DHBW
Duale Hochschule
Baden-Württemberg

Clientseitige Webframeworks wie Angular, ReactJS und OpenUI5

Seminararbeit

für die Prüfung zum

Bachelor of Science (B.Sc.)

des Studiengangs Wirtschaftsinformatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

Verfasser	Sebastian Greulich, Fabio Krämer
Kurs	WWI16B2
Betreuer	Prof. Dr. Ratz, Dietmar
	Prof. Dr. Pohl, Philipp
	Schulmeister-Zimolong, Dennis
Abgabe	07.01.2019

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich meine Seminararbeit mit dem Thema: „*Clientseitige Webframeworks wie Angular, ReactJS und OpenUI5*“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, 07.01.2019

Sebastian Greulich

Fabio Krämer

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1. Einleitung

Autor: Sebastian Greulich

1.1. Projektumfeld

Im Zuge des Projektes „Stochastische Unternehmensbewertung mittels Methoden der Zeitreihenanalyse“, kurz SUMZ, soll die Software „business horizon“ weiterentwickelt werden. Mit dieser Anwendung soll es möglich werden, auf Basis von betriebswirtschaftlichen Daten den zukünftigen Unternehmenswert zu bewerten. In „business horizon“ wird momentan das Webfrontendframework Angular verwendet. Diese Seminararbeit soll in einem Vergleich mit ReactJS und OpenUI5 bewerten, ob Angular aus verschiedenen Gesichtspunkten die passendste Oberflächentechnologie für das Projekt darstellt. Im Zuge dieser wissenschaftlichen Arbeit soll eine Handlungsempfehlung für das zukünftige Webfrontendframework ausgesprochen werden.

1.2. Aufbau

Im folgenden Kapitel „Webframeworks“ sollen die technischen Grundlagen zum Verständnis der zu vergleichenden Technologien gelegt werden. Hierbei wird beispielsweise auf die verwendeten Programmiersprachen JavaScript und TypeScript, aber auch auf verschiedene Webarchitekturen eingegangen.

Im weiteren Verlauf der vorliegenden Seminararbeit werden die einzelnen clientseitigen Webframeworks AngularJS, ReactJS und OpenUI5 beschrieben. In den Beschreibungen werden die allgemeinen Rahmenbedingungen der einzelnen Frameworks dargelegt. Diese bilden beispielsweise die verwendete Lizenz oder auch die Anforderungen an die Entwicklungsumgebung. Darüber hinaus wird in jeder Beschreibung die Architektur des Frameworks beleuchtet. Hierzu gehören sowohl die notwendigen Dateikomponenten, als auch das Definieren und Reagieren auf Events. Zum Abschluss jeder Beschreibung wird auf die möglichen Einsatzgebiete der Technologien eingegangen.

Nach den detaillierten Beschreibungen der Webframeworks wird in Abschnitt 6.2 erörtert, welche Technologie sich am besten zur Verwendung in „business horizon“ eignet.

2. Grundlagen

Autor: Sebastian Greulich

Dieses Kapitel hat zum Ziel, ein allgemeines Verständnis von Webframeworks, Webarchitekturen und JavaScript mit seinen Erweiterungen und Obermengen (TypeScript, JSX) zu vermitteln. Zuerst werden in Abschnitt 2.1 Webframeworks im Allgemeinen vorgestellt. Im Anschluss daran werden Black-Box- und White-Box-Frameworks miteinander verglichen und die allgemeinen Vor- und Nachteile von Frameworks beschrieben. Nach den Webframeworks werden in Abschnitt 2.2 verschiedene Architekturmöglichkeiten im Webumfeld erläutert, dazu gehören unter anderem das MVC-Modell und Single-Page-Applikationen. Den Abschluss des Grundlagenkapitels bildet der Abschnitt 2.3, eine Vorstellung der Websprache JavaScript und der Weiterentwicklung von ihr: TypeScript.

2.1. Webframeworks

2.1.1. Allgemeines

Webframeworks bilden die Rahmenstruktur für clientseitige Oberflächenentwicklungen. Sie sind eine Sammlung von Methoden, die Funktionalitäten durch einen Methodenaufruf ermöglichen und somit die Struktur der Oberflächen bestimmen. Webframeworks haben sich aufgrund ihrer einfachen Implementierung etabliert. Mit dem Einsatz von Webframeworks muss der Entwickler nicht Experte in jedem Teilgebiet seiner Entwicklung sein, sondern kann schon vorgefertigte und getestete Elemente übernehmen und spezielle Funktionalitäten von ihnen nutzen. Beispielsweise nimmt das Weboberflächenframework *bootstrap* dem Entwickler ab, sich um die Implementierung responsiven Fähigkeiten einer Webanwendung zu kümmern.(vgl. Schatten 2010, S. 312 ff.)

Ein Webframework nutzt hierbei ein Grundprinzip der Informatik: die Wiederverwendung. Bei der Wiederverwendung wird vermieden redundanten Quellcode zu schreiben. Dies bietet zahlreiche Vorteile. Zum einen muss sich der Entwickler nicht mehrmals die gleichen Gedanken zur Problemlösung machen, zum anderen muss der Entwickler bei einem gefundenen Fehler nur an einer zentralen Stelle und nicht bei jeder Verwendung nachbessern.(vgl. ebd., S. 302 ff.)

2.1.2. Black-Box- und White-Box-Webframeworks

Bei Frameworks unterscheidet man zwischen Black-Box- und White-Box-Frameworks. Bei einem **Black-Box-Frameworks** ist der Entwickler gezwungen sich auf vorgefertigte Methoden zu verlassen. Er kann das Framework nur einbinden aber keine logischen Veränderungen daran vornehmen. Bei einem **White-Box-Framework** stehen im Gegensatz dazu oftmals nur abstrakte oder leere Methoden zur Verfügung. Diese Methodenhülsen muss der Entwickler in seiner Software mit Inhalt füllen. Bei einem Vergleich dieser beiden Frameworkarten lässt sich feststellen, dass der Entwickler bei einem White-Box-Framework dieses im Ganzen verstanden haben muss um die Lücken mit seinem speziellen anwendungsfallspezifischen Code zu füllen. Wohingegen das Black-Box-Framework eine Standardlösung darstellt, welche sich nicht näher auf den speziellen Anwendungsfall anpassen lässt, aber dafür nur eine geringe Einarbeitungszeit seitens des Entwicklers fordert.(vgl. Kojarski et al. 2006, S. 2)

2.1.3. Vor- und Nachteile von Webframeworks

Zum Abschluss des Kapitels „Webframeworks“ werden einige Vor- und Nachteile bei der Verwendung dieser kritisch betrachtet.

Zuerst wird auf die Nachteile bei der Verwendung von Webframeworks eingegangen:

Jedes Framework hat einen hohen Grad an **Komplexität**. Diese ist notwendig, um verschiedensten, teils komplexen Anforderungen gerecht zu werden. Doch auch der Anwender muss die Funktionsweise des Frameworks zumindest teilweise verstanden haben, um es effizient in seiner Entwicklung einsetzen zu können.

Bei der Nutzung eines Frameworks ergeben sich zwangsläufig **Abhängigkeiten** von diesem. Dies führt dazu, dass der Entwickler sich an eine bestimmte vorgegebene Architektur halten muss. Darüber hinaus muss der Frameworknutzer darauf vertrauen, dass der Frameworkentwickler seine Software ausreichend testet bevor er es an die Nutzer verteilt. Sonst ist die Funktionalität, auf welche der Entwickler vertraut, nicht gewährleistet.

Viele Frameworks (oft im Open-Source-Umfeld) leiden häufig an **mangelnder Dokumentation**. Dies macht es dem Entwickler schwer, sich in das Framework einzuarbeiten und es von Beginn an effizient zu nutzen.

Sowohl die Komplexität als auch die Abhängigkeit können auf der einen Seite als Nachteil gesehen werden. Jedoch auf der anderen Seite kann durch das umfangreiche Funktionsangebot sehr viel Zeit und Aufwand gespart werden.

Ein Framework wird normalerweise im objektorientierten Umfeld genutzt. Somit gibt es dem Entwickler schon ein bestimmtes **Architekturmuster** vor und er ist gezwungen, seine Webanwendung modular und nicht spontan aufzubauen.

Aufgrund der Generik eines Webframeworks ist es möglich, dieses für viele verschiedenste Anwendungsfälle **wiederzuverwenden**. Damit spart sich der Entwickler wertvolle Zeit und Aufwand.

Webframeworks sind auf **Erweiterbarkeit** ausgelegt. Man kann den Quellcode von den meisten Frameworks untersuchen sowie beliebig verändern und erweitern. Dies bietet den Vorteil, falls man als Entwickler einen anderen Datentyp benötigt oder in eine Funktion einen weiteren Parameter hinzufügen möchte, kann man das Framework an die speziellen Bedürfnisse anpassen.

Durch **Inversion of Control** bleibt der Kontrollfluss auf Seiten des Frameworks. Somit wird die Steuerung zur Ausführung bestimmter Unterprogramme an das Framework abgegeben und der Entwickler kann sich voll und ganz auf die Implementierung seiner Geschäftslogik konzentrieren.

Zuletzt ist der Vorteil durch **Standardisierung** anzuführen. Bestimmte Frameworks haben sich in ihrem Umfeld zum Standard entwickelt. Die Frameworkstandardisierung verkürzt die Einarbeitungszeit von Entwicklern in diesem Umfeld, da wiederholt aufkommende Herausforderungen immer auf dieselbe Weise gelöst werden können.(vgl. Schatten 2010, S. 313 ff.)

