Philipp Treffinger BSc

Entwurf eines Web-Content-Management-Systems

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Softwareentwicklung – Wirtschaft

eingereicht an der

Technischen Universität Graz

Betreuer

Univ.-Prof. Dipl.-Ing. Dr. techn. Frank Kappe

Institute of Interactive Systems and Data Science

Graz, Oktober 2018

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Graz, _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

DANKSAGUNG

Ich möchte mich an dieser Stelle bei Herrn Univ.-Prof. Dipl.-Ing. Dr. techn. Franz Kappe bedanken, der mir ermöglicht hat, diese Arbeit zu verfassen.

Ein besonderer Dank gilt auch Karl Kling BSc, Matthias Reischer MSc, Dr. David Treffinger, Mag. Elisa Hochegger und Antonia Prohammer BSc, die mich während der Studienzeit unterstütz haben und mir auch eine große Hilfe beim Erstellen der Masterarbeit waren.

Danke auch an meine Eltern, Mag. Rudolf und Ida Treffinger, die mir das Studium ermöglicht haben.

KURZZUSAMMENFASSUNG

Der Anspruch an eine Administration von Webseiten ist heutzutage geprägt von unterschiedlichsten Zielgruppen und Anwendungsbereichen. Das wiederum fordert Spezialisierungen von Content-Management-Systemen.

In dieser Arbeit werden die Planung und die Umsetzung eines Content-Management-Systems für Webseiten erklärt und diskutiert. Es wird festgestellt, wie sich ein möglichst guter Kompromiss zwischen Benutzerfreundlichkeit, Funktionalität und Flexibilität sowohl im Frontend als auch im Backend finden lässt. Dabei werden alle wesentlichen Entscheidungen diskutiert und argumentiert.

Zu Beginn werden grundsätzliche Aufgaben von Content-Management-Systemen erörtert und gängige Web-Content-Management-Systeme analysiert, miteinander verglichen und Stärken und Schwächen eruiert. Anschließend wird auf die verwendeten Technologien eingegangen und die Entscheidungen für diese erklärt. Danach werden der Aufbau und die Umsetzung des Projekts erklärt und diskutiert. Im letzten Teil der Arbeit wird eine mögliche Weiterführung des Projekts vorgestellt.

ABSTRACT

The present requirements of website administration are influenced by different target groups and application areas, demanding for a specialisation in content management systems.

In this paper, the planning and implementation of a content management system for websites is being explained and discussed. It is being determined, how a good compromise between usability, functionality and flexibility in the front end as well as the back end can be reached. In the course of this all essential decisions are being discussed and reasoned.

In the first part of the paper the basic tasks of content management systems are being argued and common web content management systems are being analysed and compared, evaluating possible strengths and weaknesses. Subsequently the applied technologies are explained in more detail and the decision making is justified. Afterwards the project is introduced, focusing on its structure and implementation. In the last part of the paper a possible continuation of the project is presented.

INHALTSVERZEICHNIS

1. EINLEITUNG

Die Idee für dieses Projekt entstand durch meine Arbeit als Webentwickler und Webdesigner, vor allem die Kombination dieser beiden Tätigkeiten war ausschlaggebend für die Themenwahl.

Ich habe bei meiner Tätigkeit als Webdesigner erfahren, dass es aus technischer und/oder zeitlicher Sicht oft kaum möglich ist, das Designkonzept im vollen Umfang mit möglichst vielen Details zu realisieren. Gründe dafür gibt es viele und viele davon werden vermutlich immer Kompromisse fordern, angefangen von den Inhalten, die dem Designkonzept nicht entsprechen, bis hin zu der Vielzahl an Endgeräten, auf denen das Design funktionieren muss. Für mich persönlich war die Wartbarkeit von Dritten ein besonders großes Hindernis, um mein Designkonzept so zu realisieren, dass der Webauftritt, trotz unterschiedlichen Inhalts, dauerhaft diesem entspricht. Außerdem hatte für mich die Einfachheit der WCMS-Benutzung eine gleich hohe Priorität wie das Design und das Anwendungskonzept des Frontends. So war es mein Ziel, auf der einen Seite dem Besucher des Webauftritts einen möglichst guten Kompromiss zwischen Design, Übersicht und Anwenderfreundlichkeit zu bieten, auf der anderen Seite musste ein dementsprechendes Konzept für die Personen gefunden werden, die den Inhalt generieren und warten. In diesem Fall war ein Kompromiss zwischen Einfachheit und Funktionsvielfalt zu finden, der aber möglichst viele Anwendungsfälle abdecken sollte. Die bestehenden Web-Content-Management-Systeme, mit denen ich gearbeitet oder mit denen ich mich beschäftigt habe, wurden diesen Aufgaben nicht immer gerecht, und daraus wiederum resultierte die Idee für diese Arbeit.

Das Konzept für das Projekt entstand schon vor ca. 5 Jahren und wurde immer wieder durch Erfahrungen, sowohl im Anwendungsbereich als auch in der Webentwicklung, verändert, und alte Konzepte wurden vor der Umsetzung oder nach kurzer Zeit der Erprobung wieder verworfen. Auch ein anfänglicher Idealismus, alles selbst zu entwickeln (Framework etc.), war ein großes Hindernis, in absehbarer Zeit ein derartiges Projekt zu realisieren, wenngleich die aufgewendete Zeit viel Erfahrung brachte. Erst die genaue Planung, die daraus abgeleiteten Teilschritte und die Entscheidung dieses Projekt als Teil meiner Masterarbeit zu realisieren, machte diesen großen Schritt zur Umsetzung möglich.

Diese Arbeit stellt nur das Konzept und ein mögliches Umsetzungskonzept dar. Eine vollständige Umsetzung würde den Rahmen dieser Arbeit nicht entsprechen. Ein anderer Ansatz wäre, ein bestehendes WCMS zu erweitern, um mein Anwendungskonzept zu realisieren. Ich entschied mich aber gegen diesen Weg, da es schwierig ist, ein einheitliches und performantes System zu schaffen, wenn man so weitreichend in ein bestehendes System eingreift. Außerdem würde es meines Erachtens nach dazu führen, dass das Benutzererlebnis negativ beeinflusst wird, wenn zwei Konzepte in einem System vereint werden.

2. Zielsetzung und Forschungsfragen

Zielsetzung dieser Arbeit ist die Entwicklung eines Konzepts und die teilweise Umsetzung eines Web-Content-Management-Systems für private Webseiten bis hin zu Webauftritten von mittleren Unternehmen. Das Web-Content-Management-System soll dabei einen guten Kompromiss zwischen Benutzerfreundlichkeit, Funktionalität und Flexibilität finden. Dabei spielt nicht nur das Konzept, sondern auch die Wahl der verwendeten Technologien eine tragende Rolle. Um dieses Ziel zu erreichen, sollen auch bestehende Web-Content-Management-Systeme analysiert werden.

Daraus ergeben sich folgende Forschungsfragen, die aufeinander aufbauen und in dieser Arbeit beantworten werden sollen.

Frage 1

Welches Konzept für ein Web-Content-Management-System bietet einen guten Kompromiss zwischen Benutzerfreundlichkeit, Funktionalität und Flexibilität im Frontend, im Backend und in der Programmierung?

Frage 2

Welche Technologien eigenen sich für die Umsetzung eines solchen Projekts?

Frage 3

Welche Webseiten lassen sich mit einem solchen Konzept abbilden und welche nicht?

Frage 4

Welche Vorteile und Nachteile ergeben sich aus dem Konzept im Vergleich mit heutigen gängigen Web-Content-Management-Systemen?

2.1 Methode

Zuerst werden allgemeine Anforderungen an ein Content-Management-System analysiert und heute gängige Web-Content-Management-Systeme vorgestellt sowie deren Vor- und Nachteile eruiert. Danach werden die Anforderungen an das hier entwickelte System definiert und diskutiert. Dann wird das Konzept erklärt und dabei argumentiert, warum es den geforderten Anspruch erfüllt. Anschließend werden die verwendeten Technologien vorgestellt und die Umsetzung erklärt. Am Ende werden die Vor- und Nachteile des Systems diskutiert und ein Vergleich mit gängigen Systemen aufgestellt.

3. GRUNDLAGEN

3.1 CONTENT

Es wird immer wieder versucht, verschiedene Konzepte zu entwickeln, um zwischen Daten, Information, Content und Wissen zu differenzieren, aber diese Konzepte scheitern meist an der Abgrenzung der einzelnen Begriffe. Wichtig ist in diesem Zusammenhang vor allem der Unterschied zwischen Rohdaten und Content. Der Hauptunterschied liegt in der Erstellung und in der Nutzung. So umfasst der Prozess zur Generierung von Content verschiedene Teilprozesse wie modellieren, bearbeiten, überprüfen, vergleichen, kontrollieren. Daraus resultieren wiederum viele Versionen des Contents, der unter Umständen auch immer wieder Veränderungen unterliegt. Außerdem ist der Mensch dabei ein wesentlicher Faktor. Viele Informationen in einem Artikel werden durch subjektive Einflüsse unterschiedlich wiedergegeben und der Content wird bewusst oder unbewusst interpretiert. Content wird erstellt, um von einem anderen Menschen in Zukunft – egal in welcher Form – gelesen zu werden. Dabei ist natürlich unerheblich, wo sich diese Person befindet. Anders wäre es beispielweise bei einem Online-Einkauf. Diese Daten werden nicht generiert, um in Zukunft von einem anderen Menschen konsumiert zu werden. Sie fließen höchstens in die Erstellung eines Reports ein, sie sind also nicht zukunftsorientiert (Vgl. Barker, 2016).

Deane Parker definiert Content wie folgt:

„Content is information produced through editorial process ultimately intended for human consumption via publication.“ (Barker, 2016, S. 5)

Er sieht darin auch die zwei wichtigsten Aufgaben eines CMS: das Managen und das Erstellen von Content und dessen Veröffentlichung und Auslieferung. Auch von der Definition von Wikipedia, wobei nicht der Anglizismus „Content“ sondern „Inhalt“ verwendet wird, lassen sich die zwei oben genannten Aufgaben ableiten.

„Als Inhalt wird in Publikationen, Kunstwerken, der Kommunikation und den Medien, der Informationsgehalt verstanden, der sich an den Endnutzer oder das Publikum wendet und Träger von Bedeutungen ist. Inhalt selbst ist immateriell, auf einen Empfänger gerichtet und kontextabhängig, jedoch bedingt von materiellen Trägern und sozialen sowie historischen Aspekten.“ (Wikipedia - Medieninhalt, 2017)

3.1.1 Erstellung von Content

Bei der Erstellung von Content gibt es auch eine Vielzahl unterschiedlicher Fragestellungen, die zu Beginn bereits beantwortet werden müssen. Unter anderem:

- Um welches Thema handelt sich?
- Welche Zielgruppen kommen in Frage?
- Von welchem Blickwinkel aus soll das Thema behandelt werden?
- Wie lange soll der Content sein?
- Wird der Content veröffentlicht?
- Wer hat Zugriff auf den Content?
- Wer darf den Content bearbeiten, kontrollieren etc.?

Viele dieser Fragestellungen können von Computersystemen wegen ihrer Subjektivität nicht beantwortet werden. Das System muss es ermöglichen, diese Entscheidungen dem Autor zu überlassen und den Content darauf basierend zu verarbeiten. Der Prozess der Content-Generierung ist niemals vollständig abgeschlossen. Im Gegensatz dazu unterliegt der Prozess des Onlineeinkaufs objektiven Kriterien, und die daraus resultierenden Daten müssen nicht evaluiert werden, sondern sind eindeutig und unterliegen keiner Interpretation.

3.2 CONTENT-MANAGEMENT-SYSTEM (CMS)

Ein CMS ist eine Software, die die Aufgabe übernehmen soll, Content effizient zu verwalten und zu veröffentlichen. Solche Systeme sind meistens serverbasierend und erlauben es, von unterschiedlichen Nutzern unterschiedlich benutzt zu werden. Die Daten liegen oft ebenfalls auf dem Server, auf dem die Software läuft oder werden separat abgelegt. Die Software soll die Benutzer bei der Erstellung vom Content optimal unterstützen. Dabei sollen verschiedene Werkzeuge bereitstehen, um die einzelnen Teilschritte (Validierung, Versionierung, Modellierung etc.) zu erleichtern. Außerdem stehen die Veröffentlichung und die Auslieferung im Fokus eines CMS und auch dafür gibt es verschiedene Konzepte und Spezialisierungen. Es gibt keine genaue Definition wie ein CMS zu funktionieren hat und welche Aufgaben es genau erfüllen soll. Es gibt viele Ansätze und Anwendungsfälle, die durchaus ihre Berechtigung haben. „Best Practice“ gibt es nur in ihrer Umsetzung (Datenmodell, Infrastruktur etc.).

3.2.1 Kernaufgaben eines CMS

Content-Management-Systeme müssen unterschiedlichen Ansprüchen gerecht werden. Folgende Fragestellungen dienen zur Orientierung und als Hilfe für einen Systementwurf.

Rechtesystem

- Wer sieht den Content?
- Wer kann ihn bearbeiten?
- Wer kann ihn löschen?
- Wer darf in veröffentlichen?

Verknüpfungen

- Wie hängt Content zusammen?
- Welche Daten (Bilder, Grafiken etc.) sind mit dem Content wie verknüpft?

Verwaltung

- Wo liegen die Daten?
- Wo werden sie angezeigt?
- Wie werden sie angezeigt und dargestellt?
- Welche Priorität hat der Content?

Versionierung

- Wie viele Versionen des Contents gibt es?
- Wer hat Änderungen durchgeführt?
- Welche Änderungen wurden von Version zu Version durchgeführt?

Organisation

- Wie kann ich den Content finden (zum Beispiel Tagging, Gruppierung)?
- Wie ist der Content einzuordnen beziehungsweise kategorisiert?

3.2.2 Arten von Content-Management-Systemen

Es gibt verschiedene Arten von CMS. Ich möchte hier kurz auf 5 eingehen. Natürlich sind sowohl die Ansätze als auch die Aufgaben der verschiedenen Arten ähnlich. Es handelt sich hierbei nur um Spezialisierungen. Ein CMS kann auch für mehrere Anwendungszwecke eingesetzt werden, so kann man zum Beispiel mit Typo3[2] mehrere dieser Spezialisierungen bedienen.

3.2.3 Web-Content-Management (WCM)

Ein solches CMS wird vor allem für die Erstellung und Veröffentlichung von Content auf Webseiten verwendet. Dabei fällt auch der Darstellung und dem Design eine tragende Rolle zu. Das System soll helfen, Inhalte so zu generieren, dass sie mit dem Design, den Darstellungsmöglichkeiten und der Struktur des Frontends übereinstimmen, ohne das Anwendungskonzept des Frontends zu beeinträchtigen. Diese Diskrepanz bestmöglich zu lösen ist meines Erachtens eines der schwierigsten Disziplinen bei dem Entwurf eines WCMS. Die Anpassungsfähigkeit des Designs, der Struktur und des Anwendungskonzepts muss so gestaltet sein, dass möglichst selten Anpassungen am Content vom Autor gefordert werden. Und wenn eine solche Anpassung dennoch notwendig ist, dann muss dieser Umstand gut kommuniziert sein. Diese Arbeit handelt von der Konzeption eines solchen WCMS. Eines der verbreitetsten Vertreter eines WCMS ist WordPress[3] mit einem Markanteil von 59,9% (Wikipedia - WordPress, 2018).

3.2.4 Enterprise Content Management (ECM)

In diesem Fall geht es vor allem um firmenrelevanten Content, weniger um dessen Veröffentlichung und Auslieferung. Verschiedene Mitarbeiter schreiben Memos, Berichte etc. oder laden wichtige Dokumente/Dateien hoch. Dabei spielen auch die Rechteverteilung, der Zugriff und das Datenmanagement eine übergeordnete Rolle. ECM schließt natürlich WCMS mit ein. Der Branchenverband AIIM International definiert ECM wie folgt:

[2] https://typo3.org (Stand 6.10.2017)
[3] https://de.wordpress.com (10.10.2017)

„Enterprise-Content-Management umfasst die Technologien zur Erfassung, Verwaltung, Speicherung, Bewahrung und Bereitstellung von Content und Dokumenten zur Unterstützung organisatorischer Prozesse.“ (AIIM, 2017). Vertreter solcher Lösungen sind Censhare[4] oder als Open-Source-Variante Alfresco[5].

