PDV-SYSTEME

FACHHOCHSCHULE ERFURT UNIVERSITY OF APPLIED SCIENCES

Evaluation moderner Webtechnologien für die Entwicklung eines modularen Supportportals

Master Projekt

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. -Witschaftsinform. (FH) Nico Kaiser
Gutachter der Fachhochschule	Prof. Rolf Kruse

Erklärung

Ich, Sebastian Rieger, versichere hiermit, dass ich das vorliegende Masterprojekt 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 vernetzten Welt wie unserer werden unablässig neue und bessere Web-Technologien entwickelt. Diese neuen Technologien bringen eine bessere Programmierfreundlichkeit mit sich, sind aber zum anderen durch neue Ansätze auch schneller als ältere Technologien.

Moderne Webanwendungen müssen eine Vielzahl an Kriterien erfüllen. Eine gute Webseite sollte heute möglichst auf einem mobilen Endgerät, sowie auf einem Desktop-PC oder Smart-TV darstellbar sein. Um dies zu ermöglichen, werden besondere Anforderungen an das Design gestellt, wobei auch möglichst alle gängigen Browser unterstützt werden müssen. Ein weiterer Punkt ist, dass Webseiten sich heute so flüssig und performant wie ein natives Programm verhalten sollen. Hierbei ist es zudem wichtig, vorhandene Hardware anzusprechen und mit ihr zu interagieren.

Das vorliegende Projekt soll den theoretischen Grundstock für eine weitere Arbeit bilden, in der das Support-Portal der PDV System Erfurt GmbH neu entwickelt wird. Dabei, soll evaluiert werden, welcher der gängigen Portalserver für ein Supportportal am besten geeignet ist. Außerdem soll untersucht, wie es möglich ist, modulare Erweiterungen für einen solchen Server zu entwickeln, um unnötige Datenredundanz zu vermeiden.

Es soll zudem untersucht werden, wie und ob es möglich ist, diese verschiedenen Technologien in möglichst modularen Erweiterungen unterzubringen, um eine möglichst performante und zu gleich komplexe und umfangreiche Webseite zu entwickeln. Durch die Verwendung neuer Technologien soll außerdem eine möglichst große Funktionsvielfalt erreicht werden. Dieses Oxymoron aufzulösen, ist der Schwerpunkt der vorliegenden Arbeit, obwohl doch schon jetzt ersichtlich ist, dass für beide Seiten ein Kompromiss gefunden werden muss. Wie dieser aussehen kann, wird im weiteren Verlauf eine wichtige Rolle spielen.

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.

Durch einen modularen Aufbau, sollen außerdem die Schnittstellen zwischen verschieden Erweiterungen vereinfacht werden. Zusätzlich soll es möglich werden, nur bestimmte Erweiterungen zu verwenden, wenn für den jeweiligen Use-Case nicht alle Erweiterungen 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 Supportportal bieten zu können. Dieses System soll 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. Enterprise *Content Managemente System* (CMS)-Servers aufgebaut werden. Hierfür werden im folgenden einige Möglichkeiten genauer betrachtet.

Ein CMS-Server, welcher für das Projekt herangezogen wird, muss die folgenden Eigenschaften aufweisen:

- Webseiten müssen frei gestaltbar sein.
- Es muss die Möglichkeit bestehen, Anwendungen für den Server zu entwickeln.
- Das System muss Open-Source sein, um gegebenenfalls 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.
- Das System muss unter einer SQL-Datenbank wie MySQL oder MariaDB Lauffähing sein.
- Des sollen einfache Web-Technologien wie HTML, CSS, PHP und JavaScript verwenden werden.

Auf die Betrachtung reiner CMS-Systeme wie zum Beispiel „Alfresco (Java-Tomcat)“ wird an dieser Stelle verzichtet, da diese nicht die gewünschten Anforderungen eines Portalservers erfüllen und zumeist nicht als Webanwendung zur Verfügung stehen. Ein Vergleich verschiedener reiner CMS-Systeme ist in der Bachelorarbeit „Konzept und prototypische Implementierung eines übergreifenden Dokumenten- und Medienmanagements“ zu finden. [Rie15]

Soll ein CMS-System eingesetzt werden, muss eine Firma zuerst abwägen, welches System zur Verwendung kommen soll. Die Webseite „CMS Matrix“[1] listet die meisten bekannten Systeme auf, welche dort auch direkt verglichen werden können. Nach einer ersten internen Gegenüberstellung werden nun die Systeme TYPO3, NEOS, Drupal und Joomla genauer betrachtet.

2.1 TYPO3

TYPO3 ist ein CMS-System für Internetseiten und basiert in der neusten Version 8.2 auf der PHP Version 7. Es wurde 1998 von Kasper Skårhøj, als Open Source Projekt entwickelt, und seit der Version 7.6, welche zugleich eine *Long Term Support* (LTS) Version ist, unter dem Namen TYPO3 CMS vertrieben. [Wik16a]

TYPO3 spaltet sich intern in ein Backend und ein Frontend, wobei im Backend nicht nur Einstellungen für das Frontend vorgenommen werden. Das TYPO3 Backend arbeitet als eine Art eigene Webseite, welche es erlaubt, anfallende Aufgaben vom Frontend abzuarbeiten. Interne Mitarbeiter arbeiten somit innerhalb des TYPO3 Bachends, wobei Anwender im Frontend mit der Webseite interagieren.

Es ist möglich, unter Verwendung von HTML, CSS, JavaScript, PHP, Extbase und Fluid eigene Erweiterungen für den Portalserver zu entwickeln. Die jeweiligen Erweiterungen wirken im Frontend von TYPO3 wie ganz normale Webseiten. Auf Extbase und Fluid wird im Kapitel 3 genauer eingegangen.

[1] `http://www.cmsmatrix.org/`

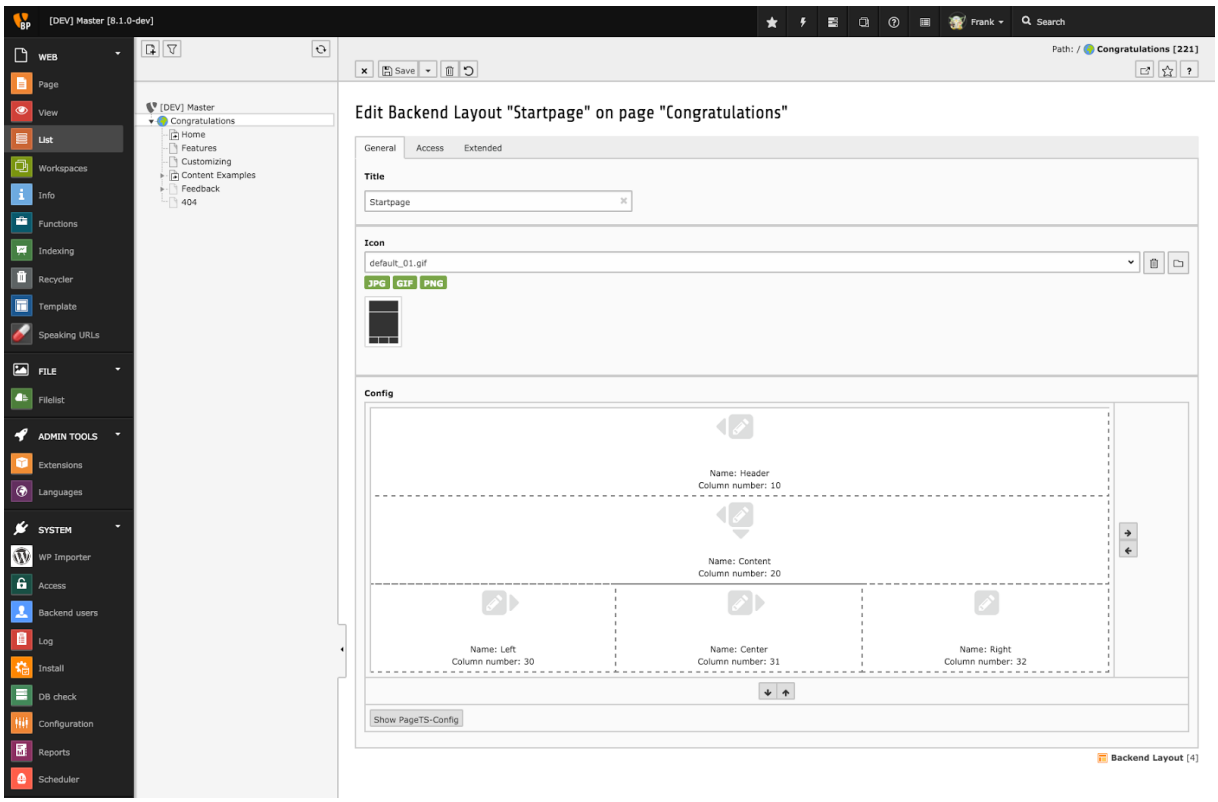

Abbildung 1: TYPO3 Backend im Seitenbearbeitungsmodus [Tea16]

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 garantiert TYPO3 eine stabile und lauffähige Software.

