PDV-SYSTEME

FACHHOCHSCHULE ERFURT UNIVERSITY OF APPLIED SCIENCES

Evaluation moderner Webtechnologien für die Entwicklung eines modularen Supportportals

Master Projekt

für die Prüfung zum

Master of Science

des Studienganges Angewandte Informatik

an der

Fachhochschule Erfurt

von

Sebastian Rieger

Abgabedatum 01.03.2017

Bearbeitungszeitraum	24 Wochen
Matrikelnummer	10286908
Ausbildungsfirma	PDV Systeme
	Erfurt
Betreuer der Ausbildungsfirma	Dipl. -Inform. FH Nico Kaiser
Gutachter der Fachhochschule	Prof. Rolf Kruse

Erklärung

Ich, Sebastian Rieger, versichere hiermit, dass ich die vorliegende Bachelorarbeit mit dem Thema

Evaluation moderner Webtechnologien für die Entwicklung eines modularen Supportportals

selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Ort Datum

Unterschrift

Inhaltsverzeichnis

1 Einleitung

In einer vernetzen Welt wie unserer, werden unablässig neue und bessere Web-Technologien entwickelt. Diese neuen Technologien bringen zum einen eine bessere Programmierfreundlichkeit mit sich, aber sie sind zum anderen durch neue Ansätze auch schneller als ältere Technologien.

Heutige Webanwendungen müssen eine Vielzahl an kriterien erfüllen. Eine gute Webseite sollte heute möglichst auf einen Handy, sowie auf einem Desktop-PC oder Fernseher darstellbar sein. Um dies zu ermöglichen, müssen besondere Anforderungen an das Design gestellt werden. Natürlich muss sie auch von möglichst allen Browsern anzeigbar sein. Ein weitere Punkt ist, dass Webseiten sich heute so flüssig und performant wie ein natives Programm verhalten sollen. Hierbei ist es Natürlich auch wichtig vorhande Hardware anzusprechen und mit ihr zu interagieren.

Das Ziel dieser Arbeit ist es herauszufinden, wie unter zu Hilfenahme moderner Technologien alle diese kriterien möglichst gut erfüllt werden können. Es sollen Programmieransätze wie Google Polymer, AngularJS, HTML5, CSS3, PHP7 und Google Dart unter dem Portalserver vereint werden, welcher die Webseite beziehungsweise die Anwendung bereitstellt.

Es soll geprüft werden, wie und ob es möglich ist diese verschiedenen Technoliegen in möglichst modularen Portalserver-Erweiterungen unterzubringen, um eine möglichst performate und zu gleich leistungsstarke Webseite zu entwickeln. Dieses Oxymoron aufzulösen ist der Schwerpunkt der vorliegenden Arbeit, obwohl doch schon jetzt ersichtlich ist, dass ein auf beiden Seiten ein Kompromiss gefunden werden muss. Wie dieser genau aussieht, wird im weiteren Verlauf eine wichtige Rolle spielen.

Diese Arbeit soll der theoretischen Grundstock für eine weitere Arbeit sein, in der das Support-Portal der PDV System Erfurt GmbH neu entwickelt wird. Im Verlauf dieser Arbeit, wird evaluiert werden, welcher der gängigen Portalserver für ein Supportportal am besten geeignet ist. Außerdem wird geprüft werden, wie es möglich ist, modulare Erweiterungen für einen solchen Server zu entwickeln, um unnötige Datenredundanz zu vermeiden.

Durch einen modularen Aufbau, soll außerdem die Kommunikation zwischen verschieden Erweiterungen vereinfacht werden. Des weiteren soll es möglich werden, nur beliebige Erweiterungen zu verwenden, wenn für den jeweiligen Use-Case niccht alle benötigt werden.

In den nun folgenden Abschnitten werden Progammierbeispiele und Hinweise gegeben, wie eine solche Neuentwicklung unter den Gesichtspunkten Performance, Umsetzbarkeit und Usability vorgenommen werden kann. Um Mitarbeitern und Kunden ein möglichst performantes Support-portal bieten zu können, welches den Arbeitsalltag eines jeden Nutzers erleichert.

2 Portalserver/ CMS-Systeme im Vergleich

Das zukünftige Supportportal der PDV Systeme GmbH soll auf Basis eines Portalservers bzw. *Content Managemente System* (CMS)-Servers aufgebaut werden. Hierfür werden im folgenden einige Möglichkeiten genauer betrachtet.

Ein Portalserver, welcher für das Projekt heran gezogen wird muss die folgenden Eigenschaften aufweisen.

- Webseiten müssen frei gestalltbar sein
- Es muss die Möglichkeit bestehen Anwendungen für den Server zu entwickeln
- Das System muss Open-Source sein, um ggf. in den Quellcode eingreifen zu können
- Die Nutzer-Community sollte möglichst groß sein, damit Probleme leicht diskutiert und behoben werden können
- Der Server muss die Möglichkeit bieten Dateien zu verwalten, welche als Download oder Kontent in das Portal einfließen
- Lauffähing unter einer SQL-Datenbank wie MySQL oder MariaDB

Auf die Betrachtung reiner CMS-System wird an dieser Stelle verzichtet, da diese nicht die gewünschten Anforderungen eines Portalservers erfüllen. Ein Vergleich verschiedener reiner CMS-Systeme ist in der Bachelorarbeit „Konzept und prototypische Implementierung eines übergreifenden Dokumenten- und Medienmanagements“ zu finden. [Rie15]

2.1 Typo3

Typo3 ist ein Verwaltungssystem für Internetseiten. Es basiert in der neusten Version 8.2 auf der PHP Version 7. Seit der Version 7.0 welche zugleich eine *Long Term Support* (LTS) Version ist, wird es unter dem Namen Typo3 CMS vertrieben. [Wik16a]

Es ist möglich unter Verwendung von PHP, Extbase und Fluid eigene Erweiterungen für den Portalserver zu entwickeln. Die jeweiligen Erweiterungen wirken im Fronten von Typo3 wie ganz normale Webseiten. Auf Extbase und Fluid wird im Kapitel 3 genauer eingegangen.

Ein Vorteil von Typo3 ist, dass es eines der am häufigsten verbreiteten Portalserver auf dem Markt ist. Selbst große Firmen wie „Sixt“ setzten auf Typo3 bei der Erstellung ihrer Internetportale. Durch die große Verbreitung von Typo3 ist auch die Community um die Software sehr groß und man findest schell zu fast jedem Problem im Internet eine Lösung.

Typo3 kann im Zusammenspiel mit MySQL, MariaDB, PostgreSQL oder Oracle als Datenbank betrieben werden. Nur mit Hilfe einer dieser Datenbank im Hintergrund ist Typo3 stabil Lauffähing und kann produktiv eingesetzt werden.

Durch ein übersichtliches Backend (siehe Abbildung 1), ist es auf für Laien möglich qualitativ Hochwertige Webseiten zu erstellen, ohne viel Kenntnis von Webtechnologien wie HTML oder CSS zu haben.

Ein Upload von Dateien für den Download beziehungsweise als Seiten-Kontent ist ebenfalls unter Typo3 möglich. Zusätzlich dazu ist es möglich das Nutzer Dateien auf den hochladen. Hierdurch entsteht auch die Möglichkeit ein Austauschportal für Dateien zu schaffen. Nutzer können so in einer sicheren Umgebung sensible Daten untereinander oder mit Mitarbeitern austauschen. [Lob16]