2.2. Webarchitekturen

2.2.1. MVC-Modell

Das Entwurfsmuster MVC stammt aus den 1970er Jahren. Es wurde ursprünglich für Desktopanwendungen entwickelt, wird nun aber auch sowohl im Web, als auch bei mobilen Anwendungen verwendet. Die Idee hinter dem MVC-Konzept ist die Trennung von Bestandteilen einer Anwendung in jeweils eigene Verantwortlichkeiten. Somit erreicht man voneinander unabhängige Teile, welche eigenständig und austauschbar sind. Mit dem MVC-Konzept vermeidet man darüber hinaus eine Vermischung von Aufgaben und der damit verbundenen Logik.(vgl. R. Steyer 2017, S. 7 ff.)

Der MVC-Ansatz umfasst 3 Komponenten. Die Erste davon ist das Model. Es steht für das zu präsentierende Datenmodell einer Anwendung. Die Zweite Komponente stellt die View dar. Sie ist für die Präsentation der im Model enthaltenen Daten zuständig. In den Verantwortungsbereich der View fällt außerdem das Anzeigen von Benutzerinteraktionsmöglichkeiten wie beispielsweise ein Button. Die letzte Komponente bildet der Controller. Er bildet die Steuerungseinheit der Anwendung und kümmert sich um die Logik einer Applikation. Zu seinen Aufgaben gehört die Vermittlung zwischen dem Model und der View sowie der Interaktion mit dem Benutzer. Der Controller muss sicherstellen, dass alle Anpassungen in der View konsistent auf das Datenmodell angewandt werden.(vgl. Magnucki 2017, S. 7 ff.)

In älterer Software befinden sich das Model und der Controller typischerweise auf dem Server, jedoch bei neueren Entwicklungen werden diese beiden Komponenten direkt in den Browser geladen. Der Vorteil hierbei ist, dass der Internettraffic reduziert und die Performance der Anwendung erhöht wird.(vgl. R. Steyer 2017, S. 7 ff.)

2.2.2. Single-Page-Applications

Früher wurde bei jeder Interaktion eine Nachricht an den Server geschickt und eine neue Webseite als Antwort zurückgesendet. Dies war aufgrund von langsamen Browsern die einzige Möglichkeit, auf Benutzereingaben zu reagieren. Im Laufe der Zeit wurden die Webbrowser immer performanter und damit wurde es komfortabel möglich, mittels JavaScript die Webseite ohne den Server zu verändern.

Um das Jahr 2005 fing die Technologie AJAX (Asynchronous JavaScript and XML) an, sich als Webkommunikationsstandard zu etablieren. Hierbei war die Technik, die AJAX zu Grunde liegt nicht neu, denn es ist lediglich eine Kombination bestehender Technologien gepaart mit einer XMLHttpRequest. Durch AJAX wurde es somit möglich, asynchrone Anfragen an den Webserver zu senden und die erhaltenen Daten in die bereits geöffnete Webseite zu integrieren. Nun war die Singe-Page-Application geboren. Das Grundgerüst einer Webseite musste bei dieser Technik nur beim initialen Aufruf geladen werden. Heutzutage nutzen fast alle Webframeworks die AJAX-Technik um die Ladezeiten der Webseiten zu verkürzen, falls nur kleine Änderungen vorliegen. Ein weiterer Vorteil ergibt sich darüber hinaus durch die Asynchronität, denn falls eine schlechte Internetverbindung vorliegt zeigt der Browser während der Ladezeit nicht eine weiße Ladeseite sondern der Benutzer kann die Webinhalte weiter konsumieren.(vgl. Fink 2014, S. 4; vgl. Jäger 2008, S. 7 ff.)

2.2.3. Websockets

Mit den bisher beschriebenen Technologien musste der Client immer eine Anfrage an den Server schicken. Doch bei bestimmten Anwendungsfällen ist eine persistente und bidirektionale Verbindung zum Server notwendig. Beispielsweise will man bei Chat-Anwendungen oder Live-Sport-Ticker immer in Echtzeit die aktuellen Informationen bereitgestellt bekommen. Doch die bisher gängigen HTTP-Verbindungen sehen diese Art der Verbindung nicht vor. Auf ihrer Basis müsste der Benutzer die Webseite immer manuell aktualisieren. Mittels eines Websockets kann eine solche persistente Client-Server-Verbindung geöffnet werden. Ein Websocket basiert auf einem eigenen Protokoll, dieses muss zuerst über HTTP-Anfragen verhandelt werden. Wenn Client und Server der Verbindung zugestimmt haben, besteht zwischen den beiden Parteien ein dauerhafter „Tunnel“, über diesen können Informationen mit geringer Latenz ausgetauscht werden. Im Vergleich zu einer herkömmlichen HTTP-Verbindung wird der überflüssige Header eingespart, dies führt zu weniger Internettraffic. Wenn eine der beiden Parteien die dauerhafte Verbindung trennen möchte, muss sie der Anderen eine „Close“-Nachricht senden.(vgl. Fink 2014, S. 11 ff.)

2.3. JavaScript

Autor: Fabio Krämer

Zur Entwicklung einer clientseitigen Anwendung wird die Skriptsprache JavaScript eingesetzt. Der Sprachstandard von JavaScript wird durch das kontinuierlich weiterentwickelt ECMAScript spezifiziert. Durch die Verwendung der Obermenge TypeScript und der Spracherweiterung JSX kann die Entwicklung erleichtert und zudem können Funktionen, die nicht Teil des Sprachstandard von JavaScript sind, eingesetzt werden. Im Folgenden werden die mit JavaScript zusammenhängende Sprachen EcmaScript, TypeScript und JSX näher erläutert.

2.3.1. Der Sprachstandard ECMAScript

ECMAScript spezifiziert den Sprachstandard von JavaScript. Dieser wird seit dem Jahr 1997 Jahren von der European Computer Manufactures Association (kurz: ECMA) weiterentwickelt. Zunächst wurden die Versionen durchnummeriert (ES1, ES2, ES3, ES4, und ES5). Im Jahre 2015 wurde beschlossen, dass jährlich eine neue Version von ECMAScript erscheinen soll. Daher tragen die nachfolgenden Versionen das Veröffentlichungsjahr im Namen (ECMAScript2015, ECMAScript2016, ECMAScript2017, ECMAScript2018, . . .).

Die neusten Browser unterstützen meist den aktuellsten ECMAScript Sprachstandard. Allerdings verwendet nicht jeder Benutzer einen neuen Browser. Um die Kompatibilität einer Webanwendung zu gewährleisten, muss der Code in eine von den meisten Browsern unterstütze Version transpiliert werden.(vgl. Woiwode et al. 2018, S. 27 ff.; vgl. Terlson 2018; vgl. M. Steyer et al. 2017, S. 13 ff.)

Im Folgenden wird auf die, für die in Kapitel 3,4 und 5 vorgestellten Frameworks, relevantesten Sprachfeatures eingegangen.

In ECMAScript2015 wurden die Variablentypen let zur Eingrenzung des Geltungsbereichs einer Variable und const zur Deklaration einer Konstanten eingeführt. Zudem können seit ECMAScript2015 auch Klassen und Module in JavaScript definiert werden. Eine Klasse kann mehrere Eigenschaften und Methoden enthalten. Zudem können Klassen voneinander erben.

Jede Datei ist ein eigenes Modul. Module fassen zusammengehörige Codeeinheiten zusammen und können Interfaces, Klassen oder Variablen bereitstellen, die wiederum von anderen Modulen verwendet werden können.(vgl. Woiwode et al. 2018, S. 34 f.; vgl. M. Steyer et al. 2017, S. 19 ff.)

Wenn Java-Script Module in einer Webanwendung verwendet werden, wird ein Modul-Loader gebraucht. Dieser löst verschiedene Abhängigkeiten auf und erstellt daraus eine Ausgabedatei. Die Verwendung des Modul-Loader *Webpack* wird sowohl für Angular als auch für React Anwendungen empfohlen. *Webpack* unterstützt zudem Hot Module Replacement. Änderungen am Code werden zur Laufzeit übernommen. Dadurch muss eine Anwendung nach einer Änderung nicht komplett neu gebaut werden.(vgl. Woiwode et al. 2018, S. 13, 21; vgl. Zeigermann et al. 2016, S. 9, 295–301; vgl. Hlushko et al. 2018)

Seit ECMAScript2017 können Dekoratoren für die Angabe von Metainformationen zu einer Klasse verwendet werden. Dies wird beispielsweise von dem in Kapitel 3 vorgestellten Framwework Angular zur Kennzeichnung und Konfiguration der unterschiedlichen Bestandteile verwendet.(vgl. Woiwode et al. 2018, S. 30 ff.)

2.3.2. Die Obermenge TypeScript

TypeScript ist eine von Anders Hejlsberg bei Microsoft entwickelte Sprache. Diese erweitert die bestehende ECMAScript Version um weitere Sprachelemente und bildet somit eine Obermenge von JavaScript (siehe Abbildung 1). Ein Transpilierer übersetzt TypeScript in JavaScript.

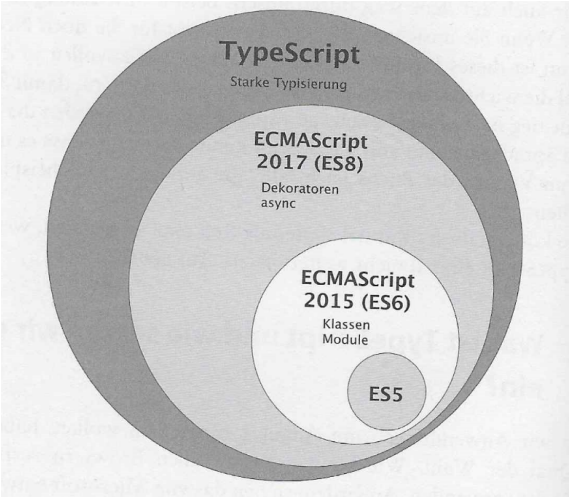

Abbildung 1.: Beziehung zwischen ECMAScript und TypeScript
Quelle: Woiwode et al. (2018, S. 28)

TypeScript ergänzt JavaScript unter anderem um ein stärkeres Typsystem. Hierdurch können Typfehler bereits zur Compilezeit erkannt und Tools zur Codeanalyse (automatische Codevervollständigung, Refactoring-Unterstützung, ...) eingesetzt werden.(vgl. Woiwode et al. 2018, S. 27 ff.; vgl. M. Steyer et al. 2017, S. 13 ff.; vgl. Zeigermann et al. 2016, S. 10)TypeScript ermöglicht zudem die Verwendung von Interfaces. (vgl. Woiwode et al. 2018, S. 40 f.)