3.2.5 Digital Asset Management (DAM)

DAM ist ein CMS speziell für die Verwaltung von digitalen Inhalten wie Bilder, Grafiken, Textdateien etc. Der Fokus dabei liegt auf der Erhebung und Verwaltung von Metadaten, der Aufbereitung, der Standardisierung und des Datenmanagements. Ein kommerzieller Vertreter einer solchen Software wäre Canto Cumulus[6] und eine Open-Source-Alternative dazu Phraseanet[7].

3.2.6 Records Management

Records Management dient der Speicherung und Verwaltung von verschiedenen Geschäftsvorgängen und -ergebnissen. Dabei spielen vor allem die Zugriffskontrolle und die Speicherung von Daten eine wichtige Rolle. Dafür gibt es auch eine ISO Norm (ISO 15489). In dieser ist das Records Management wie folgt definiert: „als Führungsaufgabe wahrzunehmende, effiziente und systematische Kontrolle und Durchführung der Erstellung, Entgegennahme, Aufbewahrung, Nutzung und Aussonderung von Schriftgut einschließlich der Vorgänge zur Erfassung und Aufbewahrung von Nachweisen und Informationen über Geschäftsabläufe und Transaktionen in Form von Akten.“ (Wikipedia - Schriftgutverwaltung, 2017)

3.2.7 Learning-Management-System (LMS)

Bei einem solchen CMS steht das Verwalten von Lerninhalten und dessen Organisation im Vordergrund. Außerdem ist auch die Vermittlung der Lehrinhalte an die lernende Person ein zentraler Punkt. Das kann zum Beispiel spielerisch stattfinden oder durch die Lösung von

[4] https://www.censhare.com/de (10.10.2017)
[5] https://www.alfresco.com/de/ (10.10.2017)
[6] https://www.canto.com/de/ (Stand 26.11.2017)
[7] https://www.phraseanet.com/en/phraseanet/product (Stand 26.11.2017)

gestellten Aufgaben, die anschließend von der lehrenden Person bewertet werden. Viele Universitäten und Schulen verwenden solche Plattformen, um den Lehrbetrieb zu entlasten. Diese Systeme sind meist sehr komplex und zeichnen sich durch ein komplexes Rechte- und Verwaltungssystem aus.

Im Wesentlichen ist die Entscheidung, welches CMS verwendet wird, abhängig davon, wo sich das CMS positioniert, welche Kernaufgaben es erfüllt, um welchen Content es sich handelt, wie komplex es ist und sein muss und welche Anwendungsfälle es abbilden kann. Außerdem spielen Sicherheit und bestehende Infrastruktur eine wesentliche Rolle.

3.3 USABILITY

3.3.1 Definition

Usability (Benutzerfreundlichkeit) ist ein oft diskutierter Begriff, von dem es viele Definitionen gibt. Auf die Softwareentwicklung bezogen geht es im Grunde um das Anwendungskonzept eines Programms. Wie können bestimmte Aufgaben mit dem Programm möglichst einfach und komfortabel erledigt werden? Dabei spielen aber auch das Benutzererlebnis und emotionale Aspekte eine wichtige Rolle. Usability-Ergonomie ist Teil einer jeden Produktentwicklung und ist daher sehr breitgefächert. Die Disziplin der Usability ist von größter Bedeutung, da sie über das Scheitern oder den Erfolg eines Produkts entscheidet. Auch wenn ein Produkt technisch perfekt ist, scheitert es, wenn das Anwendungskonzept umständlich und nicht durchdacht ist. Dazu ein Zitat von Jakob Nielsen:

„Bad Usability Equals No Customers“ (Nielsen, 1998)

Die Internationale Organisation für Normung (ISO) definiert Usability wie folgt:

„Usability eines Produktes ist das Ausmaß, in dem es von einem bestimmten Benutzer verwendet werden kann, um bestimmte Ziele in einem bestimmten Kontext effektiv, effizient und zufriedenstellend zu erreichen.“ (Stieglitz, 2008)

Michael Gralak und Thorsten Stark schreiben in ihrem Buch „Schnelleinstieg App Usability“ aus dem Jahr 2015, dass es darum ginge, mit wenig Aufwand und Belastung die gewünschten Aufgaben erfüllen zu können. Hierbei definieren sie „Belastung“ anhand einer Checkliste.

- Kognitiv – benötigter Lernaufwand, um alles zu verstehen.
- Motorisch – das Durchführen der benötigten Gesten und Klicks.
- Psychisch – Wirkung auf den Benutzer im Allgemeinen, über Datensicherheit bis hin zu den verwendeten Farben und ihrer Wirkung.

Sie sind der Meinung, dass folgende Begriffe zum Thema Usability immer im Hinterkopf behalten werden müssen: Nutzerfreundlichkeit, Benutzer, Ziel und Effektivität. Für sie fasst es Prof. Dr. Simon Nestler von der Technischen Universität München (TUM) am besten zusammen: „Die zentralen Erkenntnisse der Usability-Forschung sind auf jede Software anwendbar, insbesondere auch für Webanwendungen:

- Achten Sie auf Ihren Anwender.
- Verstehen Sie seine Bedürfnisse.
- Unterstützen Sie seine Ziele."

Vgl. (Gralak & Thorsten Stark, 2015)

3.3.2 Usability im Wandel

Usability ist ständigen Innovationen unterworfen. Die Anwendungskonzepte ändern sich immer wieder und werden einfacher oder erweitern sich. Trotz allem ist es von großer Bedeutung, bei dem Entwurf des Anwendungskonzeptes auf die Konventionen etablierter Konzepte zu achten und sich auch daran zu orientieren. So wäre es zum Beispiel keine gute Idee, eine App für iOS[8] zu entwickeln, bei dem die Geste zum Vergrößern und Verkleinern von Bildern anders gestaltet ist, als es ein User von iOS gewohnt ist. Er müsste sich neu daran gewöhnen und wäre auch ständig mit zwei unterschiedlichen Konzepten konfrontiert. Der Vorteil daran ist, dass man bestimmte Konventionen voraussetzen kann. Usability muss aber auch nicht immer selbsterklärend sein. Es geht darum, dass die Software ein Werkzeug darstellt, um die Ziele des Benutzers effizient zu erreichen beziehungsweise um ihn dabei möglichst gut zu unterstützen. So ist das Erstellen von Polierplänen eines Architekten keine triviale Aufgabe und schwer umsetzbar mit einer sich selbsterklärenden Software. Hier muss sich der Benutzer eingehend damit beschäftigen und sich dafür ausbilden lassen. Dabei muss die Software übersichtlich, stabil und zuverlässig funktionieren, damit ein Benutzer, der darin

[8] https://de.wikipedia.org/wiki/IOS_(Betriebssystem) (Stand 3.10.2017)

geschult ist, möglichst flexibel und detailliert arbeiten und zu guten Ergebnissen kommen kann.

3.3.3 Usability im Web

Die Struktur einer Webseite lässt sich in drei Teilbereiche einteilen: den Inhalt einer Seite, also den Content, die visuelle Gestaltung, also das Design, und die Anordnung der Seiten zueinander, also die Struktur (Content-Design, Page-Design und Site-Design) (vgl. Nielsen, 1998).

3.3.3.1 Content

Der Content im Kontext der Usability stellt die Informationen dar, nach denen der Benutzer sucht. Damit man diese auch vorfindet, müssen sie dementsprechend aufbereitet werden. Informationen, die nicht gefunden werden können, sind nutzlos. Der Content sollte auch logisch verknüpft und klar strukturiert sein. Nur so ist es dem Benutzer möglich, die Informationen zu finden, die er benötigt beziehungsweise zu weiterführenden Informationen zu gelangen, ohne einen hohen Ressourceneinsatz. Je schneller und genauer der Benutzer die Informationen, die er benötigt, auffindet, desto effizienter ist das Content Management der Webseite.

3.3.3.2 Visuelle Gestaltung

Die Aufgabe des Designers ist es, den Inhalt ansprechend darzustellen und dafür zu sorgen, dass der Text und die Medieninformationen möglichst ansprechend konsumiert werden können. Die Darstellung des Contents ist stark abhängig vom Inhalt. Ein Photograph, der seine Bilder, oder ein Künstler, der seine Werke präsentieren will, wird ganz andere Ansprüche an seinen Webauftritt stellen als ein Blogger oder ein Journalist, der diverse Artikel veröffentlichen will. Die visuellen Gestaltungsmöglichkeiten sind vielseitig, und daher sollte der Content eine Entsprechung im Design finden.

3.3.3.3 Struktur

Die Struktur entscheidet darüber, wie gut der User sich auf der Webseite orientieren beziehungsweise navigieren kann. Die Verknüpfung zu den einzelnen Seiten, der Aufbau der Webseite, der Überblick über den Content und die Navigationsmöglichkeiten sind dabei entscheidend. Aber auch hier ist die Art des Inhaltes maßgeblich. Eine Navigation oder sogar die ganze Struktur können in bestimmten Fällen ungewöhnlich und ineffizient sein, wenn sie Teil des Konzeptes der Seite sind. Solche Webseiten liefern erfahrungsgemäß meist wenig Content und die visuelle Komponente steht an erster Stelle.

4. Analyse bestehende WCMS

Es gibt eine unüberschaubare Anzahl an WCMS. Viele davon sind Open-Source-Projekte. Darunter sind WCMS, die breit gefächert eingesetzt werden können, und jene, die sich auf einen bestimmten Einsatzzweck spezialisiert haben. Das bekannteste und weitverbreitetste WCMS ist mit großem Abstand WordPress[9]. Es dominiert mit einem Markanteil von fast 60% (siehe Abbildung 1). Ich habe mir vier WCMS ausgesucht, die ich näher erläutern möchte. Dabei stehen Einsatzzweck, Sicherheit und allgemeine Vor- und Nachteile im Vordergrund des Vergleichs.

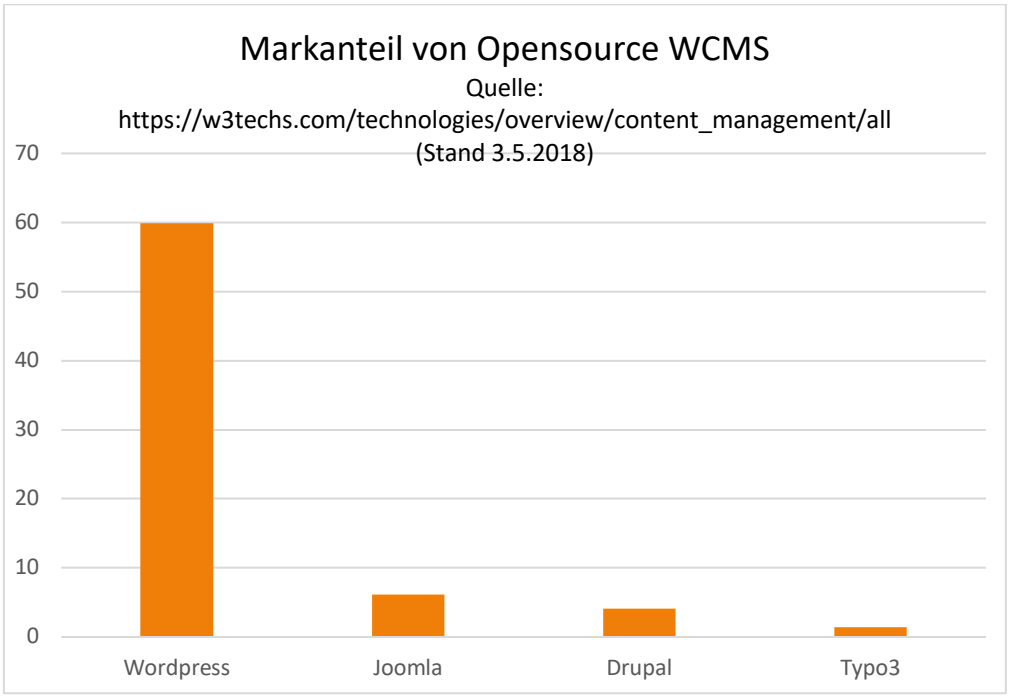

Abbildung 1: Markanteil WCMS

[9] https://de.wordpress.com/ (Stand 3.10.2017)

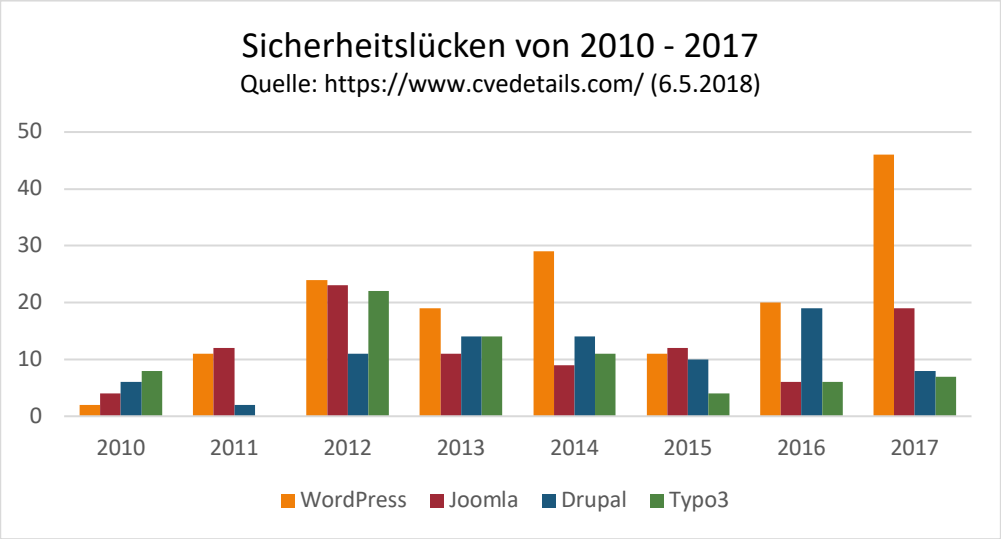

Abbildung 2: Sicherheitslücken von 2010–2017

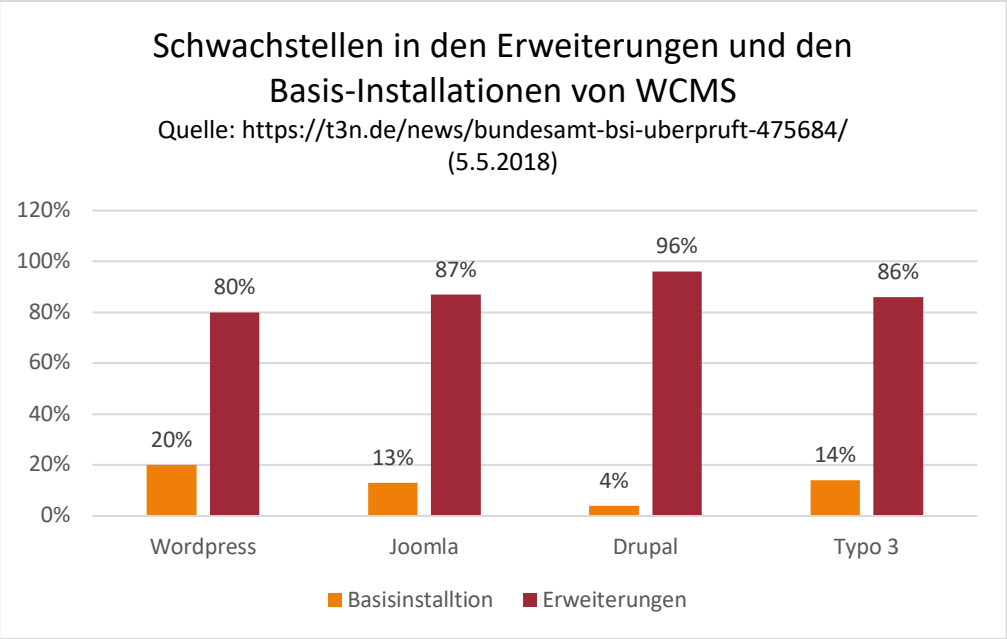

Abbildung 3: Schwachstellen in Erweiterungen und Basis-Installationen von WCMS

4.1 WordPress

Die erste stabile Version von Wordpress erschien 2004 und steht aktuell mit der Version 4.9.5 zur Verfügung (Wikipedia - WordPress, 2018). Ursprünglich als Blog konzipiert, wächst das System stätig und wird mittlerweile für unterschiedlichste Anwendungen eingesetzt.