Durch das übersichtliche Backend (siehe Abbildung 1) ist es auch für Anwender 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, dass Nutzer Dateien auf den Server vom Frontend aus hochladen. Hierdurch entsteht die Möglichkeit, ein Austauschportal für Dateien zu schaffen. Nutzer können so in einer sicheren Umgebung sensible Daten untereinander oder mit Mitarbeitern austauschen. [Lob16]

2.2 Neos

Neos ist ein relativ neuer Internet-Portalserver und ein Fork von TYPO3 CMS, 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 komplette Neuimplementierung erfordert, wurde der Grundstein für Neos gelegt. Neben der Entwickung von TYPO3 CMS wurde die Entwicklung von Neos begonnen. Ziel war es, einen Nachfolger für TYPO3 zu entwickeln. Dies gelang jedoch nicht wie vorgesehen. [Wik16a]

Früh wurde festgestellt, dass sich beide Projekte in unterschiedliche Richtungen entwickeln. 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. Mit der Veröffentlichung der Version 2.0 wurde der Name außerdem in Neos geändert. Einfach gesagt, ist NEOS 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 Anwender. 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 kleiner. Dies ist ein klarer Nachteil zu TYPO3 CMS.

Ähnlich wie bei TYPO3 CMS ist es in Neos möglich, sogenannte Plugins zu entwickeln. Diese basieren auf dem „TYPO3 Flow“-Framework, welches zusammen mit Neos entwickelt wurde. Flow ist ein Framework, welches es erlaubt, Templates mit Hilfe von „Fluid“ zu entwickeln. Auf das Fluid-Framework wird im Abschnitt 3.6 näher eingegangen. [Wik16b]

Ähnlich wie TYPO3 CMS benötigt auch Neos eine SQL-Datenbank zur Datenhaltung und ist Quelloffen verfügbar. Nach der Abspaltung von 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 Versionen ist langsamer als die von TYPO3 CMS.

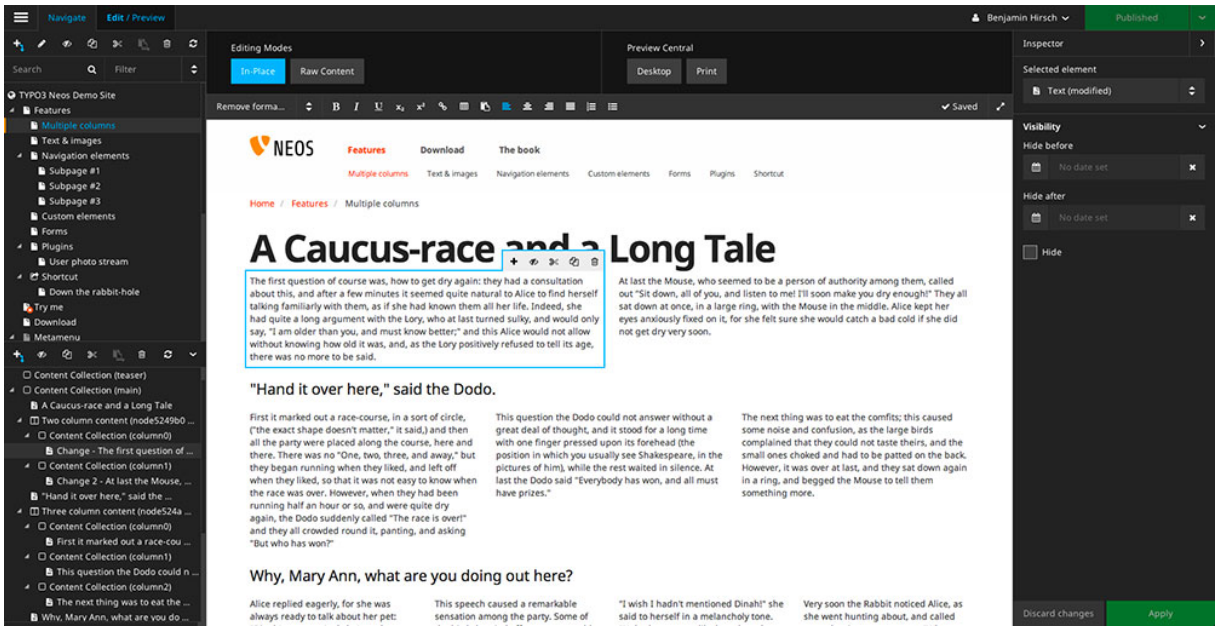

Abbildung 2: Neos Backend im Seitenbearbeitungs-Modus [HDB14]

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

2.3 Joomla

Joomla ist ein weiteres CMS-System für die Erstellung von Webseiten, welches 2005 erstmalig veröffentlicht wurde. 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 Neos verglichen werden, da es einen ähnlichen Ansatz zur Erstellung von Webseiten aufgreift.

Ein Vorteil gegenüber Neos ist, dass Joomla weiter verbreitet ist und eine beachtliche Community hat, welche bei Neos geringer ausfällt. Nachteilig an Joomla ist jedoch, dass es eine Bearbeitung von Web-Inhalten nur im Backend zulässt. Dies bedeutet wiederum, dass 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 Gestaltung der Webseiten ist zwar gegeben, jedoch ist die Erstellung von Erweiterungen wesentlich schwieriger als unter TYPO3 CMS. Es gibt momentan keine garantierte Weiterentwicklung des Portalservers. Auch wenn Joomla den SQL-Standard unterstützt, ist es es nicht offiziell für die Verwendung mit MariaDB freigegeben.

2.4 Drupal

Drupal ist ein CMS-System, welches 2001 veröffentlicht wurde. Es ist momentan in der Version 8.2.3 erhältlich. Das Hauptaugenmerk von Drupal liegt auf der Organisation von Webseiten. Es unterstützt verschiedene Datenbanken wie MySQL, MariaDB, oder PostgreSQL. Mit einem Marktanteil von ca. 2,2% ist Drupal jedoch nicht sehr weit verbreitet, obwohl 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 unterstützt PHP 7. Mit Hilfe so genannter Module kann es erweitert werden. Die Module bassieren auf dem MVC-Prinzip und ähneln in der Programmierung TYPO3-Extensions. Module in Drupal sind aber den TYPO3-Extensions um einiges unterlegen, da sie keinen Ersatz für das Fluid-Framework bieten und es so nicht möglich ist, Abfragen oder Kontrollstrukturen innerhalb von Templates zu realisieren. [Dru16a]

Drupal bietet keinen Support für Enterprise-CMS-Funktionalitäten von Hause aus. Somit müssen viele Use-Cases wie zum Beispiel das Workflow-Management selbst programmiert werden.

2.5 Auswertung der Möglichkeiten

In der folgenden Tabelle werden die bereits genannten Systeme noch einmal aufgezählt und ausgewertet.

Anforderung	Priorisierung	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
Basiert auf Webtechnologien	sehr hoch	✓	✓	✓	✓
Frontend Editing	gering	✓	✓	X	X

Tabelle 1: Tabellarischer Vergleich der betrachteten Systeme

Joomla hat durchaus seinen Charm, doch 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]

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

Leider bietet Joomla keine aktuelle Roadmap und eine Weiterentwicklung ist fraglich. Aus diesem Grund sollte es nicht für die Entwicklung eines neuen Portals verwendet werden, welches zukunftssicher und lange lauffähig 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 Neos TYPO3 CMS in nichts nachsteht wurde sich dennoch gegen die Verwendung von Neos entschieden, da eine vorranschreitende Entwicklung nicht gewährleistet ist. Im Punkto Zukunftssicherheit ist TYPO3 CMS am besten aufgestellt.

Der Vergleich zeigt, dass TYPO3 CMS das beste System istm 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.

Natürlich gibt es noch viele weitere Punkte, in denen die Systeme verglichen werden können. Da aber alle für die Entwicklung eines Supportportals wichtigen Punkte betrachtet wurden, muss an dieser Stelle auf die „CMS Matrix“[2] verwiesen werden.