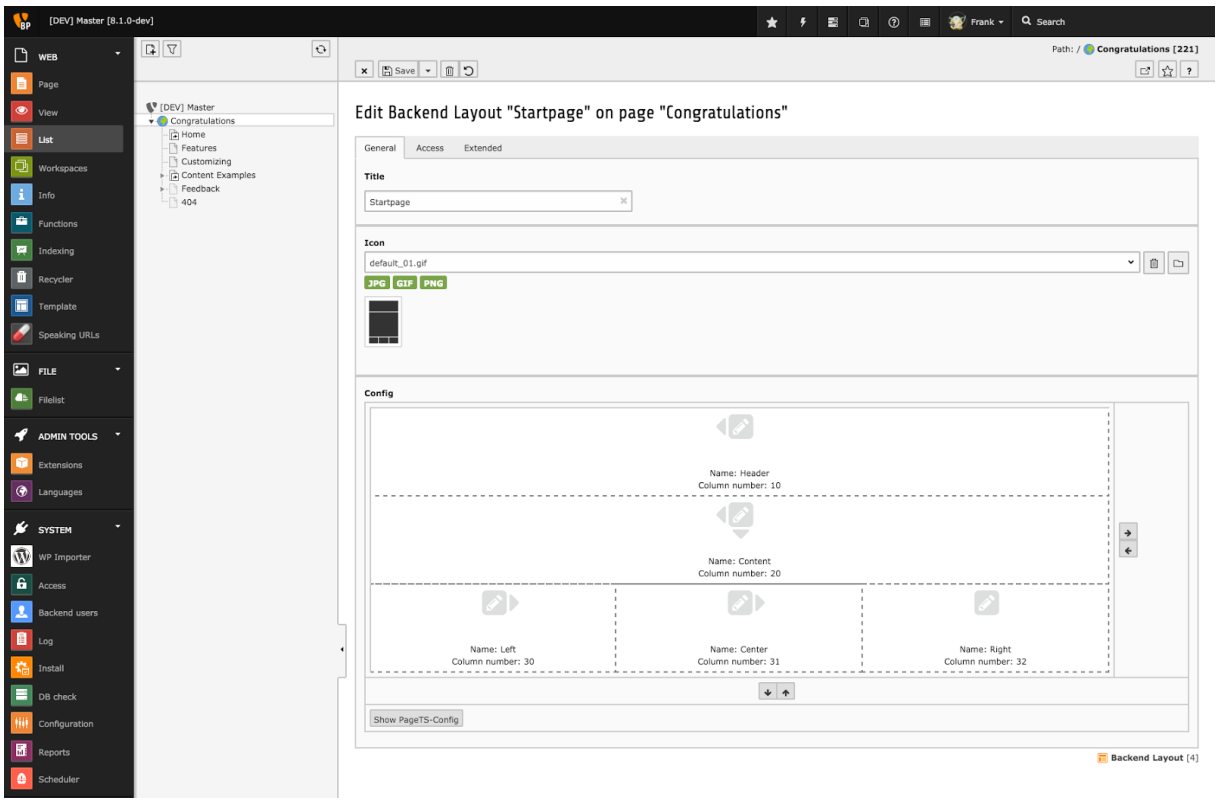


Abbildung 1: Typo3 Backend im Seitenbearbeitungs Modus [Tea16]

2.2 Typo3 Neos

Typo3 Neos ist ein relativ neuer Internetportal Server, welcher 2012 aus dem Wunsch heraus entstand Typo3 zukunftssicher zu gestalten.

Als erkannt wurde, dass eine zukunftssichere Typo3 Entwickung basierend auf dem *Model View Controller* (MVC)-Prinzip eine koplette Neuimplementierung erfordert wurde der Grundstein für Typo3 Neos gelegt. Neben der Entwickung von Typo3 CMS wurde die Entwicklung von Typo3 Neos begonnen. Das Ziel war es, einen Nachfolger für Typo3 zu entwickeln. Dies gelang jedoch nicht wie vorgesehen. [Wik16a]

Früh wurde festgestellt das sich diese die beiden Projekte in unterschiedliche Richtungen entwickeln. Typo3 Neos ist heute in der Version 2.0 erhältlich und hat sich zum Ziel gesetzt Webseiten im gegensatz zu Typo3 live im Frontend zu bearbeiten. Einfach gesagt ist Typo3 ein Internet-Portalserver welcher auf dem *What you see is what what you get* (WYSIWYG)-Prinzip aufbaut. Es ist möglich Webseiten live oder getrennt vom Livesystem im Frontend zu erstellen. Die Frontend-Bearbeitung bingt viele Vorteile für die Webseitenerstellung für Laien. Diese können eine Seite direkt bearbeiten und Live nehmen. [Lob16]

Neos ist bei weitem nicht so verbreitet wie Typo3 CMS und die Community um das Projekt ist deutlich geringer. Dies ist ein klarer Nachteil zum großen Bruder Typo3 CMS ist.

Ähnlich wie bei Typo3 CMS ist es in Neos möglich sogenannte Plugins zu entwickeln. Diese basieren auf dem „Typo3 Flow“-Framework, welche zusammen mit Neos entwickelt wurde. Flow ist ein Framework, welches es erlaubt Templates mit Hilfe von „Fluid“ zu entwickeln. Auf die Fluid wird im Abschnitt 3.4 näher eingegangen. [Wik16b]

Ähnlich wie Typo3 CMS benötigt auch Neos eine SQL-Datenbank zur Datenhaltung und ist Quelloffen verfügbar. Nach der Abspaltung von Typo3 Neos vom Typo3 Projekt, wird es heute

von einer relativ kleinen Gemeinde entwickelt. Aus diesem Grund, gibt es keine Roadmap für das Projekt und die Veröffentlichung von neuen Version ist träger als die von Typo3 CMS.

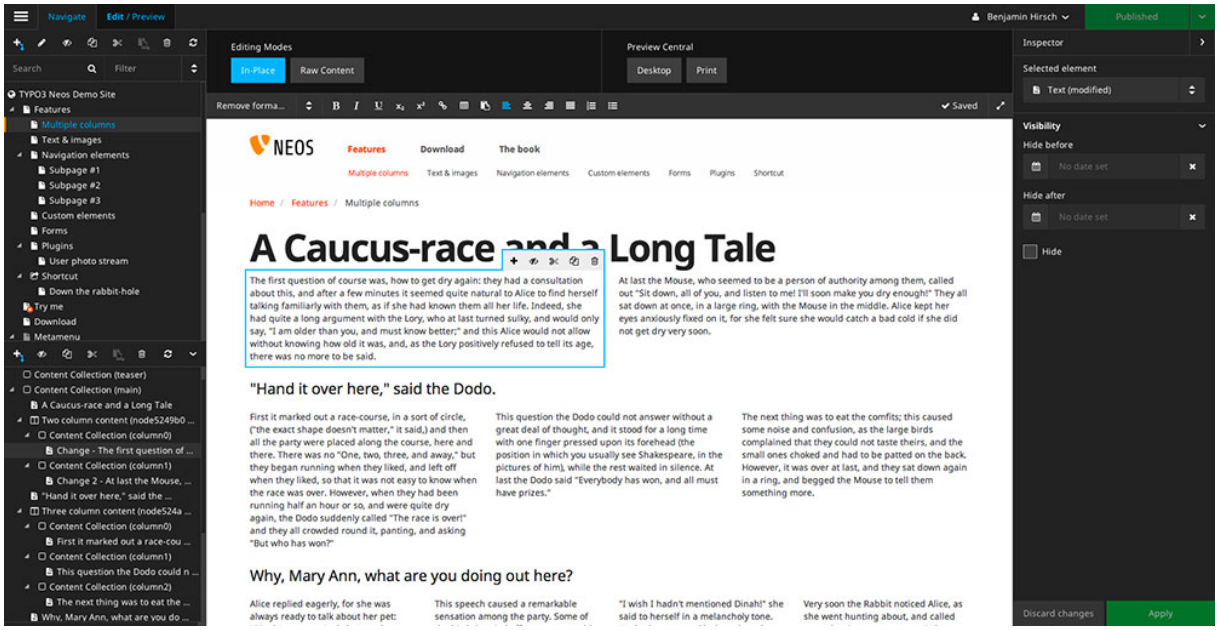


Abbildung 2: Typo3 Neos Backend im Seitenbearbeitungs Modus [HDB14]

In Abbildung 2 ist der Seitenbearbeitungs-Modus von Neos zu sehen. Es ist gut zu erkennen, das Inhalte einer Webseite direkt bearbeitet werden können. Dies macht eine Webentwicklung für Laien einfacher.

2.3 Joomla

Joomla ist ein weiteres CMS-System für die Erstellung von Webseiten. Ähnlich wie Typo3 ist es Open Source und nutzt eine MySQL-Datenbank im Hintergrund. Das System ist in PHP5 geschrieben und dient in erster Linie zur Erstellung von Webseiten. Am besten kann Joomla mit Typo3 Neos verglichen werden, da es einen ähnlichen Ansatz zur Erstellung von Webseiten aufgreift.