Folgende Basisstypen stellt TypeScript zur Verfügung: (vgl. Woiwode et al. 2018, S. 34 ff.; vgl. M. Steyer et al. 2017, S. 16 ff.)

- **number**: Ganz- oder Kommazahl
- **string**: Zeichenkette
- **boolean**: Wahrheitswert
- **Array<typ>**: typisierte Arrays
- **any**: simuliert Standardverhalten von JavaScript – beliebiger Datentyp
- **Function**: Funktion

2.3.3. Die Erweiterung JSX

Die Spracherweiterung JSX ermöglicht die Verwendung einer HTML ähnlichen Syntax in JavaScript. JSX wird durch einen geeigneten Transpilierer in gültiges JavaScript übersetzt.Zeigermann et al. (vgl. 2016, S. 10) empfiehlt an dieser Stelle die Verwendung des Compilers *Babel*.

JSX wird unter anderem im Framework React zur Beschreibung einer Benutzeroberfläche eingesetzt. Die Benutzeroberfläche wird in React nicht unter Verwendung eines Templates, sondern über JavaScript-Befehle aufgebaut. Die Verwendung von JSX Ausdrücken anstatt JavaScript-Befehlen erleichtert diesen Vorgang. Im Folgenden werden die Grundkonzepte von JSX näher erläutert.

Die Ausgabe von dynamische Informationen erfolgt unter Verwendung von JavaScript-Ausdrücken. Diese Ausdrücke müssen in ein paar geschweifter Klammern gepackt werden. Bedingungen können mithilfe des ternären Operatoren überprüft werden. Zur Ausgabe von Listen mit wiederholenden Elementen steht die für Arrays definierte JavaScript Methode map zur Verfügung.

JSX bietet zudem die Möglichkeit das Aussehen eines Elements über das style-Attribut zu verändern. Das sytle-Attribut ist ein Objekt, daher wird die CSS-Eigenschaft und der zugehörige Wert als Objekt-Literal übergeben. Zudem ist zu beachten, dass die CamelCase-Notation für die CSS-Eigenschaften verwendet werden muss. In Listing 1 wird die CSS-Eigenschaft *color* und *font-size* des HTML-Elements gesetzt.

Um XSS-Attacken zu verhindern, werden Strings in JSX automatisch escaped bzw. maskiert. Ein String wird demnach immer als Text interpretiert und dargestellt.(vgl. Zeigermann et al. 2016, S. 59 ff.; vgl. Stefanov 2017, S. 65 ff.)

```
1 const titleColor = 'red';
2 const titleFontSize = 12;
3 const helloWorld = <h1 style = {{
4 color: titleColor,
5 fontSize: titleFontSize + 'px'
6 }}> Hello, World</h1>
```

Listing 1: Beispiel für die Verwendung von JSX

3. Angular

Autor: Fabio Krämer

3.1. Allgemein

3.1.1. Einführung in das Framework

Angular ist ein von Google verwaltetes OpenSource White-Box Framework, mit dem unter anderem clientseitige Single-Page-Applikationen entwickelt werden können. Die erste Version des Angular-Frameworks hat den Namen AngularJS. Alle nachfolgenden Versionen tragen den Namen Angular, da es zwischen der ersten und zweiten Version des Frameworks grundlegende Änderungen gab.

Das Framework wird kontinuierlich weiterentwickelt. Monatlich soll eine Minor-Version und alle sechs Monate eine Major-Version erscheinen. Die aktuellste stabilste Version von Angular hat die Versionsnummer 7.1.4. (vgl. Woiwode et al. 2018, S. vii ff.; vgl. Freeman 2018, S. 3 ff.)

Angular ist in der Sprache TypeScript geschrieben. Angular kann mit TypeScript, JavaScript oder Dart genutzt werden. (vgl. Woiwode et al. 2018, S. vii f.; vgl. M. Steyer et al. 2017, S. 13)

Das in Unterabschnitt 2.2.1 beschriebene MVC Entwurfsmuster kann in einer Angular-Anwendung umgesetzt werden. Die Abbildung 2 zeigt die einzelnen Bestandteile der Anwendung. Dabei implementiert das Template die View, die Komponente den Controller und das Modell das Model. (vgl. Freeman 2018, S. 34 ff.) Im weiteren Verlauf werden die Bestandteile näher beschrieben.

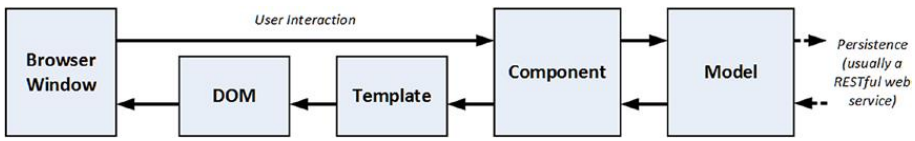

Abbildung 2.: Umsetzung des MVC-Musters in Angular
Quelle: Freeman (2018, S. 35)

3.1.2. Vorbereitung der Entwicklungsumgebung

Für die Entwicklung einer Angular-Anwendung wird unter anderem NodeJS, AngularCLI und eine geeignete Entwicklungsumgebung benötigt.

Die JavaScript Laufzeitumgebung *NodeJS* ermöglicht unter anderem das Ausführen von JavaScript Code auf dem Server. Einige Tools, die bei der Entwicklung einer Webanwendung eingesetzt werden, verwenden NodeJS. Außerdem bietet NodeJS Pakete mit wiederverwendbarem Code, die über den in integrierten Paketmanager *npm* installiert werden können. Die aktuellste Version kann von `https://nodejs.org/` heruntergeladen und installiert werden. (vgl. M. Steyer et al. 2017, S. 1 ff.; vgl. Freeman 2018, S. 7 ff.; vgl. Woiwode et al. 2018, S. 6 ff.)

Das Angular Commandline Interface (kurz: AngularCLI) unterstützt den Entwickler beim Erzeugen und Verwalten einer Angular-Anwendung. AngularCLI erzeugt unter anderem das Grundgerüst einer Angular-Anwendung und richtet den TypeScript Compiler, Werkzeuge zur Testautomatisierung und den Build-Prozess ein. Die aktuelle Version von AngularCLI kann über den Paketmanager *npm* installiert werden. (vgl. M. Steyer et al. 2017, S. 1 ff.; vgl. Freeman 2018, S. 7 ff.; vgl. Woiwode et al. 2018, S. 6 ff.)

Dies ist die Struktur einer durch AngularCLI erzeugten Angular-Anwendung:

```
├ example/
|    ├ src/
|        ├ app/
|        –app.component.css ⇒ CSS-Datei von AppComponent
|        –app.component.html ⇒ Template von AppComponent
|        –app.component.ts ⇒ Komponente AppComponent
|        –app.module.ts ⇒ Root-Modul AppModule
|        ├ assets/
|        ├ environments/
|    –index.html
|    –main.ts
|    –styles.css
|    –...
|    ├ node_modules/
|    ├ e2e/
| –...
```

Sowohl Woiwode et al. (vgl. 2018, S. 3 ff.) als auch M. Steyer et al. (vgl. 2017, S. 3 ff.) empfehlen die Verwendung der frei verfügbaren Entwicklungsumgebung Visual Studio Code. Die Entwicklungsumgebung unterstützt die Entwicklung in TypeScript und lässt sich leicht durch Plug-Ins, die die Entwicklung erleichtern sollen, erweitern. Visual Studio Code ist für Linux, Mac und Windows verfügbar und kann unter `https://code.visualstudio.com/Download` heruntergeladen werden.

3.2. Grundkonzepte

In diesem Kapitel werden die Grundkonzepte von Angular näher beschrieben. Darüber hinaus bietet Angular weitere Konzepte wie den Router, der Ansichten abhängig vom Zustand einer Anwendung lädt. Angular kann zudem auf weiteren Plattformen eingesetzt werden. Mit NativeScript lassen sich beispielsweise Native Apps für Android, IOS und Windows Phone entwickeln.(vgl. M. Steyer et al. 2017, S. 109; vgl. Woiwode et al. 2018, S. 431–440) Diese weiteren Konzepte werden unter anderem in Woiwode et al. (2018) und M. Steyer et al. (2017) näher erläutert.

Zur Beschreibung der Grundkonzepte wird die Beispielanwendung in Abbildung 3 verwendet. Im Input-Feld der Anwendung kann ein Name eingegeben werden. Dieser Name wird zur Begrüßung in der Überschrift verwendet. Eine Änderung des Namens im Input-Feld bewirkt eine Veränderung des Namens in der Überschrift.

Abbildung 3.: Screenshot Beispielanwendung

3.2.1. Angular-Module

Ein Angular-Modul fasst zusammengehörige Codeeinheiten zusammen und ermöglicht die Strukturierung einer Anwendung. Eine Angular-Anwendung kann dabei aus mehreren Angular-Modulen bestehen. Angular-Module sind nicht mit den in Unterabschnitt 2.3.1 vorgestellten JavaScript-Modulen zu verwechseln. (vgl. M. Steyer et al. 2017, S. 103 ff.; vgl. Woiwode et al. 2018, S. 301 ff.) Die Module einer

Angular-Anwendung können in Root-Module, Feature-Module und Shared-Module unterteilt werden.