[2] `http://www.cmsmatrix.org/`

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 7.6, welche aktuell (Stand Dezember 2016) noch immer unterstützt wird. Gleichzeitig 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 Text 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.

Bis zum erscheinen der neuen LTS-Version, soll auch die Seitenbearbeitung im Frontend, wie sie bei NEOS vorhanden ist umgesetzt werden soll. Hierfür wird der Code von NEOS angepasst und zurückportiert.

3.2 Frontend und Backend

Im folgenden Abschnitt wird kurz die Funktionsumfang des Frontends und des Backends von TYPO3 beschrieben.

3.2.1 Frontend

Das TYPO3 Frontend existiert momentan nur zum Anzeigen von Webseiten, redakationellen Inhalten und Extensions. Durch das einbinden von TypoScript und verschiedner Systemextensions ist es möglich ein generisches Menü oder eine Frontendanmeldung beziehungsweise Registrierung zu realisieren.

3.2.2 Backend

Innerhalb des TYPO3 Backends können sehr viele Funktionen realisiert werden. Es können Seiten hinzugefügt, entfernt oder bearbeitet werden. Die Bereitstellung von redaktionellen Inhalten ist natürlich auch möglich. TYPO3 bietet auch die Möglichkeit eine sehr umfangreiche Nutzeradministrierung für Backend und Frontendnutzer durchzuführen. Auch Workflows können innerhalb des TYPO3 Backends abgebildet werden.

3.3 TypoScript

TypoScript ist eine eigens von Kasper Skårhøj 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].

Weiterhin ist es möglich mit Hilfe von TypoScript TYPO3 zu konfigurieren, ohne hierfür große Programmierkentnisse zu besitzen.

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 Ordner noch die 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 überschrieben 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. [Lob16]

3.4 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.

[3] `https://docs.typo3.org/typo3cms/TyposcriptReference/`

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 Controller 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.

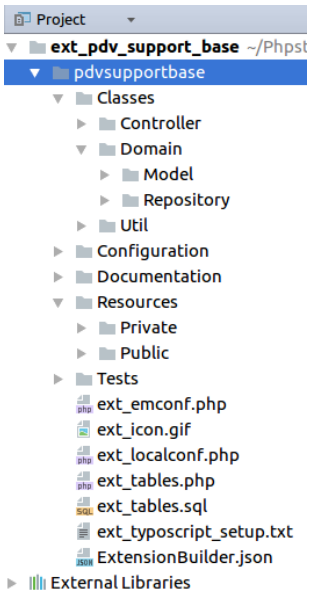

Abbildung 3: Extension Struktur

- **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 Methoden bereit, um Standardabfragen zu realisieren.

Um den Grundstock für eine neue eigene Extension zu legen, kann der Extension-Builder[4] verwendet werden. Dieser baut unter Angabe des Domain-Modells und der Controller 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[5] nachgelesen werden. Lobacher beschreibt in verschiedenen Büchern die Grundlagen für die Programmierung anhand verschiedener TYPO3 Versionen.

3.5 Das TER

Das *TYPO3 Extension Repository* (TER)[6] ist eine von der TYPO3 Association bereitgestellte Sammlung von öffentlichen Extensions für TYPO3. Hier kann jeder Entwickler seine Extension für andere bereitstellen. Die Verwendung von fremden Extensions aus dem TER geschieht jedoch immer auf eigene Verantwortung.

[4] `https://typo3.org/extensions/repository/view/extension_builder`
[5] Typo3 Extbase
[6] `https://typo3.org/extensions/repository/`

3.6 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 3.4) gibt, existiert auch ein Fluid Template, sofern kein Redirect zu einer anderen Aktion durchgeführt wird.

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 3.4).

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

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

Listing 1: Beispiel für ein ViewHelper

Im HTML-Beispiel Listing 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.

3.7 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.

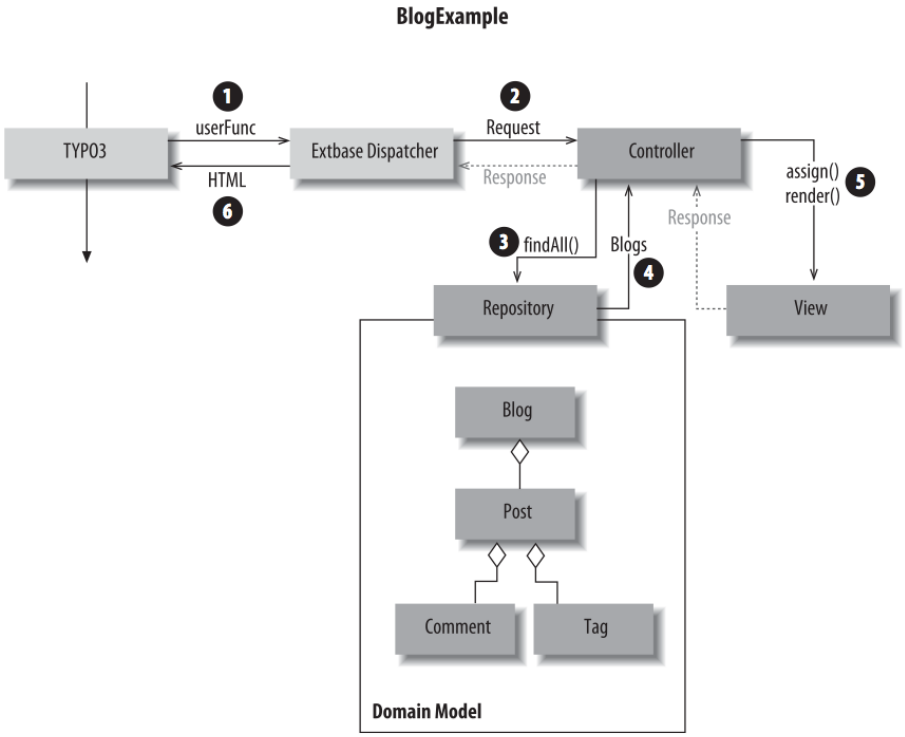

Abbildung 4: Seitenaufruf einer TYPO3 Extension [KR16]

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

Im im Folgenden soll evaluiert werden in wie weit es möglich ist, dass verschiedene Extensions zusammen arbeiten können. Das Supportportal im Ganzen soll aus verschiedenen Extensions bestehen, welche modular auf einander aufbauen.

Es sollen folgende Extensions erstellt werden:

Template	In der *Template* Extension wird das grundlegende Layout des Supportportals definiert. Zusätzlich kommen noch einige ViewHelper und Utility-Funktion hinzu, welche von mehreren anderen Extensions genutzt werden können. Alle anderen Extensions bauen auf dieser auf und sind ohne diese nicht Lauffähig.
SupportBase	Die Extension *SupportBase* stellt das grundlegende Datenmodell zur Verfügung, das andere Extensions nutzen und auch erweitern können. Zusätzlich kümmert sich diese Extension auch um das Login der Nutzer. Dies schließt die Neuanmeldung und das Ändern des Passwortes mit ein.
Polymer	Die *Polymer*-Extension stellt *Polymer*-Elemente (siehe Abschnitt 6.1) in Form von ViewHelpern (siehe Abschnitt 3.6) zur Verfügung. Sie bildet eine Sammlung von *Polymer*-Elementen, welche andere Extensions verwenden können. Alle Extensions, welche *Polymer*-Elemente verwenden, sind von ihr abhängig.
DownloadPortal	Die Extension erweitert das Datenmodell der *SupportBase* und stellt für die Nutzer von PDV-Produkten wichtige Downloads bereit. Zusätzlich bietet sie eine Abo-Funkion an, in der Nutzer sich für bestimmte Download-Kategorien anmelden können. Bei neuen Downloads werden sie dann entsprechend benachrichtigt.
SupportPortal	Die *SupportPortal*-Extension ist das Herzstück des allumfassenden Supportportals. Mit ihr ist es möglich Calls zu Problemen oder Software-Bugs zu öffenen und mit Supportmitarbeitern in Kontakt zu treten.
KnowledgeBase	Die *KnowledgeBase* ist, wie der Name schon sagt, ein Verzeichnis mit bekannten Problemen und Hinweisen zum verwenden der PDV-Software. Sie bezieht sich auf Daten aus der *SupportBase*.
DataExchange	Um den Datenaustaus zwischen Kunden zu gewährleisten, soll die Extension *DataExchange* geschaffen werden. Über sie ist es möglich, dass PDV-Mitarbeiter Kunden Daten zur Verfügung stellen und umgekehrt.
AdminSchulung	Die *AdminSchulung* ist eine Extension, welche es der PDV Systeme GmbH erlaubt Schulungen und Prüfungen für Systemadministratoren online durchführen zu können.