WordPress hat eine große Community, die hunderte von Plug-Ins zur Verfügung stellt. So lässt sich WordPress beliebig erweitern. Außerdem gibt es hunderte von freien und kostenpflichtigen Themes, die einen professionellen Webauftritt ohne fundierte Kenntnisse erlauben. Texte werden in WordPress als Beiträge oder Seiten erstellt, die mit Kategorien versehen werden können. Diese werden dann in chronologischer Reihenfolge angezeigt. Bilder, Videos, Audiodateien und Dokumente werden in einer zentralen Mediathek verwaltet. Diese können dann zu Beiträgen und Seiten hinzugefügt werden. Das Menü lässt sich per Drag-and-Drop zusammenstellen. Zudem gibt es noch eine Verwaltung von Plug-Ins und Themes. Die Rechteverwaltung ist eher rudimentär. Es gibt nur fix vorgegebene Rollen.

4.1.1 Einsatzmöglichkeiten

Durch die vielen Plug-Ins lässt sich das WCMS für viele Projekte einsetzten, die Möglichkeiten reichen von Blogs über Webshops bis hin zu Webauftritten für kleine bis mittlere Unternehmen. Aber nicht für jedes Projekt, das mit WordPress umgesetzt wird, ist es auch die beste Wahl. Die einfache Benutzung, die vielen Plug-Ins und die große Verbreitung führen dazu, dass viele zu WordPress greifen, ohne sich mit den vielleicht besser geeigneten WCMS zu beschäftigen. Ich selbst hatte immer wieder mit bestehenden Projekten zu tun, die unter einer solchen Fehlentscheidung litten.

4.1.2 Sicherheit

Die große Verbreitung von WordPress wirkt sich negativ auf die Sicherheit aus. Ein weit verbreitetes System ist natürlich ein sehr beliebtes Ziel für Hacker. Eine aktive Community reagiert zwar schnell auf Sicherheitslücken, aber viele solcher Lücken befinden sich oft in Plug-Ins, die nicht mehr oder nur sehr sporadisch gewartet werden. Außerdem muss der Administrator das System ständig aktualisieren, um die bestmögliche Sicherheit gewährleisten zu können.

4.1.3 Vorteile

- Sehr einfache Installation (ca. 5 Minuten)
- Große Auswahl an Plug-Ins und Themes

- Einfache Benutzeroberfläche
- Viele Anwendungsmöglichkeiten
- Große Community

4.1.4 Nachteile

- Beliebtes Ziel für Hacker
- Nur rudimentäre Benutzerverwaltung
- Schlechte Performance
- Komplexität, Benutzerfreundlichkeit und Performance sind stark abhängig von der Anzahl und der Qualität installierter Plug-Ins

4.2 JOOMLA

Die erste stabile Version von Joomla wurde 2005 veröffentlicht (Wikipedia - Joomla, 2018) und ist mittlerweile das zweitbeliebteste WCMS. Die Updatepolitik von Joomla ist unregelmäßig, aber sicherheitsrelevante Updates werden sehr schnell geliefert. Joomla bietet schon in der Basisinstallation einen großen Funktionsumfang, aber es stehen auch viele Erweiterungen, die von einer großen Community bereitgestellt werden, zur Verfügung. Ebenfalls stehen viele kostenlose sowie kostenpflichtige Themes bereit. Die Benutzerverwaltung macht zwar detaillierte Rechteeinstellungen möglich, ist dabei aber nicht sehr einfach zu bedienen. Diese eher kompliziert gestalteten Einstellungsmöglichkeiten findet man auch an anderen Stellen im System. Die Struktur der Webseite besteht aus einzelnen Beiträgen, die sich in Kategorien einteilen lassen. Joomla bietet auch die Möglichkeit, jede Seite in einem anderen Design darstellen zu lassen.

4.2.1 Einsatzmöglichkeiten

Joomla ist schon ohne Erweiterungen sehr vielseitig. Dennoch ist die Bedienung nicht ganz so einfach wie bei WordPress und bedarf schon mäßig erfahrener Anwender. Es eignet sich ebenfalls (vor allem durch die detaillierte Benutzerverwaltung) für größere und vielseitigere Projekte – auch im Businessbereich. Die vielen Erweiterungen der Community tragen dazu bei,

dass sich das System ohne großen Aufwand auf unterschiedliche Anwendungszwecke hin spezialisieren lässt.

4.2.2 Sicherheit

Auch bei Joomla sind die Erweiterungen das größte Sicherheitsrisiko, 87% aller Sicherheitslücken gehen auf diese zurück (siehe Abbildung 3). Die Entwickler der Basisinstallation reagieren schnell auf Sicherheitslücken und stellen schnell passende Patches zur Verfügung, die sich einfach über das Backend einspielen lassen.

4.2.3 Vorteile

- Gute Erweiterbarkeit
- Detaillierte Benutzerverwaltung
- Sehr flexibles Designen möglich
- Die Basisinstallation bring schon viel Sicherheit mit sich

4.2.4 Nachteile

- Medienverwaltung nicht zeitgemäß
- Viele Einstellungen sind kompliziert und unübersichtlich

4.3 DRUPAL

Drupal[10] erschien im Mai 2000 und steht aktuell mit der stabilen Version 8.5.1 zur Verfügung (Wikipedia - Drupal, 2018). Drupal unterscheidet sich in vielerlei Hinsicht von Joomla und WordPress. Es ist eine Art „WCMS Baukastensystem“. Es basiert nicht auf Seiten, sondern es werden sogenannte Inhaltstypen definiert, die mit verschiedenen Feldern bestückt werden können. Mit Views, die sich im Core-Modul befinden, können Datenbankabfragen ganz ohne Programmierkenntnisse erstellt werden. Durch diese Eigenschaften ist Drupal fast kompromisslos flexibel. Zusätzliche Module bieten die Entwickler auf ihrer Homepage an.

[10] https://www.drupal.org/ (Stand 5.5.2018)

4.3.1 Einsatzmöglichkeiten

Aufgrund der hohen Flexibilität von Drupal sind den Einsatzmöglichkeiten kaum Grenzen gesetzt. So lässt sich das WCMS sehr gut an verschiedene Bedürfnisse anpassen. Die Programmschnittstelle von Drupal ist schon in der Basisinstallation sehr ausführlich, lässt sich aber mit Hilfe der verfügbaren Module sehr einfach erweitern. Für Drupal 8 stehen 12.627 (Stand Februar 2017) Module zur Verfügung. Das Spektrum reicht von einfachen Funktionen bis hin zu komplexen, per grafischer Benutzeroberfläche konfigurierbaren Erweiterungen (vgl. Wikipedia - Drupal, 2018, Aufbau und Funktionen). So lassen sich sowohl Weblogs als auch große Online-Communitys realisieren.

4.3.2 Sicherheit

Sicherheitslücken der Basisinstallation werden schnell geschlossen. Module stehen nur auf der offiziellen Seite zur Verfügung und werden dort auf Sicherheit überprüft. Sollten Lücken nicht geschlossen werden, dann werden Module auch gelöscht. Außerdem versucht man, Probleme innerhalb der großen Drupal-Community gemeinsam zu lösen. Aber auch bei Drupal sind hinzugefügte Module das größte Sicherheitsrisiko (siehe Abbildung 3).

4.3.3 Vorteile

- Hohe Flexibilität
- Hohe Sicherheitsstandards
- Gute API für Entwickler
- Große Community

4.3.4 Nachteile

- Lange Einarbeitungszeit
- Erfordert fortgeschrittene Kenntnisse

4.4 TYPO3

Typo3 ist 1998 erschienen und steht aktuell in der stabilen Version 9.2.0 zur Verfügung. Es ist das größte und komplexeste WCMS meiner Auswahl. Die Einrichtung einer Webseite ist wesentlich schwieriger im Vergleich zu Joomla oder Wordpress aber auch im Vergleich zu Drupal. Es eignet sich demnach vor allem für mittlere bis große Projekte, da es für kleine Webseiten überdimensioniert ist und zusätzlich mehr Leistung des Servers voraussetzt. Als Templatesprache kommt TypoScript zum Einsatz. Erweiterungen können ebenfalls nachgeladen werden. Genaue Zahlen sind auf Grund unterschiedlicher Versionen schwer zu finden, aber für Typo3 dürfte es mit Abstand am wenigsten Erweiterungen geben. Dieser Umstand lässt sich vermutlich mit der Verbreitung und der Zielgruppe erklären (siehe Abbildung 1).

4.4.1 Einsatzmöglichkeiten

Typo3[11] ist ein komplexes und für große Projekte ausgelegtes WCMS, welches viele Bedürfnisse wie Multisite-Management mit zentraler Verwaltung des Contents (Content-Syndication) erfüllt. Die Benutzerverwaltung lässt umfangreiche Einstellungen zu. Geeignete Projekte für Typo3 sind zum Beispiel Webauftritte von Universitäten, Online-Markplätze und größere Newsportale.

4.4.2 Sicherheit

Sicherheitsupdates werden in Typo3 schnell geliefert und lassen sich über die Benutzeroberfläche einspielen. Sie werden auch noch für die älteren Versionen 6.2 und 7 ausgeliefert. Typo3 hat zwar in diesem Vergleich am meisten Schwachstellen, daraus lassen sich aber auf Grund der Datenerhebung keine fundierten Aussagen treffen (vgl. Bundesamt für Sicherheit in der Informationstechnik). Aber auch bei Typo3 sind die meisten Sicherheitslücken in den Erweiterungen zu finden (siehe Abbildung 3).

[11] https://typo3.org/ (Stand 12.10.2018)

4.4.3 Vorteile

- Hohe Flexibilität
- Für große und komplexe Projekte geeignet
- Viele Funktionen

4.4.4 Nachteile

- Hoher Ressourcenverbrauch
- TypoScript verlangt viel Einarbeitungszeit

4.5 DISKUSSION

Jedes der vorgestellten WCMS hat seine Besonderheiten. Dabei fällt auf, dass Joomla und Wordpress hinsichtlich ihres Einsatzzwecks am ehesten miteinander zu vergleichen sind. Typo3 ist in diesem Vergleich der „Außenseiter“, da sein Funktionsumfang für mittlere bis große Projekte ausgelegt und für kleine Projekte eigentlich gänzlich ungeeignet ist. Drupal sticht damit hervor, dass es eine Art „Baukastensystem“ mit einer ausgereiften API darstellt. Somit ist es nicht für Benutzer mit keinen oder wenigen Vorkenntnissen ausgelegt. Der von mir vorgestellte Ansatz für ein WCMS soll sich zwischen Drupal auf der einen Seite und WordPress/Joomla auf der anderen Seite positionieren. Es soll die Benutzerfreundlichkeit der Content-Generierung und Medienverwaltung von WordPress mit der Flexibilität von Drupal verbinden. Dabei sind natürlich auch Einschränkungen in Kauf zu nehmen, die im Laufe der Arbeit noch erläutert werden.

5. Sektionsbasierender Ansatz

5.1 Einleitung

Das WCMS soll einen sektionsbasierenden Ansatz verfolgen, der meines Erachtens nach viele (aber natürlich nicht alle) Designkonzepte abbilden kann. Dabei ist es wichtig, dass das WCMS ein klares Konzept verfolgt, um dem Anspruch der Benutzerfreundlichkeit gerecht werden zu können. Verschiedene Konzepte innerhalb eines WCMS können leicht zu einer Vermengung dieser führen. Daraus wiederum resultiert eine große Komplexität und ein uneinheitliches Bedienungskonzept und Designkonzept. Diesen Effekt kenne ich von WordPress, wenn viele unterschiedliche Plug-Ins eingesetzt werden. Außerdem wäre ein solches WCMS auch in seiner Programmierung komplex und damit fehleranfällig, schwer zu warten und wenig performant.

5.2 Konzept

Der Aufbau der Webseite verläuft in diesem WCMS vertikal. Die Elemente (hier Sektionen) können dabei pro Seite untereinander beliebig angeordnet werden. Jede Sektion hat dabei unterschiedliche Eigenschaften und eine sich unterscheidende Strukturierung. Dadurch ergibt sich auch ein unterschiedliches Konzept, die Sektionen zu administrieren. Aus diesem Grund wird im Backend die Administrierung auch auf Basis dieser Sektionen erfolgen. Dabei gibt es Inhaltstypen, die am ehesten mit denen von Drupal verglichen werden können, mit dem Unterschied, dass diese Inhaltstypen nicht im Backend konfiguriert werden, sondern der Entwickler stellt eine Administration für einen Inhaltstypen zur Verfügung, in der der generierte Inhalt in Kategorien eingeteilt wird. Jede Sektion, die einer Seite hinzugefügt wurde, wird ebenfalls mit einer eigenen Administration ausgestattet. Unter anderem möchte ich damit verhindern, dass Tags Teil der Administrierung im Editor werden[12], diese würden die Administration schwieriger und unübersichtlicher machen, außerdem würden sie Programmierkenntnisse erfordern. Durch eine eigens entwickelte Administration kann die Sektion auch performanter sein, ohne dem User zu viel abzuverlangen, sofern der Entwickler auch darauf achtet. In diesem Konzept sind auch Standardadministrationen für eine Sektion und für einen Inhaltstypen vorgesehen, für die sich der Entwickler der Sektion entscheiden

[12] Das Plug-In „MotoPress Content Editor“ verwendet beispielsweise Tags in WordPress (https://motopress.com Stand 24.5.2018).

kann. Wenn eine Sektion mit dem Inhaltstyp einer solchen Standardadministration übereinstimmt, dann kann in der Administration der Sektion über Kategorien entschieden werden, welche Inhalte relevant für die Sektion werden. In diesem Fall besteht die sektionsbezogene Administration nur aus der Auswahl der Kategorien. Die Administrierung der Inhalte erfolgt somit über die Administration des Inhaltstypen. Wenn es eine solche Übereinstimmung nicht geben sollte, dann kann die Administration der Sektion auch für die Generierung der Inhalte selbst zuständig sein.

5.3 Bestandteile im Detail

5.3.1 Seiten

Eine Seite in diesem WCMS ist eine Anordnung von vertikal angeordneten Sektionen. Möchte man eine Seite mit einer Anzeige von Nachrichten und darunter eine Anzeige von Angeboten, dann muss man die Seite einfach mit den entsprechenden Sektionen versehen. Eine Seite kann über einen Link aufgerufen werden und zeigt die der Seite zugeordneten Sektionen an. Jede Seite soll als Single-Page-Applikation realisiert sein. Daher wird die Webseite nur dann vollständig geladen, wenn man zwischen Seiten wechselt, nicht aber während der Benutzung einer Sektion, bei der Benutzerinteraktionen möglich sind, wie beispielsweise bei einem Kontaktformular.

5.3.2 Container

Jeder Seite kann ein Container zugewiesen werden. Dem Container werden ebenfalls Sektionen hinzugefügt, welche als Rahmen dienen und bei allen Seiten angezeigt werden, denen der Container zugeordnet ist. Besonders geeignete Sektionen für einen Container wären zum Beispiel eine Navigation, ein Footer oder ein Header.

5.3.3 Sektionen

Eine Sektion ist ein Element einer Seite. Die Sektionen werden auf einer Seite vertikal angeordnet. Sie verfügen im Backend entweder über eine individuelle Administration oder über eine Standardadministration.

5.3.4 Widgets

Widgets können Sektionen zugeordnet werden, die auch Widgets unterstützen. Ein Widget könnte zum Beispiel ein Kalender sein, der rechts oder links neben dem Inhalt einer Sektion angezeigt wird. Nicht jede Sektion ist dafür geeignet. So wäre eine Slideshow, die die ganze Bildschirmbreite benötigt, nicht dafür geeignet Widgets zu unterstützen. Daher muss der Programmierer einer Sektion darüber entscheiden, ob seine Sektion für Widgets geeignet ist oder nicht.

