkrete Ablageort der Adapter nicht vorgegeben ist. Auf der gleichen Verzeichnisebene wie core-app und my-adapters befindet sich die glue-config.js Datei, die im nächsten Absatz näher beschrieben wird.

Referenzierung der entwickelten Adapter-Klassen Listing 11 zeigt die Implementierung der glue-config.js Datei für die exemplarisch entwickelte Web-Anwendung unter Verwendung der CKAN-API des europäischen Datenportals. Zunächst müssen die zuvor entwickelten Adapter unter Angabe der entsprechenden relativen Pfade in die Datei importiert werden. Anschließend wird ein JavaScript-Objekt exportiert, das die Basis-URL der zu verwendenden Schnittstelle und die zuvor importierten Adapter-Klassen enthält.

```
1 import datasetAdapter from './edp-impl/src/datasets';
2 import distributionAdapter from './edp-impl/src/distributions';
3 import datastoreAdapter from './edp-impl/src/datastore';
4
5 export default {
6   api: {
7     baseUrl: 'https://www.europeandataportal.eu/data/api/3/action/',
8   },
9   adapter: {
10    datasetAdapter,
11    distributionAdapter,
12    datastoreAdapter,
13  },
14 };
```

Listing 11: Konfigurationsdatei zum definieren der Schnittstelle und zu verwendenden Adapter

Registrierung der Adapter mit Vue-inject Um die im Konfigurations-Objekt exportierten Adapter in die Kern-Anwendung unkompliziert referenzieren zu können, wird das Vue-inject Paket eingesetzt. Vue-inject ist ein speziell für Vue entwickeltes Dependency-Injection-Tool, das ein einfaches Einspeisen von Abhängigkeiten in Vue-Komponenten zur Laufzeit der Anwendung ermöglicht. Dafür müssen die jeweiligen Abhängigkeiten in diesem Fall die in der glue-config.js definierten Adapter, im Vue-injector registriert werden. Dabei werden außerdem ein Name sowie gegebenenfalls Parameter für den zu registrierenden Service übergeben. Listing 12 zeigt den Code der service.js Datei, in der zunächst das exportierte JavaScript-Objekt der glue-config.js Datei importiert wird. Anschließend wird die darin enthaltene Angabe der Schnittstellen-URL als konstante im Vue-injector registriert (siehe Z. 5), die bei der letztlichen Registrierung der Adapter als Konstruktor-Parameter mit übergeben wird (siehe Z. 7-8).

```
1 import injector from 'vue-inject';
2
3 import glueConf from '../../../glue-config';
4 // Register baseURL as a constant
5 injector.constant('baseUrl', glueConf.api.baseUrl);
6 // Register adapters
7 injector.service('DatasetAdapter', ['baseUrl'], glueConf.services.
      datasetAdapter);
8 injector.service('DistributionAdapter', ['baseUrl'], glueConf.services.
      distributionAdapter);
```

Listing 12: services.js Datei zum registrieren der Adapter

Verwendung der registrierten Adapter in Vue-Komponenten Innerhalb von Vue-Komponenten können in Vue-inject registrierte Services unter Angabe des verwendeten Namens verwendet werden. Die Vue-Komponente dient dabei in diesem Fall nur als Vermittler und leitet den

zur Verfügung gestellten Adapter an das entsprechende Vuex-store-Modul weiter. Listing 13 zeigt einen Auszug der View-Logik innerhalb der Vue-Komponente, die für das Auflisten der Datensätze zuständig ist. Zuerst wird der bei der Registrierung angegebene Name DatasetAdapter im Feld dependencies angegeben (siehe Z. 3). Die registrierte Abhängigkeit kann anschließend innerhalb der Komponente verwendet werden. Innerhalb des created-hooks der Methode die automatisch ausgeführt wird sobald die Vue-Komponente geladen wurde wird zuerst die Vuex-action useService aufgerufen und ihr der zu verwendende Adapter als Parameter übergeben (siehe Z. 10). Folgend wird der Adapter im jeweiligen Vuex-store-Modul hinterlegt und kann von anderen Actions genutzt werden.

```
1  export default {
2     name: 'datasets',
3     dependencies: 'DatasetAdapter',
4     components: { ... },
5     data() { ... },
6     computed: { ... },
7     methods: { ... },
8     watch: { ... },
9     created() {
10      this.useService(this.DatasetAdapter);
11      this.initPage();
12      this.initQuery();
13      this.$nextTick(() => {
14        this.loadDatasets({}).then(() => {
15          this.initFacets();
16        });
17      });
18      window.addEventListener('scroll', this.onScroll);
19    },
20    beforeDestroy() {
21      window.removeEventListener('scroll', this.onScroll);
22    },
23  };
24 </script>
```

Listing 13: Verwendung eines Adapters in der Datasets-Vue-Komponente

Verwendung der Adapter im Vuex-store Listing 14 zeigt einen Auszug aus dem Vuex-store-Modul für Datensätze. Mit dem Aufruf der Action useService (siehe Z. 36-38) wird der übergebene Service lediglich der Mutation SET_SERVICE übergeben (siehe Z. 46-48) und dadurch im state-Objekt hinterlegt. Während der weiteren Laufzeit der Anwendung kann der Service nun jederzeit über die Getter-Funktion getService (siehe Z. 5) abgerufen werden.