Die Extensions *Template*, *SupportBase* und *Polymer* stellen grundlegende Extensions dar, von denen alle Support-Extensions abhängig sind. Die Support-Extensions *DownloadPortal*, *SupportPortal*, *KnowledgeBase*, *DataExchange* und *AdminSchulung* sind alle voneinander unabhänig und können modular eingesetzt werden.

Innerhalb der vorliegenden Arbeit sollen nicht alle Extensions programmiert werden, da dies aus Gründen des Aufwandes nicht möglich ist. Es soll lediglich Anhand der Extensions *Template*, *Polymer*, *SupportBase* und *DownloadPortal* gezeigt werden, dass modulare Extensions unter Typo3 möglich sind und wie ein Lösungsansatz aussehen kann.

Im weiteren Verlauf wird ebenso auf die genaue Programmierung der einzelnen Extensions verzichtet. Es wird sich auf die Schwerpunkte bezogen, welche wichtig sind um die Extensions modular aufzubauen und wie es möglich ist, moderne Web-Technologien wie *Polymer* oder AngularJS in Typo3 Extensions einzubauen (siehe Kapitel 6).

4.1 Datenbasis

Im Verlauf der Arbeit wurde schon erwähnt, dass die verschiedenen Extensions modular aufgebaut werden sollen. Hierfür muss natürlich auch eine grundlegende Datenbasis geschaffen werden. Diese Basis stellt die Extension *Support Base* zur Verfügung. In Abbildung 5 ist die Datenbasis einmal grafisch dargestellt.

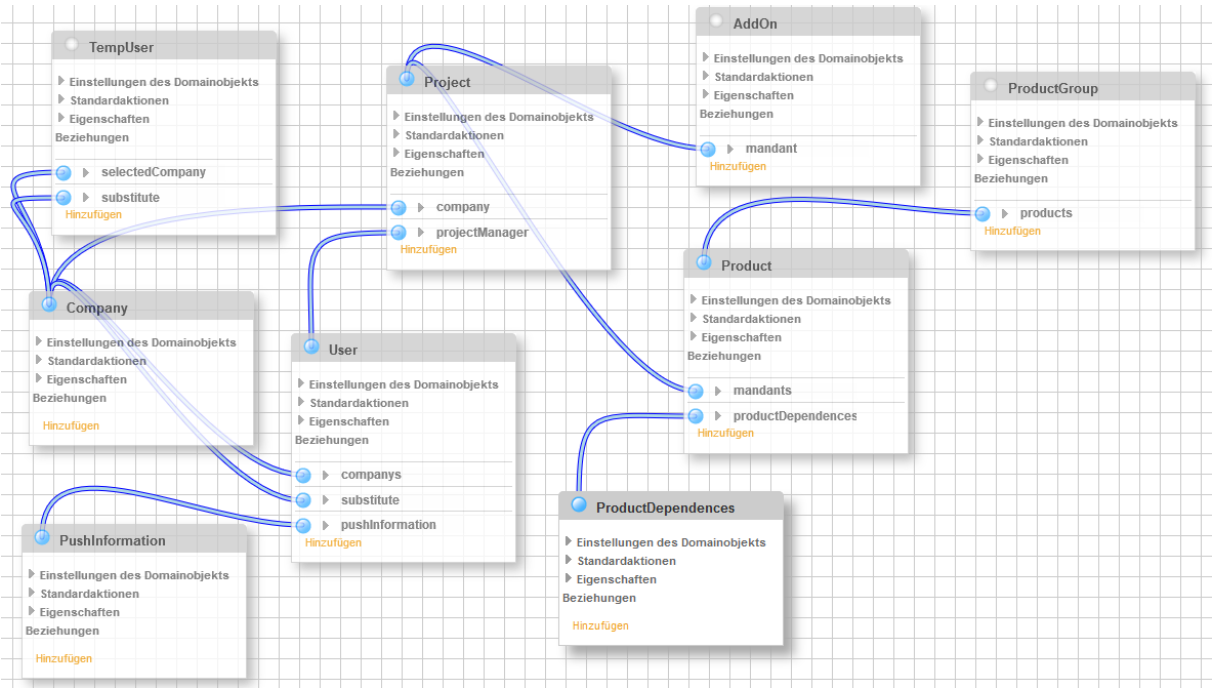

Abbildung 5: Datenbasis der *Support Base* Extension

Die Datenbasis besteht insgesamt aus neun Klassen, wobei die Klassen *TempUser*, *User*, *PushInformation* und *Company* die Frontend-Nutzer abbilden. *Projekt*, *AddOn*, *ProductGroup*, *Product* und *ProductDependences* stellen die Datenbasis für Produktentwicklungen der PDV Systeme GmbH dar.

TempUser ist eine Hilfsklasse, welche innerhalb der Registrierung verwendet wird. Solange der Nutzer nicht freigegeben ist, sind seine Daten als „TempUser“ gespeichert. Sobald er Freigegeben wurde, werden die entsprechenden Daten zu einer Instanz der Klasse *User* ungewandelt und gespeichert. Ein „User“ gehört einem Unternehmen an und kann zusätzlich auch Stellvertetungen (*substitue*) für andere Firmen übernehmen. Zusätzlich werden für einen „User“ auch Informationen gespeichert, welche für die Übertragung der Push-Nachrichten benötigt werden (*PushInformation*).

Produkte (*Product*) innerhalb der PDV werden für Mandanten erstellt, welche innerhalb eines Projekts (*Project*) teilnehmen. Innerhalb eines Projekts können mehrere Firmen als Mandanten

angegeben werden. Produkte wiederum sind in Produktgruppen (*ProductGroup*) angeordnet und können verschiedene Abhängigkeiten (*ProductDependences*) haben.

Zusätzlich zum eigentlichen Produkt, können Projekte auch „AddOns“ (*AddOn*) enthalten, welche ein Produkt erweitern.

Die Klasse *User* erbt von der TYPO3 eigenen Klasse *FrontendUser*. Diese Erweiterung muss innerhalb der Extensionkonfiguration `Configuration/TCA/Overrides` angegeben werden. TYPO3 erweitert bei der Installation der Extension dann die Datenbanktabelle der „FrontendUser“ um die entsprechenden Felder.

Das Datenmodell der *Downloads* Extension wiederum ist in Abbildung 6 zu sehen und erweitert das Datenmodell der *Support Base*.