Ein Vorteil gegenüber Neos ist, das Joomla weit verbreitet ist und eine beachtliche Community hat, welche bei Neos geringer ausfällt. Nachteilig an Neos ist jedoch, das es eine Bearbeitung von Web-Inhalten nur im Backend zulässt. Dies bedeutet wiederum, das mehr Erfahrung benötigt wird um eine Webseite zu erstellen.

Weitere Nachteile sind zum einen die kaum existente Roadmap, welche zur Zeit der Bearbeitung nicht aktuell war. Zum anderen ist die Entwicklung von Erweiterungen für Joomla nicht so gut strukturiert wie es bei Typo3 mit Fluid der Fall ist.

Eine freie Gestalltung der Webseiten ist zwar gegeben, aber für die Entwicklung eines komplexes Supportportals wie es später einmal entstehen soll ist Joomla nicht oder nur bedingt geeignet. Es gibt momentan keine garantierte Weiterentwicklung des Portalservers und eine Unterstützung von MariaDB ist nicht offiziell gegeben.

Auch wenn Joomla durchaus seinen Charm hat, so ist es für die Entwicklung eines Supportportals, welches modular aufgebaut werden soll nicht zu empfehlen. Dies bedingt sich zum einen aus der abgelaufen Roadmap, aber auch durch die schlechtere Erweiterungsentwicklung im Gegensatz zu Typo3 CMS oder Neos. [Joo16a] [Joo16b]

2.4 Drupal

Drupal ist ein CMS-System welches 2001 veröffentlicht wurde. Das Hauptaugenmerk von Drupal liegt auf der Organisation von Webseite un unterstützt verschiedene Datenbanken wie MySQL, MariaDB, oder PostgreSQL. Mit einem Marktanteil von ca. 2,2% ist Drupal nicht sehr weit verbreitet, ob wohl es mit einer sehr großen Community wirbt. Das System von Drupal, ist in erster Linie für die Gestaltung und die Administration von Webseiten gedacht. Es bietet aber auch CMS-Funktionalität, wie das Verwalten von Dateien. [Dru16b]

Drupal ist in PHP geschrieben und kann mit Hilfe so genannter Module erweitert werden. Die Module bassieren auf dem MVC-Prinzip und ähneln in der Programmierung etwas an Typo3-Extensions. Module in Drupal sind aber den Typo3-Extensions um einiges unterlegen, da sie kein ersatz für das Fluid-Framework bieten und es so nicht möglich ist Abfragen oder ähnliches in Templates unterzubringen. [Dru16a]

2.5 Auswertung der möglichkeiten

In der folgenden Tabelle werden die, zum Begin des Kapitels 2, genannten noch einmal aufgezählt und ausgewertet.

Anforderung	Priorisierung	Typo3 Neos	Typo3 CMS	Joomla	Drupal
Frei gestalltbar	sehr hoch	✓	✓	✓	✓
Eigene Erweiterungen	sehr hoch	✓	✓	✓X	✓X
Open-Source	gering	✓	✓	✓	✓
Große Community	hoch	✓	✓	X	✓X
Verwaltung von Dateien	sehr hoch	✓	✓	✓	✓
Unterstützt MySQL/MariaDB	hoch	✓	✓	X	✓
Gesicherte Weiterentwicklung	sehr hoch	✓X	✓	✓X	✓X

Tabelle 1: Tabellarischer Vergleich der betrachteten Systeme

Joomla und Drupal haben beide ihre eigenen Ansätze und Vorzüge, so kann Drupal mit besonders vielen Datenbanken zusammen arbeiten und Joomla ist sehr gut für die Erstellung von Webseiten geeignet.

Leider bietet Joomla keine aktuelle Roadmap und eine Weiterentwicklung ist fraglich. Aus diesen Grund, sollte es nicht für die Entwicklung eines neuen Portals verwendet werden, welches zukunftssicher und lange Lauffähing sein soll.

Drupal bietet zwar eine stetige Weiterentwicklung anhand einer Roadmap, ist aber leider in Bezug auf die Entwicklung von Erweiterungen Typo3 CMS und Neos unterlegen.

Auch wenn Typo3 Neos Typo3 CMS in nichts nachsteht, so wurde sich dennoch gegen die Verwendung von Neos entschieden, da eine vorranschreitende Entwicklung nicht gewehrleistet ist. Im Punkto Zukunftssicherheit ist Typo3 CMS am besten aufgestellt.

Der gezeigte Vergleich zeigt, dass Typo3 CMS das beste System ist um ein modulares Supportsystem für die PDV Systeme Erfurt GmbH zu entwickeln, weshalb im weiteren Verlauf der Arbeit auf die Entwicklung mit diesem System eingegangen wird.

3 Typo3

Im Abschnitt 2.1 wurde schon grundlegend auf Typo3 eingegangen. Es stellte sich im Abschnitt 2.5 heraus, dass Typo3 die besten Möglichkeiten bietet um ein modulares Supportsystem aufzubauen.

Im folgenden Kapitel wird genauer auf die Verwendung von Typo3 CMS (im weiteren als Typo3 bezeichnet) eingegangen. Hierbei wird auf die Roadmap von Typo3 eingegangen und gezeigt wie eine Typo3 Extension grundlegend entwickelt wird.

3.1 Typo3 8.5

Die momentan aktuelle LTS-Version von Typo3 ist die Version 7, welche aktuell (Stand Dezember 2016) noch immer unterstützt wird. Gleichzeitg wird eine neue LTS-Version entwickelt, welche iterativ innerhalb der Version 8.x entsteht. Momentan ist die aktuelle Version 8.5, welche zum Beispiel einen neuen *Real Time Editor* (RTE)-Editor mitbringt. [Typ16]

Im Verlauf des Jahres 2017, soll dann die neue LTS-Version von Typo3 erscheinen. Da sich die Entwicklung des Supportportals vorraussichtlich bis Ende 2017 erstrecken wird, ist es nur logisch die Neuentwicklung auf den momentanen Zwischenversionen zu entwickeln, um bei der Fertigstellung möglichst auf einer neuen LTS-Version aufsetzen zu können.

3.2 Extensions

Es ist möglich beinahe jede benötigte Funktion in Typo3 einzubauen. Dies geschieht über so genannte Extensions, welche den Funktionsumfang von Typo3 erweitern oder verändern können. Das Grundsystem, welches nach der Installation vorzufinden ist, basiert zum Teil schon auf Extensions, den sogenannten System-Extensions. Diese Extensions bringen grundlegende Funktionalitäten in das Grundsystem ein und erweitern dieses.

Das Auslagern von Funktionalitäten in Extensions hat mehrere Vorteile. Zum einen entlastet es den System-Kern und macht diesen schlanker und übersichtlicher. Zum anderen können Fehler in Extensions behoben werden, ohne das ganze System zu updaten.

Konzepte von Typo3 Extensions sind zum einen die objektorienterte Programmierung und das MVC-Prinzip.

Objektorienterte Programmierung sagt aus, dass alles und jedes im Programm auf Klassen basiert. Lediglich PHP-Funktionalitäten wie finale Klassen oder die Sichtbarkeit „private“ wird nicht unterstützt.

Der gesammte PHP-Quellcode einer Extension ist Objektbasiert, nicht nur Modellklassen, sondern auch Konstroller oder Repositorys sind als Klassen abgebildet.

Model-View-Controller ist ein sehr bekanntes Strukturierungsmittel in der Informatik und kommt auch bei Typo3 Extensions zum Einsatz. In einem Extension-Projekt, müssen alle Klassen innerhalb fester Pfade abgelegt werden.

In Abbildung 3 ist die Struktur der Basis-Extension für das Supportportal zu sehen.

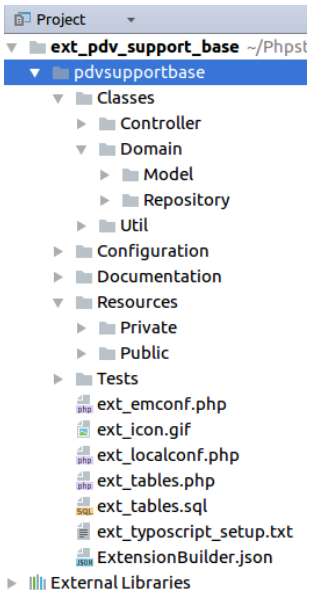


Abbildung 3: Extension Struktur

Modellklassen, welche nach dem MVC-Prinzip reale Objekte verkörpern, sind unter Classes/Domain/Model zu finden.