Die tatsächliche Verwendung des Services, also des Adapters, findet innerhalb der Action loadDatasets statt (siehe Z. 10-40). Wesentlich ist dabei, dass in Zeile 22 zunächst der zu verwendende Service über die oben erwähnte Getter-Funktion abgerufen und anschließend in Zeile 23 durch den Aufruf der im Adapter definierten get-Methode verwendet wird. Sobald die Methode eine Promise zurückgibt, werden die im response-Objekt enthaltenen Daten, die dem in Kapitel 4.3.7 definierten Datenmodell entsprechen müssen, auf die zuständigen

Mutationen verteilt und im state-Objekt abgelegt. Die abgelegten Daten können nun in den entsprechenden Vue-Komponenten abgerufen und dargestellt werden.

```
// ...
const state = { ... };
const getters = {
  getDatasets: state => state.datasets,
  getService: state => state.service,
  // ... more getters
};

const actions = {
  loadDatasets(
    { commit, state },
    {
      query = getters.getQuery(state),
      facets = getters.getFacets(state),
      limit = RESULTS_PER_PAGE,
      offset = (getters.getPage(state) - 1) * limit,
      append = false,
    },
    ) {
    return new Promise((resolve, reject) => {
      commit('SET_OFFSET', offset);
      const service = getters.getService(state);
      service.get(query, facets, limit, offset)
      .then((response) => {
        commit('SET_AVAILABLE_FACETS', response.availableFacets);
        commit('SET_DATASETS_COUNT', response.datasetsCount);
        if (append) commit('ADD_DATASETS', response.datasets);
        else commit('SET_DATASETS', response.datasets);
        resolve();
      })
      .catch((error) => {
        reject(error);
      });
    });
  },
  useService({ commit }, service) {
    commit('SET_SERVICE', service);
  },
  // ... more actions
};

const mutations = {
  SET_DATASETS(state, data) {
    state.datasets = data;
  },
  SET_SERVICE(state, service) {
    state.service = service;
  },
  // ... more mutations
```

```
50 };
```

Listing 14: Auszug des vuex-store-Moduls für Datensätze

5.2. Anpassung des Designs

Das Design der Web-Anwendung kann in den in Kapitel 4.3.6 genannten Grenzen angepasst werden. Dafür werden zwei Sass-Dateien bereitgestellt, in denen innerhalb der Kern-Anwendung verwendeten Sass-Variablen neue Stile zugewiesen werden können. Die zu manipulierenden Sass-Dateien befinden sich auf der gleichen Verzeichnisebene wie die im vorigen Kapitel erläuterte glue-config.js:

```
my-projekt
├── core-app
├── my-adapters
├── glue-config.js
├── _custom_bulma_variables.sass
└── _custom_variables.sass
```

Listing 15 und Listing 16 zeigen Auszüge der anpassbaren Sass-Dateien. In _custom_bulma_variables.sass können Bulma-Variablen verändert werden, die anschließend auch in _custom_variables.sass zur Verfügung stehen. In _custom_variables.sass selbst befinden sich die in Vue-Komponenten verwendeten Variablen die sinntragend nach dem Schema <Vue-Komponente>-<Element>–<state/device>_<style> benannt wurden, um die Verwendung der Variable zu suggerieren.

```
1 // 2. Primary colors
2
3 //$primary: $turquoise
4 $primary: #009374
5
6 $info: $blue
7 $success: $green
8 $warning: $yellow
9 $danger: $red
10
11 $light: $white-ter
12 $dark: $grey-darker
13 //...
```

Listing 15: Auszug aus _custom_bulma_variables.sass

```
1 // self-defined helper variables
2 $primary-dark: darken($primary, 6)
3 $primary-light: lighten($primary, 6)
4
5 $secondary: $white
6 $secondary-invert: $dark
7
8 $topbar_background-color: $primary
9 $topbar_background-color-invert: $secondary
```

```
10
11
12 // Variables used by vue-components
13 $navbar_background-color: $secondary
14 $navbar_background-color-invert: invert($navbar-background-color)
15 $navbar-nav-item_background_color: $secondary
16 $navbar-nav-item_color: $primary
17 $navbar-nav-item--hover_background_color: darken($navbar-background-color,
       6)
18 $navbar-nav-item--active_color: $primary
19 $navbar-nav-item--hover_border-color: $primary
20 $navbar-nav-item--active_border-color: $primary
21
22 $navbar_background-color--mobile: $primary
23 $navbar_color--mobile: $secondary
24 $navbar-nav-toggle--hover--mobile: darken($navbar_background-color--mobile,
       6)
25 // ...
```

Listing 16: Auszug aus _custom_variables.sass

In der in Listing 17 abgebildeten Sass-Datei wird mithilfe der Import-Funktion von Sass eine Bundle-Datei erstellt. Dabei ist zu beachten, dass die Präsenz der verwendeten sichergestellt wird, indem zuerst die Dateien _custom_bulma_variables_default.sass und _custom_variables_default.sass importiert werden. Anschließend können die anpassbaren Dateien importiert werden, in denen die bereits zuvor angelegten Variablen gegebenenfalls überschrieben werden.

```
1 // import Bulmas Sass Functions to use them inside variable files.
2 @import "~bulma/sass/utilities/functions"
3
4 // import Default/Fallback style variables
5 @import "custom_bulma_variables_default"
6 @import "custom_variables_default"
7
8 // Custom Bulma Design Framework variables
9 @import "../../../../custom_bulma_variables"
10 // Custom variables
11 @import "../../../../custom_vars"
```

