THM | CAMPUS GIESSEN | MNI
Mathematik, Naturwissenschaften und Informatik
TECHNISCHE HOCHSCHULE MITTELHESSEN

Bachelorthesis

Traditionelle CMS vs. Flat-File CMS
Ein umfassender Vergleich

zur Erlangung des akademischen Grades

Bachelor of Science

eingereicht im Fachbereich Mathematik, Naturwissenschaften und Informatik an der Technischen Hochschule Mittelhessen

von

Chiara Michelle Funke

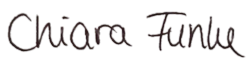
2. August 2023

Referent: Prof. Dr. Peter Kneisel

Korreferent: Kevin Linne

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.



Gießen, den 2. August 2023 Chiara Michelle Funke

Diese Bachelorarbeit befasst sich mit einem umfassenden Vergleich zwischen traditionellen Content Management Systemen (CMS) und Flat-File CMS. Zu Beginn werden die grundlegenden Komponenten von CMS erläutert und die Unterschiede in der Datenstruktur zwischen den beiden Ansätzen beleuchtet. Anhand verschiedener Kriterien, wie Kosten, Lizenzierung, Entwicklerfreundlichkeit und Aktualisierung, werden bekannte CMS verglichen und bewertet. Die Umsetzung erfolgt durch die Implementierung eines Beispielprojekts eines Custom Post Types für Bücher mithilfe des Falt-File Systems Kirby und des traditionellen Systems WordPress.

Die Ergebnisse zeigen, dass die Wahl des geeigneten Systems von spezifischen Faktoren wie dem Anwendungsszenario und der Datenmenge abhängt. Die Arbeit identifiziert Kirby, Grav, Drupal, Joomla, Typo3, Statamic und WordPress als die besten CMS basierend auf den verglichenen Kriterien.

Inhaltsverzeichnis

1 Einführung

Content Management Systeme, häufig abgekürzt als CMS, sind heutzutage ein wichtiger Bestandteil der Webentwicklung, da sie es ermöglichen, Webseiten einfach und effizient zu erstellen und zu verwalten. CMS bieten eine Vielzahl von Funktionen, die es Nutzern ermöglichen, Inhalte zu erstellen, zu bearbeiten und zu veröffentlichen, ohne tiefgreifende technische Kenntnisse zu besitzen.

Jeden Tag interagieren die meisten Menschen mit Content Management Systemen, ohne es überhaupt zu bemerken oder sich dessen bewusst zu sein. Wenn eine Nachrichtenseite oder eine andere Webseite mit regelmäßig aktualisierten Inhalten besucht wird, wird das CMS im Hintergrund aktiv, um sicherzustellen, dass der Inhalt effizient organisiert und präsentiert wird.

In den vergangenen Jahren hat sich jedoch eine neue Art von CMS entwickelt – das sogenannte Flat-File CMS. Im Gegensatz zu traditionellen CMS, die auf Datenbanken zur Speicherung von Inhalten und Konfigurationseinstellungen angewiesen sind, speichern Flat-File CMS Inhalte und Konfigurationen in einfachen Textdateien.

In dieser Bachelorarbeit wird sich mit den verschiedenen Aspekten der beiden CMS-Ansätze auseinandergesetzt, darunter deren technische Merkmale, Vor- und Nachteile sowie Anwendungsfälle. Des Weiteren wird ein umfassender Vergleich zwischen traditionellen und Flat-File CMS durchgeführt.

Die Ergebnisse dieser Untersuchung werden dabei helfen, ein umfassendes Verständnis für die Unterschiede und Gemeinsamkeiten zwischen traditionellen und Flat-File CMS zu erlangen, eine fundierte Antwort auf die Frage zu liefern, welche Vor- und Nachteile traditionelle und Flat-File CMS jeweils bieten und eine informierte Entscheidung darüber zu treffen, welcher Ansatz am besten den Anforderungen von Webseiten und digitalen Inhalten gerecht wird und welche Aspekte bei der Auswahl eines CMS für bestimmte praktische Szenarien zu berücksichtigen sind.