Controller sind unter Classes/Controller zu finden.

Views, welche bei Webseiten als HTML-Dateien realisiert werden, sind unter Resources/Private zu finden. Hier wiederum sind sie im Unterordner Templates zu finden. Eine Besonderheit bei Extensions ist, dass wiederkehrende Views so genannte „Partials“ im Unterordner Partials zu finden sind. Partials werden innerhalb von Templates gerendert und dargsestellt.

Eine weitere Besonderheit sind die Repository-Klassen, welche unter /Classes/Domain/Repository zu finden sind. Sie stellen die Persistenzschicht zwischen Anwendung und Datenbank dar. Alle Lese- oder Schreiboperationen werden innerhalb der Repositorys durchgeführt. Typo3 stellt innerhalb der Repositorys viele dynamische Methode bereit, um Standardabfrage zu realisieren.

Um den Grundstock für eine neue eigene Extension zu legen, kann der Extension-Builder[1] verwendet werden. Dieser baut unter Angabe des Domain-Modells und der Constroller die Grundextension auf, die dann mit Inhalt gefüllt werden kann.

Eine genauere Beschreibung, aller Komponenten einer Extension und wie eine Extension grundlegend zu Programmieren ist, kann in den Büchern von Patrick Lobacher[2] nachgelesen werden. Lobacher beschreibt in verschiedenen Büchern die Grundlagen für die Programmierung anhand verschiedener Typo3 Versionen.

3.3 TypoScript

TypoScript ist eine eigens für Typo3 entwickelte Konfigurationssprache mit der es möglich ist, eine Typo3 Extension zu konfigurieren. Eine Sprachdefinition zu TypoScript findet sich in der TypoScript Reference[3].

Das eigentliche TypoScript für jede Extension ist unter Configuration/TypoScript (siehe Abbildung 3) in der Datei setup.txt zu finden. Zusätzlich zu der Konfiguration findet sich im selben

[1] `https://typo3.org/extensions/repository/view/extension_builder`
[2] Typo3 Extbase
[3] `https://docs.typo3.org/typo3cms/TyposcriptReference/`

Ordner noch eine Datei constants.txt, in welcher globale Konstanten für die Extension definiert werden können. [Tho07]

Konstanten, welche in der constants.txt, zu finden sind, können aber zusätzlich innerhalb von Typo3 geändert werden. Dies bietet bei richtiger Programmierung einer Extension eine große Flexibilität. Zum Beispiel können in einer, entsprechend zur vorliegenden Arbeit, entwickelten Extension über Konstanten verschiedene Dinge angepasst werden. Es ist so möglich ohne Eingriff in den Quellcode, Links im Footer der Extension zu ändern. (Mehr zu entwickelten Extension siehe Kapitel 4)

Wie genau Extension mit Hilfe von TypoScript konfiguriert werden können, ist in der offizellen Dokumentation oder in den Büchern von Patrick Lobacher genauer beschrieben.

3.4 Fluid

Fluid ist eine Typo3 System Extension, welche ein Framework für das Rendern von Templates bereit stellt. Zu jeder Aktion, welche es innerhalb eines Controllers (siehe 3.2) gibt existiert auch ein Fluid Template, sofern kein Redirect zu einer anderen Aktion durchgeführt wird.

Hat die Aktion des Controllers alle für das Template benötigten Daten ermittelt, so werden diese Fluid übergeben. Fluid wiederum bestimmt anhand der Namen des Controllers und der Aktion, das jeweilige Template und rendert es mit den entsprechenden Daten.

Fluid ermittelt das entsprechende Template wie folgt Resources/Private/Templates/«Controller Name»/«Aktion Name».html.

Innerhalb eines Template können verschiedene Partials angegeben sein, welche wiederkehrende Teile enthalten (siehe 3.2).

Was Fluid jedoch so mächtig macht, sind die sogenannten „ViewHelper“. ViewHelper sind Tags, welche Fluid anweisen, PHP-Routinen auszuführen. Diesen PHP-Routinen können für ihrere Arbeit Daten übergeben werden und geben an Fluid, dass entsprechende Ergebnis zurück, welches dann wiederum im Template durch den eigentlichen ViewHelper-Tag ersetzt wird.

```
1 <f:for each="{tempUsers}" as="tempUser">
2     <tr>
3         <td>{tempUser.firstName} {tempUser.lastName}</td>
4     </tr>
5 </f:for>
```

Listing 1: Beispiel für ein ViewHelper

Im HTML-Beispiel 1 ist ein Ausschnitt aus einem Template zu sehen, welcher einen ViewHelper für eine For-Schleife zeigt. Beim rendern des Templates, wird für jeden Eintrag innerhalb von „tempUsers“ der Inhalt des Tags f:for dupliziert. Zusätzlich ersetzt Fluid die Platzhalter in den geschweiften Klammern mit den realen Werten.

3.5 Schematische Darstellung eines Seitenaufrufs

In den vorangegangen Abschnitten wurden die einzelnen Bestandteile und der interne Aufbau einer Typo3 Extension beschrieben. Zur besseren Verdeutlichung des Seitenaufrufs einer Typo3 Extension, stellt die Abbildung 4 den Zusammenhang noch einmal dar.

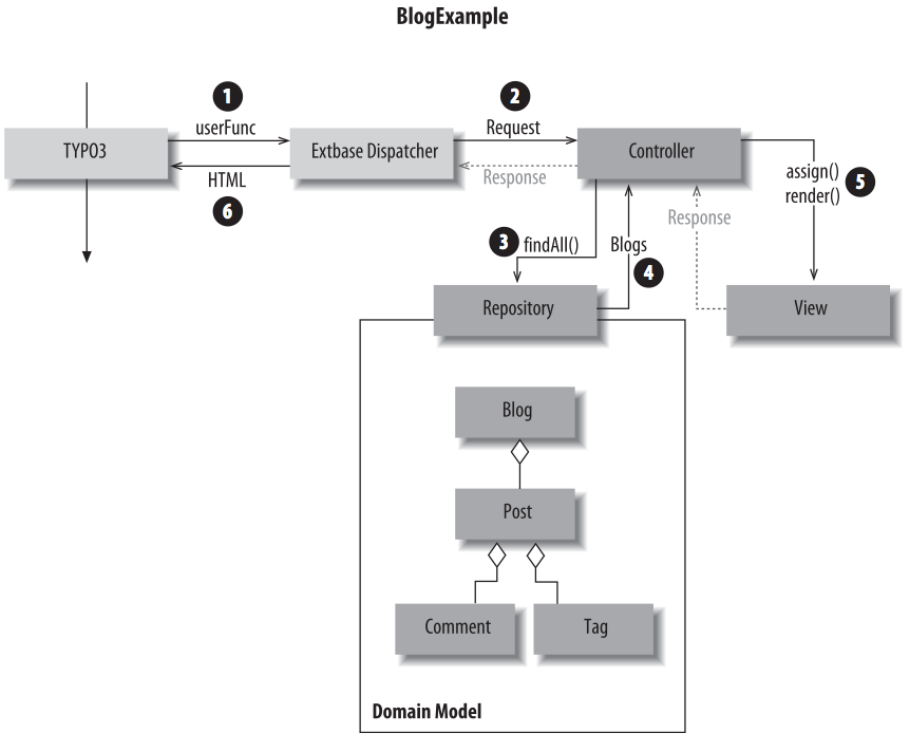


Abbildung 4: Seitenaufruf einer Typo3 Extension[KR16]

1. Nutzer ruft eine Funktion auf
2. Extbase Dispatcher leitet Request an entsprechenden Controller/ Aktion
3. Aktion nutzt Repositorys um Daten zu suchen
4. Repository liefert Daten zurück
5. Aktion ruft den View (Fluid) auf und übergibt die Daten
6. Fluid rendert das Template mit den Daten und gibt das HTML zurück

4 Modulare Extensions

5 Neue Webtechnologien

Im folgenden Kapitel wird auf neue Webtechnologien eingegangen, welche für die Verwendung eines modularen Supportportals in Frage kommen.