Jede Angular-Anwendung hat genau ein Root-Modul. Dieses Modul konfiguriert die Angular-Anwendung. Wenn ein Client eine auf Angular basierte Seite anfordert, dann schickt der Server den Inhalt der *index.html* als Antwort an den Client zurück. Der Client führt daraufhin die im HTML-Dokument enthaltenen Skript-Elemente aus. Dabei wird die Angular-Plattform initialisiert und das Root-Modul übergeben. (vgl. M. Steyer et al. 2017, S. 60; vgl. Freeman 2018, S. 226 ff.) Das Root-Modul der Beispielanwendung ist das Modul *AppModule* siehe Listing 2.

Feature Module ermöglichen die Gruppierung einer Anwendung in Anwendungsfällen. Mithilfe von Shared-Module können die Teile einer Anwendung zusammengefasst werden, die unabhängig vom Anwendungsfall verwendet werden können. (vgl. Freeman 2018, S. 528 ff.; vgl. Google 2018a; vgl. M. Steyer et al. 2017, S. 105 ff.)

Module werden im Allgemeinen durch den Dekorator *@NgModule* gekennzeichnet. Ein Modul kann verschiedene weitere Module über die Eigenschaft *import* importieren und damit die bereitgestellten Funktionalitäten verwenden. Die vom Modul verwendeten Direktiven, Komponenten und Pipes werden in der Eigenschaft *declarations* angegeben. Jedes Root-Modul muss die Eigenschaft *bootstrap* führen. Diese Eigenschaft spezifiziert die Komponente, die beim Starten der Anwendung geladen werden soll.

Das Modul *AppModule* in Listing 2 importiert die Module *NgModule*, *BrowserModule* und *FormsModule* und deklariert die zugehörigen Komponenten *AppComponent* und *HelloComponent*. Beim Starten der Anwendung soll die Komponente *AppComponent* aufgerufen werden.

```
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4  import { AppComponent } from './app.component';
5  import { HelloComponent } from './hello.component';
6
7  @NgModule({
8    imports:      [ BrowserModule, FormsModule ],
9    declarations: [ AppComponent, HelloComponent ],
10   bootstrap:    [ AppComponent]
11 })
12 export class AppModule { }
```

Listing 2: Beispiel Angular-Anwendung: Das Root-Module

3.2.2. Angular-Komponenten

Komponenten sind Klassen, die Daten und Logik zur Anzeige in den zugehörigen Templates bereitstellen. Diese ermöglichen die Aufteilung einer Angular-Anwendung in logisch getrennte Teile. (vgl. Freeman 2018, S. 401)

Eine Komponente wird durch den Dekorator *@Component* gekennzeichnet und kann über verschiedene Dekorator-Eigenschaften konfiguriert werden. Die Eigenschaft *selector* identifiziert das HTML-Element, dass durch diese Komponente repräsentiert wird. Zur Anzeige der bereitgestellten Daten kann entweder ein Inline-Template *template* definiert oder auf ein externes Template *templateUrl* verwiesen werden. (vgl. Google 2018b; vgl. Freeman 2018, S. 405; vgl. M. Steyer et al. 2017, S. 47 ff.)

Die Beispielanwendung enthält die Komponenten *AppComponent* (siehe Listing 3) und *HelloComponent* (siehe Listing 4).

```
import { Component } from '@angular/core';

@Component({
selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: [ './app.component.css' ]
})
export class AppComponent  {
  name: string;
}
```

Listing 3: Beispiel Angular-Anwendung: Die Komponente AppComponent

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'hello',
  template: `<h1>Hello {{name}}!</h1>`,
  styles: [`h1 { font-family: Lato; }`]
})
export class HelloComponent  {
  @Input() name: string;
}
```

Listing 4: Beispiel Angular-Anwendung: Die Komponente HelloComponent

3.2.3. Templates

Zur Darstellung von Komponenten nutzt Angular Templates. Ein Template besteht aus HTML Code, der um Angular spezifische Konzepte wie Direktiven, Datenbindungsausdrücke und Pipes erweitert wird. (vgl. Google 2018b; vgl. M. Steyer et al. 2017, S. 52)

Mit Direktiven kann einem Element zusätzliches Verhalten hinzugefügt werden. (vgl. M. Steyer et al. 2017, S. 265; vgl. Freeman 2018, S. 401) In Angular werden folgende drei Arten von Direktiven unterschieden. (vgl. Google 2018d)

- Strukturelle Direktiven
- Attribut-Direktiven
- Komponenten

Die strukturellen Direktiven ändern die Struktur des zugehörigen HTML-Elements, indem sie HTML-Elemente hinzufügen oder entfernen. Hierfür verwenden die strukturellen Direktiven Templates, die beliebig oft gerendert werden. (vgl. M. Steyer et al. 2017, S. 269 ff.; vgl. Freeman 2018, S. 365) Beispiele für strukturellen Direktiven aus Angular sind (vgl. Freeman 2018, S. 261 ff.):

ngIf Fügt dem HTML-Dokument Inhalt hinzu, wenn die Bedingung wahr ist.

ngfor Fügt für jedes Item einer Datenquelle den gleichen Inhalt dem HTML-Dokument hinzu.

ngSwitch Fügt dem HTML-Dokument, abhängig vom Wert eines Ausdrucks, Inhalt hinzu.

Das Verhalten und das Aussehen des zugehörigen HTML-Elements kann durch die Attribut-Direktiven verändert werden. Diese Direktiven fügen oder entfernen dem zugehörigen HTML-Element Attribute. (vgl. ebd., S. 339) Beispiele für Attribut-Direktiven aus Angular-JS (vgl. ebd., S. 249 ff.):

ngStyle Mit dieser Direktive können unterschiedliche Style-Eigenschaften dem Element hinzugefügt werden.

ngClass Weißt dem Element ein oder mehrere Klassen hinzu.

Richtung	Syntax	Verwendung
One-way Komponente -> View	{{Ausdruck}} [Ziel]="Ausdruck"	Interpolation, Eigenschaft, Attribut, Klasse, Style
One-way Komponente <- View	(Ziel)="Ausdruck"	Events
Two-way	[(Ziel)]="Ausdruck"	Formular

Tabelle 1.: Arten von Data-Bindings in Angular

Mittels einer Komponente kann einem HTML-Element eine View hinzugefügt werden. Komponenten sind nämlich Direktiven mit einer eigenen View. (vgl. M. Steyer et al. 2017, S. 265)

Die von Angular bereitgestellten Direktiven (engl. Built-In Directives) können durch selbst entwickelte Direktiven erweitert werden. (vgl. Freeman 2018, S. 261)

Der Datenaustausch zwischen der Komponente und dem Template erfolgt durch Datenbindungsausdrücke. Ein Datenbindungsausdruck bindet einen JavaScript-Ausdruck an ein Ziel. Das Ziel kann entweder eine Attribut-Direktive oder eine Eigenschaft des zugehörigen HTML-Elements sein. Der JavaScript-Ausdruck ermöglicht den Zugriff auf die Eigenschaften und Methoden der Komponente. (vgl. Freeman 2018, S. 237 ff.; vgl. M. Steyer et al. 2017, S. 52 f.; vgl. Google 2018c) Es gibt insgesamt drei Arten von Data-Bindings, die anhand der Flussrichtung der Daten unterschieden werden können.

Mit Pipes können die Daten vor der Ausgabe sortiert, formatiert oder gefiltert werden. (vgl. M. Steyer et al. 2017, S. 83 ff.)