Listing 17: Sass-Datei _all.sass zum Bündeln aller definierten Variablen

Abschließend wird wie in Listing 18 dargestellt ein komplettes Bundle erstellt, dass das gesamte Bulma-Design-Framework sowie die weiteren definierten Sass-Variablen enthält. Innerhalb der main.js, dem Einstiegspunkt der Applikation, kann das Bundle nun importiert werden und sorgt so dafür, dass die angepassten Bulma-Stile durch die Benutzung der von Bulma spezifizierten Klassen und Elemente verwendet werden. Das reine Variablen-Bundle (siehe Listing 17) kann außerdem explizit im Style-Bereich von Vue-Komponenten importiert werden, um die Verwendung der darin definierten Sass-Variablen zu ermöglichen. Ein Auszug

aus dem Style-Bereich der Datensatz-Komponente wird in Listing 19 dargestellt.

```
1 // Import Bulma's core
2 @import "~bulma/sass/utilities/all"
3 // Import Customized variables
4 @import "./variables/all"
5 // Import Rest of Bulma
6 @import '~bulma'
```

Listing 18: Sass-Datei custom_bulma.sass zum erstellen des Bulma-Bundles

```
1 <style lang="scss" scoped>
2   @import '../styles/variables/all';
3
4   .datasets {
5     padding: $datasets_padding;
6     .dataset {
7       background-color: $datasets-dataset_background-color;
8       border-top: $datasets-dataset_border-top;
9       border-right: $datasets-dataset_border-right;
10      border-bottom: $datasets-dataset_border-bottom;
11      border-left: $datasets-dataset_border-left;
12      border-radius: $datasets-dataset_border-radius;
13      box-shadow: $datasets-dataset_box-shadow;
14      color: $datasets-dataset_color;
15      margin: $datasets-dataset_margin;
16      padding: $datasets-dataset_padding;
17      transition: $datasets-dataset_transition;
18      &:hover {
19        border-color: $datasets-dataset--hover_border-color;
20        box-shadow: $datasets-dataset--hover_box-shadow;
21        cursor: $datasets-dataset--hover_cursor;
22      }
23      // ...
24    }
25    // ...
26  }
27 </style>
```

Listing 19: Style-Bereich der Dataset.vue Komponente

Aufgetretene Probleme Befindet sich unter dem angegebenen Dateipfad im Variablen-Bundle (siehe Listing 17 Z. 9, 11) zum Importieren der angepassten Stil-Variablen keine entsprechende Datei, schlägt der Build der Anwendung fehl, da die angegebene Datei nicht gefunden werden konnte. Auch nach intensiver Recherche konnte keine Möglichkeit gefunden werden Imports nur durchzuführen, wenn die entsprechend zu importierende Datei auch existiert. Um die Robustheit der Anwendung zu erhöhen, sollte bei der Weiterentwicklung der Anwendung der Entwurf überdacht und dahingehend optimiert werden. Alternativ wäre eine umfangreiche und exakte Dokumentation nötig, die Entwickler über das erläuterte Verhalten

aufklärt und genaue Anweisungen und Hinweise gibt, wie das Design der Anwendung verändert werden kann.

5.3. Tests

Um die Testbarkeit der Anwendung zu veranschaulichen, wurden exemplarische Unit- und Integrations-Tests implementiert, da eine vollständige Abdeckung der Anwendung durch Tests den Rahmen dieser Arbeit überstiegen hätte.

Die Unit-Tests testen die Komponenten der Kern-Anwendung selbst. Die Integrations-Tests dagegen testen zu diesem Zeitpunkt nur Anwendungen die auf der Kern-Anwendung basieren. Um die Kern-Anwendung isoliert mit end-to-end-Tests abdecken zu können, müsste zunächst eine umfangreiche Testumgebung eingerichtet werden, die das Abrufen darzustellender Daten anhand von Mock-Adaptern imitiert. Derzeit werden stellvertretend die exemplarisch entwickelten Adapter für eine CKAN-API verwendet.

5.3.1. Unit-Tests

Werkzeuge Für die Implementierung der Unit-Tests wurden die Werkzeuge Karma, Mocha, Sinon und Avoriaz verwendet. Die Funktion von Karma und Mocha wurde im Kapitel 4.2.1 näher erläutert. Mit Sinon können Platzhalter-Funktionen definiert werden, um lediglich zu überprüfen ob diese ausgeführt wurden, ohne sie tatsächlich auszuführen. Avoriaz ist eine Bibliothek, die hilfreiche Funktionen zum Testen von vue-Komponenten bietet.

Testen von vue-Komponenten Um Vue-Komponenten, die Zugriff auf einen vuex-store besitzen mit Unit-Tests zu testen, ist es zunächst erforderlich, dass der verwendete vuex-store als Mock-Store innerhalb des Test-Pakets definiert wird. In den einzelnen Tests wird die zu testende Komponente dann mit dem zuvor definierten Mock-Store initialisiert und dieser anstelle des eigentlichen vuex-stores während des Tests verwendet. In Kombination mit Sinon kann anschließend getestet werden, ob bestimmte Actions des Stores nach der Initialisierung der Komponente oder zu einem anderen Zeitpunkt aufgerufen wurde.