5.1 Web Components

Web Components sind eine neue Technologie zur Erstellung von Web-Seiten. Die Spezifikation des W3C für Web Components sehen, wie der Name sagt, vor Komponenten speziell für Web-Seiten zu erstellen. Diese Komponenten können entweder vom Programmierer selbst oder aus Fremdquellen stammen.

Durch die Verwendung soll die Programmierung von Webtechnologien vereinfacht werden. Komponenten kapseln spezielle Elemente wie zum Beispiel Tabellen oder Button mit ihren Funktionen zusammen in ein Element. Komponenten können also auf einer Vielzahl von Technologien aufbauen. Die Nutzung einer Komponente durch einen Programmierer geht somit auch ohne die Kenntnis aller in der Komponente verwendeten Technologien. Ein Programmiere muss sich nicht um die Funktion oder den Aufbau einer Komponente kümmern. Er kann sie einfach benutzen.

Dies bringt zum einen den Vorteil, dass Programmierer sich nicht mehr mit allen Technologien beschäftigen und auskennen muss. Zum anderen können Komponenten schnell in bestehende und neue Projekte integriert werden. Hierdurch kann die Entwicklungszeit einer Web-Seite im Idealfall drastisch gesenkt werden. [Ihl13]

Web Components bestehen nach der Definition des W3C aus fünf bestandteilen:

- Templates
- Decorators
- Custom Elements
- Shadow DOM
- Imports

Templates sind HTML-Teile, welche beim rendern einer Web Komponente zum tragen kommen. Sie werden innerhalb der Komponente definiert (siehe Listing 2) und sind innerhalb von HTML5 standardisiert.

Decorators beschreiben das Aussehen eines Templates innerhalb einer Komponente mit Hilfe von CSS. Zum aktuellen Zeitpunkt sind sie jedoch nicht spezifiziert.

Custom Elements sind Komponenten, die vom Programmierer selbst erstellt wurden. Sie bilden den Grundstein für wiederverwendbare und eigene Web Components.

Shadow DOM ist eine HTML-Struktur, welche vom normalen HTML-DOM aus nicht beeinflusst werden kann (siehe Abschnitt 5.1.1).

Imports sind Anweisungen, die dem Browser sagen welche Komponenten er zusätzlich laden muss. Sie sind essentiell, damit Web Components überhaupt in eine DOM-Struktur eingebunden werden können. Imports sind wie Templates ebenfalls schon in HTML standardisiert. [CG13]

5.1.1 Shadow DOM

Damit Komponenten nicht durch CSS-Klassen von der eigentlichen Web-Seite oder untereinander beeinflussen, wurde der Shadow DOM erfunden.

Der Shadow DOM ist ein konstrukt um Komponenten vor dem eigentlichen HTML-DOM zu verbergen. Dieses vorgehen ist notwendig, um Komponenten verwenden zu können, denn Komponenten sollen sich werder untereinander Beeinflussen noch sollen sie von der Web-Seite beeinflusst werden.

Komponenten im Shadow DOM werden vor dem eigentlich HTML-DOM verborgen, wie der Name schon sagt, liegen sie im Schatten. Der Browser rendert also zuerst das HTML-DOM und anschließend den Shadow DOM, ohne das dieser vom eigentlichen DOM beeinflusst wird. JQuery aufrufe auf der Hautseite zum Beispiel, können sich nicht auf Elemente im Shadow beziehen, da sie diese nicht kennen.

Zur aktuellen Stunde, können alle modernen Browser den Shadow DOM umsetzten. [Pol16b]

5.2 Google Polymer

Das Polymer Projekt wird seit 2012 von Google entwickelt und basiert auf dem Web Component-Standard des W3C. Polymer ist eine Library, welche mit Hilfe des Shadow DOM arbeitet.

Die erste stabile Version von Polymer wurde im März 2016 veröffentlicht und ist stark von AngularJS (siehe Abschnitt 5.3) beeinflusst worden.

Zur Zeit der Bearbeitung ist Polymer in der Version 1.0 veröffentlicht. Jedoch wird momentan schon an der Version 2.0 gearbeitet, welche auch als Beta verfügbar ist.

In den folgenten Abschnitten wird nun weiter auf Polymer in der Version 1.0 eingegangen. [Pol16a]

5.2.1 Wozu Polymer Elemente

Polymer Elemente versuchen sogenannten Boiler-Code unter Zuhilfenahme von Web Komponenten zu vermeiden. Als Boiler-Code wird die Verschmelzung verschiedener Programmiersprachen innerhalb einer Datei oder eines Elements bezeichnet. Boiler-Code ist häufig in Web-Seiten zu finden, was historsch bis heute bedingt ist. Eine weiter gängiger Name für Boiler-Code im Web ist „DOM Pollution“, was so viel aussagt wie, das der DOM mit verschiedenen Sprachen verschmutzt ist.

Um eine HTML-Element zu erstellen wird als Grundgerüst HTML verwendet. Zusätzlich enthält das HTML-Element aber auch CSS um das aussehen zu beeinflussen. Um ein HTML-Element dynamisch oder interaktiv zu gestalten wird meist zusätzlich auch noch Java-Skript oder AngularJS verwendet. Typischer weise findet mal alle drei Sprachen in einer HTML-Datei oder in getrennten Dateien.

Beide Ansätze haben jedoch Nachteile. Bei der Vereinigung verschiedener Sprachen in einer Datei, leidet meist die Übersicht, denn CSS, Java-Skript und zugehöriges HTML muss nicht an festen Positionen sein. So kommt es öfter vor, dass zusammengehörige Elemente innerhalb einer Datei weit auseinander stehen. Hier leidet die Übersicht und Zusammenhänge sich wenn überhaupt oft nur schwer für fremde Programmierer zu erkennen.

Der Ansatz die verschiedenen Sprachen zu trennen, beseitigt zwar das durcheinander innerhalb der HTML-Dateien, jedoch entstehen hier wieder neue Probleme. Wenn alle Sprachen strikt in verschiedenen Dateien sind, ist es oft nur schwer nachzuvollziehen, welches Java-Skript zu welchem HTML-Element gehört. Für außenstehende Programmiere ist dies oft eine unlösbare

Aufgabe. Nachteilig ist auch, dass immer alle Dateien benötigt werden, um das Gesamtbild verstehen zu können.

5.2.2 Was macht Polymer aus

Polymer wiederum geht, im Gegensatz zu den beiden im Abschnitt 5.2.1 genannten, einen dritten sehr eleganten Ansatz. Den der Web Components. Grundsätzlich kommen bei Polymer verschiedene Programmiersprachen und Konzepte wieder zusammen in eine HTML-Datei.

Der entscheidende Unterschied jedoch ist, das HTML, CSS und Skripte ihre festen Bereiche haben, in denen sie implementiert werden. Dies gestaltet die HTML-Datei des jeweiligen Elements sehr übersichtlich und wurde so in den Web Components vom W3C spezifiziert.

Ein weiterer Vorteil ist, dass in der Web-Seite, in der das Element Verwendung findet nur noch ein einzelner Tag zu sehen ist, der auf das Polymer Element verweist.

Damit der Browser das entsprechende Element mit Hilfe der Polymer Library parsen kann muss vor der Verwendung das jeweilige Element über den HTML-Link-Tag eingebunden werden.

Das Polymer Projekt bietet eine ganze Reihe fertige Elemente, welche einfach verwendet werden können[4].

Wem diese Elemente mit ihren Funktion noch nicht reichen, kann entweder die bestehenden Elemente erweitern oder neue Elemente erstellen.

5.2.3 Ein eigenes Polymer Element erstellen

Um zu zeigen, wie einfach es ist auf Basis von Polymer Elemente selbst zu erstellen folgt in den nun kommenden Abschnitten nun Beispiel, welches im Kapitel 6 in Typo3 eingebunden wird.

```
1  <link rel="import" href="../../bower_components/polymer/polymer.html">
2  <dom-module id="button-up">
3      <template>
4          <button id="scrollToTopFAB"
5                  class="btn-floating␣btn-large␣waves-effect␣waves-light␣waves-
                       circle␣half-out-button␣tooltipped␣fix-scroll-button"
6                  data-position="top" data-delay="50" data-tooltip="Nach␣oben␣
                       Scrollen"
7                  on-click="handleClick">
8              <i class="material-icons">navigation</i>
9          </button>
10     </template>
11     <script>
12         Polymer({
13             is: "button-up",
14             handleClick:
15                 function (e){
16                     $("html,␣body").animate({scrollTop: 0}, "slow");
17                 },
18         });
19     </script>
20 </dom-module>
```

Listing 2: Beispiel für ein eigenes Polymer-Element

Als erstes wird der Polymer-Link hinzugefügt, welcher immer vorhanden sein muss. Danach folgt der Tag „dom-module“, welcher die genaue Beschreibung des Elements enthält. Im „template“-Tag wird das HTML eingefügt, welches das Element beinhaltet. Hierauf kann optional ein „style“-Tag folgen, in welchem der Style des HTML-Elements per CSS angegeben wird.