Im weiteren Verlauf der Arbeit wird auf die wissenschaftliche Methodik und den Aufbau der Arbeit eingegangen, um die Schritte und Herangehensweise transparent zu machen. Außerdem werden die spezifischen Fragestellungen und Ziele der Arbeit definiert (Kapitel 1.2) und die Vorgehensweise der Forschung beschrieben (Kapitel 1.4). In der Abgrenzung wird erläutert, welche Themenbereiche behandelt werden und welche nicht (Kapitel 1.3). Abschließend wird die Struktur der Arbeit vorgestellt, um dem Leser einen Überblick über den Aufbau zu geben.

1.1 Problembeschreibung und Motivation

Der Bedarf an Webseiten wächst stetig und damit einhergehend steigen auch die Marktanteile von CMS von Jahr zu Jahr. Insbesondere WordPress, das weltweit bekannteste CMS, hat in den vergangenen Jahren eine beeindruckende Entwicklung durchlaufen. Im Jahr 2023 wurden im Vergleich gegenüber dem Jahr 2018 etwa ein Drittel mehr Webseiten mit WordPress erstellt [W3T23f, Nie18] (siehe Anhang A). Auch Shopify, ein weiteres beliebtes CMS, hat seit 2018 eine enorme Steigerung der Nutzung um das Vierfache verzeichnet [W3T23f, Nie18]. WordPress bleibt unangefochten an der Spitze und hält mit einem Marktanteil von 64 % deutlich den ersten Platz, gefolgt von Shopify auf dem zweiten Platz [W3T23e] (siehe Abbildung 1.1).

Im Gegensatz dazu werden Flat-File Content Management Systeme wesentlich seltener genutzt. Ein Beispiel hierfür ist Kirby, ein vergleichsweise neues CMS, das im Jahr 2012

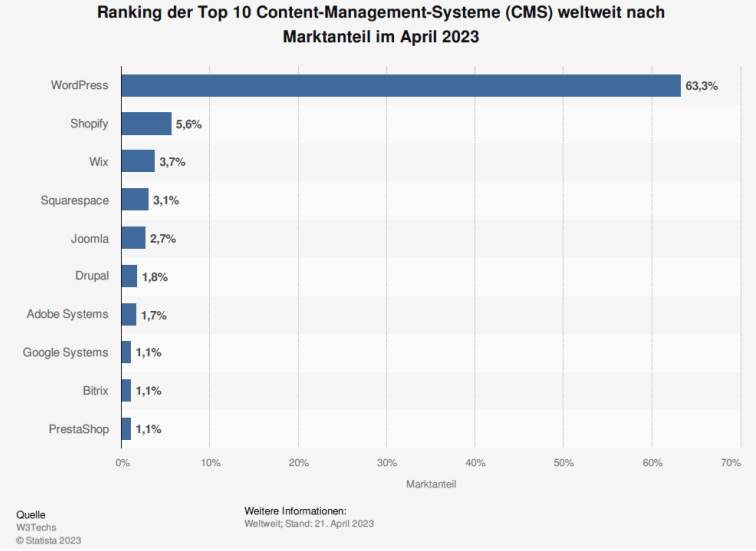


Abbildung 1.1: Marktanteile CMS 2023,
Quelle: W3Techs

eingeführt wurde. Obwohl die Nutzung von Kirby im Vergleich zu anderen CMS gering ist und weniger als 0,1 % aller mit Content Management Systemen erstellten Webseiten ausmacht, ist jedoch ein Anstieg in der Nutzung von 31 % innerhalb des letzten Jahres zu verzeichnen [W3T23g] (siehe Anhang A).

Im Vergleich zu anderen Content Management Systemen hat das Flat-File-System Kirby eine geringere Verbreitung und wird insgesamt von weniger Webseiten und solchen mit geringerem Datenverkehr genutzt [W3T23g] (siehe Anhang A).