Listing 20 zeigt einen Auszug aus der Datasets.spec.js Datei in der Unit-Tests zum Testen der Datensatz-Komponente definiert sind. Zunächst wird der Mock-Store definiert (siehe Z. 10-28), der bei der Initialisierung der Komponente durch die von Avoriaz bereitgestellte shallow-Funktion mit übergeben wird (siehe Z. 31-36 und Z. 40-35). Anschließend wird mittels der expect-Funtion ein erwartetes Verhalten definiert. In dem abgebildeten Fall wird in Zeile 37 erwartet, dass die Action useService nach der Initialisierung genau einmal ausgeführt wird. In Zeile 47 wird außerdem erwartet, dass die Action loadDatasets genau einmal ausgeführt wurde. Weicht das tatsächliche vom erwarteten Verhalten ab, gilt der Test als nicht bestanden. Entspricht das Verhalten der definierten Erwartung, gilt der Test als bestanden.

```
1 describe('Datasets.vue', () => {
2   // Store Variables
3   let actions = {};
4   let getters = {};
5   let store = {};
```

```
6   // $route stub
7   const $route = { query: { query: '' } };
8   // Create a mock-store
9   beforeEach(() => {
10    actions = {
11      loadDatasets: sinon.stub(),
12      useService: sinon.stub(),
13      // ...more actions
14    };
15    getters = {
16      getDatasets: () => [...],
17      // ...more getters
18    };
19    store = new Vuex.Store({
20      modules: {
21        datasets: {
22          namespaced: true,
23          state: {},
24          actions,
25          getters,
26        },
27      },
28    });
29  });
30  it('called action useService() once when mounted', () => {
31    shallow(Datasets, {
32      store,
33      globals: {
34        $route,
35      },
36    });
37    expect(actions.useService.calledOnce).to.equal(true);
38  });
39  it('called action loadDatasets() once when mounted', () => {
40    shallow(Datasets, {
41      store,
42      globals: {
43        $route,
44      },
45    });
46    Vue.nextTick(() => {
47      expect(actions.loadDatasets.calledOnce).to.equal(true);
48    });
49  });
```

Listing 20: Unit-Test für Datasets.vue

5.3.2. Integrations-Tests

Wie in Kapitel 4.2.1 im Rahmen des Werkzeugs Nightwatch beschrieben, kann durch Integrations-Tests das Verhalten eines Anwenders während der Benutzung einer Web-Anwendung für

verschiedene Browser simuliert werden. Zusätzlich kann überprüft werden, ob die Anwendung auf die simulierten Aktionen korrekt reagiert.

Listing 21 zeigt einen exemplarisch entwickelten Integrationstest für die Datensatz-Komponente. In Zeile 6 wird zunächst die URL der zu testenden Seite aufgerufen. Anschließend wird geprüft, ob innerhalb von fünf Sekunden ein Element mit der ID app und ein Element mit der Klasse dataset auf der Seite sichtbar sind. Des Weiteren wird ein Klick auf das erste Element mit der Klasse dataset simuliert und darauf geprüft, ob in einem Zeitfenster von fünf Sekunden ein Element mit der Klasse dataset-details sichtbar ist. Besteht die Anwendung alle Überprüfungen wird der Test erfolgreich beendet und gilt als bestanden. Schlägt eine Überprüfung fehl wird der Test abgebrochen und gilt als nicht bestanden.

Eine umfassende Überprüfung aller Browser die alle Bereiche der Anwendung abdeckt hätte die Rahmen dieser Arbeit überstiegen, weshalb der beschriebene Test nur für den Browser Chrome erfolgreich ausgeführt und bestanden wurde. Allerdings wurde die Anwendung in allen Browsern, die in der Anforderung an die Browser Kompatibilität genannt werden (vgl. Kapitel 3.3), händisch getestet. Dabei konnten keine Inkompatibilitäten festgestellt werden. Bei einer Weiterentwicklung der Anwendung sollten die händisch ausgeführten Tests automatisiert werden, indem eine umfassende Abdeckung durch Integrations-Tests für unterstützte Browser realisiert wird.

```
1  describe('Datasets.vue', () => {
2  module.exports = {
3    'e2e tests': function (browser) {
4      const devServer = browser.globals.devServerURL
5      browser
6        .url(`${devServer}/#/datasets`)
7        .waitForElementVisible('#app', 5000)
8        .waitForElementVisible('.dataset', 5000)
9        .click('div.dataset:nth-of-type(1)')
10       .waitForElementVisible('.dataset-details', 5000)
11       .end();
12   }
13 }
```

Listing 21: Integrations-Test für Datasets.vue

6. Fazit

6.1. Zusammenfassung und Bewertung

Ziel dieser Arbeit war es eine Kern-Anwendung in Form einer Single-Page Application zu entwerfen und zu entwickeln, die als Basis für Oberflächen von Open Data Plattformen dient. Dabei sollten die verwendete Schnittstellen, das Design und die Inhalte auf dieser Basis entstehender Web-Anwendungen anpassbar sein, ohne den Quellcode der Kern-Anwendung selbst zu manipulieren.

Dafür wurden sowohl die funktionalen Anforderungen der Nutzer solcher Open Data Web-Anwendungen und der Entwickler dieser Anwendungen bestimmt als auch die nicht funktionalen Anforderungen an die Basis-Applikation formuliert. Anhand der Anforderungen wurden anschließend geeignete Werkzeuge zur Realisierung einer solchen Applikation ermittelt und Konzepte erarbeitet, um daraus resultierende Problemstellungen zu lösen. Letztlich wurde anhand des Entwurfs eine Prototypische Web-Anwendung entwickelt, die den Einsatz der Kern-Applikation während der zukünftigen Entwicklung von darauf basierenden Open Data Plattform-Oberflächen verdeutlicht.