[4] https://elements.polymer-project.org/

CSS für ein Element Die Definition von CSS hat nur Einfluss auf das jeweilige Template. Elemente außerhalb des Element-DOM werden von dem hier definiertem CSS nicht beeinflusst.

Es ist jedoch umgekehrt möglich, CSS-Klassen zu nutzen, im globalen CSS beschrieben sind. Eine erneute Einbindung der CSS-Datei in das Polymer-Element ist nicht notwendig. Die Verwendung von CSS-Abhängigkeiten in Polymer ist jedoch unschön, da dies ein Element gegebenen Falls auf Web-Seiten beschränkt, welche diese bestimmte CSS-Datei verwenden. Es ist somit in den meisten Fällen besser keine CSS-Abhängigkeiten in Elementen zu verwenden, vor allem dann wenn diese öffentlich sind.

Im Listing 2 wurden CSS-Abhängigkeiten zu MaterializeCSS (Abschnitt 5.5.2) aufgebaut, eine Begründung hierfür ist im Kapitel/Abschnitt (?????) zu finden.

Element-Script Der „script“-Tag enthält das jeweilige Skript zum Element. Es ist JSON-Notation gehalten und basiert auf Callbacks.

Die Registrierung eines Elements steht immer zu begin des Skripts. Dieser Teil muss also immer vorhanden sein, damit das jeweilige Element überhaupt dargestellt werden kann.

```
Polymer({
    is: "button-up",
)};
```

Listing 3: Polymer-Element Registrierung

Handler werden in Polymer verwendet um einzelne Skripte aufzurufen. Innerhalb des Handlers kann ein Java-Skript definiert sein, das ausgeführt wird, wenn der jeweilige Handler getriggert wird.

```
handleClick:
    function (e){
        $("html, body").animate({scrollTop: 0}, "slow");
    },
```

Listing 4: Polymer-Element Handler

Damit ein Handler überhaupt zum Tragen kommt, muss er innerhalb des HTML-Tags für das dass jeweilige Skript sein soll registriert werden.

```
on-click="handleClick"
```

Listing 5: Polymer-Element Handler Registration

Attribute, welche das Polymer-Element haben soll, müssen ebenfalls im Skript definiert werden.

```
properties: {
    owner: {
        type: String,
            value: "Daniel"
    }
}
```

Listing 6: Polymer-Element Attribute

Das Attribut „owner“ wird im Beispiel als String definiert, und hat den Standardwert „Daniel“. Jedes im Skript definiertes Attribut lässt sich bei Verwendung des Elements wie ein normales XML-Attribut verwenden. Wird bei Verwendung des Elements das „owner“-Attribut nicht angegeben, so ist es „Daniel“. Um den Inhalt des Attributs im Template zu verwenden, muss es in doppelten geschweiften Klammern aufgerufen werden. So kann zum Beispiel ein B-Tag wie folgt geschrieben werden.

```
<b>{{owner}}</b>
```

Listing 7: Polymer-Element Attribute benutzen

Der Aufruf dieses Polymer-Elements muss wie im Beispiel umgesetzt werden

5.2.4 Zusammenfassung

Polymer bietet wie gezeigt wurde eine echte Alternative zum normalen Web-Seiten-Boiler-Code. Durch die Verwendung von Polymer-Elementen entsteht keine nennenswerte Verzögerung beim parsen einer Web-Seite durch den Browser. Es bietet sich daher an, bei Neuentwicklungen auf Polymer zu setzen.

Ob und wie eine Intergration von Polymer in Typo3 möglich ist, wird im Kapitel 6 näher beschrieben.

5.3 AngularJS

AngularJS oder kurz Angular ist ein von Google entwickeltes Javascript Framework, welches sich sehr gut für Singe-Page-Web-Sites eignet. Angular ist nach dem *Model View ViewModel* (MVVM)-Prinzip aufgebaut, welches eine Trennung zwischen Darstellung und Logik bietet und arbeitet Client-Seitig.

Die Darstellung wird über HTML abgebildet, welches Angular durch eigene Attribute ergänzt. Die Logik wiederum wird durch Javascript auf Basis von Angular abgebildet.

Angular ist in der Lage das HTML-DOM entsprechend der Befehle der Logik umzubauen ohne dafür auf JQuery zurückgreifen zu müssen.

Aufgaben, wie das Suchen oder Sortieren einer Tabelle sind typische Anwendungsbeispiele von Angular. [TB14]

Die momentan aktuelle Version von Angular ist 1.5.6, welche auch in dieser Arbeit verwendet wird. Zur Zeit wird aber auch der Release von Angular 2.0 vorbereitet, wobei die neue Version nicht abwertskompatibel ist, da sich sehr viel an der Codebasis geändert hat.

Durch das neue Framework, soll das ohnehin schon schnelle Framework noch einmal an Geschwindigkeit zunehmen.

Im folgenden Beispiel der W3C-School wird gezeigt wie Angular eingesetzt werden kann.

```
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + "␣" + lastName}}

</div>
```

Listing 8: Angular View im HTML-DOM [W3C16]

Die im Div-Tag zu sehenden Attribute „ng-app“ und „ng-controller“ werden von Angular genutzt. „ng-app“ gibt dabei an, wie die Angular Applikation heißen soll. „ng-controller“ gibt den Namen des Controllers an, welcher für das HTML-Element zum Einsatz kommen soll.

Innerhalb des Div-Tags, welcher die Applikation darstellt, sind zwei Input-Tag zu sehen, welche das Angular-Attribut „ng-model“ beinhalten. „ng-model“ sagt aus, dass diese Elemente zum Datenmodell gehören und welchen Namen sie tragen.

In doppelten geschweiften Klammern werden Modell-Attribute, welche zur Laufzeit von Angular durch die konkreten Werte der Modell-Attribut ersetzt werden.

```
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "John";
    $scope.lastName= "Doe";
});
</script>
```

Listing 9: Angular ViewModel und Model im HTML-DOM [W3C16]

Im Listing 9 ist das ViewModel beziehungsweise das Model des im Listing 8 gezeigten Views zu sehen.

Als erster Schritt, wird der zu referenzierende Angular View angegeben und in der Variable „app“ abgelegt. Daraufhin wird ein Controller innerhalb der Applikation, unter Verwendung des Namens, angelegt.

Innerhalb des Controllers werden dann für die Model-Attribute angegeben. Im Beispiel sind das die Werte „John“ und „Doe“.

Wird nun die entsprechende HTML-Seite im Browser aufgerufen, füllt Angular die beiden Input-Tags mit den Standardwerten und ersetzt auch die Platzhalter in geschweiften Klammern. Ändert ein Nutzer nun den Inhalt der Input-Tags, so passt Angular automatisch das HTML-Dom an und aktualisiert die Werte innerhalb der geschweiten Klammern.

Das vom Browser geparste Dom enthält keine Angaben darüber, wo sich Platzhalter befinden. Einzig Angular weis, welche Elemente bei einer Änderung angepasst werden müssen. [W3C16]

Durch die neue Version 2.0 von Angular ändert sich die Verwendung und das Paradigma dramatisch. Wärend Angular 1 noch auf das MVVM-Prinzip setzt, geht Angular 2 den Weg der Web Komponenten. Da bald eine neue Version erscheint, kann zum aktuellen Zeitpunkt noch nicht gesagt werden wie lange Angular 1 noch unterstützt wird und ob der Einsatz noch sinnvoll ist.

Eine Alternative zu Angular 1 wäre das Java-Script-Framework Aurelia[5] welches einen ähnlichen Umfang wie Angular 1 bietet.