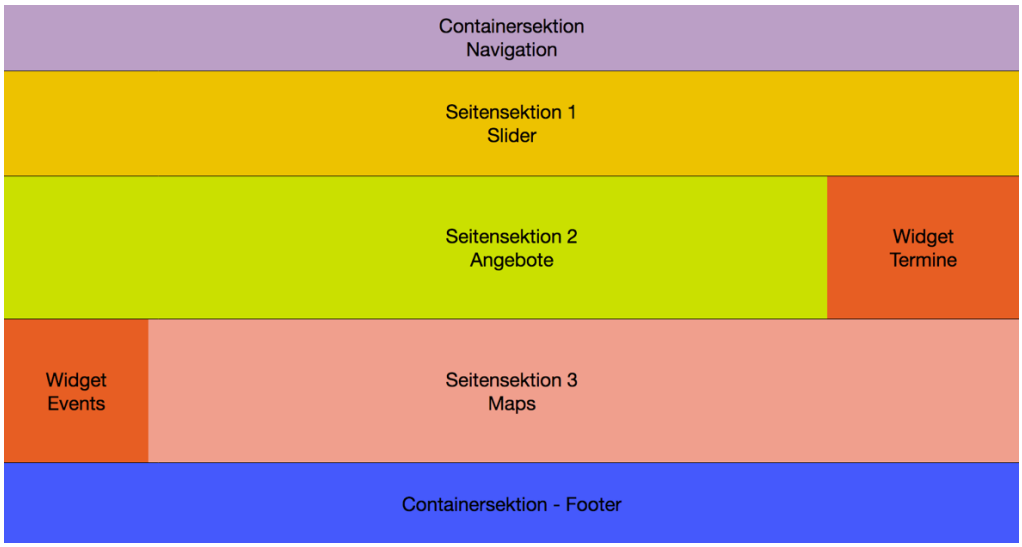

Abbildung 4: Vertikaler Seitenaufbau mit Sektionen

5.3.5 Media Center

Die Media Center ist ein Modul zur Verwaltung diverser Multimediadateien wie Bilder, Grafiken, Dokumente, Videos und Musik. Viele Sektionen werden Multimediadateien einbinden. Dabei will der Benutzer keine redundanten Dateien am Server ablegen, beziehungsweise nicht jede Datei, die am Server bereits gespeichert ist, wird gerade benötigt.

Die Media Center soll den Benutzer dabei unterstützen seine Multimediadateien zu organisieren und zu verwalten und sie jeder Sektion zur Verfügung zu stellen, die diese unterstützt.

Ein weiterer großer Vorteil der einheitlichen Verwaltung von Multimediadateien ist der Aufwand, den sich der Programmierer erspart, wenn er die Organisation und Verwaltung von Multimediadateien pro Sektion nicht selbst lösen muss. Außerdem soll die Media Center den Programmierer dabei unterstützen nur bestimmte Multimediadateien mit bestimmten Eigenschaften zuzulassen, um die Funktion seiner Sektion nicht negativ zu beeinflussen.

6. EINGESETZTE TECHNOLOGIEN

6.1 EINLEITUNG

Das WCMS soll einfach installiert und konfiguriert werden können. Der Fokus liegt dabei auf gut dokumentierte Frameworks und Bibliotheken. Es soll gut erweiterbar sein, daher muss es auch für Dritte leicht möglich sein, sich in das System einzulesen. Die Auswahl bekannter und erprobter Technologien wird die Einarbeitungszeit und die Entscheidung für dieses WCMS positiv beeinflussen. Außerdem sind Sicherheit und Erweiterbarkeit des Projekts stark abhängig von der Wartung und Weiterentwicklung der eingesetzten Technologien. Gerade wenn die Entwicklung eines solchen WCMS nur von wenigen Entwicklern durchgeführt wird, sind die Mittel für eigene Frameworks im Front- oder Backend nicht ausreichend und gefährden auch erheblich die Sicherheit.

6.2. SERVERSEITIGE TECHNOLOGIEN

6.2.1 Serverseitige Programmiersprache

Die Verbreitung von PHP[13] als serverseitige Programmiersprache ist nach wie vor am größten und bei den meisten Webhosting-Angeboten enthalten, und es ist sehr einfach PHP auf ein eigenes System zu installieren. Durch die weite Verbreitung von PHP stehen einem auch eine große Anzahl fundierter Frameworks zu Verfügung, die eine große aktive Community haben und ebenfalls regelmäßig gewartet werden. Zusätzlich gehört PHP bei fast allen Webhosting-Anbietern zur Standardkonfiguration. Daher bietet sich PHP als serverseitige Sprache an.

[13] http://www.php.net/ (Stand 5.5.2018)

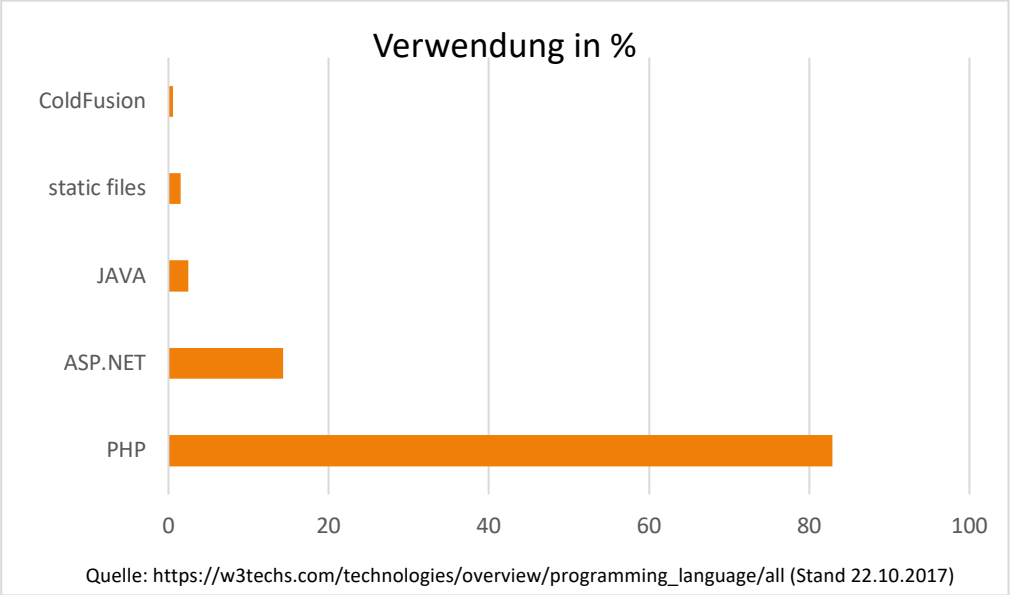

Abbildung 5: Verwendung in % von serverseitigen Programmiersprachen

6.2.2 PHP-Framework

In dem Entscheidungsprozess für ein Framework sollte auf jeden Fall dessen Verbreitung eine tragende Rolle spielen. Die Verbreitung eines Frameworks sagt einiges über folgen Punkte aus:

- Wie groß ist die Community, die Erweiterungen und Hilfestellungen zur Verfügung stellt?
- Wie gut wird das Framework weiterentwickelt und gepflegt?
- Wie einfach ist es, sich in das Framework einzuarbeiten, und wie detailliert ist die Dokumentation?
- Wie flexibel ist das Framework einsetzbar?
- Wie skalierbar ist die damit entwickelte Applikation auch in Zukunft?

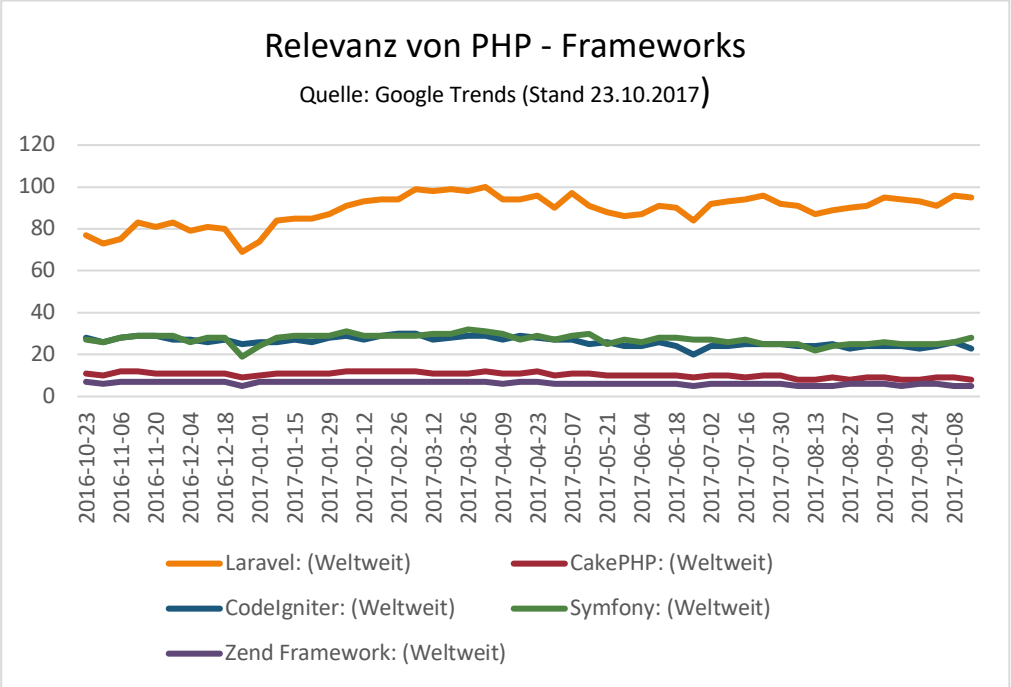

Abbildung 6: Relevanz von PHP-Frameworks

Laravel[14] ist das Framework mit der größten Verbreitung mit steigender Tendenz. Aber für die Wahl des Frameworks müssen noch andere Punkte beachtet werden. Auch das am meisten benutzte Framework muss nicht für jeden Anwendungszweck optimal sein. Folgende Argumente haben dazu geführt, dass ich mich letztlich für CakePHP[15] entschieden habe.

- Das Framework wird gut gewartet und hat eine klare und gute Dokumentation[16].
- Es gibt einen klaren Workflow vor, der auch gut dokumentiert ist.
- Die Namenskonvention für Dateien, Datenbanktabellen und Tabellenfelder sind klar vorgegeben.
- Es beinhaltet eine sehr gute ORM Erweiterung.
- Es beinhaltet ein gutes Plug-In-System, das einen guten modularen Aufbau ermöglicht. Das wiederum macht es leicht, verschieden Module zu teilen, und begünstigt eine

[14] https://laravel.com/ (Stand 17.11.2017)
[15] https://cakephp.org/ (Stand 17.11.2017)
[16] https://book.cakephp.org/3.0/en/index.html (Stand 17.11.2017)

klare Trennung der einzelnen Module. Außerdem hält es den Applikationsordner sauber.

- CakePHP hat ein einfach zu konfigurierendes Routing. Die Routingkonfiguration ist in einer Datei abgelegt und so bleibt sie sehr übersichtlich. Außerdem lassen sich automatische Routen einstellen, sodass Änderungen der Routing-Konfiguration eher selten erfolgen müssen.

Eines der wichtigsten Argumente für mich bei dieser Entscheidung war der strikte Workflow, die strikte Namenskonvention und das Plug-In-System. Das Projekt soll dafür ausgelegt sein, dass man es leicht erweitern kann und dass Module von anderen Entwicklern leicht eingebunden werden können. Dafür sind die oben genannten Eigenschaften maßgeblich.

6.2.3 Datenbank

MySQL (Pröll, Zangerle, & Gassler, 2015) ist nach wie vor das am weitesten verbreitete Datenbanksystem. Dadurch gehört MySQL bei den meisten Webhosting-Anbietern zur Standardkonfiguration und lässt sich auch sehr einfach auf einen eigenen Server einrichten. Die Anforderungen an die Datenbank sind bei diesem CMS eher gering, und daher spielen andere Gesichtspunkte als die Verbreitung eine untergeordnete Rolle.

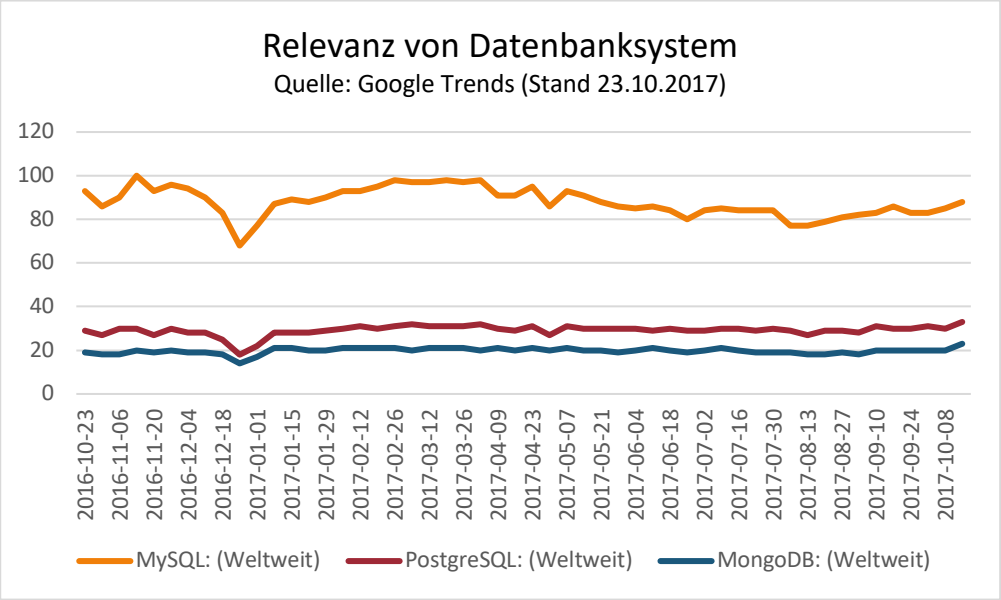

Abbildung 7: Relevanz von Datenbanksystem

6.2.4 Webserver

Für die Verwendung dieses CMS ist es unerheblich, ob nginx[17], Apache 2[18] oder eine andere vergleichbare Webserversoftware zum Einsatz kommt.

6.3. CLIENTSEITIGE TECHNOLOGIEN

6.3.1 Frontend

Im Frontend soll die Auswahl der eingesetzten Technologien dem Entwickler möglichst freistehen. Dabei sollte es unerheblich sein, ob man das Frontend als Single-Page-Applikation umsetzen will oder als klassische Webanwendung. CakePHP verwendet PHP auch als Templatesprache und bietet ebenfalls gute Möglichkeiten, Daten per AJAX/XML nachzuladen.

[17] https://nginx.org/en/ (Stand 2.11.2017)
[18] https://httpd.apache.org/ (Stand 2.11.2017)

6.3.2 Backend

Ich habe mich für Inspinia[19] als Theme für das Backend entschieden. Inspinia ist eine durchdachte Sammlung verschiedenster HTML5, JavaScript und CSS3 Komponenten. Das Theme wird in verschiedenen Varianten angeboten. Ich habe mich für die AngularJS[20]-Variante entschieden, da ich das Backend als Single-Page-Applikation realisieren möchte und dank der Direktiven von AngularJS ist es einfach möglich, jQuery[21] Plug-Ins nahtlos zu integrieren. Außerdem hat AngularJS eine weite Verbreitung unter den MVC-JavaScript-Frameworks (siehe Abbildung 9) und es gibt eine gute Dokumentation und eine große Community für diverse Hilfestellungen. Zu dem Zeitpunkt als die Entwicklung begann, war bereits Angular 2[22] veröffentlicht, aber Inspinia war noch nicht in Angular 2 verfügbar. Der Grund dafür war, dass es für zu viele jQuery-Plug-Ins noch keine geeigneten Angular-2-Komponenten gab. Zudem ist AngularJS einfacher zu erlernen und passt somit auch besser ins Konzept. Die Performance Vorteile von Angular 2 gegenüber AngularJS würden in diesem Backend kaum zu tragen kommen. In der von mir gewählten Version von Inspinia wird Grunt[23] als Build-System eingesetzt und als Paketmanager npm[24] und bower[25]. Inspinia gibt es mit AngularJS auch als Variante mit dem Task-Runner gulp.js[26]. Gulp.js lässt sich zwar flexibler konfigurieren, gibt aber demnach auch weniger einen einheitlichen Workflow vor als Grunt. Da dieses Projekt einen möglichst leichten Einstieg und ebenfalls einen möglichst einheitlichen Workflow vorgeben soll, habe ich mich für den Task-Runner Grunt entschieden.