Das Ergebnis der Arbeit erfüllt die Anforderungen an das Projekt hinreichend und stellt eine sinnvolle Lösung einer Basis-Applikation für zu entwickelnde Open Data Plattform-Oberflächen dar. Dennoch besteht weiterhin Optimierungsbedarf für die entwickelten Lösungen zur Anpassung des Designs und der Inhalte von Elementen der Oberfläche. Das Design der Web-Anwendung ist zwar unkompliziert anpassbar, allerdings bietet Sass nicht die Möglichkeit Dateien nur dann zu importieren, sofern diese auch existieren. Folglich schlägt der Build-Prozess der Anwendung fehl, falls die Datei unter dem angegebenen Pfad nicht aufzufinden ist. Des Weiteren können Inhalte von Komponenten nicht angepasst werden ohne die Kern-Anwendung selbst zu manipulieren, sondern es müssen zu diesem Zweck bereitgestellte Vue-Komponenten direkt angepasst werden. Trotzdem sind die eingesetzten Vue-Slots sehr gut für die Individualisierung von Vue-komponenten geeignet, auch wenn die Möglichkeit zur Auslagerung der anzupassenden Komponenten wünschenswert gewesen wäre.

Als besonders herausfordernd erwies sich die Entkopplung des jeweils eingesetzten Backends zur Datenbeschaffung vom Rest der Anwendung. Anfänglich war es die Idee, die konkreten Anfragen an das Backend in den Actions des vuex-stores innerhalb der Kern-Anwendung zu definieren. Die Parameter für die Server-Anfragen sowie die Form der abgerufenen Daten sollten über eine Konfigurationsdatei definiert werden können. Dadurch solle die Kern-Anwendung in der Lage sein die Anfrage wunschgemäß durchzuführen und die erhaltenen Daten interpretieren und darstellen zu können. Dieser Denkansatz musste zunächst verworfen werden, da Konfiguration für den benutzenden Entwickler sehr aufwändig wäre und gleichzeitig fehleranfällig wäre. Zuzüglich müsste die Kern-Anwendung bereits Vorannahmen treffen in welcher Form eine Anfrage an das einzusetzende Backend gestellt werden muss. Letztlich wurde klar, dass sich mit diesem Denkansatz keine zufriedenstellende Lösung realisieren lassen würde. Die Problemstellung wurde darauf grundlegend neu betrachtet. Das Resultat der erneuten Problemanalyse war, dass die Interfaces der Kern-Anwendung und des Backends nicht miteinander kompatibel sind. Die Kern-Anwendung kennt also weder die Form in der Anfragen an das noch unbekannte Backend gestellt werden müssen, noch die Form in der Daten zurückgeliefert

werden. Resultierend aus dieser Erkenntnis wurden Parallelen zum Adapter-Pattern erkannt, die letztlich zur entworfenen Lösung führten. Nämlich, dass die Entwickler einer auf der Kern-Anwendung basierenden Web-Applikation Adapter implementieren müssen, die Anfragen an das eingesetzte Backend durchführen und die abgerufenen Daten gemäß eines vorgegebenen Datenmodells aufbereiten.

Eine weitere besondere Herausforderung stellte die Auswahl und der Einsatz einer Vielzahl von modernen Frontend-Werkzeugen dar. Es musste bei der Recherche stets sichergestellt werden, dass die Technologien untereinander kompatibel waren und ob die explizite Aufgabe für die das Werkzeug eingesetzt werden soll mit dessen Hilfe hinreichend gut erfüllt werden kann. Des Weiteren mussten die ausgewählten Tools verstanden und im Kontext einer Vue-Webpack-Applikation sinnvoll verwendet werden.

6.2. Ausblick

Es ist festzustellen, dass die entstandene Applikation eine solide Grundlage für zu entwickelnde Oberflächen von Open Data Plattformen bietet. Auch wenn für die Anforderungen an die Flexibilität der Elemente oder an die Anpassbarkeit des Designs noch Optimierungsbedarf besteht, kann die Applikation bereits zu diesem Zeitpunkt sinnvoll eingesetzt werden, um Open Data Plattform-Oberflächen mit vergleichsweise wenig Aufwand prototypisch zu realisieren und beispielsweise Schnittstellen zur Datenbeschaffung praxisnah zu testen.

Für einen weitreichenden Einsatz der Basis-Applikation wäre es sinnvoll weitere häufig benötigte Features wie eine Datenvorschau für verschiedene Formate von Distributionen oder eine Detail-Suche für Datensätze hinzuzufügen. Zur Optimierung der Benutzerfreundlichkeit der Oberfläche für Online-Nutzer wären zudem Usability-Tests angebracht, um die auf bereits existierenden Open Data-Oberflächen basierten Oberflächenstrukturen zu optimieren. Des Weiteren sollte eine ausführliche, unmissverständliche Dokumentation zur Verwendung der Kern-Applikation angefertigt werden, um Entwicklern ein komfortables und zeitsparendes Werkzeug für die Entwicklung einer Open Data Frontend-Anwendung zu bieten. Darüber hinaus ist eine vollständige Abdeckung der Anwendung durch Unit- und Integrations-Tests zu empfehlen, um den reibungslosen Betrieb der Anwendung nach dem Hinzufügen weiterer Features zu gewährleisten.

Durch eine kontinuierliche Weiterentwicklung der Applikation könnte schließlich erreicht werden, dass auch Entwickler mit wenig Erfahrung in der Frontend-Entwicklung dazu im Stande wären, eine vollständige Single-Page Application zur Darstellung von Open Data Metadaten zeitsparend zu entwickeln und anschließend zu betreiben.

Literatur

[Fla06] Flanagan, David: *JavaScript - The Definitive Guide*. 5th edition. O'Reilly, 2006. – 497 S.