In Bezug auf die Konkurrenz unter den Flat-File-Systemen hat Kirby jedoch eine starke Marktposition, insbesondere im Vergleich zu Pico, das fast gar nicht mehr genutzt wird [W3T23c]. Im Vergleich zu HTMLy wird Kirby knapp achtmal häufiger genutzt, auch von Webseiten mit höherem Datenverkehr als HTMLy [W3T23b] (siehe Anhang A). Auffällig ist auch, dass die Nutzung von HTMLy seit einem Jahr konstant bleibt [W3T23b]. Bludit wird im Vergleich zu Kirby nur halb so oft eingesetzt, beide Systeme haben jedoch bei Webseiten einen vergleichbaren Datenverkehr [W3T23b]. Grav wird circa dreimal so häufig genutzt wie Kirby, insbesondere von Webseiten mit etwas höherem Datenverkehr als bei Kirby [W3T23b]. Statamic hat im letzten Jahr einen großen Anstieg in der Nutzung erfahren – diese ist um ein Drittel angestiegen. Dieses System wird, ebenfalls wie Grav, kapp dreimal häufiger genutzt als Kirby [W3T23b]. Im Vergleich zu WordPress hat Kirby eine weitaus geringere Verbreitung [W3T23d] (siehe Anhang A).

Kirby hat jedoch eine treue Benutzerbasis. Es wechseln nur wenige Webseiten, die Kirby nutzen, auf WordPress (2,86 %) [W3T23a]. Ebenso hat nur ein kleiner Anteil die Nutzung von WordPress kürzlich auf Kirby umgestellt (0,95 %). Insgesamt wechselten 2,29 % der Webseiten von einem anderen CMS zu Kirby, während 3,44 % von Kirby zu einem anderen CMS wechselten [W3T23a] (siehe Abbildung 1.2). Dies zeigt, dass es einen gewissen Grad an Dynamik und Veränderung in der Nutzung von Content Management Systemen gibt.

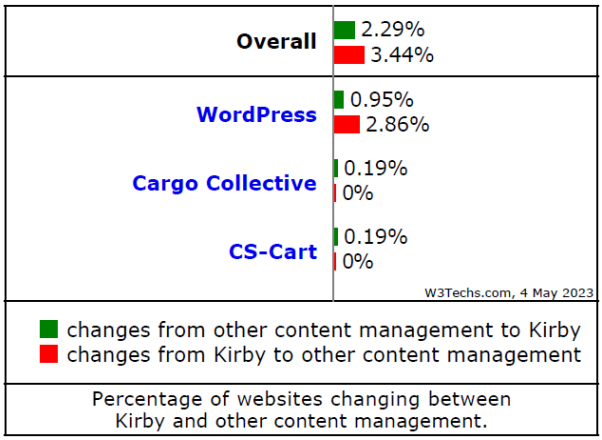


Abbildung 1.2: CMS Wechsel – Kirby,
Quelle: W3Techs

Version 3 von Kirby wird derzeit mit 60 % am häufigsten genutzt, gefolgt von Version 2 mit 30 % der Nutzer und Version 1 mit 10 % der Nutzer. Beliebte Webseiten, die Kirby verwenden, sind beispielsweise Sygic.com [W3T23g] (siehe Anhang A).

Angesichts des steigenden Trends, bei dem immer mehr Unternehmen von traditionellen CMS wie WordPress auf Flat-File CMS wie Kirby umsteigen [Bus23, Kou, Mic16], ist es von Bedeutung, diese beiden Arten von Content Management Systemen eingehend zu vergleichen und Unterschiede herauszufiltern. Es ist außerdem wichtig, spezifische Anwendungsfälle und Szenarien zu analysieren, in denen der Einsatz von Flat-File CMS oder traditionellen CMS am besten geeignet ist. Durch eine umfassende Analyse können Vor- und Nachteile der beiden Ansätze identifiziert und fundierte Entscheidungen für die Verwendung des jeweils passenden CMS in verschiedenen Situationen getroffen werden.

1.2 Ziele dieser Arbeit

Das Ziel dieser Arbeit ist es, einen Überblick über die Grundlagen von Content Management Systemen zu bekommen eine umfassende Untersuchung der Gemeinsamkeiten und Unterschiede zwischen Flat-File CMS und traditionellen CMS durchzuführen.

Durch diese Analyse sollen die Stärken und Schwächen beider Ansätze beleuchtet werden, um als Entscheidungshilfe für die Auswahl des geeigneten CMS zu dienen. Zudem soll ein umfassender Überblick über die Features, sowie Vor- und Nachteile von Content Management Systemen gegeben werden. Außerdem sollen mögliche Kriterien für einen Vergleich herausgefiltert werden.