Aus Zeitgründen kann leider an dieser Stelle nicht auf die Verwendung von Aurelia und Angular 2 eingegangen werden.

5.4 HTML5

5.5 CSS3

Cascading Style Sheets (CSS) ist eine Designsprache für HTML und ist deshalb eine der Hauptkomponenten der Webentwicklung. Mit Hilfe von CSS ist es möglich einzelne HTML-Elemente oder Gruppen zu stylen. Seit der ersten Version die 1993 erschien, wird CSS kontinuirlich weiterentwickelt und ist heute auf praktisch jeder Web-Seite im einsatz.

Auf der Basis von CSS entwickelten sich im laufe der Jahre immer mehr Frameworks, welche einen Designansatz umsetzten und fertige Klassen für die Verwendung bereitstellen. Eines der heute am häufigsten anzutreffenden CSS-Frameworks ist „Bootstrap“.

Innerhalb der PDV Systeme Erfurt wurde beschlossen das neue Supportportal im „Material Design“ aufzubauen. Als „Material Design“ werden Gestaltungsrichtlinen von Google bezeichnet, welche angeben wie eine Android-Applikation oder eine mobile Web-Seite für Android aussehen sollte.

[5] www.aurelia.io

Das „Material Design“ ist ein flaches Design, und geht von der Metapher aus, dass der Bildschirm Papier ist. Jedes Element soll sich also wie reales Papier verhalten und auch so aussehen. Obwohl das Design sehr schlicht gehalten ist, so kann mit viel Farbe gearbeitet werden. [Mat16a]

Die nun folgenden Betrachtungen beziehen sich auf die Umsetzung im „Material Design“.

5.5.1 Bootstrap

Bootstrap ist ein CSS Framework, welches von „Twitter“ entwickelt wird und das unter der „MIT-Lizenz“ frei erhältlich ist. Durch den modularen Ansatz von Bootstrap ist es sehr leicht möglich das Framework um eigene Style-Anweisungen zu ergänzen. [Boo16] [Wik16c]

Bootstrap baut auf dem Less-Parser auf. Less ist eine Sprache, die es sich zum Ziel gesetzt hat das schreiben von CSS möglichst einfach und effizient zu machen. Geschriebender Less-Code muss in CSS geparst werden.

Das Bootstrap Framework kann entweder direkt als fertiges CSS-Framework oder als Less-Code heruntergeladen werden.

Ein Nachteil von Bootstrap ist, dass es nicht leichtgewichtig ist. Zwar kann es sehr gut erweitert werden, dies macht jedoch das Framework auch Schwerfällig. Soll zum Beispiel ein Bootstrap-Template für das „Material Design“ verwendet werden, muss zunächst das „standard Framework“ eingebunden werden. Ein Template wie „Material Design for Bootstrap“ überschreibt dann nach der Einbindung zum Teil das standard Framework und ergänzt es um eigene Klassen. Der Vorteil bei diesem Vorgehen ist ganz klar, dass ein bestehendes Template relativ leicht ausgetauscht werden kann, ohne dass der eigentliche HTML-Code der Web-Seite angepasst werden muss.

Dieser Vorteil wird schnell zum Nachteil, wenn eine Web-Seite erstellt werden soll bei der es auf Geschwindigkeit ankommt. Durch das überscheiben des standard Frameworks wird zusätzliche Zeit beim parsen der Web-Seite benötigt, was vorallem bei älteren PCs auffällt. Zusätzlich steigt der Overhead beim laden einer Seite, da mehr CSS geladen wird, als tatsächlich benötigt wird.

Standardmäßig enthält Bootstrap zum Beispiel keinen Date-Picker, welcher jedoch in einem Supportportal unerlässlich ist. Auch „Bootstrap Material“ enthält keinen Date-Picker im „Material Design“. Durch den modularen Aufbau von Bootstrap kann natürlich schnell und einfach ein entsprechendendes Template ergänzt werden. Dies bedeutet aber wieder zusätzlichen Overhead. [Mat16b]

5.5.2 MaterializeCSS

Ein anderes Framework, welches die Design-Richtlinien des „Material Design“ umsetzt ist das noch recht neue Framework „MaterializeCSS“. MaterializeCSS ist ein eigenständiges Framework, welches versucht alle gegebenen Gestaltungsrichtlinen möglichst elegant und performat umzusetzen.

Die MaterializeCSS Quellen sind mit *Syntactically Awesome Stylesheets* (SASS) geschrieben, und müssen vor der Verwendung in CSS komiliert werden. Dieses Vorgehen hat meherer Vorteile. So ist es zum einen sehr einfach möglich das Framework um eigene Style-Objete zu erweitern. Zum anderen hat es den Vorteil, dass die Farbgebung des gesamten Frameworks an einer Stelle zusammen gefasst ist. Soll also die Farbgebung geändert werden, so werden in den SASS-Quellen des Frameworks die Farben zentral angepasst. Nach einer erneuten kompilierung stehen dann die Farben im gesamten Framework zur Verfügung. Ohne dieses Vorgehen müssten die entsprechenden Farben an unzähligen Stellen innerhalb der CSS-Datei angepasst werden. [Mat16c] Ein weitere Vorteil von MaterializeCSS ist, das viele Templates wie ein Date-Picker im „Material Design“ schon vorhanden sind. Diese Templates können ohne zusätzlichen Overhead verwendet werden.

Auch wenn MaterializeCSS noch sehr jung im Vergleich zu Bootstrap ist, so ist es doch eine gute alternative, wenn eine Web-Seite im „Material Design“ umgesetzt werden soll.

5.5.3 Vergleich beider CSS-Frameworks

Vergleichspunkte	Bootstrap Material	MaterializeCSS
Abhängigkeiten	• Bootstrap • JQuery	• JQuery
Umfang	• Standard Layout • Dialoge	• Standard Layout • Dialoge • Date-Picker • Card-Views • Side-Navs • viele weitere
Farbanpassung	über CSS Klassen (17 Farben)	Farben werden in SASS-Quellcode definiert (kein Limit)
Icons	Verwendung von Material Icons	Verwendung von Material Icons
Subjektive Geschwindigkeit	teilweise Träge und Langsam	immer schnell und flüssig

Tabelle 2: Vergleich von Bootstrap Material und MaterializeCSS

????? Hier geht es noch weiter

5.6 PHP7

5.7 Google Dart

5.8 Service Worker

6 Zusammenspiel der Technologien

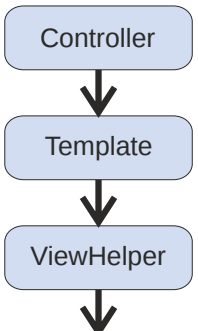
6.1 Integration von Polymer in Typo3

Im Abschnitt 5.2 wurde ausführlicher auf die Verwendung und die Erstellung von Polymer-Elementen eingegangen. Nun soll untersucht werden wie sich Polymer möglichst elegant in Typo3 integrieren lässt.

Der beste Weg Polymer-Elemente in Typo3 zu integrieren ist, diese in ViewHelper zu kapseln (Siehe Kapitel ?????).

Das Ziel ist es also, einen ViewHelper zu erstellen, welcher einen ViewHelper-Tag in einen Polymer-Tag umwandelt.

In Abbildung 5 (rechts) ist die Aufrufhirarchie zu sehen, welche innerhalb des Projektes entwickelt wurde, um Polymer Elemente elegant in Typo3 einzuarbeiten.



Die beim laden einer Seite aufgerufene „Action“ eines bestimmten „Controllers“ wird aufgerufen. Der „Controller“ stellt alle zum Rendern des Templates benötigte Daten bereit und übergibt sie dem Template.

Das Template wiederum beinhaltet einen für das Polymer Element entwickelten ViewHelper, welcher innerhalb des Templates aufgerufen wird.

Das Fluid-Framework ruft beim Rendern eines Templates den entsprechenden ViewHelper auf.

Der ViewHelper wiederum gibt den Link zum entsprechenden Polymer Element und den Polymer Tag zurück.

Ist das Template fertig gerendert, so wird es an den aufrufenden Client gesendet.

Der Client wiederum ersetzt das Polymer Element durch HTML.