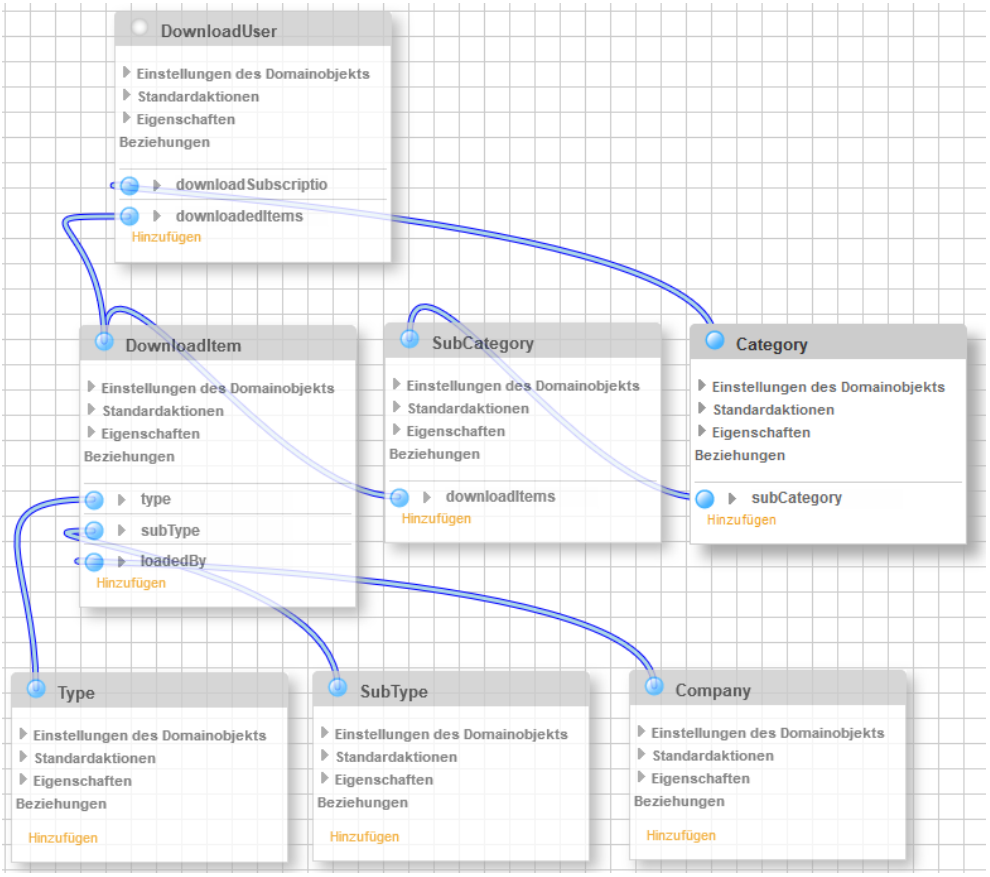

Abbildung 6: Datenbasis der *Downloads* Extension

Die *Downloads* Extension beinhaltet die Klassen *DownloadUser*, *DownloadItem*, *Category*, *SubCategory*, *Type* und *SubType*. Die Klasse *Company* gehört zur *Support Base* Extension und wurde hier nur zur Veranschaulichung hinzugefügt.

DownloadUser erweitert die Klasse *User* der *Support Base* Extension und fügt ihr weitere Attribute hinzu. Ein „DownloadItem“ kann in verschiedene Kategorien (*Category*) und Unterkategorien (*SubCategory*) eingeordnet sein. Zusätzlich enthält ein „DownloadItem“, zur besseren Gliederung noch einen Typ (*Type*) und einen Untertyp (SubType).

Um nachzuverfolgen zu können, welche Downloads von Nutzer und Firmen, getätigt wurden, werden hier zusätzliche Informationen gespeichert. Es kann nachverfolgt werden, welcher Nutzer welches „DownloadItem“ geladen hat. Zusätzlich wird ein Vermerk an das „DownloadItem“ angefügt um nachvollziehen zu können, welche Firmen den Download bezogen haben. Diese Informationen werden von Supportmitarbeitern benötigt, um möglichst schnell sehen zu können, ob eine Firma schon die neusten Änderungen der Software bezogen hat.

Weitere Extensions, welche zur Zeit noch nicht existieren, sollen dieses Prinzip der modularen Datenhaltung weiterverfolgen.

4.2 Rendering

Im folgenden Abschnitt soll dargestellt werden, wie es unter TYPO3 möglich ist ein modulares Rendering aufzubauen. Dies geschieht am Beispiel der oberen Menüleiste des des Templates.

Abbildung 7: Navigationsleiste wenn Nutzer ein nicht eingeloggt ist

Die *Template*-Extension stellt die grundlegende Menüleiste siehe Abbildung 7 dar. Um die oben beschrieben Problematik zu lösen, wurde ein ViewHelper entwickelt, welcher innerhalb der Menüleiste eingebunden werden kann.

Ohne eine Extension, welche das Login bereitstellt (*SupportBase*), kann die *Template*-Extension also keinen Login darstellen. Alle Elemente wie zum Beispiel Buttons, welche modale Dialoge zum Einloggen triggern, müssen also beim Rendern vom Template gesucht und eingebunden werden. Dies betrifft natürlich auch die modalen Dialoge selbst.

Um das eben beschriebene Problemen zu lösen, wurde ein *ViewHelper* entwickelt, welcher innerhalb der Menüleiste eingebunden werden kann. Wird nun das Template mit der Menüleiste gerendert, wird der *ViewHelper* aufgerufen, welcher die entsprechend Elemente in die Menüleiste rendert.

Der *ViewHelper* muss nun alle installierten Extensions nach Elementen durchsuchen, welche innerhalb der Menüleiste gerendert werden sollen. Hierbei muss natürlich darauf geachtet werden, dass es in der Menüleiste drei Arten von Elementen gibt.

- Elemente, die immer geladen werden müssen.
- Elemente, die angezeigt werden, wenn der Nutzer eingeloggt ist.
- Elemente, die angezeigt werden, wenn der Nutzer nicht eingeloggt ist.

Damit die entsprechenden Elemente gefunden und im passenden Fall geladen werden, wurden drei Annotationen geschaffen, mit denen die entsprechenden HTML-Dateien beginnen müssen.

- `@additionHTML`
- `@authenticated`
- `@notAuthenticated`

Um die Suche nach entsprechenden HTML-Dateien zu beschleunigen, wurde festgelegt, dass Elemente nur in Extensions gesucht werden, welche mit „pdv“ beginnen. Somit durchsucht der ViewHelper nur PDV eigene Extensions, welche nach der Konvention mit „pdv“ beginnen. Damit die Suche noch schneller von statten geht, wurde außerdem festgelegt, dass Elemente innerhalb eines festgelegten Pfades zu finden sind. Konkret bedeutet das, dass Elemente innerhalb von Extensions im Pfad `Resources/Private/NavBar` abgelegt werden müssen.

Damit der *ViewHelper* die entsprechenden Element-Arten an den richtigen Stellen einbringt, kann dieser mit Hilfe des Attributs *section* gesteuert werden.

- `add`
- `Auth`
- `noAuth`

```
<div xmlns:f="http://typo3.org/ns/TYPO3/Fluid/ViewHelpers"
     xmlns:t="http://typo3.org/ns/PdvTemplate/ViewHelpers">
    <f:security.ifAuthenticated>
        <f:then>
            <t:navBar section="Auth"></t:navBar>
        </f:then>
    </f:security.ifAuthenticated>
</div>
```

Listing 2: ViewHelper-Aufruf

Im Codebeispiel 2 wird der Aufruf des *ViewHelpers* schematisch dargestellt. Über den *Standard-ViewHelper* `ifAuthenticated` geprüft, ob der Nutzer angemeldet ist. Ist dies der Fall, wird wiederum der eben beschriebene `NavBarViewHelper` mit dem Attribut `section=" Auth"` aufgerufen. Somit sucht der *ViewHelper* beim Rendern nun nach allen Elementen, die angezeigt werden müssen, wenn ein Nutzer angemeldet ist.

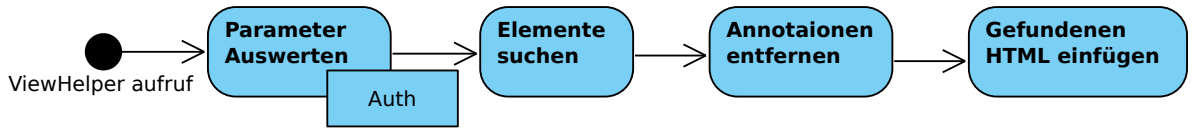

Abbildung 8: Darstellung des NavBarViewHelper

In Abbildung 8 ist die Funkion des `NavBarViewHelper` noch einmal schematisch dargestellt. Der *ViewHelper* wird mit einem Parameter aufgerufen. Dieser wird zuerst ausgewertet, um zu bestimmen, welche Elemente benötigt werden. Anschliesend werden im nächsten Schritt die entsprechenden Elemente gesucht. Als nächstes wird die Annotationen aus den geladenen HTML-Elementen entfernt. Zum Schluss werden alle geladenen Elemente anstelle des *ViewHelpers* innerhalb des angeforderten HTML-Templates eingefügt.

TYPO3 besitzt sehr intelligente Caching-Mechanismen, die ein solches Vorgehen erlauben. Denn durch die Nutzung von Caches erstellt TYPO3 nicht bei jedem Seitenaufruf das Template neu.

Abbildung 9: Navigationsleiste wenn ein Nutzer eingeloggt ist

Ist der Nutzer eingeloggt, so ändern sich die Symbole in der Menüleiste auf der linken Seite (siehe Abbildung).

5 Neue Webtechnologien

Im folgenden Kapitel wird auf neue und zum Teil noch nicht fertig spezifizierte Webtechnologien eingegangen. Die vorgestellten Technologien HTML5, CSS3, PHP7, Web Components, Angular JS, Google Polymer und die Service Worker sollen bei der Entwicklung des Supportportals Verwendung finden.

5.1 HTML5

HTML5 ist die aktuelle Fassung, der Hypertext-Auszeichnungssprache HTML, welche vom W3C am 28. Oktober 2014 standardisiert wurde.