[Int15] International, Ecma: ECMAScript 2015 Language Specification. (2015), S. XVIII

[Luc10] Lucke, Christian P. G. v.: Open Government Data - Frei verfügbare Daten des öffentlichen Sektors. (2010), S. 2–3

[Uni15] Union, Europäische: DCAT Application Profile for data portals in Europe Version 1.1. (2015), S. 3

Quellen Online

[Ang] Angular. Angular repository. https://github.com/angular/angular. besucht am 25.08.2017.

[bab] babel. Babel - the compiler for writing next generation javascript. https://babeljs.io/. besucht am 18.08.2017.

[Bema] Joshua Bemenderfer. Understanding component slots with vue.js. https://alligator.io/vuejs/component-slots/. besucht am 30.09.2017.

[Bemb] Joshua Bemenderfer. Using vue.js devtools. https://alligator.io/vuejs/vue-devtools/. besucht am 25.09.2017.

[Cin14] Bilal Cinarli. An introduction to css pre-processors: Sass, less and stylus. https://htmlmag.com/article/an-introduction-to-css-preprocessors-sass-less-stylus, 2014. besucht am 18.08.2017.

[Cro] Vivian Cromwell. Between the wires: An interview with vue.js creator evan you. https://medium.freecodecamp.org/between-the-wires-an-interview-with-vue-js-creator-evan-you-e383cbf57cc4. besucht am 28.08.2017.

[da117] da14. Gulp vs grunt vs webpack: Comparison of build tools / task runners. https://da-14.com/blog/gulp-vs-grunt-vs-webpack-comparison-build-tools-task-runners, 2017. besucht am 18.08.2017.

[Dat] Refsnes Data. Node.js npm. https://www.w3schools.com/nodejs/nodejs_npm.asp. besucht am 19.08.2017.

[Deu] Sebastian Deutsch. Die flux architektur und react. http://reactjs.de/posts/die-flux-architektur-und-react. besucht am 08.09.2017.

[Dou16] Brian Douglas. Prerendering explained. www.netlify.com/blog/2016/11/22/prerendering-explained/, 2016. besucht am 11.08.2017.

[Dri] Drifty. Build amazing native apps and progressive web apps with ionic framework and angular. https://ionicframework.com/. besucht am 25.08.2017.

[ESL] ESLint. About - eslint - pluggable javascript linter. https://eslint.org/docs/about/. besucht am 18.08.2017.

[Faca] Facebook. Application architecture for building user interfaces. http://facebook.github.io/flux/docs/in-depth-overview.html#content. besucht am 08.09.2017.

[Facb] Facebook. Learn the basics. https://facebook.github.io/react-native/docs/tutorial.html. besucht am 30.08.2017.

[Facc] Facebook. React - a javascript library for building user interfaces. https://facebook.github.io/react/. besucht am 30.08.2017.

[Facd] Facebook. React documentation. https://facebook.github.io/react/docs/hello-world.html. besucht am 30.08.2017.

[Face] Facebook. Repository of react. https://github.com/facebook/react. besucht am 30.08.2017.

[Facf] Facebook. Repository of react native. https://github.com/facebook/react-native. besucht am 30.08.2017.

[Facg] Facebook. Sites using react. https://github.com/facebook/react/wiki/sites-using-react. besucht am 30.08.2017.

[Fach] Facebook. Tutorial: Intro to react. https://facebook.github.io/react/tutorial/tutorial.html. besucht am 30.08.2017.

[Faci] Facebook. Who's using react native? https://facebook.github.io/react-native/showcase.html. besucht am 30.08.2017.

[Fou] Node.js Foundation. Über node.js. https://nodejs.org/de/about/. besucht am 08.09.2017.

[fWuE] Bundesministerium für Wirtschaft und Energie. Open data: Mit öffentlichen daten digitale wirtschaft fördern. https://www.bmwi.de/Redaktion/DE/Artikel/Digitale-Welt/open-data.html. besucht am 09.08.2017.

[Gec] Minko Gechev. Ahead-of-time compilation in angular. http://blog.mgechev.com/2016/08/14/ahead-of-time-compilation-angular-offline-precompilation/. besucht am 18.09.2017.

[Gim16] Alberto Gimeno. How javascript bundlers work. https://medium.com/@gimenete/how-javascript-bundlers-work-1fc0d0caf2da, 2016. besucht am 18.08.2017.

[Gmb] Workshops Europe GmbH. Dependency injection und der injektor. https://angularjs.de/buecher/angularjs-buch/dependecy-injection/. besucht am 08.09.2017.

[Gooa] Google. Angular - dependency injection. https://angular.io/guide/dependency-injection. besucht am 25.08.2017.

[Goob] Google. Angular - documentation. https://angular.io/docs. besucht am 25.08.2017.

[Gooc] Google. Angular - tutorial: Tour of heroes. https://angular.io/tutorial. besucht am 25.08.2017.

[Good] Google. Angular cli. https://cli.angular.io/. besucht am 25.08.2017.

[Gooe] Google. Angular material. material.angular.io/. besucht am 25.08.2017.

[Goof] Google. Angular quickstart. https://angular.io/guide/quickstart. besucht am 25.08.2017.

[Goog] Google. Angular release schedule. https://github.com/angular/angular/blob/master/docs/RELEASE_SCHEDULE.md. besucht am 25.08.2017.

[Gooh] Google. Explore angular resources - education. https://angular.io/resources. besucht am 25.08.2017.

[Groa] CKAN Association Steering Group. About ckan. https://ckan.org/about/. besucht am 16.08.2017.