[19] https://wrapbootstrap.com/theme/inspinia-responsive-admin-theme-WB0R5L90S (Stand 2.11.2017)
[20] https://angularjs.org/ (Stand 2.11.2017)
[21] https://jquery.com/ (Stand 2.11.2017)
[22] https://v2.angular.io/docs/ts/latest/ (Stand 2.11.2017)
[23] https://gruntjs.com/ (Stand 2.11.2017)
[24] https://www.npmjs.com/ (Stand 2.11.2017)
[25] https://bower.io/ (Stand 2.11.2017)
[26] https://gulpjs.com/ (Stand 2.11.2017)

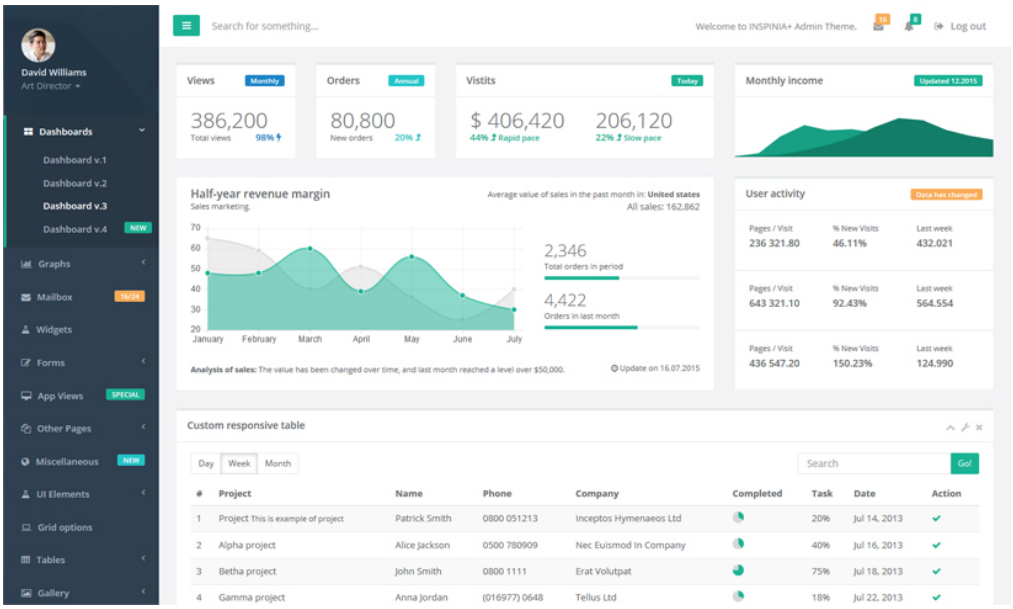

Abbildung 8: Inspinia[27]

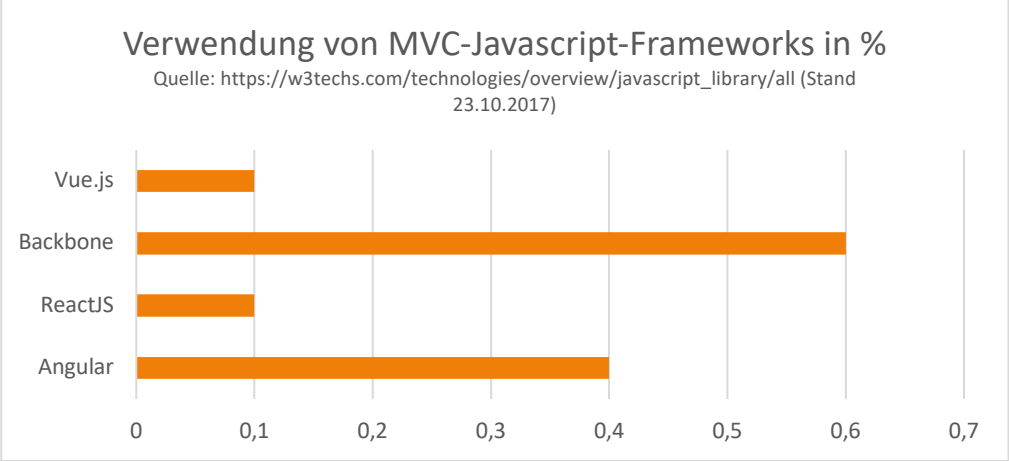

Abbildung 9: Google Trends - JavaScript Frameworks

6.4 Diskussion

Die Auswahl von Programmiersprachen, Frameworks etc. ist oft subjektiv. Es kommt viel darauf an, wie vertraut einem eine Programmiersprache ist oder wie oft man mit einem Framework schon zu tun hatte. In diesem konkreten Fall sind sicher viele andere Kombinationen von Sprachen und Frameworks denkbar, die ein ähnlich gutes Setup ergeben

[27] http://webapplayers.com/inspinia_admin-v2.6/img/dashbard4_1.jpg (Stand 3.10.2017)

würden. Eine Kombination aus Laravel und vue.js[28] hätte sicher ein vergleichbar hohes Potenzial ergeben.

[28] https://vuejs.org (Stand 24.9.2018)

7. TECHNISCHE UMSETZUNG

Im Zuge dieser Arbeit wird nicht das ganze Konzept umgesetzt, da es den Rahmen übersteigen würde. Außerdem werde ich mich bei der Umsetzung nur auf das Backend konzentrieren. Folgende Punkte werden Teil der Umsetzung sein:

- Einrichtung und Konfiguration von CakePHP 3
- Konfiguration von Inspinia
- Basismodule
- Datenbankdesign
- Verwaltung von Seiten
- Verwaltung von Beiträgen
- Verwaltung von Kategorien
- Medienverwaltung (Media Center)
- Zwei Sektionen

7.1 EINRICHTUNG UND KONFIGURATION VON CAKEPHP 3

CakePHP 3 basiert auf Ruby on Rails und sie teilen sich auch die zwei Grundätze „Don't repeat yourself" (Wikipedia - Don't repeat yourself, 2018) und „Konvention vor Konfiguration" (Wikipedia - Ruby on Rails, 2018). Mit „Dont't repeat yourself" ist gemeint, dass versucht wird, möglichst wenig redundanten Code zu produzieren. „Konvention vor Konfiguration" ist ein Softwaredesign-Paradigma, welches besagt, dass man sich viel Konfigurationsaufwand ersparen kann, wenn man sich an bestimmte Konventionen hält. Daher ist die Konfiguration von CakePHP 3 auch denkbar einfach.

7.1.1 Routing

```
47
48      Router::prefix( name: 'backend', function ($routes) {
49
50        $routes->setExtensions(['json', 'xml', 'pdf', 'html']);
51        $routes->connect('/', ['controller' => 'Main', 'action' => 'index']);
52        $routes->fallbacks(DashedRoute::class);
53
54        Router::prefix( name: 'section', function ($routes) {
55
56          $routes->setExtensions(['json', 'xml', 'pdf', 'html']);
57          $routes->fallbacks(DashedRoute::class);
58
59        });
60
61      });
62
```

Abbildung 10: CakePHP-Routing Konfiguration (/config/routing.php)

Der Aufruf der Domain soll natürlich zur eigentlichen Webseite führen. Um ins Backend zu gelangen, soll der Link zur Domain um „/backend“ erweitert werden. Um das zu erreichen, stellt CakePHP das sogenannte „Prefix Routing“ zur Verfügung (Foundation, Cake Software, 2018, S. 174).

Der Link, um ins Backend zu kommen, sieht dann beispielsweise so aus:

http//www.meine-domain.at/backend

Für den Aufruf einer Methode einer Sektion würde der Link wie folgt aussehen:

http://www.meine-domain.at/backend/section/name-der-sektion/name-der-methode

Inspinia ist als Single-Page-Applikation realisiert, daher wird die Kommunikation mit dem Server fast ausschließlich über AJAX[29] Aufrufen stattfinden, die JSON[30] als Antwortformat erwarten, deshalb wird die entsprechende Extension geladen. Nach der Konfiguration der Default-Route wird definiert, dass die Standardkonfiguration für das Routing der übrigen Aufrufe verwendet werden soll. Auch dieses Standardrouting folgt dem Grundsatz „Konvention vor Konfiguration“.

[29] https://de.wikipedia.org/wiki/Ajax_(Programmierung) (Stand 24.5.2018)
[30] https://de.wikipedia.org/wiki/JavaScript_Object_Notation (Stand 24.5.2018)

7.1.2 Mysql Konfiguration

Die Konfiguration der Datenbankverbindung findet bei CakePHP 3 in der Datei /config/app.php statt.

7.1.3 AppController

Im AppController fürs Backend wird die Componente „Auth“ (Foundation, Cake Software, 2018, S. 67) geladen. Für die Anmeldung wird die Standardauthentifizierung gewählt und als Formularfelder „email“ und „password“ definiert, so wie die entsprechenden Felder in der Datenbank heißen. Die zweite Änderung im AppController deaktiviert das Rendern der Seite in ein Layout (Foundation, Cake Software, 2018, S. 269), da es sich – wie oben erwähnt – fast ausschließlich um AJAX Aufrufe handelt.

Damit ist die Konfiguration für das Backend von CakePHP abgeschlossen. Für die weitere Umsetzung halte ich mich in diesem Projekt an den Workflow von CakePHP. Das betrifft sowohl die Strukturierung serverseitig als auch die Konventionen bei der Namensgebung der Datenbanktabellen und Datenbankfelder.

7.2 Abstract Section

Im Namespace „App\Controller\Backend\Section“ befinden sich alle Controller für die einzelnen Sektionen. Jede Sektion muss von der abstrakten Klasse „Section“ abgeleitet werden. Die Klasse Section selbst ist wiederrum vom AppController abgeleitet. Sie beinhaltet folgende Methoden, die überschrieben werden können:

- „edit“ – Wird aufgerufen, um das Template der Administration für die Sektion zu laden.
- „info“ – Wird aufgerufen, um das Template für die Informationen über die Sektion zu laden.
- „remove“ – Wird aufgerufen, wenn eine Sektion von einer Seite entfernt wird.
- „add“ – Wird aufgerufen, wenn eine Sektion einer Seite hinzugefügt wird.

7.3 DATENBANK

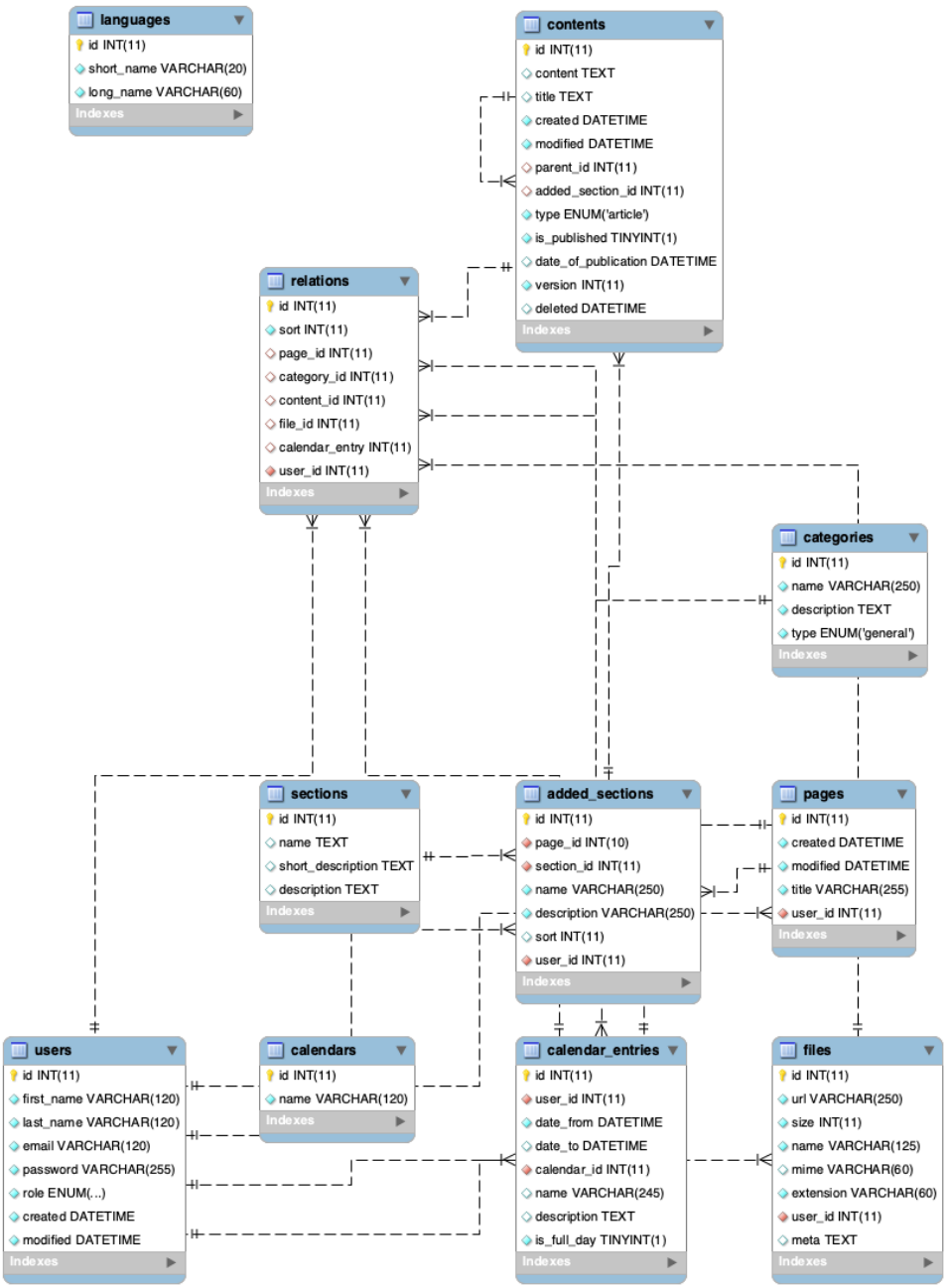

Abbildung 11: Datenbankdesign

Das Datenbankdesign soll möglichst einfach gehalten sein. Ich möchte hier nur auf Besonderheiten eingehen, da es meines Erachtens ohnehin zum Großteil selbsterklärend ist.

7.3.1 Relations

Die Tabelle „relations“ speichert möglichst alle Verbindungen zwischen einzelnen Tabellen. Dafür gibt es ein Feld für den Primary Key aller notwendigen Tabellen. Somit kann zum Beispiel die Tabelle „contents“ mit Dateien, Kategorien etc. verknüpft werden.

7.3.2 Users

In „users“ befinden sich alle Benutzer des Backends. Wenn möglich wird auch zu jedem Datensatz die ID des Users gespeichert, der gerade angemeldet ist, um nachvollziehen zu können, wer was gemacht hat. Außerdem könnte damit gewährleistet werden, dass zum Beispiel nur der Autor eines Beitrages diesen auch wieder verändern kann.

7.3.3 Languages

Die Tabelle „languages“ speichert die unterstützen Sprachen. Felder wie zum Beispiel das Feld „title“ der Tabelle „contents“ unterstützt mehrere Sprachen. Somit ist das Feld „title“ in CakePHP über das Model (Foundation, Cake Software, 2018, S. 378) als „json“ Feld deklariert (Foundation, Cake Software, 2018, S. 383). Dadurch speichert CakePHP ein Array automatisch als JSON-String und konvertiert den JSON-String beim Abrufen der Daten wieder in ein Array. Wenn jetzt ein neuer Titel für einen Beitrag auf Deutsch gespeichert wird, dann wird ein Objekt in JavaScript erstellt mit der Eigenschaft „DE“, die den Titel des Beitrages hält. Gespeichert wird diese Information dann als JSON-String. Damit ist es auch beim Auswählen einer anderen Sprache möglich, den Inhalt sofort übersetzt anzuzeigen.

7.3.4 Contents

Die Tabelle „contents“ ist dafür gedacht, alle möglichen Inhalte zu speichern, sofern die Tabelle für den Inhaltstypen passend ist. Das Feld „parent_id“ verlinkt auf dieselbe Tabelle. Damit werden die verschiedenen Versionen zum Beispiel eines Beitrages gespeichert.