Die erste wichtige Neuerung ist die Angabe des Dokumententyps, welche nun nur noch `<!DOCTYPE html>` lautet.

`section`, `header` und `footer` sind nur ein paar Beispiele für die neuen Struktur-Elemente in HTML5, welche es erlauben, das Dokument besser zu Strukturieren.

HTML5 bietet aber nicht nur neues. Es erweitert zum Teil auch bestehende Elemente, wie den `input`-Tag, in welchem nun auch Typdefinitionen wie zum Beispiel Telefonnummern oder E-Mail-Adressen standardisiert sind. Dies ermöglicht dem Browser eine Validierung von gekennzeichneten Tags vorzunehmen ohne auf Java-Skript zurückgreifen zu müssen.

Abbildung 10: HTML5 Logo des W3C

Da innerhalb der Arbeit nicht auf alle Neuerungen eingegangen werden kann, muss an dieser Stelle auf die HTML5-Definition des W3C[7] verwiesen werden. [HBF14] [HTM16]

5.2 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, ein neues 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.

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“.

[7] `https://www.w3.org/TR/2014/REC-html5-20141028/`

5.2.1 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. Geschriebener 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 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]

5.2.2 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 mehrere Vorteile. So ist es zum einen sehr einfach möglich, das Framework um eigene Style-Objekte 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 und sich zut Zeit noch in der Beta befindet, so ist es doch eine gute Alternative, wenn eine Web-Seite im „Material Design“ umgesetzt werden soll.

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

5.2.3 Vergleich beider CSS-Frameworks

In der Tabelle 2 wurden die beiden Frameworks Bootstrap Material und MaterialiezeCSS gegenübergestellt. Beim Umfang und der Farbanpassung stellt sich ganz Klar heraus, dass MaterialiezeCSS Bootstrap Material weit überlegen ist. Ebenso zeigt die Tabelle, dass MaterialiezeCSS abgesehen von JQuery keine weiteren Abhängigkeiten besitzt.

Zusätzlich lässt sich Subjektiv sagen, dass MaterialiezeCSS schneller und flüssiger arbeitet als Bootstrap Material.

Aus den vorgelegten Gründen wurde innerhalb der PDV Systeme GmbH entschieden für das neue Supportportal MaterialiezeCSS in der Version 0.97.0 (aktuelle Version) zu verwenden. Für die Verwendung innerhalb der PDV Systeme wurde das Framework an die Farben der PDV Systeme angepasst.

5.3 PHP7

PHP7 ist die aktuelle Version der Programmiersprache PHP welche im Dezember 2015 veröffentlicht wurde. PHP7 ist nicht abwärtskompatibel zu alten Versionen und durch eine Neuimplementierung bis zu 30 Prozent schneller als Vorgängerversionen. PHP ist eine Serverseitige-Programmiersprache für die Erstellung von dynamischen Webseiten.[PHP17]

Neue Features in PHP7 sind zum Beispiel die Engine-Exceptions, welche es ermöglichen, innerhalb von PHP PHP-Engine-Exceptions abzufangen. [Sha17]

Da TYPO3 in den aktuellen 8.x-Version auf PHP7 bassiert, muss die Programmiersprache zwingend verwendet werden, um TYPO3-Extensions zu erstellen.

5.4 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 müssen. 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 5) und sind 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 5.4.1).

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]

5.4.1 Shadow DOM

Damit Komponenten nicht durch CSS-Klassen von der eigentlichen Web-Seite oder sich 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 eigentlichen 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 Hauptseite zum Beispiel, können sich nicht auf Elemente im Shadow beziehen, da sie diese nicht kennen.

Aktuell, können alle modernen Browser den Shadow DOM umsetzten. [Pol16b]

5.5 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. Somit ist JQuery nicht mehr notwendig.

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

Die momentan aktuelle Version von Angular ist, abgesehen von 2.x.x Versionen, 1.5.6, welche auch in dieser Arbeit verwendet wird. Angular 2 ist seit September 2016 veröffentlicht, wird aber in dieser Arbeit noch nicht Verwendet, da sich Angular 2 viel mehr an Entwickler von „Single-Page-Applikation“ richtet.

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 3: 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-Tags 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, gesetzt.

```
<script>
var app = angular.module('myApp', []);
```

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

Listing 4: Angular ViewModel und Model im HTML-DOM [W3C16]

Im Listing 4 ist das ViewModel beziehungsweise das Model des im Listing 3 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 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[8] 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.

5.6 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 5.5) 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]

5.6.1 Wozu Polymer Elemente

Polymer Elemente versuchen den so genannten 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

[8] www.aurelia.io

Web ist „DOM Pollution“, was so viel aussagt wie, dass der DOM mit verschiedenen Sprachen verschmutzt ist.

Um ein 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 sind 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.

5.6.2 Was macht Polymer aus

Polymer wiederum geht, im Gegensatz zu den beiden im Abschnitt 5.6.1 genannten, einen dritten sehr eleganten Ansatz und zwar 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[9].

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

[9] `https://elements.polymer-project.org/`

5.6.3 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 ein 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 5: 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.

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, welche 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 gegebenenfalls 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 5 wurden CSS-Abhängigkeiten zu MaterializeCSS (Abschnitt 5.2.2) aufgebaut, eine Begründung hierfür ist im Abschnitt 5.2.3 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 Beginn des Skripts. Dieser Teil muss also immer vorhanden sein, damit das jeweilige Element überhaupt dargestellt werden kann (siehe 5 Zeile 12 und 13).

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 (siehe 5 Zeilen 14 bis 18).

Damit ein Handler überhaupt zum Tragen kommt, muss er innerhalb des HTML-Tags, für das das jeweilige Skript sein soll, registriert werden (siehe 5 Zeile 7).

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

```
1 properties: {
2     owner: {
3         type: String,
4             value: "Daniel"
5     }
6 }
```

Listing 6: Polymer-Element Attribute

Das Attribut „owner“ wird im Beispiel als String definiert und hat den Standardwert „Daniel“. Jedes im Skript definierte 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.

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

Listing 7: Polymer-Element Attribute benutzen

Der Aufruf dieses Polymer-Elements muss wie im Beispiel 7 umgesetzt werden.

5.6.4 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.

5.7 Service Worker

Als Service Worker werden Skripte bezeichnet, welche vom Browser im Hintergrund ausgeführt werden. So können über Service Worker zum Beispiel Push-Nachrichten empfangen oder Hintergrund-Syncronisationen durchgeführt werden. Die entsprechenden Skripte werden beim Laden einer Webseite mit ausgeliefert und vom Browser ausgeführt.

Service Worker sind bis heute noch nicht vollständig standardisiert, bieten aber ein großes Potential, weshalb namenhafte Softwareunternehmen wie Google oder Facebook sie schon seit längeren im großen Stil einsetzen. Da Service Worker hoch sensieble Daten wie zum Beispiel Nachrichten transportieren, sind sie nur unter verwenden von HTTPs einsetzbar. [Gau17]

Service Worker werden zur Zeit schon den aktuellen Versionen von Chrome, Firefox und Opera unterstützt. Für Safari und Edge, ist die Unterstützung momentan noch in der Entwicklung. [Can17] [Is117]

5.7.1 Lifecycle eines Service Workers

Im folgenden Abschnitt wird der Lebenszyklus eines Service Workers beschrieben.

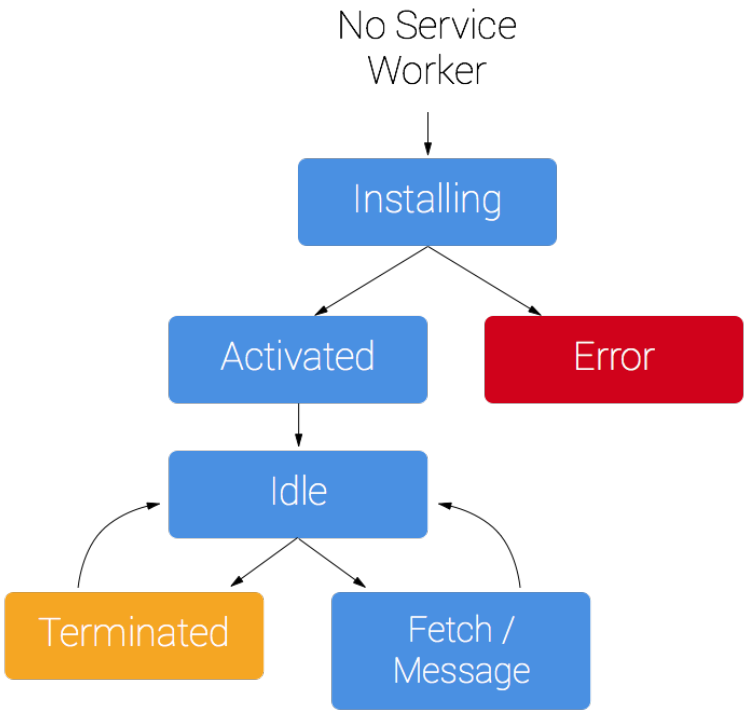

Abbildung 11: Lifecycle von Service Worker [Gau17]

In Abbildung 11 ist der Lifecycle von Service Workern dargestellt. Nach dem Installieren ist der Service Worker aktiv und bereit zum Arbeiten. Geht beim Installieren des Workers etwas schief, so gelangt er in einen Fehlerzustand und wird abgeschaltet. Dies kann zum Beispiel passieren, wenn der verwendete Browser keine Service Worker unterstützt.