[Grob] CKAN Association Steering Group. Ckan features. https://ckan.org/features/. besucht am 16.08.2017.

[Inf] Staatsbetrieb Sächsische Informatikdienste. Metadaten. http://www.opendata.sachsen.de/609.htm. besucht am 05.10.2017.

[Kra] Stefan Krause. A comparison of the perfomance of a few popular javascript frameworks. https://github.com/krausest/js-framework-benchmark#about-the-benchmarks. besucht am 29.08.2017.

[Kra17] Stefan Krause. Results for js web frameworks benchmark. http://www.stefankrause.net/js-frameworks-benchmark6/webdriver-ts-results/table.html, 2017. besucht am 25.08.2017.

[Kö15] Paul Kögel. React einsteiger tutorial. http://reactjs.de/posts/react-tutorial, 2015. besucht am 30.08.2017.

[Lie] Deborah Liebig. Was macht ein webentwickler? https://www.get-in-it.de/arbeitswelt/it-berufe/was-macht-ein-webentwickler. besucht am 08.09.2017.

[Lip] Klaus Lipinski. Datensatz. http://www.itwissen.info/Datensatz-data-record.html. besucht am 05.10.2017.

[Mar16] Danny Markov. 30 learning resources for mastering angular 2. https://tutorialzine.com/2016/09/30-learning-resources-for-mastering-angular-2, 2016. besucht am 25.08.2017.

[McC15] Ben McCormick. Es5, es6, es2016, es.next: What's going on with javascript versioning? https://benmccormick.org/2015/09/14/es5-es6-es2016-es-next-whats-going-on-with-javascript-versioning/, 2015. besucht am 08.09.2017.

[McK13] Craig McKeachie. Choosing a javascript mvc framework. http://www.funnyant.com/choosing-javascript-mvc-framework/, 2013. besucht am 07.09.2017.

[Mei14] Daniel Meixner. Was ist eigentlich... dependency injection (di)? https://blogs.msdn.microsoft.com/dmx/2014/10/14/was-ist-eigentlich-dependency-injection-di/, 2014. besucht am 08.09.2017.

[Moc] Mocha. Mocha - the fun, simple, flexible javascript test framework. http://mochajs.org/. besucht am 25.09.2017.

[Mor17] Josh Morony. Beginners guide to getting started with ionic 2. www.joshmorony.com/beginners-guide-to-getting-started-with-ionic-2/, 2017. besucht am 25.08.2017.

[Moz17] Mozilla. Manipulieren des browser-verlaufes - web-entwickler leitfäden | mdn. developer.mozilla.org/de/docs/Web/Guide/DOM/Manipulating_the_browser_history, 2017. besucht am 11.08.2017.

[MW17] Peter Dierx Michael Wanyoike. A beginner's guide to npm — the node package manager. https://www.sitepoint.com/beginners-guide-node-package-manager/, 2017. besucht am 08.09.2017.

[Nag15] Kazushi Nagayama. Official google webmaster central blog: Deprecating our ajax crawling scheme. webmasters.googleblog.com/2015/10/deprecating-our-ajax-crawling-scheme.html, 2015. besucht am 11.08.2017.

[Nem17] Gergely Nemeth. What is node.js end-to-end testing? https://blog.risingstack.com/end-to-end-testing-with-nightwatch-js-node-js-at-scale/, 2017. besucht am 25.09.2017.

[Neu17] Alexander Neumann. Cross-plattform: Ionic-framework in version 2.0 erschienen. www.heise.de/developer/meldung/Cross-Plattform-Ionic-Framework-in-Version-2-0-erschienen-3607691.html, 2017. besucht am 25.08.2017.

[nIa] npm Inc. What can you make with 475,000 building blocks? https://www.npmjs.com/. besucht am 19.08.2017.

[nIb] npm Inc. What is npm? docs.npmjs.com/getting-started/what-is-npm. besucht am 19.08.2017.

[Nig] Nightwatch. Nightwatch overview. http://nightwatchjs.org/gettingstarted. besucht am 25.09.2017.

[Nih17] Akshay Nihalaney. Real world angular - part 8: Just ahead of in time. https://blog.realworldfullstack.io/real-world-angular-part-8-just-ahead-of-in-time-ae2d3cc89656, 2017. besucht am 25.08.2017.

[Pat] Patreon. Evan is creating vue.js. https://www.patreon.com/evanyou. besucht am 28.08.2017.

[Pfu] Andreas Pfund. Was sind metadaten? definition und beispiel. http://andreas-pfund.de/definition/metadaten/metadaten.php. besucht am 05.10.2017.

[Pro17] ISA Programme. Dcat application profile for data portals in europe. joinup.ec.europa.eu/asset/dcat_application_profile/asset_release/dcat-ap-v11, 2017. besucht am 21.08.2017.

[Ran] Dean Ranft. Javascript frameworks 2017. blog.dkd.de/javascript-frameworks-2017/#fn:4. besucht am 22.08.2017.

[Sac15] Rathes Sachchithananthan. Gulp - wie du dank taskrunner besser durchstartest. https://web-und-die-welt.de/web/gulp-wie-du-dank-taskrunner-besser-durchstartest/, 2015. besucht am 18.08.2017.

[Sal] Amir Salihefendic. Flux vs. mvc (design patterns). https://medium.com/hacking-and-gonzo/flux-vs-mvc-design-patterns-57b28c0f71b7. besucht am 18.09.2017.