```
1 <hello name="{{name}}"></hello>
2 <label for="name">Name:</label>
3 <input name="name" [(ngModel)]="name">
```

Listing 5: Beispiel Angular-Anwendung: Das Template

3.2.4. Services

Laut Freeman (vgl. 2018, S. 474) kann jedes Objekt, dass durch Dependency Injection verwaltet und verteilt wird, als Service bezeichnet werden. Services stellen

wiederverwendbare Routinen oder Daten zur Verfügung, die von Direktiven, Komponenten, weiteren Services und Pipes verwendeten werden können. (vgl. Freeman 2018, S. 467 ff.; vgl. M. Steyer et al. 2017, S. 89)

Services verringern die Abhängigkeiten zwischen den Klassen, durch Dependency Injection. Hierdurch können Beispiel Unit-Tests einfacher durchgeführt werden. (vgl. Freeman 2018, S. 469)

Um einen Service zu verwenden, muss dieser entweder global in einem Modul oder lokal in einer Komponente registriert werden. Dies geschieht durch Einrichten eines Providers beim Modul oder der Komponente. Der Provider verknüpft ein Token mit einem Service. Ein Service kann eine Klasse, ein Wert, eine Funktion, eine Factory oder eine Weiterleitung sein. Aus diesem Grund gibt es unterschiedliche Provider.

Sobald ein Service global registrierte wurde, steht dieser auch in weiteren Modulen zur Verfügung. Ein lokal registrierter Service kann dahingegen nur von der jeweiligen Komponente und den direkten und indirekten Kindkomponenten verwendet werden.

Zur Nutzung eines Services muss die jeweilige Klasse den Service importieren und im Konstruktor deklarieren. Beim Erzeugen einer Instanz der Klasse injiziert Angular den jeweiligen Service.(vgl. M. Steyer et al. 2017, S. 92 ff.; vgl. Freeman 2018, S. 474 ff.)

4. ReactJS

Autor: Fabio Krämer

4.1. Allgemein

4.1.1. Einführung in das Framework

ReactJS ist eine von Facebook entwickelte JavaScript White-Box Framework zur Entwicklung von Benutzeroberflächen. Im Gegensatz zu Angular ist ReactJS ein reines View-Framework. Das Framework wird unter anderem bei Facebook, Instagram, Netflix, Airbnb und dem Content Management System Wordpress eingesetzt. Es bietet einige Vorteile bei der Entwicklung von Anwendungen mit großen Benutzeroberflächen mit Daten, die sich häufig verändern.

Das Framework ist in JavaScript geschrieben und kann mit JavaScript oder einer in JavaScript übersetzbare Sprache wie TypeScript verwendet werden.(vgl. Gackenheimer 2015, S. 1 ff.; vgl. Zeigermann et al. 2016, S. 3 ff.)

4.1.2. Vorbereitung der Entwicklungsumgebung

Für die Entwicklung einer React-Anwendung wird unter anderem *NodeJS* (siehe Unterabschnitt 3.1.2), ein Übersetzer (bspw. *Babel* siehe Unterabschnitt 2.3.3), ein Modul-Loader (bspw. *Webpack* siehe Unterabschnitt 2.3.1) und die React-Bibliothek benötigt. Die React-Bibliothek bestehendes aus *react* und *react-dom* kann über den Paketmanager *npm* installiert werden. (vgl. Stefanov 2017, S. 92 ff.; vgl. Zeigermann et al. 2016, S. 8 ff.)

4.2. Grundkonzepte

Zur Erklärung der Grundkonzepte wird das Hello Beispiel aus Abschnitt 3.2 herangezogen.

Das Framework React kann nur zum Bilden von Benutzeroberflächen verwendet werden. Darüber hinaus bietet es keine weiteren Funktionen. Für die Entwicklung einer größeren Anwendung muss beispielsweise eine Router-Implementierung eingebunden

und ein Architekturmodell definiert werden. Sowohl Zeigermann et al. (2016) als auch Stefanov (2017) empfehlen die Verwendung des Architekturmodells Flux und der Implementierung Redux. Flux beschreibt ein Anwendungsmodell, bei dem die Daten ausschließlich in eine Richtung fließen. (vgl. Zeigermann et al. 2016, S. 8; vgl. Zeigermann et al. 2016, S. 11; vgl. Stefanov 2017, S. 167)

Facebook bietet unter dem Namen React Native zudem eine Native Version von React an. Mit dieser können Apps für die unterschiedlichsten Plattformen entwickelt werden.(Facebook 2019)

4.2.1. Einbinden von React

Eine React-Anwendung benötigt ein Wurzelelement auf einer HTML-Seite. Unter diesem HTML-Element rendert React die Benutzeroberfläche der React-Anwendung. Das Rendern bezeichnet dabei den Vorgang, der schlussendlich zum Einfügen von HTML-Elementen im DOM führt. Dieser Vorgang wird in Unterabschnitt 4.2.3 näher betrachtet.

Die Benutzeroberfläche einer React-Anwendung wird durch React-Elemente repräsentiert. Ein React-Element ist ein JavaScript-Objekt. Es ist unveränderbar und kann nur durch Erzeugen eines neuen Elements aktualisiert werden.

Das Listing 6 zeigt die HTML-Seite der Beispielanwendung. Die Benutzeroberfläche der Beispiel React-Anwendung aus Listing 7 soll unter dem Element mit der ID *app* gerendert werden. Dies erfolgt durch Aufruf der Methode „ReactDOM.render() “in Listing 7. Diese veranlasst React zum Rendern des im ersten Parameter definierten React-Elements in das im zweiten Parameter angegebenen Wurzelelements. Das React-Element wird unter Verwendung der Spracherweiterung JSX (siehe Unterabschnitt 2.3.3) definiert.

```
<!DOCTYPE html>
<html>
  ...
  <body>
    <div id="app"></div>
    <script src="src/app.js"></script>
  </body>
  ...
</html>
```

Listing 6: Beispiel React-Anwendung: HTML-Datei

In diesem Beispiel wird der Methode die Komponente App, die ein React-Element zurückliefert, übergeben. Auf die Komponenten wird in Unterabschnitt 4.2.2 näher eingegangen. (vgl. Zeigermann et al. 2016, S. 4 ff., 26 ff.; vgl. Facebook 2018c; vgl. Facebook 2018a; vgl. Stefanov 2017, S. 2 ff.)

```
import "./styles.css";
import React from "react";
import ReactDom from "react-dom";
import Hello from "./hello";

class App extends React.Component {
  render() {
    return (
      <ErrorBoundary>
        <Hello name="World" />
      </ErrorBoundary>
    );
  }
}

class ErrorBoundary extends React.Component {
  ...
}

ReactDom.render(<App />, document.getElementById("app"));
```

Listing 7: Beispiel React-Anwendung: Aufruf der Render-Methode

4.2.2. React-Komponenten

Komponenten sind das zentrale Element in ReactJS. Sie enthalten sowohl die Logik als auch die zugehörige Anzeige. Eine Komponente kann entweder als Klasse oder als Funktion (engl. functional components) implementiert werden.

Eine Funktion, die eine React-Komponente implementiert, muss genau ein React-Element als Rückgabewert zurückgeben. Die Klasse muss von *React.Component* erben und die Methode *render()*, die ein React-Element zurückgibt, implementieren. Der Klassenname oder der Funktionsname entspricht dem Namen der Komponente.(vgl. Zeigermann et al. 2016, S. 80 ff.) Das Listing 8 zeigt eine Komponente Hello, die als Komponentenklasse implementiert ist.

```
import "./styles.css";
import React from "react";
import ReactDom from "react-dom";

export default class Hello extends React.Component {
  constructor(props) {
    super(props);
    this.state = { name: this.props.name };
  }