7.4 KONFIGURATION VON INSPINIA

Inspinia bietet eine Reihe an Konfigurationen für verschiedene Frameworks und Task-Runner an. In diesem Fall habe ich mich für das Framework AngularJS und den Task-Runner Grunt entschieden (siehe Kapitel 6.3.2 Backend). Sowohl Grunt als auch die AngularJS-Struktur wurden schon von den Entwicklern von Inspinia vorkonfiguriert. Außerdem wurden Boostrap 3, UI Boostrap und viele andere nötige Plug-Ins und Bibliotheken vorkonfiguriert und dem Projekt hinzugefügt. Einige AngularJS-Direktiven werden von den Entwicklern selbst geschrieben, um diverse jQuery-Plug-Ins nahtlos dem Projekt hinzufügen zu können.

7.4.1 Struktur von Inspinia

- app
    - less
    - scripts
    - bower_components
    - styles
    - views
    - index.html

7.4.2 Design

Im Ordner „less“ befinden sich alle Less-Dateien für das Stylesheet des Projekts. Die Datei „customer.less“ ist dafür gedacht, das Projekt um eigene Designelemente zu ergänzen. Die anderen Less-Dateien erlauben es, das bestehende Design zu modifizieren. Die Less-Dateien werden von Grunt in einer CSS-Datei zusammengefasst und dabei minimiert[31] und unter styles/style.css gespeichert.

7.4.3 Javascript

Im Ordner „scripts“ befinden sich die JavaScript -Dateien für dieses Projekt. In den Dateien „app.js“, „controller.js“, „directives.js“ und „config.js“ befindet sich entsprechend aufgeteilt

[31] https://en.wikipedia.org/wiki/Minification_(programming) (Stand 24.5.2018)

das Hauptmodul. Alle weiteren Module werden dem Hauptmodul hinzugefügt, um dem ganzen Projekt zur Verfügung zu stehen. Das Routing der Single-Page-Applikation befindet sich in der „config.js“. Im Ordner „bower_components“ befinden sich alle von dem Paketmanager bower geladenen JavaScript-Bibliotheken.

7.4.4 Templates

Im Ordner „views“ befinden sich diverse HTML-Templates wie zum Beispiel für die Sidebar und die Hauptnavigation. In dieses Projekt werden manche Templates, wie zum Beispiel die Hauptnavigation, in ein CakePHP-Template umgewandelt, um es dynamisch rendern zu können.

7.4.5 Konfigurationsdateien

Auf gleicher Ebene wie der Ordner „app“ befinden sich die Konfigurationsdateien für Grunt, npm und bower.

7.4.6 Bereitstellung des Projekts

Mit dem Konsolenbefehl „grunt build“ lässt sich die Applikation für den Echtbetrieb bereitstellen. Dabei werden alle Javascript und LESS-Dateien in jeweils zwei entsprechenden Dateien zusammengefügt und minimiert. Die „index.html“ wird ebenfalls minimiert und die neuen Dateien eingebunden.

7.4.7 Eigene Module

Grundsätzlich halte ich mich an die Struktur und den Workflow von Inspinia. Für jedes weitere Angular-Modul[32] wird ein eigener Ordner im Verzeichnis „scripts“ erstellt, der den Namen des Moduls trägt. Das Modul selbst wird wie folgt benannt: „name-des-modules.module“. In jedem Modul-Ordner befinden sich die Dateien „controllers.js“, „directives.js“, „services.js“ und „module.js“. In der Datei „controller.js“ befinden sich alle Angular-

[32] https://docs.angularjs.org/guide/module (Stand 24.9.2018)

Controller[33] dieses Moduls, in der Datei „directives.js“ alle Angular-Direktiven[34], in der Datei „services.js“ alle Angular-Services[35] und Angular-Provider[36] und in der Datei „module.js“ die Definition des Modules und alle Angular-Filter[37]. Die einzelnen Dateien werden dann in der „index.html“ eingebunden und das Modul dem Hauptmodul hinzugefügt.

7.4.8 Clientseitige Entwicklung

Inspinia wird vom dem vorkonfigurierten Task-Runner Grunt geliefert. Mit dem Konsolenbefehl „grunt live“ lässt sich ein eigner lokaler Server starten, der das Theme anzeigt und den Browser bei Änderungen von relevanten Dateien aktualisiert und die Less-Dateien kompiliert. Grundsätzlich lässt sich damit gut arbeiten, nur die Kommunikation mit dem Server ist in der Entwicklungsphase problematisch. CakePHP müsste in diesem Fall auf einem eigenen lokalen Server laufen. Wenn der Klient jetzt eine Anfrage an den Server stellt, würde diese Anfrage aufgrund der SOP (Same-Origin-Policy)[38] untersagt. Dieses Problem würde sich zwar mit CORS (Cross-Origin Resource Sharing)[39] lösen lassen, aber folgende Gründe sprechen dagegen:

- CORS ist für den Echtbetrieb weder erwünscht noch notwendig und würde somit nur für die Entwicklung konfiguriert werden müssen. Für den Echtbetriebt müsste die Konfiguration wieder deaktiviert oder entfernt werden.

- Um Cookies verwenden zu können, notwendig unter anderem für die Anmeldung zum Backend, müsste AngularJS dementsprechend konfiguriert werden. Auch hier stellt sich die Frage, ob eine solche Konfiguration vorgenommen werden soll, wenn sie für den Echtbetrieb nicht erwünscht ist.

[33] https://docs.angularjs.org/guide/controller (Stand 24.9.2018)
[34] https://docs.angularjs.org/guide/directive (Stand 24.9.2018)
[35] https://docs.angularjs.org/guide/services (Stand 24.9.2018)
[36] https://docs.angularjs.org/guide/providers (Stand 24.9.2018)
[37] https://docs.angularjs.org/guide/filter (Stand 24.9.2018)
[38] https://de.wikipedia.org/wiki/Same-Origin-Policy (Stand 24.9.2018)
[39] https://de.wikipedia.org/wiki/Cross-Origin_Resource_Sharing (Stand 24.9.2018)

Um diesem Problem zu entgehen, wird bei der Entwicklung diesbezüglich auf Grunt verzichtet und die „index.html“ direkt im Browser aufgerufen. Dadurch muss die Applikation im Browser zwar manuell vollständig aktualisiert werden, aber sie verhält sich auch weitgehend gleich wie im Echtbetrieb.

7.5 BASISMODULE

7.5.1 Das Core-Modul

Das Core-Modul beinhaltet die Basisfunktionalität des Projects. Es soll nicht projektspezifische Funktionalität enthalten, sondern vielmehr allgemein gehaltene Funktionalität beziehungsweise Funktionalität, die die Kommunikation mit CakePHP erleichtert. Der wichtigste Angular-Service ist der Http-Service.

7.5.1.1 Der Http-Service

Der Http-Service ist ein Wrapper über dem von AngularJS mitgelieferten $http-Provider[40]. Er kümmert sich um die einfache Generierung von Anfragen, das Error-Handling und die Umwandlung von relativen Links zu absoluten. Die Anfragengenerierung ist dabei auf CakePHP zugeschnitten, aber es lassen sich auch Anfragen auf herkömmlichem Weg stellen.

Eine Anfrage mit dem AngularJS eigenen Provider sieht so aus:

```
78
79    var Http = {
80      method : 'post',
81      url : 'http://localhost/Lt/my-controller/my-action.json',
82      data : { id : 5 }
83    };
84
85    $http(Http).then(function(Resonse) {
86
87      console.log(Response.data);
88
89    });
90
```

Abbildung 12: POST Anfrage mit $http

[40] https://docs.angularjs.org/api/ng/service/$http (Stand 24.9.2018)

Mit dem Http-Provider des Core-Moduls so:

```
84
85  $http(Http).then(function(Resonse) {
86
87    console.log(Response.data);
88
89  });
90
91  Http.post({ controller : 'my-controller', action : 'my-action.json' }, { id : 5 }, function(Response) {
92
93    console.log(Response.data);
94
95  });
96
```

Abbildung 13: POST Anfrage mit dem Http-Service

7.5.1.2 Das Main-Modul

Das Main-Modul stellt allgemeine projektspezifische Funktionalität zur Verfügung wie zum Beispiel den Languages-Service, der die möglichen Sprachen verwaltet, in der die Webseite angezeigt werden kann. Außerdem befindet sich dort die notwendige Funktionalität für den Footer, den Header und die Navigation des Backends.

7.5.1.3 Das User-Modul

Das User-Modul stellt die Funktionalität zur Verfügung, die notwendig ist, um sich im Backend anmelden und abmelden zu können und die userbezogenen Daten zu halten. Dabei wird der Cookie-Provider[41] von AngularJS verwendet, um die userbezogenen Daten auch nach dem Neuladen der Anwendung im Browser zu halten.

[41] https://docs.angularjs.org/api/ngCookies/service/$cookies (Stand 24.9.2018)

7.6 Übersicht aller AngularJS-Services

Abbildung 14: Übersicht aller AngularJS-Services

7.7 Verwaltung von Seiten

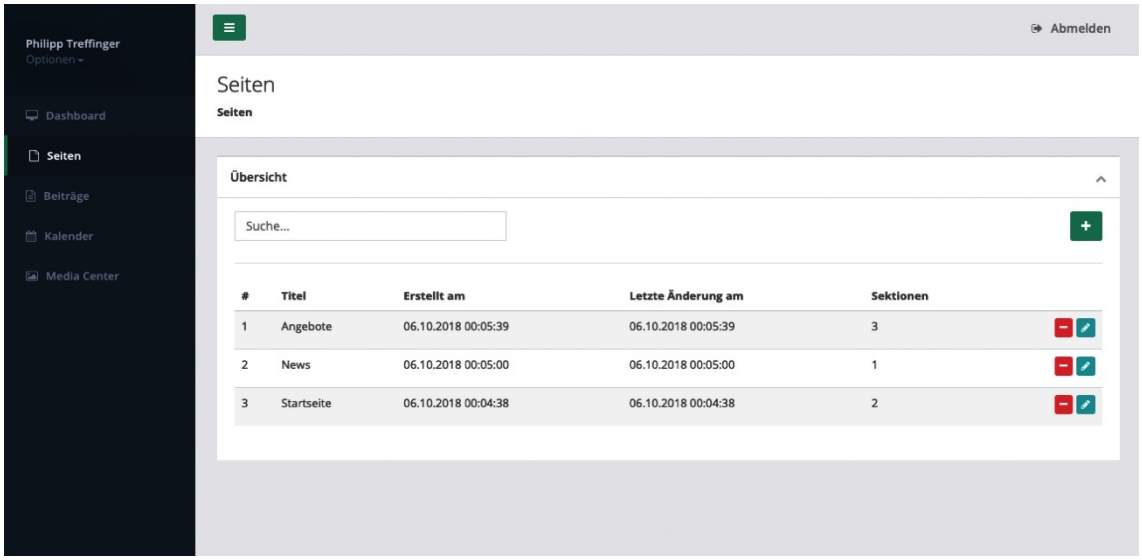

Abbildung 15: Verwaltung von Seiten

Unter den Menüpunkt „Seiten“ können die einzelnen Seiten der Webseite verwaltet werden. Der Funktionsumfang begrenzt sich in dieser Version auf die Erstellung, das Löschen und die Anzeigen von Seiten. Das Pages-Modul stellt dabei den Service „Pages“ dem Projekt zur Verfügung, der alle Seiten der Webseite hält. Die Daten bezieht er über den PagesController vom Server.

7.8 Bearbeiten einer Seite

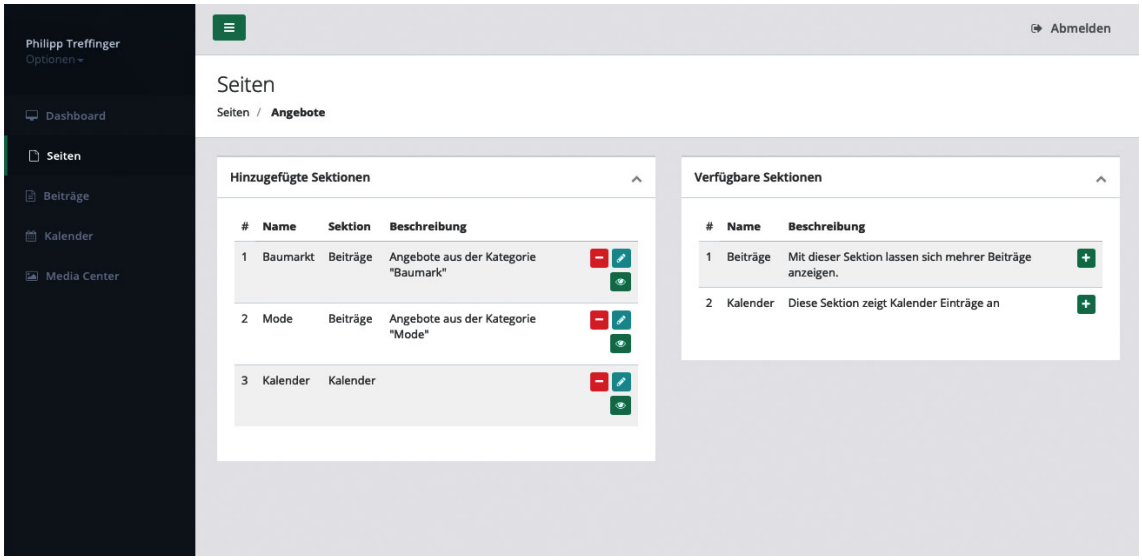

Abbildung 16: Bearbeiten einer Seite

Das Bearbeiten einer Seite gliedert sich in zwei Container. Im linken Container befinden sich die der Seite bereits hinzugefügten Sektionen, im rechten die zur Verfügung stehenden Sektionen. Die Programmierung dafür befindet sich im Page-Modul, das selbst keine Services bereitstellt. Im Backend nimmt der PageController die entsprechenden Anfragen entgegen.

7.9 Administrierung einer Sektion

Abbildung 17: Konfigurieren einer Sektion

In dieser Version steht nur eine Standardadministration für die Sektion dem Entwickler zur Auswahl. In dieser lassen sich eine Kategorie und ein Inhaltstyp auswählen. Beim Klicken auf den Button, der zur Administrierung der Sektion führt, wird die Methode „edit“ des entsprechenden Controllers aufgerufen, um das Template für die Administrierung zu laden.

7.10 Inhaltstypen

Die Version für diese Arbeit beinhaltet zwei Inhaltstypen. Auch für Inhaltstypen gibt es Elemente, die allen Inhaltstypen zu Verfügung stehen und die der Entwickler einbinden kann. In dieser Version steht ein Panel zum Anzeigen und zum Verknüpfen von Kategorien und die Media Center zum Einbinden von Bildern, Dokumenten etc. zur Verfügung.

7.10.1 Beiträge

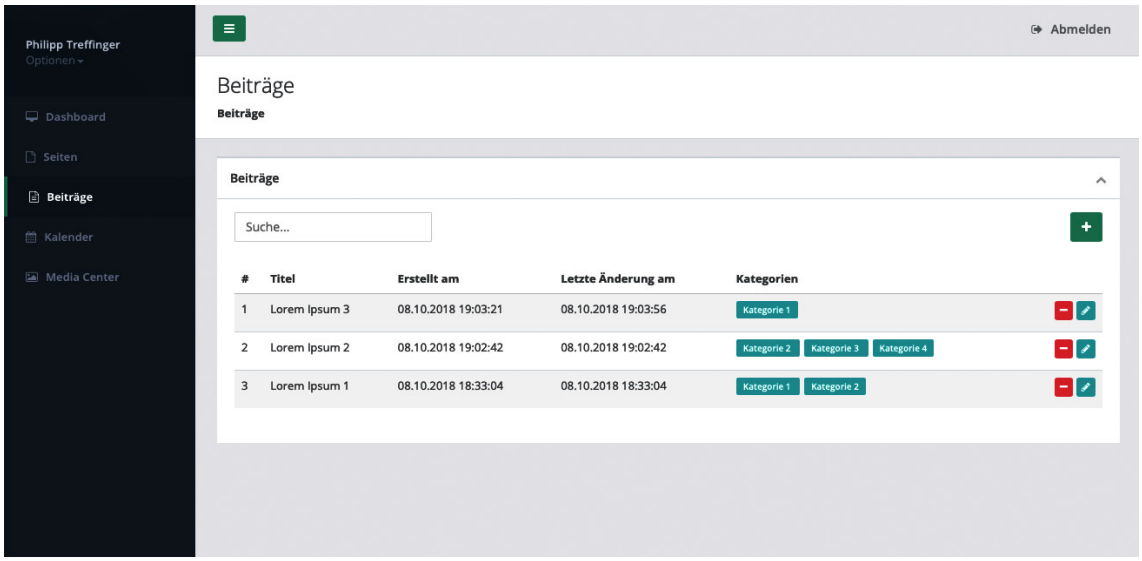

Abbildung 18: Inhaltstyp „Beitrag“

Unter „Beiträge“ in der Hauptnavigation können die bestehenden Beiträge verwaltet, neue hinzugefügt und gelöscht werden. Im Articles-Modul ist der „Articles“-Service implementiert, der alle Beiträge hält.