Im Laufe der Arbeit werden die folgenden Forschungsfragen ausführlich behandelt und beantwortet und jeweils entsprechend mit den Beschriftungen F1 bis F4 referenziert:

[F1] Welche Komponenten bilden die Grundlage von CMS und wie sind sie definiert?

[F2] Welche Merkmale und Arbeitsweisen kennzeichnen traditionelle CMS?

[F3] Was sind Flat-File CMS und wie ist deren Funktionsweise?

[F4] Welches sind mögliche Kriterien für einen Vergleich von traditionellen CMS und Flat-File CMS?

Die Ergebnisse sollen eine Definition von CMS geben und praktische Empfehlungen für die Entwicklung von Webseiten unter Verwendung von bestimmten Content Management Systemen bieten.

1.3 Abgrenzung

In dieser Arbeit wird ein Vergleich zwischen traditionellen CMS und Flat-File CMS durchgeführt, wobei nicht tiefergehend auf andere Kategorien wie Digital Experience Plattformen, Headless CMS, Hybrid CMS, Cloud CMS, Static Site Generatoren oder Decoupled CMS eingegangen wird.

Das Beispielszenario konzentriert sich auf die Erstellung und Programmierung von Post Types mit WordPress und Kirby und berücksichtigt keine anderen Arten von Anwendungen oder CMS-Systemen.

1.4 Vorgehensweise

Um die oben genannten Ziele zu erreichen, wird diese Arbeit in einen theoretischen und einen praktischen Teil unterteilt.

Der theoretische Teil der Arbeit basiert auf einer umfassenden Literaturrecherche, bei der relevante wissenschaftliche Artikel, Fachzeitschriften und Online-Ressourcen analysiert werden. Dadurch kann eine Einführung in Content Management Systeme und auch Flat-File CMS gegeben werden, wodurch die Fragen „Welche Komponenten bilden die Grundlage von CMS und wie sind sie definiert?“ [F1], „Welche Merkmale und Arbeitsweisen kennzeichnen traditionelle CMS?“ [F2] und „Was sind Flat-File CMS und wie ist deren Funktionsweise?“ [F3] beantwortet werden. Außerdem werden die Unterschiede der Speicherarten erarbeitet und zusätzlich dazu bekannte CMS vorgestellt. Diese Ergebnisse definieren die Vergleichskriterien, die im Teil „Analyse und Vergleich“ betrachtet werden [F4].

Im praktischen Teil der Arbeit wird eine Implementierung eines Beispielszenarios durchgeführt und dokumentiert. Die Ergebnisse der Analyse werden anhand der vorab definierten Vergleichskriterien in einer strukturierten Übersicht zusammengefasst und bewertet. Diese Informationen und Ergebnisse sollen den Anwendern und Entwicklern dabei helfen, zu entscheiden, welches CMS für spezifische Projekte am besten geeignet ist.

Das Beispielszenario wird die Erstellung und Programmierung eines Post Types sein, der es ermöglicht, mehrere Einträge mit demselben Aufbau und denselben Feldern zu erstellen.

1.5 Struktur der Arbeit

Die Bachelorarbeit unterteilt sich in sechs Kapitel.

Die Arbeit beschäftigt sich im ersten Kapitel mit der Problemstellung und Motivation. Es wird erläutert, warum ein Vergleich zwischen traditionellen Content Management Systemen und Flat-File CMS von Relevanz ist und welche Ziele dadurch verfolgt werden sollen. Außerdem werden die Methodik und Vorgehensweise beschrieben, um die definierten Ziele zu erreichen und es wird ein grober Überblick darüber gegeben, was in dieser Ausarbeitung behandelt wird.

Das zweite Kapitel wird eine Einführung in grundlegende Konzepte und Allgemeinem von Content Management Systemen geben. Außerdem werden die Komponenten betrachtet und definiert, die die Grundlage von CMS bilden.

Im dritten Kapitel werden die Definition und Funktionsweise von traditionellen CMS und Flat-File CMS erläutert, um ein allgemeines Verständnis für die Systeme aufzubauen. Anschließend werden die Unterschiede der Speicherarten beleuchtet und bekannte traditionelle CMS und Flat-File CMS vorgestellt.