  onChange(event) {
    this.setState({ name: event.target.value });
  }

  render() {
    return (<div>
      <h1>Hello {this.state.name}</h1>
      <input onChange={e => this.onChange(e)} />
    </div>);
  }
}
```

Listing 8: Beispiel React-Anwendung: Die Komponente Hello

Durch Eigenschaften (engl. Properties) lässt sich das Aussehen und Verhalten einer Komponente von außen beeinflussen. Einer Komponente können Eigenschaften (engl. Properties) in Form eines Objektes übergeben werden. Das Propertie-Objekt kann von der Komponente nicht verändert werden.

Bei Komponentenfunktionen wird das Objekt der Funktion und bei Komponentenklassen dem Konstruktor der Klasse übergeben. Nach der Weitergabe des Objekts an die Oberklasse React.Component stehen die Properties über die Instanzvariable *props* zur Verfügung. (vgl. Zeigermann et al. 2016, S. 24 f., 83–88; vgl. Stefanov 2017, S. 12–17)

Die Komponentenklasse bietet gegenüber der Komponentenfunktion einige zusätzliche Funktionen. Eine Komponente, die als Klasse implementiert wird, hat einen Zustand (engl. State). Dieser wird in der Instanzvariable *state* gehalten und kann nur von der Komponente gelesen und verändert werden. Die Änderung des States sollte hauptsächlich über die Methode *setState()* erfolgen. Diese Methode erwartet ein Objekt mit KeyValue-Paaren oder eine Callback-Funktion. Durch den Aufruf der Methode wird der State mit dem bisherigen State zusammengeführt.(vgl. Zeigermann et al. 2016, S. 24 f., 89–93; vgl. Stefanov 2017, S. 17 f.)

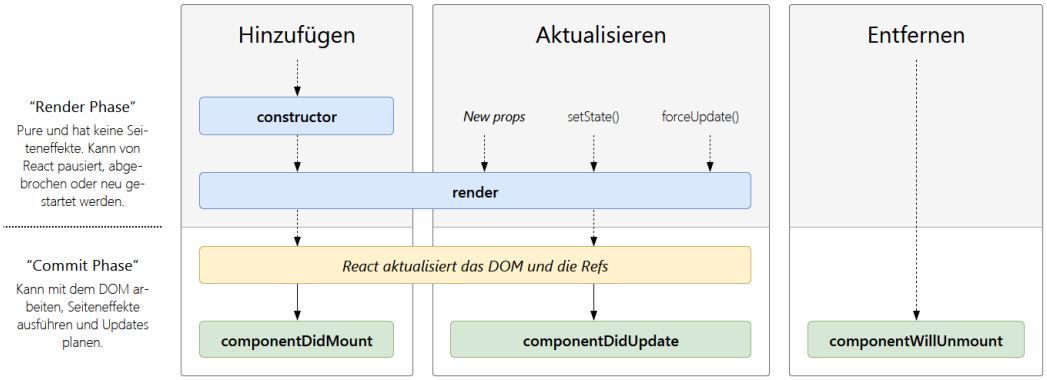

Abbildung 4.: Lebenszyklus einer React-Komponente
Quelle: Maj (2018)

Die Komponente Hello in Listing 8 verwendet einen über die Properties übergebenen Namen als initialen Wert für den im State gehaltenen Namen. Bei Änderungen im Inputfeld wird das Event OnChange ausgelöst, die Methode OnChange aufgerufen und der vom State gehaltene Namen über die Methode setState geändert. Der Aufruf der Methode führt zum erneuten Rendern der Anwendung. Der in der Überschrift angezeigte Name wird angepasst.

Eine Komponente hat einen gewissen Lebenszyklus. In diesen Zyklus kann bei Verwendung einer Komponentenklasse durch Überschreiben von Lebenszyklus-Methoden (engl. lifecycle-methods) eingegriffen werden. Die Abbildung 4 zeigt die gebräuchlichsten Lebenszyklus-Methoden einer Komponente.(vgl. Zeigermann et al. 2016, S. 96–100; vgl. Facebook 2018b)

In einer Komponente können weitere Komponenten eingebunden werden. Eine Komponente wird dabei durch ihren Namen identifiziert. Das Einbinden erfolgt durch Verwendung der Komponente in der Ausgabe. Dieser können Attribute über die Properties mitgegeben werden.(vgl. Zeigermann et al. 2016, S. 111 ff.) In diesem Beispiel wird die Komponente Hello aus Listing 8 und die Komponente ErrorBoundary aus Listing 7 in der Komponente App verwendet.

4.2.3. Rendern von Komponenten

Eine Änderung an den Properties oder am State einer Komponente veranlasst React zum erneuten Rendern. Beim Rendern wird das von einer Komponente bereitgestellte React-Element allerdings nicht direkt in ein DOM-Element umgewandelt und in das native DOM eingesetzt. An stattdessen wird beim Aufruf der Render-Methode eine virtuelle Repräsentation des DOM-Baums generiert.

Anschließend vergleicht React den virtuellen DOM-Baum vor und nach der Änderung. Auf Grundlage der ermittelten Unterschiede generiert React eine minimale Anzahl an DOM-Operationen. Diese Operationen werden dann auf den nativen DOM-Baum angewandt und stellen somit den gewünschten Zustand her. Das Rendern ist ein asynchroner Vorgang. Änderungen werden zum Teil zusammengefasst und nicht sofort angewandt.

React-Elemente sind einfache und leichtgewichtige JavaScript-Objekte. Daher können die Änderungen zwischen den zwei virtuellen DOM-Bäumen schnell ermittelt werden. Durch dieses Verfahren werden teure DOM-Manipulationen reduziert und damit die Performance der Web-Anwendung verbessert.(vgl. Zeigermann et al. 2016, S. 23 f., 60 f., 90 f.; vgl. Stefanov 2017, S. 53 f.; vgl. Facebook 2018c)

5. OpenUI5

Autor: Sebastian Greulich

5.1. Allgemeines

5.1.1. Einführung in das Framework

Das dritte, im Rahmen dieser Seminararbeit betrachtete clientseitige Webframework heißt OpenUI5. Es ist ein Open-Source-Projekt der SAP SE und wurde im Dezember 2013 unter der Apache-Lizenz 2.0 veröffentlicht. Diese Lizenz erlaubt das freie Verwenden, Verteilen und Modifizieren der Software, durch diese Eigenschaften ist sie mit der GNU General Public License 3 kompatibel. Das Pendant zu OpenUI5 ist SAPUI5, es ist die proprietäre Version von OpenUI5. SAPUI5 hat den gleichen Funktionsumfang wie OpenUI5 und wird von der SAP SE kostenfrei an ihre Kunden vertrieben. Der Vorteil von SAPUI5 ist ein umfangreicher Softwaresupport, welcher für OpenUI5 nicht verfügbar ist. Da man für den Einsatz von SAPUI5 SAP-Kunde sein muss wird dieses Produkt im Rahmen dieser Seminararbeit nicht weiter betrachtet.(vgl. Magnucki 2017, S. 8)

Das OpenUI5-Framework basiert auf HTML5, CSS3 und JavaScript (TypeScript ist nicht vorgesehen, da es in keiner offiziellen Dokumentation erwähnt wird). Die OpenUI5-Standardsteuerelemente sind laut Dokumentation mit folgenden Browserversionen kompatibel. Dazu zählen der Internet Explorer ab Version 9, Firefox ab Version 17 und Google Chrome ab Version 1. Beim Scripting hängt die Kompatibilität zum Browser an der jeweils verwendeten ECMAScript-Version. Es wird kein offizielles Tool zum Transpilieren bereitgestellt. (vgl. SAP 2013, S. 6)

5.1.2. Auswahl der Entwicklungsumgebung

Prinzipiell kann für die Entwicklung von OpenUI5-Anwendungen jeder beliebige Texteditor verwendet werden. Für größere Webentwicklungsprojekte wird jedoch auf der OpenUI5-Dokumentationshomepage die Entwicklungsumgebung Eclipse

empfohlen. Für diese wurde von der SAP das „SAPUI5 Application Development“-AddOn entwickelt. In diesem AddOn sind alle notwendigen Bibliotheken zur OpenUI5-Entwicklung enthalten. Darüber hinaus dient es als Assistent zur Erstellung neuer UI5-Entwicklungsprojekte. Der Assistent erstellt nach seinem Durchlauf eine angepasste Dateistruktur.(vgl. Antolovic 2014, S. 109 ff.)

5.1.3. Typische Einsatzgebiete

OpenUI5 besitzt eine sehr umfassende und übersichtliche Dokumentation unter `www.openui5.hana.ondemand.com/#/api`. Diese umfasst viele Anwendungsbeispiele aus der Praxis. Hierbei ist es möglich, den verwendeten Code zu analysieren und in der eigenen Anwendung wiederzuverwenden. OpenUI5 wurde von der SAP entwickelt um die veralteten SAP ERP-Transaktionen zu ersetzen. In diesem Umfeld hat OpenUI5 auch die größten Stärken. Beispielsweise ist es sehr komfortabel möglich Single-Screen-Geschäftsanwendungen zu erstellen. Diese Anwendungen haben ihren Fokus zumeist auf dem Anzeigen, Pflegen und Auswerten von Daten aus einer Datenbank. Jedoch ist es aufgrund der Verwendung von HTML5 auch möglich, verschiedenste multimediale Inhalte wie beispielsweise Audio oder Video wiederzugeben. Einen weiteren Vorteil von OpenUI5 stellt die Responiveness der Oberfläche dar. Das Framework beinhaltet Funktionen, welche dafür sorgen, dass die Weboberfläche auf jeder Geräteklasse optimal dargestellt wird. Bei der Datenanbindung beherrscht OpenUI5 unter anderem die OData-Technologie, diese ermöglicht es Daten im JSON-Format asynchron über AJAX zu übertragen. Das auf HTTP basierende OData-Protokoll ist ein von Microsoft entwickelter offener Standard zur Datenübertragung. Der letzte Vorteil von OpenUI5 ist die mögliche Verwendung des MVC-Modells. Dieses vereinfacht die Übersichtlichkeit und Wartbarkeit der Webanwendung.(vgl. Magnucki 2017, S. 26 ff.)

5.2. Architektur

5.2.1. Einbinden von OpenUI5

In diesem Abschnitt wird beschrieben, wie OpenUI5 ohne die, vom Assistenten generierten MVC-Dateifragmente initialisiert. Allgemein ist OpenUI5, wie in den Technischen Grundlagen in Unterabschnitt 2.1.2 beschrieben, ein clientseitiges JavaScript Black-Box-Framework. Dies bedeutet, man kann das Framework mittels

<script>-Befehl in ein HTML-Dokument importieren. Hierbei können wichtige Interfaceparameter übergeben werden, wie beispielsweise das zu verwendende Theme. Dies kann man in Listing 9 anschaulich sehen.

```
<!DOCTYPE html>
  <html>
  ...
    <head>
      <script id="sap-ui-bootstrap"
        src=https://openui5.hana.ondemand.com/resources/sap-ui-core
            .js
        data-sap-ui-theme="sap_belize">
      </script>
    </head>
  ...
  </html>
```

Listing 9: Beispiel für das Einbinden von OpenUI5

Nachdem die Bibliothek in das Dokument integriert wurde, kann man in einem JavaScript-Teil des HTML-Dokumentes verschiedene UI-Elemente generieren und diese auf der Webseite anzeigen lassen. Dieses Verfahren eignet sich jedoch aufgrund der schlechten Strukturierungsmöglichkeiten nur für sehr kleine und kompakte Webanwendungen mit geringem Funktionsumfang.(vgl. Antolovic 2014, S. 136 ff.)

5.2.2. Dateifragmente

Wie im vorherigen Unterabschnitt 5.2.1 erwähnt, existiert für OpenUI5 ein AddOn, welches beim Anlegen des Projektes alle wichtigen Bibliotheken und eine MVC-Dateistruktur erstellt. Die wichtigsten Dateielemente dieser Struktur lauten:

- Index.html
- Component.js
- View.xml
- Controller.js

Auf die Funktion und Struktur der Elemente wird im Verlauf dieses Kapitels detailliert eingegangen. Hierbei ist wichtig, dass die beschriebenen JavaScript- und XML-Dateien aus mehreren Komponenten bestehen und auch anders benannt werden können. Die folgenden Dateien werden lediglich als Beispiele für eine empfohlene Struktur herangezogen.

Die index.html-Datei ist der erste Schritt für den Browser, beim Öffnen einer Webseite. Sie ist zwingend nötig und darf nicht umbenannt oder gelöscht werden. In der Datei index.html werden allgemeine Parameter wie beispielsweise Dateipfade oder Kompatibilitätsinformationen festgelegt. Darüber hinaus wird von ihr auf die anderen Bestandteile der Webanwendung verwiesen.

In der Component.js werden alle Datenverbindungen beschrieben. Hierzu gehören beispielsweise einfach eingebundene JSON-Dateien oder auch Verbindungen zu einem ODATA-Webservice, welcher dem UI5-Frontend Daten zur Anzeige bereitstellt. Hierbei kann OpenUI5 im Standard sowohl JSON-, als auch mit XML-Dateien verarbeiten. Darüber hinaus wird in der Component.js festgelegt, welche Art der Datenbindung zum Webservice besteht. Möglich ist hierbei nur das bloße Anzeigen (unidirektionale Bindung), aber zusätzlich auch das Verändern auf der Datenbank (bidirektionale Bindung).

Die View.xml ist von der Struktur eine XML-Datei. Wie ihr Name schon aussagt, wird in ihr die visuelle Anzeige beschrieben. Die XML-Datei eignet sich in ihrem Aufbau sehr gut, eine grafische Oberfläche mit vorgegebenen Steuerungselementen zu strukturieren. Denn in ihr kann man verschiedene Anzeigeelemente klar erkennen und voneinander trennen. Dazu gehören zum Beispiel Header, Footer oder einzelne Container mit verschiedenem Inhalt. Mittels, im OpenUI5-Standard enthaltenen, Router ist es möglich, verschiedene Views miteinander zu verbinden und somit eine Webanwendung mit mehreren Unterseiten zu schaffen. Die View in UI5 ermöglicht darüber hinaus auch die einfache Datenbindung zur Datenbank ohne zusätzliche Programmierung im Controller. Mit dem folgenden Listing 10, einer Produktliste, wird die große Stärke von OpenUI5 nochmals unterstrichen.