7.10.1.1 Bearbeiten eines Beitrages

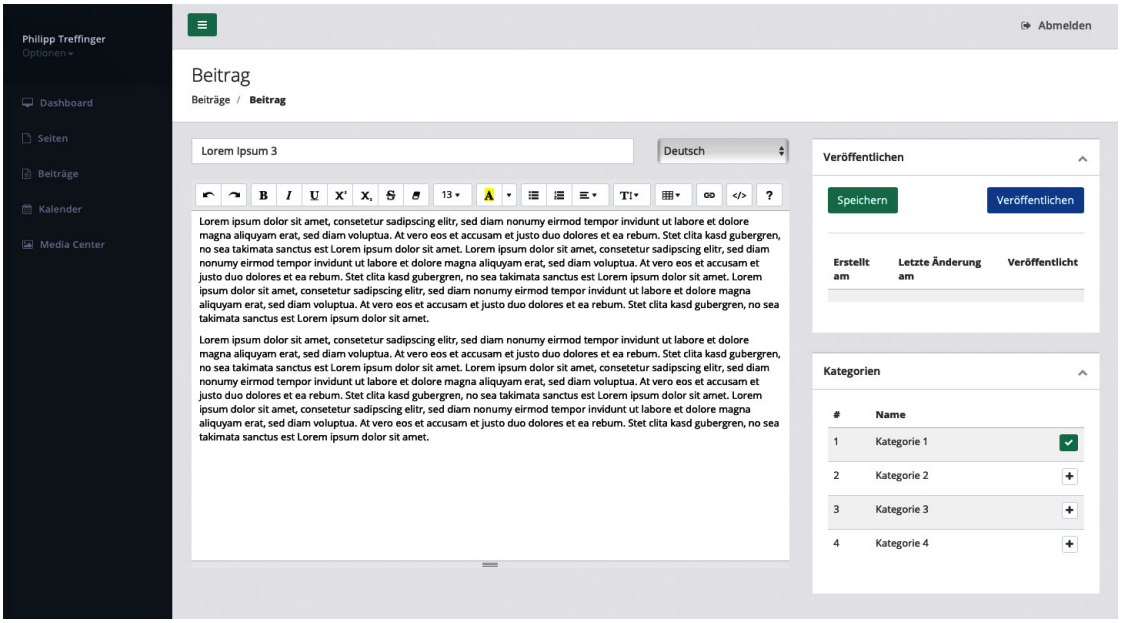

Abbildung 19: Beitrag bearbeiten

Ein Betrag besteht aus einem Titel und einem Text, der mit Hilfe eines WYSIWYG[42]-Editors erstellt wird. Da ich einen schlanken, einfach zu verwendenden und mit Boostrap kompatiblen Editor verwenden wollte, habe ich mich für den summernote[43]-Editor entschieden. Im Panel „Veröffentlichen“ lässt sich der Beitrag veröffentlichen oder nur abspeichern. Darunter befinden sich die verschiedenen Versionen des Beitrages, auf die auch zurückgestellt werden kann. Die Beiträge werden in der Tabelle „contents“ gespeichert. Bei jedem Speichern wird die alte Version in einen neuen Datensatz abgelegt und eine Referenz auf den aktualisierten Beitrag erstellt. So ändert sich die ID des aktuellen Beitrages nicht, was wiederum eine etwaige Referenz auf den Beitrag nicht beeinflusst.

7.10.2 Kalender

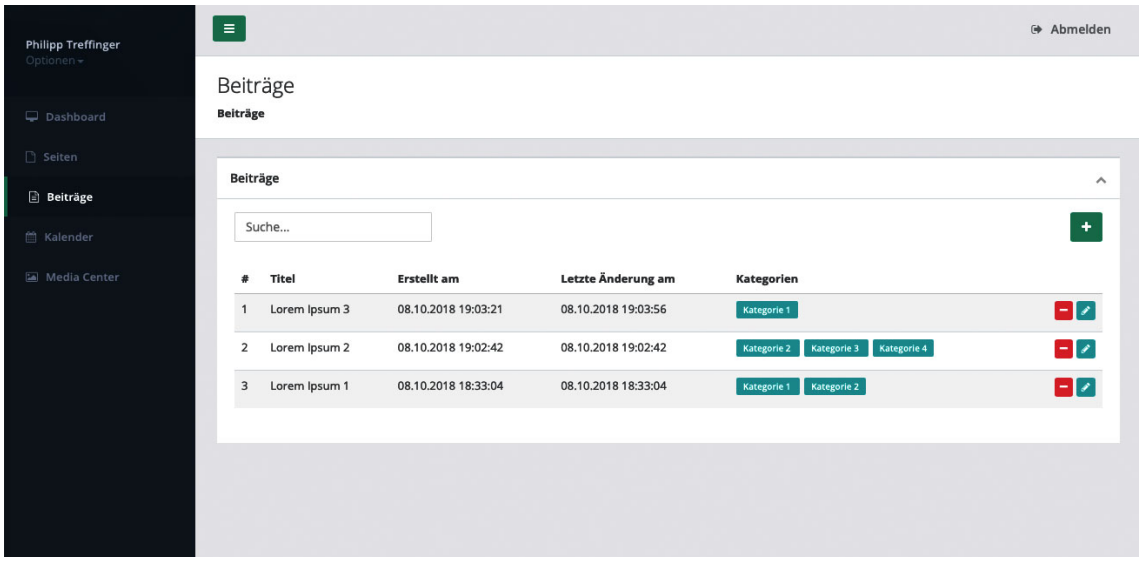

Abbildung 20: Inhaltstyp „Kalender“

Ein Kalendereintrag besteht aus einem Datum mit Uhrzeit beziehungsweise einer Zeitspanne, dabei kann wie üblich gewählt werden, ob es sich um ganze Tage handelt oder die Uhrzeit relevant ist. Außerdem lassen sich ein Titel und ein Text definieren und der Kalendereintrag

[42] https://de.wikipedia.org/wiki/WYSIWYG (Stand 24.9.2018)
[43] https://summernote.org/ (Stand 24.9.2018)

lässt sich in Kategorien einteilen. Zusätzlich lassen sich noch Dateien mit einem Kalendereintrag verknüpfen.

7.10.3 Media Center

In dieser Version der Media Center reduziere ich mich auf die notwendigsten Funktionen. Dazu gehören:

- Hochladen von Dokumenten, Bildern und sonstigen Dateien per Klick oder per Drag-and-Drop[44]
- Skalierung der hochgeladenen Bilder am Server
- Speichern notwendiger Metadaten
- Anzeigen der hochgeladenen Dateien mit Filtermöglichkeiten
- Dateien einem Inhaltstypen zuweisen

Zum Hochladen der Datei entschied ich mich für das Plug-In „angular-file-upload“[45], da es einfach zu integrieren und gut dokumentiert ist. Das Plug-In unterstützt sowohl Multiple-Upload als auch Drag-and-Drop. Zum Skalieren der Bilder am Server verwende ich das Plug-In „gumlet/php-image-resize“[46], damit lassen sich Bilder mit wenigen Zeilen Code in verschiedenen Größen und Qualitätsstufen skalieren. Beim Hochladen der Datei wir diese an die Methode „uploadFile“ des MediaCenter-Controllers geschickt. Der MediaCenter-Controller nützt die FileHandler-Componente, die sich um das Abspeichern diverser Dateien kümmert. In dieser Componente werden die notwendigen Meta-Daten aus der Datei gelesen, sofern diese vorhanden sind (Dateiformat, Koordinaten, Autor etc.). Bilder werden in 5 verschiedenen Größen gespeichert, um Ladezeiten verkürzen zu können. Skaliert wird immer auf die längste Seite:

- Originalgröße
- 1920x1920 Pixel
- 1024x1024 Pixel

[44] https://de.wikipedia.org/wiki/Drag_and_Drop (Stand 26.9.2018)
[45] https://github.com/nervgh/angular-file-upload/ (Stand 26.9.2018)
[46] https://github.com/gumlet/php-image-resize (Stand 26.9.2018)

- 800x800 Pixel
- 300x300 Pixel

Alle Dateien werden im Ordner „uploads/files“ gespeichert, und es wird je nach Hochladedatum ein Unterordner angelegt (Jahr/Monat), damit nicht zu viele Dateien in einem Ordner liegen. Der Dateinamen wird analysiert und wenn nötig unbenannt. Nach dem Abspeichern wird ein Datensatz in der Tabelle „files“ erstellt, der die nötigen Daten enthält und auch die Referenzen auf die Datei/Dateien. Danach wird der Datensatz retourniert, um die Datei im Browser anzeigen zu können. Im Browser werden die Datensätze vom Service „Files“ gehalten, der das Modul „mediaCenter“ bereitstellt.

Im Frontend kennt die Media Center zwei Modi. Zum ersten gelangt man über die Hauptnavigation. Hier können die hochgeladenen Dateien verwaltet und die Media Center erweitert werden. Der zweite Modus kommt dann zum Tragen, wenn Dateien zum Beispiel einem Inhaltstypen oder einer Sektion zugeordnet werden sollen. In diesem Modus ist die Hauptnavigation ausgeblendet und es besteht eben die Möglichkeit, per Mouseklick eine Datei zu verknüpfen oder eine Verknüpfung zu lösen. Mit dem Button „Schließen“, der sich rechts oben befindet, beendet man diesen Modus.

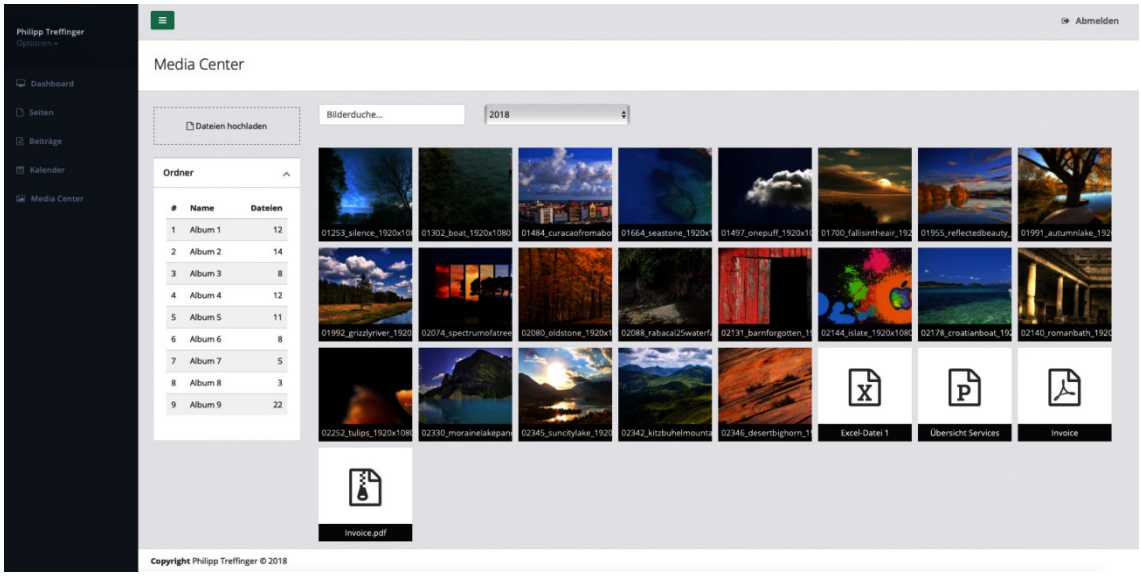

Abbildung 21: Media Center im reinen Verwaltungsmodus

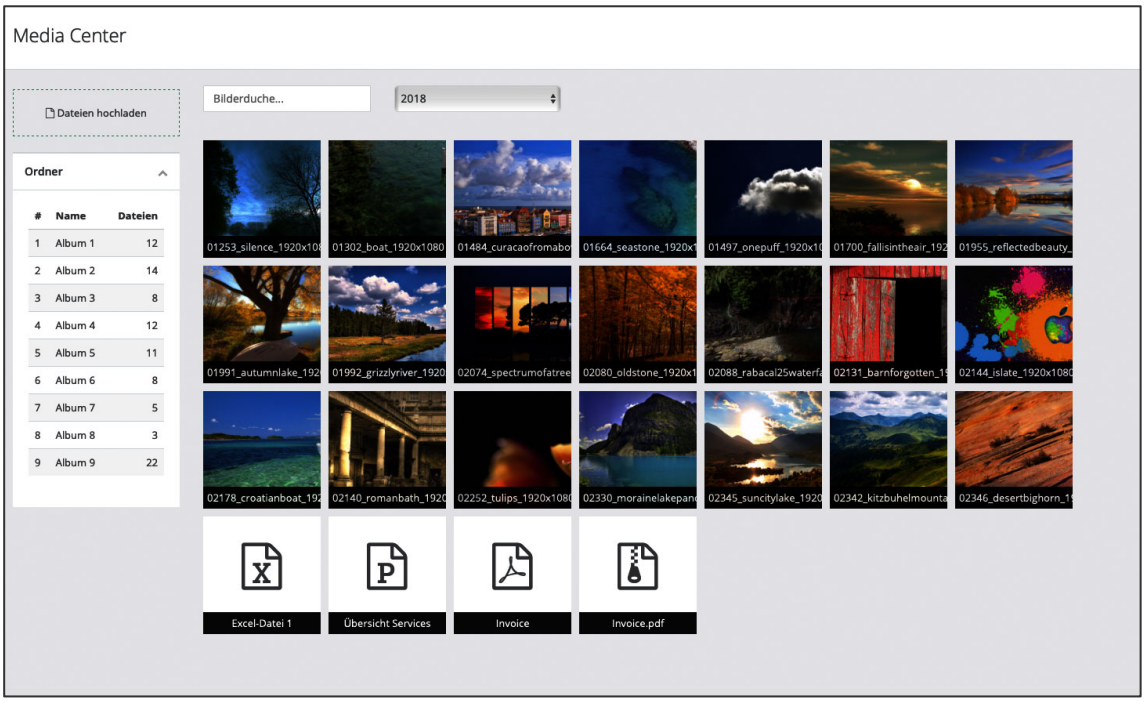

Abbildung 22: Media Center im Zuordnungsmodus

7.10.3.1 Verknüpfen von Dateien

Wenn man Dateien beispielsweise mit dem Inhaltstypen „Beiträge“ verknüpfen will, dann fügt man mit Hilfe der mediaCenterPanel-Direktive das entsprechende Panel der Administration hinzu. Das Panel zeigt alle schon verknüpften Dateien an und erstellt einen Button, der zur Media Center führt. Die Direktive wird mit folgenden Parametern initialisiert:

- Id des Beitrages
- Name des id-Feldes der Tabelle „relations“ (in diesem Beispiel „content_id“)
- Die Rücksprungsroute (in diesem Beispiel die Route zurück zum Beitrag)
- Ein Konfigurationsarray (optional)

Nachdem in der Media Center Dateien mit dem Beitrag verknüpft oder Verknüpfungen aufgelöst wurden, löst der MediaCenter-Controller das Event „closeMediaCenter“ aus und

übergibt dabei den aktuellen Stand der Verknüpfungen. Die Direktive der Beitragsadministration hört auf dieses Event und aktualisiert die Anzeige.

7.11 SEKTIONEN

7.11.1 Beiträge

Mit der Sektion „News“ können Beiträge auf der Seite angezeigt werden. Dabei kann definiert werden, ob alle Beiträge oder nur Beiträge einer Kategorie angezeigt werden sollen.