[Sen16] Peleke Sengstacke. Javascript transpilers: What they are and why we need them. https://scotch.io/tutorials/javascript-transpilers-what-they-are-why-we-need-them, 2016. besucht am 18.08.2017.

[Sha] Paul Shan. Mvc vs flux – which one is better? http://voidcanvas.com/flux-vs-mvc/. besucht am 18.09.2017.

[Sil] Adam Silver. The disadvantages of single page applications. https://adamsilver.io/articles/the-disadvantages-of-single-page-applications/. besucht am 08.09.2017.

[Sta17a] StatCounter. Desktop browser version market share worldwide july 2017. gs.statcounter.com/browser-version-market-share/desktop/worldwide/#monthly-201707-201707-bar, 2017. besucht am 10.08.2017.

[Sta17b] StatCounter. Search engine market share worldwide. http://gs.statcounter.com/search-engine-market-share#monthly-201701-201707-bar, 2017. besucht am 11.08.2017.

[Sug] James Sugrue. Adapter pattern tutorial. https://dzone.com/articles/design-patterns-uncovered-0. besucht am 29.09.2017.

[T.17] Ferit T. How linting and eslint improve code quality. https://hackernoon.com/how-linting-and-eslint-improve-code-quality-fa83d2469efe, 2017. besucht am 18.08.2017.

[Tho] Jeremy Thomas. Customize bulma. http://bulma.io/documentation/overview/customize/. besucht am 29.09.2017.

[Til] Michelle Tilley. What is flux? http://fluxxor.com/what-is-flux.html. besucht am 15.09.2017.

[Tur17] Lawrence Turton. Introducing vue and weex for native mobile apps. https://code.tutsplus.com/tutorials/introducing-vue-and-weex-for-native-mobile-apps--cms-28782, 2017. besucht am 28.08.2017.

[Vie17] René Viering. Single-page-anwendungen framework-unabhängig entwickeln. www.heise.de/developer/artikel/Single-Page-Anwendungen-Framework-unabhaengig-entwickeln-3633930.html, 2017. besucht am 11.08.2017.

[vuea] vuejs. Content distribution with slots. https://vuejs.org/v2/guide/components.html#Content-Distribution-with-Slots. besucht am 30.09.2017.

[vueb] vuejs. A curated list of awesome things related to vue.js. https://github.com/vuejs/awesome-vue. besucht am 28.08.2017.

[vuec] vuejs. A simple cli for scaffolding vue.js projects. https://github.com/vuejs/vue-cli. besucht am 28.08.2017.

[vued] vuejs. Vue-cli repository. https://github.com/vuejs/vue-cli. besucht am 25.09.2017.

[vuee] vuejs. Vue-loader repository. https://github.com/vuejs/vue-loader. besucht am 25.09.2017.

[vuef] vuejs. Vue webpack bundle repository. https://github.com/vuejs-templates/webpack. besucht am 25.09.2017.

[vueg] vuejs. Vuex. https://vuex.vuejs.org/en/intro.html#. besucht am 25.09.2017.

[w3s] w3sDesign. Dependency injection. http://w3sdesign.com/?gr=u01&ugr=struct. besucht am 15.09.2017.

[web] webpack. Webpack concepts. https://webpack.js.org/concepts/. besucht am 18.08.2017.

[Youa] Evan You. vue-router documentation. https://router.vuejs.org/de/. besucht am 28.08.2017.

[Youb] Evan You. Vue.js api. https://vuejs.org/v2/api/. besucht am 28.08.2017.

[Youc] Evan You. Vue.js comparison with other frameworks. https://vuejs.org/v2/guide/comparison.html. besucht am 28.08.2017.

[Youd] Evan You. Vue.js documentation. https://vuejs.org/v2/guide/. besucht am 28.08.2017.

[Youe] Evan You. Vue.js examples. https://vuejs.org/v2/examples/index.html. besucht am 28.08.2017.

[Youf] Evan You. Vue.js guide getting started. https://vuejs.org/v2/guide/#Getting-Started. besucht am 28.08.2017.

[Youg] Evan You. Vue.js introduction. https://vuejs.org/v2/guide/. besucht am 28.08.2017.

[Youh] Evan You. vuex documentation. https://vuex.vuejs.org/en/. besucht am 28.08.2017.

[Zei15] Oliver Zeigermann. Single page-anwendungen - vorteile von web und desktop kombiniert. http://www.embarc.de/single-page-anwendungen/, 2015. besucht am 09.08.2017.

[Zie] Friedel Ziegelmayer. Karma - how it works. https://karma-runner.github.io/1.0/intro/how-it-works.html. besucht am 25.09.2017.

Glossar

Datensatz Ein Datensatz ist die Zusammenfassung von Daten, die in einer direkten Beziehung zueinander stehen oder gemeinsame Merkmale haben. [Lip].

Distribution Eine Distribution ist eine spezifische Form, in welcher der Datensatz veröffentlicht wird. Jeder Datensatz kann in verschiedenen Formen verfügbar sein, die sich zum Beispiel hinsichtlich der Vertriebsform, im Dateiformat oder dem betrachteten Zeitraum unterscheiden. [Inf].

Metadaten Als Metadaten werden strukturierte Daten bezeichnet, die Informationen über andere Informationsressourcen enthalten. [Pfu].

A. Anhang

A.1. Inhalte der beigefügten CD

Die beigefügte CD enthält die folgenden Ressourcen:

- Quellcode der Anwendung
- Installationsanleitung
- Screenshots der Anwendung
- Ausarbeitung als PDF-Datei
- Online Quellen