Am Beispiel des Button-Up Elements (siehe Abschnitt 5.2.3), wird im folgenden erklärt, wie dieser Prozess im Quellcode umgesetzt wird.

```php
<?php
namespace PdvPolymer\ViewHelpers\Custom;
use TYPO3\CMS\Fluid\Core\ViewHelper\AbstractTagBasedViewHelper;
class ButtonUpViewHelper extends \TYPO3\CMS\Fluid\Core\ViewHelper\
    AbstractTagBasedViewHelper
{
    //the tagname which is also the name
    protected $tagName = 'button-up';
    /**
     * this method is used to initialize the specific tag arguments
     */
    public function initializeArguments()
    {
        parent::initializeArguments();
    }
    /**
     * generate the tag for the element and the link for the element.
     * @return string the link and the tag for the element
     */
    public function render(){
        $tag = '<link rel="import" href="typo3conf/ext/pdvpolymer/Resources/
            Public/Elements/Custom/button-up.html">';
        return $tag . $this->tag->render();
    }
}
```

Listing 10: Polymer ViewHelper

Der ViewHelper erbt von der abstrakten Klasse „AbstractTagBasedViewHelper“. Dieser ist in der ViewHelper ist anschließend in der Lage, einen HTML-Tag zu verändern.

Mit dem Attribut $tagname, wird definiert wie der neue Tag heißen soll, welcher den ViewHelper-Tag ersetzt. Dieser Tag muss nun so heißen wie das Polymer-Element.

Die Methode „render“ des ViewHelpers gibt den String zurück, der den ViewHelper-Tag ersetzt.

Die „return“-Anweisung gibt immer den Link zum entsprechenden Polymer-Element zurück, und den Polymer-Tag.

Der ViewHelper wird nun wie folgt aufgerufen.

```
<p:custom.buttonUp></p:custom.buttonUp>
```

Listing 11: Polymer ViewHelper aufrufen

Beim ausliefern der Web-Seite ersetzt Typo3 nun den ViewHelper-Tag gegen die folgenden HTML-Elemente

```
<link rel="import" href="typo3conf/ext/pdvpolymer/Resources/Public/Elements/
    Custom/button-up.html">
<button-up />
```

Listing 12: Polymer ViewHelper Ersetzung

Wird die Web-Seite nun vom Browser gerendert wird, ersetzt er diesen Polymer-Tag schließlich durch das definierte Polymer-Element.

Durch die Verwendung von ViewHelpern, kann jedes beliebige Polymer-Element auch unter Typo3 zum Einsatz kommen. Es ist also sehr elegant möglich Boiler-Code auch in Typo3-Templates und -Partial mit der Hilfe von Polymer zu umgehen.

Durch die Kapselung der Polymer-Elemente in ViewHelpern verhalten sich diese sich außerdem wie standardmäßige ViewHelper mit Boiler-Code.

Um nun Polymer-Elemente in einem Template zu verwenden, ist kein extra Wissen notwendig, was eine Verwendung so einfach wie möglich macht. Der Programmierer eines Templates muss sich keine Gedanken über Links zu Polymer-Elementen machen und muss sich auch nicht mit

Polymer auskennen. Es wird einfach der ViewHelper verwendet und alles andere passiert im Hintergrund.

6.2 MaterialzeCSS

Die Integration von MaterializeCSS in Typo3 ist genau so einfach, wie die Integration von Bootstrap oder jedem anderem CSS-Frameworks. Um MaterializeCSS in Typo3 verwenden zu können, muss im Template nur angegeben werden, welche CSS-Dateien Typo3 einbinden soll.

Zusätzlich zu der CSS-Datei benötigt MaterializeCSS noch eine eigene Java-Script-Datei und eine aktuelle Version von JQuery. Diese Dateien werden zusammen mit dem CSS im Template definiert.

Außerdem ist es möglich mit MaterialieCSS die standard Icons und Schriftarten von Google zu nutzen. Sollen dieses Sachen verwendet werden, so müssen sie nur zusätzlich zum CSS und Java-Script im Template definiert werden.

Im Listing 13 ist ein Typo-Script-Beispiel zu sehen, mit welchem MaterializeCSS in Typo3 integriert wird. Das angegebe Typo-Script an entweder in der Konfiguration einer Extension angegeben sein oder es wird direkt in einem Typo3-Template definiert.

```
1  page{
2      includeCSS{
3          icons = https://fonts.googleapis.com/icon?family=Material+Icons
4          material = EXT:pdvtemplate/Resources/Public/Materialize/css/materialize.
               css
5      }
6      includeJS{
7          query = EXT:pdvtemplate/Resources/Public/jquery-2.1.1.min.js
8          materialjs = EXT:pdvtemplate/Resources/Public/Materialize/js/materialize
               .js
9          webcomponents = EXT:pdvtemplate/Resources/Public/webcomponents-lite.js
10     }
11 }
```

Listing 13: MaterialieCSS Anbindung

6.3 Nutzer Anmeldung und Registrierung

7 Zusammenfassung

8 Fazit

9 Abkürzungsverzeichnis

CMS *Content Managemente System*

LTS *Long Term Support*

MVC *Model View Controller*

WYSIWYG *What you see is what what you get*

CSS *Cascading Style Sheets*

SASS *Syntactically Awesome Stylesheets*

MVVM *Model View ViewModel*

RTE *Real Time Editor*

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[Boo16] Bootstrap Homepage. http://getbootstrap.com/, September 2016.

[CG13] Dominic Cooney and Dimitri Glazkov. Introduction to Web Components. June 2013.

[Dru16a] Drupal. https://www.drupal.org/, Dezember 2016.

[Dru16b] Wikipedia Drupal. https://de.wikipedia.org/wiki/Drupal, Dezember 2016.

[HDB14] HDBlog TYPO3 Neos: Das CMS der Zukunft? https://www.hdnet.de/blog/typo3-neos-das-cms-der-zukunft/, Januar 2014.

[Ihl13] Jens Ihlenfeld. HTML-Elemente selber bauen. http://www.golem.de/news/web-components-html-elemente-selber-bauen-1305-99318.html, May 2013.

[Joo16a] Joomla. https://www.joomla.de/, Dezember 2016.

[Joo16b] Wikipedia Joomla. https://de.wikipedia.org/wiki/Joomla, Dezember 2016.

[KR16] Sebastian Kurfürst and Jochen Rau. Developing TYPO3 Extensions with Extbase and Fluid. Technical report, Typo3 Org., Dezember 2016.

[Lob16] Parick Lobacher. *TYPO3 Extbase: Moderne Extensionentwicklung für TYPO3 CMS mit Extbase & Fluid*. CreateSpace Independent Publishing Platform, 2 edition, Februar 2016.

[Mat16a] Material Design. Technical report, Google Inc., September 2016.

[Mat16b] Material Design for Bootstrap. Technical report, September 2016.

[Mat16c] Materialize. Technical report, September 2016.

[Pol16a] Polymer Project. https://www.polymer-project.org/1.0/, November 2016.

[Pol16b] What is shady DOM? https://www.polymer-project.org/1.0/blog/shadydom, November 2016.

[Rie15] Sebastian Rieger. Konzept und prototypische Implementierung eines übergreifenden Dokumenten- und Medienmanagements. Technical report, Karlsruher Institut für Technologie, August 2015.

[TB14] Philipp Tarasiewicz and Robin Böhm. *AngularJS : eine praktische Einführung in das JavaScript-Framework*. dpunkt-Verl., Heidelberg, 2014.

[Tea16] TYPO3 News. https://typo3.org/news/article/typo3-v81-tightening-the-screws/, September 2016.

[Tho07] Thomas. TypoScript: Eine Einführung. http://www.webworking-blog.de/content-management/typo3/typoscript-eine-einfuhrung/, Juli 2007.

[Typ16] TYPO3 News. https://typo3.org/news/article/typo3-v85-released/, Dezember 2016.

[W3C16] AngularJS Introduction. http://www.w3schools.com/angular/angular_intro.asp, November 2016.

[Wik16a] TYPO3. https://de.wikipedia.org/wiki/TYPO3, September 2016.

[Wik16b] TYPO3 Flow. https://de.wikipedia.org/wiki/TYPO3_Flow, September 2016.

[Wik16c] Bootstrap (Framework). https://de.wikipedia.org/wiki/Bootstrap_(Framework), September 2016.