Im nächsten Kapitel werden die Vergleichskriterien beschrieben, erläutert und anschließend in Bezug auf die vorgestellten traditionellen und Flat-File CMS verglichen und bewertet.

Das fünfte Kapitel beschreibt die Installationen der Programme für das Beispielszenario und setzt dieses prototypisch aus Sicht eines Entwicklers in beiden CMS um.

Im letzten Kapitel werden die Ergebnisse zusammengefasst und die Vor- und Nachteile beider Content Management Systeme aufgezeigt, um zu ermitteln, welches CMS im konkreten Anwendungsbeispiel besser geeignet gewesen ist. Außerdem werden alle Forschungsfragen kurz und prägnant beantwortet und ein Fazit gezogen.

2 Theoretischer Hintergrund

In diesem Kapitel werden die grundlegenden Konzepte und das allgemeine Wissen über Content Management Systeme vorgestellt.

2.1 Grundlagen

Content Management Systeme sind aus der heutigen Webentwicklung nicht mehr wegzudenken und bieten eine effektive Möglichkeit, Inhalte auf Webseiten zu erstellen, zu verwalten und zu veröffentlichen. Aber was genau kann unter „Content“ oder „Content Management“ verstanden werden? In diesem Teil soll ein umfassender Überblick über die Grundlagen gegeben und eine Basis für das Verständnis geschaffen werden. Dadurch soll eine Antwort auf die Frage „Welche Komponenten bilden die Grundlage von CMS und wie sind sie definiert?“ [F1] gegeben und ein Grundverständnis geschaffen werden, um im weiteren Verlauf die Forschungsfragen F2 und F3 beantworten zu können.

2.1.1 Definition von Content

Der Begriff „Content“ wird häufig im Bereich Content Management und Content Management Systeme verwendet, aber was genau kann darunter verstanden werden? Um diesen Begriff umfassender zu verstehen, ist es entscheidend, zwischen Daten, Informationen und Inhalten zu unterscheiden.

Daten sind kleine Informationsfragmente, also Rohinformationen, die Informationen, also Sachverhalte und Vorgänge darstellen, die in maschinell verarbeitbarer Form vorliegen ([Loh01, S. 3]; [Boi05, S. 7]). Sie werden von Menschen gesammelt, kombiniert und in Datenbanken gespeichert [Boi05, S. 7] und erst durch Interpretation erlangen sie Sinn und werden zu Informationen [Spö09, S. 5]. Bestimmte Informationen haben dabei einen höheren Nutzungsgrad und können als Objekt- oder Austauschgegenstand betrachtet werden [Loh01, S. 3]. Informationen werden schließlich zu Content, der als Wissen, Informationen und Dokumente zusammengefasst und als Asset betrachtet wird, wenn er einen Wert darstellt ([Loh01, S. 3]; [Spö09, S. 5]). Informationen haben einen Einfluss auf das menschliche Verhalten und können schließlich zu Wissen werden, da sie auf

individueller Wahrnehmung und Interpretation beruhen [Loh01, S. 3]. In Abbildung 2.1 ist zu sehen, wie aus Daten Informationen werden und aus diesen wiederum Content oder Wissen.