Nachdem der Worker Aktiviert wurde, geht er in den Leerlauf (Idle) bis er eine Aufgabe zugeteilt bekommt. Ist dies der Fall erledigt er Service Worker diese und geht danach wieder in den Leerlauf.

Wird der Browser beendet, wird auch die Ausführung des Workers beendet. Sobald der Browser nun erneut gestartet wird, werden auch alle aktivierten Service Worker gestartet. Nutzer müssen sich also keine Gedanken um das Starten oder Beenden der jeweilien Service Worker machen.

5.7.2 Entwicklung eines Service Workers

Zur Entwicklung eines Service Workers sollte der Google Chrome verwendet werden, da dieser die Entwicklung zur Zeit am besten unterstützt. Nur der Chrome erlaubt es momentan, einen Service Worker bei einem Seiten-Reload erneut zu installieren. Dies ist wichtig, da nun nicht nach jedem Entwicklungsschritt der komplette Browser-Cache gelöscht werden muss.

Im folgenden wird erläutert wie Push-Nachrichten über einen Service Worker empfangen werden können.

```
if ('serviceWorker' in navigator && 'PushManager' in window) {
  console.log('Service Worker and Push is supported');

  navigator.serviceWorker.register('sw.js')
  .then(function(swReg) {
    console.log('Service Worker is registered', swReg);

  })
  .catch(function(error) {
    console.error('Service Worker Error', error);
  });
} else {
  console.warn('Push messaging is not supported');
  pushButton.textContent = 'Push Not Supported';
}
```

Listing 8: Service Worker registrieren

Im Codebeispiel 8 wird überprüft, ob der aktuelle Browser Service Worker und Push-Nachrichten unterstützt. Ist dies der Fall, wird der Service Worker, welcher sich in der Datei `sw.js` befindet installiert.

```
self.addEventListener('push', function(event) {
  console.log('[Service Worker] Push Received.');
  console.log(`[Service Worker] Push had this data: "${event.data.text()}"`);

  const title = 'Push Codelab';
  const options = {
    body: 'Yay it works.',
    icon: 'images/icon.png',
    badge: 'images/badge.png'
  };

  event.waitUntil(self.registration.showNotification(title, options));
});
```

Listing 9: Service Worker Hauptfunktion

Innerhalb des Service Worker ist die im Codebeispiel 9 gezeigte Funktion dafür verantwortlich, Nachrichten zu empfangen und dem Nutzer anzuzeigen. Im Codebeispiel werden dem Nutzer Standardtexte angezeigt. Es ist aber auch möglich, Texte und Nutzdaten in der Nachricht einzubetten und diese im Worker auszulesen.

Das gesamte Beispiel zum Empfangen von Push-Nachrichten ist innerhalb der Google Developer Seiten zu finden[10].

[10] `https://developers.google.com/web/fundamentals/getting-started/codelabs/push-notifications/`

6 Zusammenspiel der Technologien

6.1 Integration von Polymer in Typo3

Im Abschnitt 5.6 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 Abschnitt 3.6).

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

In Abbildung 12 (rechts) ist die Aufrufhirarchie zu sehen, welche innerhalb des Projektes entwickelt wurde, um Polymer Elemente elegant in Typo3 einzuarbeiten. Die gezeigte Hirarchie läuft innerhalb der Abbildung 4 im Schitt fünf ab.

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.

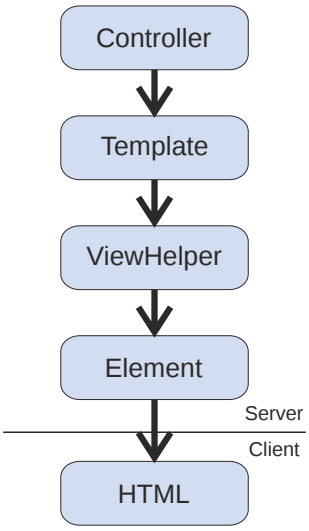

Abbildung 12: Polymer *ViewHelper* Hierarchie

Am Beispiel des Button-Up Elements (siehe Abschnitt 5.6.3), 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 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 MaterialiezeCSS 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 wird entweder in der Konfiguration einer Extension angegeben 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: MaterialiezeCSS Anbindung

6.3 Nutzer Registrierung und Anmeldung

TYPO3 liefert eine Extension zur Anmeldung und Registrierung von Fronten Nutzer bereits mit. Diese Extension genügt jedoch nicht den Anforderungen der Use Cases der PDV.

6.3.1 Use Case Registrierung

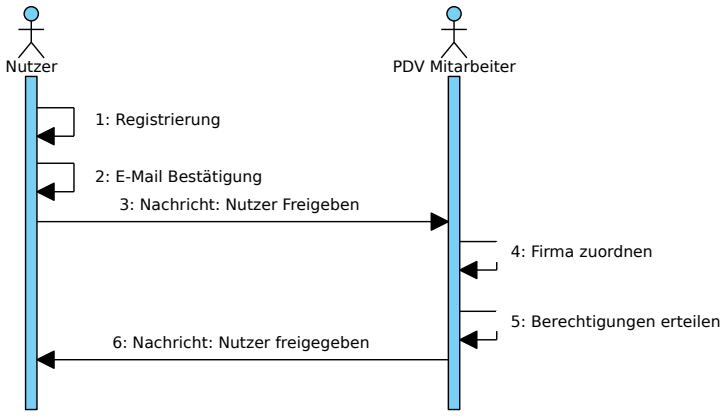

Abbildung 13: Use Case Nutzerregistrierung

In Abbildung 13 ist der Use Case zur Nutzerregistrierung zu sehen. Als erstes registriert sich der Nutzer am Portal. Nach dem Absenden erhält er automatisiert eine Nachricht zur bestätigung der E-Mail-Adresse (Double-Opt-In). Nachdem der Nutzer seine E-Mail-Adresse bestätigt hat, bekommen die Supportmitarbeiter der PDV eine Nachricht, in der sie darüber informiert werden, dass ein neuer Nutzer freigegeben werden möchte.

Ein Mitarbeiter prüft die Plausibilität der Registrierung und gibt den Nutzer für eine Firma (PDV-Kunde) frei. Anschließend muss der Mitarbeiter noch die Berechtigungen für den Nutzer setzen. Ist der Nutzer von einem Mitarbeiter freigegeben worden, bekommt er eine Nachricht, dass er sich nun anmelden kann.

Leider gibt es keine bestehende Extension im TER, die diesen Use Case auch nur im Ansatz umsetzt, weshalb entschieden wurde, die Registration für Nutzer selbst zu entwickeln.

6.3.2 Umsetzung der Nutzerregistrierung

Die Nutzerregistrierung besteht aus einem HTML5-Formular, welches durch den Browser validiert wird (siehe Abschnitt 5.1). Zusätzlich zur Registration im Fronten wurde ein Backend-Modul erstellt, welches es erlaubt, PDV Mitarbeitern die Firmen- und Rechtezuordung umzusetzen (siehe Abbildung 13. 4 und 5).

Die Firmenrichtlinien der PDV Systeme GmbH erlauben es leider nicht an dieser Stelle den gesamten Controller zur Nutzerregistrierung zu veröffentlichen, weshalb an dieser Stelle kein praktisches Beispiel gegeben werden kann.

6.3.3 Die Nutzer Anmeldung

Die TYPO3 eigene Extension *FE_Login* ist nur schwer zu konfigurieren und bietet keine Möglichkeit den Login in modale Dialoge auszulagern. Aus diesen Gründen wurde entschieden, den Login durch einen eigenen Controller innerhalb der *SupportBase*-Extension zu realisieren.