```
<List
id="ProductList"
items="{/ProductCollection}
includeItemInSelection="true"">
  <StandardListItem
  title="{Name}"
  description="{ProductId}"
  icon="{ProductPicUrl}"/>
</List>
```

Listing 10: Beispiel OpenUI5 View

Das letzte wichtige OpenUI5-Fragment stellt die Datei Controller.js dar. Sie enthält die programmspezifische Logik der UI5-Webanwendung. In ihr können mittels JavaScript die gewünschten Funktionen ausprogrammiert werden. Eine typische Funktion stellt beispielsweise, das Öffnen eines Dialogfensters beim Drücken eines zuvor in der View definierten Buttons dar. Somit fügt der Controller der einfachen View Funktionen, welche über einfache Tabellenanzeigen oder -änderungen hinausgehen hinzu. Im Listing 11 wird ein Szenario veranschaulicht, bei welchem auf Knopfdruck ein MessageToast mit Blindtext erscheint. Es ist darüber hinaus auch möglich, mehrere Controller zu verwenden und in diese weitere Frameworks zur Benutzung zu laden. Als mögliches Anwendungsbeispiel wäre hierbei das Einbinden eines Barcodescannerframeworks (zum Beispiel quagga.js) um mit einer Handykamera industrielle Barcodes zu lesen.