7.11.2 Kalender

Mit der Sektion „Kalender“ können die Kalendereinträge angezeigt werden. Dabei kann wieder definiert werden, ob alle angezeigt werden oder nur Einträge einer Kategorie.

8. AUSBLICK

Wie schon oben erwähnt, ist die für diese Arbeit entwickelte Version nur der Beginn. Die bereits beschriebenen Features „5.3.4 Widgets“ und „5.3.2 Container“ fehlen in dieser Version noch vollständig. Aber es wären auch noch einige andere Schritte für einen ersten Release notwendig. Einige davon möchte ich noch erwähnen.

8.1 BENUTZERVERWALTUNG

Eine Homepage wird natürlich oft von mehreren Personen betreut. Somit müssen auch mehrere Benutzer angelegt und verwaltet werden können. Außerdem hat nicht jeder Benutzer dieselben Rechte, daher bedarf es auch einer ausreichenden Rechteverwaltung. Die sollte aber nicht zu komplex ausfallen, wie das zum Beispiel bei Typo3 (siehe Kapitel 4.4 Typo3) der Fall ist. Das Ziel wäre eine Rechteverwaltung mit drei Rechteebenen, die global gelten, aber auch für jede Sektion individuell angepasst werden können.

8.2 DASHBOARD

Beim Anmelden am Backend sollte der User zu einem Dashboard gelangen, dass ihm diverse Informationen anzeigt und schnelle Aktivitäten ausführen lässt. Das Erstellen eines schnellen Beitrags oder eines Kalendereintrages wäre denkbar, ähnlich wie man es bei WordPress vorfindet.

8.3 FRONTEND

Die Umsetzung in dieser Arbeit schließt das Frontend vollständig aus. Für einen Release müsste eine entsprechende Funktionalität noch vollständig implementiert werden. Um ins Backend zu gelangen ist ein Prefix-Routing (siehe Kapitel 7.1.1 Routing) konfiguriert, die Controller und Templates des Frontends sollten sich im Hauptverzeichnis befinden – also ohne ein Prefix-Routing. Auch aus Sicherheitsgründen sollte das gesamte Frontend aus eigenen Controllern bestehen, damit es keine direkte Verbindung zum Backend gibt. Andernfalls müssten

Ausnahmen für die Auth-Komponente konfiguriert werden, was wiederum eine Fehlerquelle ist und dadurch ein Sicherheitsrisiko sein kann (Foundation, Cake Software, 2018, S. 236).

8.4 Media Center

In der Media Center fehlen noch diverse Funktionen. Einige möchte ich aufzählen:

- Bilder sollten geschnitten werden können, außerdem wären rudimentäre Bearbeitungsmöglichkeiten denkbar.
- Es sollte Ordner geben, in die die Dateien verschoben werden können.
- Beim Zuordnen von Dateien sollte die Reihenfolge definiert werden können.
- Wenn die Media Center zum Zuordnen von Dateien geöffnet wird, dann sollte über das Konfigurationsarray folgendes eingestellt werden können:
    - Wie viele Dateien sollen maximal zugeteilt werden können
    - Welche Dateitypen sind erlaubt
    - Erlaubte Dimension der Bilder
    - Sind ganze Ordner erlaubt oder nicht

8.5 Features & Details

Die Version dieser Arbeit bietet bei allen vorgestellten Features nur rudimentäre Funktionalität. Um das Backend aber angemessen bedienen zu können, müsste es wesentlich mehr bieten. Dazu gehören beispielsweise diverse Filter-, Such- und Sortierungsfunktionen, aber es fehlt auch an Funktionalität in diversen Features.

8.6 Menü

Es fehlt die Möglichkeit eine Navigation zu erstellen. Gedacht wäre ein eigener Inhaltstyp „Navigation“. Dieser könnte dann einer entsprechenden Sektion zugeteilt werden.

8.7 SEKTIONEN

Für einen ersten Release fehlen noch diverse Sektionen. Wichtige Sektionen für einen möglichen Release:

- Eine Sektion für Navigationen, die den noch nicht vorhanden Inhaltstyp „Navigation“ unterstützt
- Eine Sektion für einen Footer
- Verschiedene Sektionen für die verfügbaren Inhaltstypen.

8.8 SPRACHEN

Das Backend sollte mehrere Sprachen für die Benutzeroberfläche möglich machen.

9. VERGLEICH MIT BESTEHENDEN WCMS

In diesem Kapitel möchte ich Ähnlichkeiten und anhand dieser Unterschiede im Detail zu bestehenden WCMS erörtern.

9.1 WORDPRESS

Das Erstellen von Beiträgen und die Kategorien erinnern natürlich stark an WordPress. Wobei es bei WordPress von Haus aus eben nur diese Beiträge als „Inhaltstypen“ gibt, die auf der einen Seite zwar für viele verschiedene Zwecke verwendet werden, auf der anderen Seite müssen Plug-Ins aber oft neue Möglichkeiten zur Verfügung stellen um Inhalte zu generieren. Diese werden dann allerdings in den meisten Fällen nur von diesem Plug-In verwendet. Wenn jetzt viele verschiedene Plug-Ins installiert werden, die eigene „Inhaltstypen“ anbieten, wird die Administration sehr schnell sehr unübersichtlich. Deshalb soll hier zwischen einem Inhaltstyp und einer Sektion klar unterschieden werden. Dadurch soll vermieden werden, dass viele verschiedene, oft ähnliche, Inhaltstypen parallel zur Verfügung stehen. Die Media Center ist ebenfalls angelehnt an die von WordPress. Nur werden Bilder nicht direkt in den Texteditor übernommen, sondern beispielweise dem Beitrag hinzugefügt. Dadurch verhindert man negative Auswirkungen auf das Design und gibt dem Designer der Webseite die genaue Kontrolle wie und wo Multimediainhalte angezeigt werden. Durch die Möglichkeit, zu definieren, welche und wie viele Dateien hinzugefügt werden können, können ebenfalls negative Effekte auf das Design verhindert werden. Die Kategorien für Beiträge, Kalendereinträge etc. sind mit denen von WordPress direkt vergleichbar.

9.2 JOOMLA

Auch in diesem Fall ist das Kategoriensystem direkt mit dem von Joomla zu vergleichen. Seiten setzen sich bei Joomla ebenfalls aus Beiträgen zusammen. Auch hier fehlt mir die klare Unterscheidung zwischen verschiedenen Inhaltstypen. So besteht die Gefahr erhöhter Komplexität durch eine eventuell überstrapazierende Anwendung des Inhaltstypen „Beiträge“.

9.3 DRUPAL

Die Inhaltstypen dieses Systems sind auch mit denen von Drupal zu vergleichen. Der Unterschied besteht im Aufwand, einen Inhaltstypen zu erstellen. In Drupal lässt sich im Backend schnell ein solcher Inhaltstyp erstellen. In diesem System muss der Entwickler die Benutzeroberfläche, die serverseitige Programmierung und auch die Datenbank – wenn nötig – selbst entwickeln. Dieser Mehraufwand für den Entwickler soll durch eine saubere Benutzeroberfläche, eine Programmierung und eine Reduzierung des gesamten Backends gerechtfertigt werden. Außerdem ist der Entwickler flexibler bei der Gestaltung und der Funktionsweise eines Inhaltstypen.

10. DISKUSSION

Wie man anhand bestehender WCMS sehen kann, steigt die Komplexität für den Enduser mit der Flexibilität und Funktionsvielfalt. Alle hier analysierten WCMS haben ein Backend, welches vom Ersteller der Homepage, dem Administrator der Inhalte und auch vom Entwickler der Erweiterungen verwendet wird. Diese Multifunktionalität hat Auswirkungen auf die Benutzerfreundlichkeit. In diesem System muss der Entwickler neuer Sektionen, Inhaltstypen etc. diese auch selbst entwickeln – mit einem erhöhten Aufwand. Dadurch ist die Benutzeroberfläche des Backends nur für den Ersteller der Webseite und dem Administrator der Inhalte optimiert. Der Entwickler hat im Gegenzug viel Freiheit, Flexibilität und durch die Anwendung verbreiteter und erprobter Technologien eine möglichst geringe Einarbeitungszeit. Außerdem ist die Zeit, die die Anwender des Backend damit verbringen, wesentlich höher als die der Entwickler, die Erweiterungen für mehrere User erstellen. Eine weitere Vereinfachung wird durch die Einschränkung erreicht, dass die Seite ausschließlich vertikal, anhand von Sektionen, aufgebaut wird. Dadurch können viele Designkonzepte abgebildet werden, aber natürlich auch nicht alle. Diese Vorgehensweise schafft aber die Möglichkeit, die Administration einfacher zu halten. So besteht sie nur aus Inhaltstypen, Sektionen und Widgets. Elemente wie die Kategorien und die Media Center können direkt mit denen von anderen WCMS verglichen werden.

11. ZUSAMMENFASSUNG

Das Ziel dieser Arbeit ist ein Konzept und die teilweise Umsetzung eines Web-Content-Management-Systems (WCMS). Dabei stehen Benutzerfreundlichkeit, Funktionalität und Flexibilität im Vordergrund.

Die Generierung von Content kann auf Grund der Subjektivität nur eingeschränkt von Computersystemen übernommen werden. Das System soll ermöglichen, Content-bezogene Entscheidungen dem Autor zu überlassen und den Content darauf basierend zu verarbeiten.

Benutzerfreundlichkeit kann durch die Analyse des Aufwands und der Belastung, die ein User braucht, um eine gewünschte Aufgabe zu erfüllen, erreicht werden. Je mehr Flexibilität und Funktionalität ein System hat, desto komplexer wird es. Daher muss abgewogen werden, wie viele Einschränkungen man in Kauf nimmt und wie spezialisiert das System werden soll.

Das Konzept dieses WCMS sieht einen vertikalen Aufbau vor, der aus einzelnen Sektionen besteht. Die Sektionen beziehen je nach Bedarf ihren Inhalt aus den sogenannten Inhaltstypen. Inhaltstypen werden global organisiert und sind in Kategorien eingeteilt. Sollten die Sektionen zu keiner dieser Inhaltstypen passen, dann wird eine der Seite zugeordnete Sektion über eine eigene Administration verwaltet. Für die Organisation von Bildern, Dokumenten und sonstigen Dateien gibt es eine globale Media Center.

Für die technische Umsetzung kommen verbreitete Frameworks – sowohl im Frontend als auch im Backend – zum Einsatz. Dadurch wird der Einstieg in die Entwicklung vereinfacht und man kann auf viele hilfreiche Funktionen der Frameworks zurückgreifen. Ausschlagegebend für die Wahl der Frameworks war auf der einen Seite die Verbreitung und auf der anderen Seite die Unterstützung eines klaren Workflows. Die Wahl fiel auf CakePHP 3 als serverseitiges Framework, MySQL als Datenbank, AngularJS fürs Frontend, Bootstrap 3 für das Design.

Um den Rahmen dieser Arbeit nicht zu übersteigen, werden nur Teile des WCMS umgesetzt. Dazu gehören die Konfiguration der Frameworks, das Datenbankdesign, die einfache Verwaltung von Seiten, Beiträgen und Kategorien, die Media Center und zwei mögliche Sektionen.

Die Inhaltstypen kennt man in abgewandelter Form von Drupal mit dem Hauptunterschied, dass Inhaltstypen über die Administration erstellt werden und somit zwar einfacher zu

erstellen, aber recht unflexibel sind. Das Erstellen von Beiträgen, die Funktionsweise von Kategorien und die Media Center erinnern am ehesten an WordPress, aber auch an Joomla.

Dieses WCMS soll vom Entwickler mehr Erfahrung abverlangen zu Gunsten des Anwenders. So ist das Backend ausschließlich für den Administrator und den Autor gedacht. Neue Sektionen und Inhaltstypen werden vom Entwickler eigens in gängigen Frameworks entwickelt. Damit soll eine gute Benutzerfreundlichkeit erreicht werden. Der Funktionsumfang und die Flexibilität sollen mit Hilfe der eigenen Administration pro Sektion erreicht werden. Die Entwicklung in gängigen Frameworks erhöht ebenfalls die Flexibilität für den Entwickler.

ABBILDUNGSVERZEICHNIS

LITERATURVERZEICHNIS

2017. AIIM. [Online] 2017. https://www.aiim.org/.

Barker, Deane. 2016. *Web Content Management: Patterns and Best Practices.* [Hrsg.] O'REILLY Media Inc. s.l. : O'REILLY, 2016. ISBN 978-1-491-90812-9.

Bundesamt für Sicherheit in der Informationstechnik. *Sicherheitsstudie Content Management Systeme (CMS).*

c't – Magazin für Computertechnik. **Schürmann, Tim. 2016.** 25, s.l. : Heise Medien GmbH & Co. KG, 2016.

Dāsa, Rādhārādhya . 2016. *Learn CakePHP: With Unit Testing.* s.l. : Apress; Auflage: 2nd ed. (22. August 2016), 2016. ISBN 9781484212134.

Fjordvald, Martin und Nedelcu, Clement. 2018. *Nginx HTTP Server - Fourth Edition: Harness the power of Nginx to make the most of your infrastructure and serve pages faster than ever before.* s.l. : Packt Publishing; Auflage: 4th Revised edition (14. Februar 2018), 2018. ISBN: 9781788623551.

Foundation, Cake Software. 2018. CakePHP. [Online] 23. 6 2018. https://book.cakephp.org/3.0/_downloads/en/CakePHPCookbook.pdf.

Gralak, Michael und Thorsten Stark. 2015. *Schnelleinstieg App Usability.* s.l. : Franzis Verlag; Auflage: 1, 2015. ISBN: 9783645602594.

Markus Beier, Vittoria von Gizycki. 2002. *Usability: Nutzerfreundliches Web-Design (X.media.press).* s.l. : Springer; Auflage: 2002, 2002. ISBN 978-3-642-56377-5.

Nielsen, Jakob. 1998. *Designing Web Usability: The Practice of Simplicity.* s.l. : New Riders Publ; Auflage: 01 (1. Februar 1998), 1998. ISBN 9781562058104.

Passaglia , Andrea. 2017. *Vue.js 2 Cookbook: Build modern, interactive web applications with Vue.js.* s.l. : Packt Publishing (28. April 2017), 2017. ISBN: 9781786468093.

Porsche, Anton und Heiduk, Marc. 2017. *Laravel in 60 Minuten: Eine Einführung in das moderne PHP-Framework.* s.l. : epubli; Auflage: 3 (11. Juni 2017), 2017. ISBN 3745062957.

Pröll, Stefan, Zangerle, Eva und Gassler, Wolfgang. 2015. *MySQL: Das umfassende Handbuch.* s.l. : Rheinwerk Computing; Auflage: 3 (25. Mai 2015), 2015. ISBN 3836237539.

Rodewig, Klaus. 2011. *Webserver einrichten und administrieren.* s.l. : Galileo Computing; Auflage: 2 (28. Dezember 2011), 2011. ISBN 3836217082.

Stieglitz, Stefan. 2008. *Governance of Virtual Communities: instruments, mechanisms, and interdependences.* Universität Potsdam. 2008. S. 106, Dissertation. QR 760.

Tarasiewics, Philipp und Böhm, Robin. 2017. *AngularJS: Eine praktische Einführung in das JavaScript-Framework.* s.l. : dpunkt.verlag GmbH; Auflage: 1., Auflage (29. Mai 2014), 2017. ISBN 9783864901546.

2018. Wikipedia - Don’t repeat yourself. [Online] 15. 10 2018. https://de.wikipedia.org/wiki/Don%E2%80%99t_repeat_yourself.

2018. Wikipedia - Drupal. [Online] 10. 10 2018. https://de.wikipedia.org/wiki/Drupal .

2018. Wikipedia - Joomla. [Online] 10. 10 2018. https://de.wikipedia.org/wiki/Joomla .

2017. Wikipedia - Medieninhalt. [Online] 27. 11 2017. https://de.wikipedia.org/wiki/Medieninhalt.

2018. Wikipedia - Ruby on Rails. [Online] 24. 5 2018. https://de.wikipedia.org/wiki/Ruby_on_Rails.

2017. Wikipedia - Schriftgutverwaltung. [Online] 26. 11 2017. https://de.wikipedia.org/wiki/Schriftgutverwaltung.

2018. Wikipedia - WordPress. [Online] 10. 10 2018. https://de.wikipedia.org/wiki/WordPress .