Abbildung 14: Login für Frontend-Nutzer

Hat ein Nutzer das Anmeldeformular 14 Abbildung abgeschickt, gelangen die Anmeldedaten (Nutzername und Passwort) durch TYPO3 in die Controller-Action `loginAction`

```
1  public function loginAction(User $loginUser){
2
3          /** @var User $users */
4          $user = $this->userRepository->findOneByUsername($loginUser->getUserName
               ());
5
6          $objSalt = SaltFactory::getSaltingInstance(NULL);
7
8          //Checke Passwort und login
9          if($objSalt->checkPassword($loginUser->getPassword(),
10             $user->getPassword())){
11             $GLOBALS['TSFE']->fe_user->checkPid = 0;
12             $GLOBALS['TSFE']->fe_user->getAuthInfoArray();
13             $GLOBALS['TSFE']->fe_user->fetchUserRecord($info['db_user'], $user->
                   getUserName());
14
15             $GLOBALS['TSFE']->fe_user->user = $GLOBALS['TSFE']->fe_user->
                   fetchUserSession();
16             $GLOBALS['TSFE']->loginUser = 1;
17             $GLOBALS['TSFE']->fe_user->fetchGroupData();
18             $GLOBALS['TSFE']->fe_user->start();
19             $GLOBALS['TSFE']->fe_user->createUserSession($user);
20             $GLOBALS['TSFE']->fe_user->loginSessionStarted = TRUE;
21         }
22
23
24
25         $linkUrl = GeneralUtility::_GP('redirect_url');
26         HttpUtility::redirect($linkUrl);
27 }
```

Listing 14: loginAction

Im Quellcodebeispiel 14 wird der User anhand des Namens aus der Datenbank geholt (Zeile 4). Anschliesend wird geprüft, ob das angegebene Passwort mit dem in der Datenbank überein stimmt. Ist dies der Fall, so wird die Nutzer-Session in das `$GLOBALS['TSFE']`-Array geschrieben, womit der Nutzer eingeloggt ist. Anschliesend wird der Nutzer zur ursprünglichen Seite, von der aus er sich eingeloggt hat, umgeleitet.

Wie die modalen Dialoge in das Template gerendert werden, wurde im Abschnitt 4.2 erklärt.

6.4 Push-Nachrichten über Service Worker

Im Abschnitt 5.7 wurde beschrieben, wie ein Service Worker erstellt wird, um Nutzern Push-Nachrichten anzuzeigen. Nachdem die Clients konfiguriert wurden, um Nachrichten zu empfangen, muss natürlich nun auch ein Sender innerhalb der Servers umgesetzt werden.

Um das Senden von Push-Nachrichten zu ermöglichen, gibt es eine Reihe von Librarys für unterschiedliche Programmiersprachen. So auch auch eine für PHP[11], welche innerhalb von TYPO3 verwendet werden kann.

Bisher konnte das Versenden und Empfangen von Nachrichten noch nicht Praktisch umgesetzt werden, da Service Worker wie schon beschrieben nur mittels HTTPs arbeiten. Leider bietet die aktuelle Testumgebung der PDV Systeme keine Möglichkeit, eine HTTPs Verbindung umzusetzten. Aus diesem Grund kann die Einbindung der Library „web-push-php“ nur theoretisch ausgearbeitet werden.

Die Library bietet die Möglichkeit Push-Nachrichten zu erstellen und diese zu versenden. Diese Funktion soll in die *Support Base* Extension integriert werden, da verschiedene Extension in der Lage sein müssen Nachrichten zu versenden.

Es soll eine Wrapperklasse entstehen, welche unter Angabe des „Users“, einer Überschrift, einer Nachricht und einem Link automatisch eine Push-Nachricht erstellt. Die Wrapperklasse soll anhand des „Users“ selbstständig die benötigten Verbindungsdaten aus der Datenbank abrufen und die Nachricht versenden.

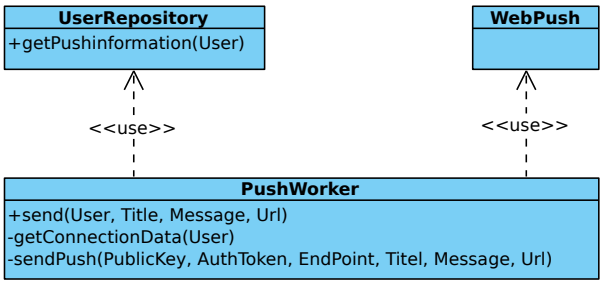

Abbildung 15: Klassendiagramm für PushWorker

In Abbildung 15 ist schematisch dargestellt, wie die Klasse PushWorker aussehen soll. Sie besitzt die öffentliche Methode `send()`, welche eine Push-Nachricht versendet. Sie wiederum ruft die Methode `getConnectionData()` auf, welche unter Verwendung der Methode `getPushinformation()` des *UserRepository* die benötigten Daten für die Push-Nachricht holt.

Sind anschliesend alle Daten vorhanden, wird die Methode `sendPush()` innerhalb der Methode `send()` aufgerufen, welche wiederum unter Verwendung der Klasse *WebPush* aus der *web-push-php*-Library die entsprechende Nachricht generiert und versendet.

[11] `https://github.com/web-push-libs/web-push-php`

7 Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Entwicklung eines modularen Supportportals für die PDV Systeme GmbH auf Basis eines existierenden CMS-Systems.

Es wurden die vier bekannten CMS-Systeme TYPO3 CMS, Neos, Joomla und Drupal anhand vorgegebener Kriterien verglichen um eins für die weitere Entwicklung auszuwählen. Die Entscheidung viel auf TYPO3 CMS, welches die Kriterien am besten erfüllt.

TYPO3 CMS wird anschliesend genauer vorgestellt. Hierbei wird der Schwerpunkt auf die Extension-Entwicklung gelegt.

In der Arbeit wird gezeigt, wie es möglich ist modulare Extensions für TYPO3 CMS zu erstellen. Hierbei wird darauf eingegangen wie das Datenmodell Extensionübergreifend aufgebaut werden kann. Außerdem ist das Rendering ein wichtiger Punkt. Hier wird gezeigt, wie es möglich ist Template-Elemente dynamisch von anderen Extensions zu laden.

Es wird sich damit befasst, wie neue Webtechnologien wie HTML5, CSS3, PHP7, AngularJS, Google Polymer und Service Workern in die aktuelle Version von TYPO3 integriert werden können. Hierfür werden die Technologien zuerst einzeln vorgestellt, um später zu zeigen, wie sie sich mit TYPO3 verbinden lassen.

8 Fazit

Es ist möglich TYPO3-Extensions modular aufzubauen und neue Webtechnologien zu integrieren. Hierfür wurden in dieser Arbeit viele Beispiele gegeben.

Neben der Arbeit entstand ein Grundgerüst für ein modulares Supportportal, welches nun nicht nur technisch auf den neusten Stand ist, es lässt sich zudem einfach erweitern.

Durch die gewonnenen Erkentnisse konnte nicht nur die Basis für ein Supportportal geschaffen werden. Es wurde Testweise auch ein weiteres Modul, das Download-Portal implementiert, um die genannten Behauptungen zu belegen.

Auf der Basis dieser Arbeit wurden weitere Projektmittel durch die PDV Systeme GmbH gewährt, um das nun begonne Projekt weiterzuführen und das neue Supportportal mit den restlichen Modulen zu vervollständigen.

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 Text Editor*

TER *TYPO3 Extension Repository*

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

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

[Can17] Can I use. http://caniuse.com/#feat=serviceworkers, Januar 2017.

[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.

[Gau17] Matt Gaunt. Web Fundamentals. Technical report, Google Inc., Februar 2017.

[HBF14] Ian Hickson, Robin Berjon, and Steve Faulkner. HTML5. Technical report, W3C, Oktober 2014.

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

[HTM16] HTML5. https://de.wikipedia.org/wiki/HTML5, Februar 2016.

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

[Is117] Is ServiceWorker ready. https://jakearchibald.github.io/isserviceworkerready/, Februar 2017.

[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.

[PHP17] PHP Wikipedia. https://de.wikipedia.org/wiki/PHP#PHP_7, Februar 2017.

[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.

[Sha17] Davey Shafik. An Exceptional Change in PHP 7.0. https://www.daveyshafik.com/archives/69237-an-exceptional-change-in-php-7-0.html, Juli 2017.

[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.