```
1 handleMessageToastPress: function(oEvent) {
2   var msg = 'Lorem ipsum dolor sit amet, consetetur sadipscing
        elitr, sed diam nonumy\r\n eirmod.';
3 MessageToast.show(msg);
4 }
```

Listing 11: Beispiel OpenUI5 Controller

6. Projektbezogener Vergleich der Webframeworks

Autor: Sebastian Greulich

6.1. Vergleichende Betrachtung

Nachdem in den vorherigen drei Kapiteln die einzelnen Technologien detailliert beschrieben wurden, werden sie nun im folgenden Kapitel miteinander verglichen.

Das erste Vergleichskriterium bildet der Umfang an Funktionen, welche direkt im Framework integriert sind. Alle drei Webframeworks erlauben es, Daten komfortabel auf der Weboberfläche anzeigen. Jedoch bieten nur Angular und OpenUI5 die Möglichkeit im Standard, eine Datenquelle mittels „Two-Way-Binding“ anzubinden(siehe Unterabschnitt 3.2.3 und Unterabschnitt 5.2.2). Bei React muss hierfür eine zusätzliche Komponente nachinstalliert werden(siehe Unterabschnitt 4.2.1). Ähnlich verhält es sich bei dem Routing zwischen mehreren Seiten der Anwendung. Angular und OpenUI5 liefern Möglichkeiten hierzu in ihrem Standard dazu aus, jedoch bei React muss hierfür wiederum eine zusätzliche Router-Implementierung eingebunden werden(siehe Unterabschnitt 4.1.2 und (vgl. Zeigermann et al. 2016, S. 8)).

Es gibt darüber hinaus weitere Unterschiede bei der empfohlenen Webarchitektur. Bei Angular und OpenUI5 wird man aufgrund des Frameworkaufbaus zur Verwendung des MVC-Modells gedrängt. Dies wird bei OpenUI5 deutlich, indem schon beim Einrichtungsassistenten die typische MVC-Struktur angelegt wird(siehe Unterabschnitt 5.2.2). Auch das Grundkonzept von Angular ist auf MVC ausgerichtet(siehe Unterabschnitt 3.1.1). Anders verhält es sich allerdings bei React, denn hier wird die Verwendung von Flux empfohlen, bei dem typischerweise die Daten nur in eine Richtung fließen(siehe Unterabschnitt 4.1.2). Dies generiert eine schnelle Performance bei React.

Ein weiteres wichtiges Merkmal bei Webframeworks ist die Datenanzeige. Bei OpenUI5 ist die Anbindung an einen Webserver über OData sehr einfach gehalten. Mit nur ein paar wenigen Codezeilen ist es möglich, Daten anzuzeigen und diese auch bei Bedarf zu ändern(siehe Listing 10). Jedoch ist man bei den Anzeigemöglichkeiten, welche einfach einzubinden sind, auf den von SAP entwickelten Standard beschränkt. Will man Daten anderweitig anzeigen muss man mit großem Aufwand andere Frameworks einbinden und den Datenaustausch mit diesen manuell implementieren. Bei

Angular wird die Datenanzeige über Komponenten gesteuert, welche die Templates mit Inhalt füllen(siehe Unterabschnitt 3.2.3). Bei Angular sind oberflächenspezifische Anpassungen komfortabler und schneller möglich als bei OpenUI5, da diese direkt im CSS vorgenommen werden können. React hat bei der Datenanzeige klare Geschwindigkeitsvorteile im Vergleich zu OpenUI5 und Angular, da das Framework die Anzahl der DOM-Manipulationen auf ein Minimum reduziert(siehe Unterabschnitt 4.2.3). Daraus folgt, dass der Webbrowser weniger Aktionen ausführen muss. Im Vergleich zu Angular und React wird bei OpenUI5 das Design der Steuerelement vorgegeben und kann nur mit erheblichem Aufwand auf eigne Bedürfnisse angepasst werden(siehe Unterabschnitt 5.2.2).

Bei der Wahl der Entwicklungsumgebung ist man bei Angular und React weitestgehend frei(siehe Unterabschnitt 3.1.2 und 4.1.2). Man sollte jedoch bei beiden darauf achten, dass die Entwicklung in TypeScript unterstützt wird, falls man diese Programmiersprache verwendet. Bei OpenUI5 ist man in dieser Hinsicht beschränkter. Prinzipiell ist jeder Texteditor dafür geeignet, jedoch werden in Eclipse spezielle AddOns angeboten, welche das Erstellen und Entwickeln an einer Anwendung erleichtern(siehe Unterabschnitt 5.1.2).

Es gibt darüber hinaus auch Unterschiede bei den Wahlmöglichkeiten der Programmiersprache. Wie im Unterabschnitt 2.3.2 erläutert bietet die von JavaScript abgeleitete Programmiersprache TypeScript zahlreiche Vorteile gegenüber ihrem Ursprung. Die Verwendung von TypeScript ist sowohl in Angular, als auch in React möglich(siehe Unterabschnitt 3.1.1 und 4.1.1). Somit können mögliche Syntaxfehler schon beim Kompilieren erkannt werden und die Kompatibilität mit veralteten Webbrowsern sichergestellt werden. Bei OpenUI5 ist TypeScript zur Entwicklung nicht vorgesehen, deshalb wird hierbei das Entwickeln nur in der Webgrundsprache JavaScript empfohlen(siehe Unterabschnitt 5.1.1). Diese wird darüber hinaus auch von Angular sowie React unterstützt.

Ein weiteres Unterscheidungsmerkmal ist die Lizenzierung. Offiziell stehen alle drei zu vergleichenden Webframeworks unter Open-Source-Lizenzen. Jedoch werden lediglich die Frameworks Angular und React von einer breiten Entwicklergemeinde vorangetrieben. Bei OpenUI5 entwickelt hauptsächlich die SAP an Verbesserungen des Frameworks. Dies liegt daran, dass UI5 hauptsächlich in Unternehmen eingesetzt wird und diese ihre Entwicklungen nur selten veröffentlichen.

Der letzte Unterschied zwischen den Webframeworks ist, dass sowohl Angular, als auch React native Versionen ihres Frameworks anbieten. Somit ist es möglich, mit ihnen native Apps für Android zu entwickeln. Bei OpenUI5 ist diese Möglichkeit nicht vorgesehen.

6.2. Handlungsempfehlung

In diesem Abschnitt werden nun nach dem Vergleich aus dem vorherigen Abschnitt 6.1 konkrete Empfehlungen zur Umsetzung in „business horizon“ ausgesprochen.

Um eine fundierte Empfehlung für eine Webtechnologie geben zu können muss zunächst die Anwendung auf ihre frontendrelevanten Eigenschaften untersucht werden. Nach einer umfassenden Analyse lassen sich einige Merkmale der Software erkennen. Sie umfasst viele unterschiedliche Komponenten und hat darüber hinaus noch einen Anmeldebildschirm. Daraus folgt eine Vielzahl von Views, zwischen denen gewechselt werden muss. Beim Routing hat sowohl Angular, als auch OpenUI5 seine Stärken. In React muss hierfür eine zusätzliche Komponente nachinstalliert werden. In „business horizon“ gibt es ausschließlich unidirektionale Webserveranbindungen. OpenUI5 hätte bei bidirektionalen Bindungen seine Vorzüge, da es diese allerdings nur selten benötigt werden sind alle betrachteten Webframeworks in diesem Punkt gleichwertig. Die Anwendung umfasst darüber hinaus viele nicht standardisiert aufgebaute Dialoge. Deshalb sollte das zu verwendende Framework viele Anpassungsmöglichkeiten der Anzeige bieten. Diese Möglichkeiten bieten Angular und React, bei OpenUI5 sind diese mit einem erheblichen Mehraufwand verbunden. Die von der SAP standardisierten Steuermöglichkeiten sind darüber hinaus nicht so eingängig als selbstentwickelte bei Angular und React. Denn beispielsweise gibt SAP im Standard lediglich neun Buttontypen vor(vgl. SAP 2019). Einen weiteren Vorteil für die Frameworks Angular und React stellt die breite Entwicklergemeinde dar. Denn es existieren zahlreiche Foren, in welchen über aufkommende Probleme beraten wird. UI5 wird zum Großteil von Unternehmen verwendet, falls diese auf etwaige Probleme stoßen wenden sie sich an den Support der SAP und teilen die Informationen nicht öffentlich im Internet. Somit würde man mit Angular und React während der Entwicklung einfacher zu Hilfen gelangen. Ein weiteres Argument für Angular und React stellen vorhandene native Versionen dar. Falls im weiteren Projektverlauf eine App benötigt wird kann man auf diese Frameworks zurückgreifen. Einen Nachteil für OpenUI5 stellen Komponenten dar, welche sich vom typischen HTML unterscheiden. Als Beispiel hierfür kann man die View heranziehen, welche in XML verfasst wird. Somit würde dies eine erhebliche Einstiegshürde für die Projektmitglieder bedeuten. Eine weitere Einstiegshürde für das Projektteam wäre auch die Verwendung des Architekturmodells Flux, denn bisher wurde in den Vorlesungen ausschließlich das MVC-Modell gelehrt. Das gewichtigste Argument für die Verwendung des Webframeworks Angular stellt das Faktum dar, dass bisher in der kompletten Anwendung „business horizon“ Angular verwendet wurde. Eine Änderung des Frameworks würde einen immensen Aufwand bedeuten. Bestärkt wird dieses Argument zusätzlich, da bei Angular bisher keinerlei Probleme aufgetreten sind.

Aufgrund der zuvor erörterten Argumente eignet sich Angular im Gesamten als Webfrontend für „business horizon“ am Besten. Es wird zudem empfohlen, dass der bestehende Code belassen wird, außer er bedarf einer schnellen Datenänderung, welche den Mehraufwand zur Implementierung von React rechtfertigt. Jedoch ist die Hürde, der Verwendung von React, bei Erweiterungen der Anwendung nicht so hoch und kann im Einzelfall in Betracht gezogen werden. OpenUI5 kommt für „business horizon“ nicht in Frage, da die zuvor erläuterten Vorteile nicht die Einstiegshürde für die Projektmitglieder rechtfertigt.

7. Fazit

Autor: Sebastian Greulich

Im Zuge dieser Seminararbeit sollten die Webfrontendtechnologien Angular, ReactJS sowie OpenUI5 mit dem Ziel analysiert und verglichen werden, um zu überprüfen, ob Angular das passendste Framework für die Anwendung „business horizon“ darstellt.

In der Analyse und dem Vergleich konnten verschiedenste Stärken und Schwächen der einzelnen Technologien erarbeitet werden. Es wurde festgestellt, dass sich Angular insgesamt am besten für die Anwendung eignet. Jedoch hat das Framework React Geschwindigkeitsvorteile, bei schnellen Datenänderungen auf der Oberfläche. Deshalb lautet die Empfehlung, dass der vorhandene Code belassen wird, außer die Geschwindigkeitsvorteile von React rechtfertigen den resultierenden Mehraufwand durch das Umschreiben des Codes. Falls jedoch bei einer Erweiterung von „business horizon“ häufige Anzeigedatenänderungen auftreten wird die Entwicklung in React empfohlen.

Somit wurde das Ziel dieser Seminararbeit erfüllt und mit der Weiterverwendung von Angular eine fundierte Handlungsempfehlung ausgesprochen.

A. Anhang

A.1. Beigabenverzeichnis

⊢ **Seminararbeit/**

| ⊢ **Latex-Files/** ⇒ *LaTeX Dateien*

| ⊢ **ads/** ⇒ *Deckblatt und Anhang*

| ⊢ **content/** ⇒ *Hauptteil*

| ⊢ **images/** ⇒ *alle verwendeten Dateien*

| ⊢ **lang/** ⇒ *Sprachdateien für LaTeX template*

| – WWI16_GREULICH_KRAEMER_Clientseitige Webframeworks.pdf

|

⊢ **Elektronische Quellen/**

– Facebook (Hrsg) 2018a, React - Add React to a Website.pdf

– Facebook (Hrsg) 2018b, React - ReactComponent.pdf

– Facebook (Hrsg) 2018c, React - Rendering Elements.pdf

– Facebook (Hrsg) 2019, React Native.pdf

– Google (Hrsg) 2018a, Angular - Attribute Directives.pdf

– Google (Hrsg) 2018b, Angular - Introduction to components.pdf

– Google (Hrsg) 2018c, Angular - Introduction to moduls.pdf

– Google (Hrsg) 2018d, Angular - Template Syntax.pdf

– Hlushko, O'Brien et al 2018, Webpack.pdf

– Magnucki (Hrsg) 2017, App Entwicklung mit SAPUI5 und OData.pdf

– SAP (Hrsg) 2013, Getting Started with SAPUI5.pdf

– SAP (Hrsg) 2019, OpenUI5 API Reference.pdf

Literatur

Antolovic, Miroslav (2014). *Einführung in SAPUI5*. 1. Aufl. SAP PRESS. Bonn: Galileo Press.

Facebook, Hrsg. (2018a). *React: Add React to a Website*. URL: https://reactjs.org/docs/add-react-to-a-website.html (besucht am 29. 12. 2018).

– Hrsg. (2018b). *React: React.Component*. URL: https://reactjs.org/docs/react-component.html (besucht am 28. 12. 2018).

– Hrsg. (2018c). *React: Rendering Elements*. URL: https://reactjs.org/docs/rendering-elements.html (besucht am 29. 12. 2018).

– Hrsg. (2019). *React Native: A framework for building native apps using React*. URL: https://facebook.github.io/react-native/ (besucht am 05. 01. 2019).

Fink, Gil (2014). *Pro Single Page Application Development : Using Backbone.js and ASP.NET*. Hrsg. von Ido Flatow. SpringerLink : Bücher. Berkeley, CA: Apress.

Freeman, Adam (2018). *Pro Angular 6*. 3. Aufl. Berkeley, CA: Apress.

Gackenheimer, Cory (2015). *Introduction to React*. The expert's voice in web development. New York, New York: Apress.

Google, Hrsg. (2018a). *Angular: Attribute Directives*. URL: https://angular.io/guide/attribute-directives (besucht am 12. 12. 2018).

– Hrsg. (2018b). *Angular: Introduction to components*. URL: https://angular.io/guide/architecture-components (besucht am 13. 12. 2018).

– Hrsg. (2018c). *Angular: Introduction to modules*. URL: https://angular.io/guide/architecture-modules (besucht am 15. 12. 2018).

– Hrsg. (2018d). *Angular: Template Syntax*. URL: https://angular.io/guide/template-syntax (besucht am 17. 12. 2018).

Hlushko, Eugene et al. (2018). *Webpack: Why webpack*. URL: https://webpack.js.org/concepts/why-webpack/ (besucht am 30. 12. 2018).

Jäger, Kai, Hrsg. (2008). *Ajax in der Praxis : Grundlagen, Konzepte, Lösungen*. Xpert.pressSpringerLink : Bücher. Berlin, Heidelberg: Springer Berlin Heidelberg.

Kojarski, Sergei und David H. Lorenz (2006). „Modeling Aspect Mechanisms: A Top-down Approach“. In: *Proceedings of the 28th International Conference on Software Engineering*. ICSE '06. Shanghai, China: ACM, S. 212–221.

Magnucki, Rico, Hrsg. (2017). *App-Entwicklung mit SAPUI5 und OData*. URL: https://mission-mobile.de/download/e-book-app-entwicklung/ (besucht am 05. 01. 2019).

Maj, Wojciech (2018). *React Lifecycle Methods diagram*. URL: http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/ (besucht am 28. 12. 2018).

SAP, Hrsg. (2013). *Getting Started with SAPUI5*. URL: https://www.selflearningsap.com/SAPUI5.pdf (besucht am 05. 01. 2019).

– Hrsg. (2019). *OpenUI5 API Reference*. URL: https://openui5.hana.ondemand.com/#/api/sap.m.ButtonType/properties (besucht am 05. 01. 2019).

Schatten, Alexander (2010). *Best Practice Software-Engineering : Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen*. Hrsg. von Stefan Biffl et al. SpringerLink : Bücher. Heidelberg: Spektrum Akademischer Verlag.

Stefanov, Stoyan (2017). *Durchstarten mit React: Web-Apps einfach und modular entwickeln*. 1. Aufl. Heidelberg: dpunkt.verlag.

Steyer, Manfred und Daniel Schwab (2017). *Angular: Das Praxisbuch zu Grundlagen und Best Practices*. 2. Aufl. Heidelberg: O'Reilly.

Steyer, Ralph (2017). *Webanwendungen mit ASP.NET MVC und Razor : Ein kompakter und praxisnaher Einstieg*. SpringerLink : Bücher. Wiesbaden: Springer Vieweg.

Terlson, Brian, Hrsg. (2018). *ECMAScript: 2018 Language Specification*. URL: https://www.ecma-international.org/ecma-262/9.0/ (besucht am 26. 12. 2018).

Woiwode, Gregor et al. (2018). *Angular: Grundlagen, fortgeschrittene Techniken und Best Practices mit TypeScript - ab Angular 4, inklusive NativeScript und Redux*. 1. Aufl. ix edition. Heidelberg: dpunkt.verlag.

Zeigermann, Oliver und Nils Hartmann (2016). *React: Die praktische Einführung in React, React Router und Redux*. 1. Aufl. Heidelberg: dpunkt.