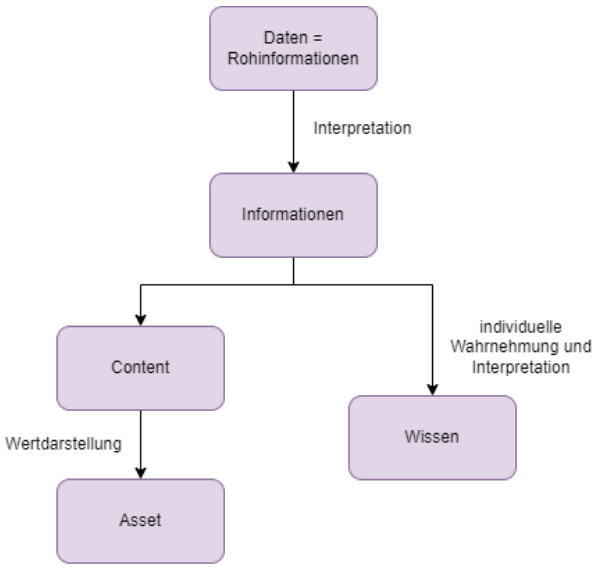


Abbildung 2.1: Daten,
Quelle: In Anlehnung an Lohr, 2001, S.4

Der Begriff Content wird oft übersetzt mit dem deutschen Begriff „Inhalt“, jedoch ist diese Beschreibung laut Nix sehr unpräzise [Nix05, S. 25]. Content bezieht sich nicht nur auf Texte, Audio- und Videodateien, sondern auch Metadaten wie Autor, Titel und Beschreibung – also jede Inhaltsart in digitaler Form ([Nix05, S. 25]; [Spö09, S. 5]). Metadaten sind "Data about Data"[Boi05, S. 11], die Informationen über den Kontext und die Bedeutung von Informationen liefern. Es handelt sich um zusätzliche Informationen, die an den Inhalt angehängt werden, um die Katalogisierung, Speicherung und den Abruf zu erleichtern. Diese Metadaten erleichtern es Computern, mit den Daten umzugehen [Boi05, S. 11].

Content kann also als die Summe von Inhalt, Layout und Struktur erklärt werden (mehr dazu in Kapitel 2.1.2), die aus einzelnen Informationen oder einzelnen Komponenten besteht [Spö09, S. 5-6, 44]. Es sind also Daten, die für einen bestimmten Zweck erstellt werden [Bar16, S. 4]. Boiko sagt, dass Informationssysteme uns helfen, mit Menschen zu sprechen, die nicht vor uns stehen [Bar16, S. 5]. Dabei geht es also darum, Informationen auf eine klare und verständliche Weise zu präsentieren und so eine Kommunikation zwischen Absender und Empfänger zu ermöglichen.

Bob Boiko beschäftigt sich in seinem Buch „The Content Management Bible“ mit der Frage, was Content ist und unterscheidet die Erstellung und Verwendung von Inhalten [Bar16, S. 3]. Inhalte werden durch einen redaktionellen Prozess erstellt [Bar16, S. 3] und

durch Veröffentlichung bereitgestellt [Bar16, S. 5]. Ein Prozess ist das, was Menschen tun, um Informationen für die Veröffentlichung an ein Publikum vorzubereiten [Bar16, S. 3]. Der redaktionelle Prozess besteht aus verschiedenen Schritten wie Modellieren, Verfassen, Bearbeiten, Überprüfen, Genehmigen, Versionieren, Vergleichen und Steuern [Bar16, S. 3]. Dabei basiert die Erstellung von Inhalten auf den Meinungen menschlicher Redakteure, die entscheiden, welches Thema behandelt wird, wer die Zielgruppe ist und wie der Inhalt angegangen wird [Bar16, S. 4]. Der redaktionelle Prozess ist jedoch nicht immer linear und bereits veröffentlichte Inhalte können später wieder verändert oder ergänzt werden. Inhalte sind also ständig in Bewegung und werden immer wieder überarbeitet und angepasst, um den Bedürfnissen des Publikums gerecht zu werden [Bar16, S. 4]. Es gibt außerdem einen Unterschied zwischen dem Management und der Bereitstellung von Inhalten, denn Inhalte werden erstellt und verwaltet, bevor sie veröffentlicht und bereitgestellt werden. Das Management umfasst die Organisation, Verwaltung und Kontrolle der verschiedenen Inhalte, die Bereitstellung hingegen bezieht sich auf die eigentliche Veröffentlichung der Inhalte über eine Webseite, soziale Medien oder andere Kanäle [Bar16, S. 5].

Zusammenfassend kann gesagt werden, dass Content ein sehr breiter Begriff ist, der sich auf die Erstellung und Verwendung von Inhalten für einen bestimmten Zweck bezieht. Diese Inhalte werden von Redakteuren erstellt und kontinuierlich angepasst, um den Bedürfnissen des Publikums gerecht zu werden. Informationssysteme helfen uns dabei, mit Menschen zu kommunizieren, die nicht direkt vor uns stehen. Sie ermöglichen eine klare und verständliche Präsentation von Informationen, um eine effektive Kommunikation zwischen Absender und Empfänger zu gewährleisten.

2.1.2 Definition von Content Management

Der Begriff Content Management wird oft im Kontext mit der Verwaltung von Webseiten verwendet [Spö09, S. 6]. Laut Nix bezeichnet er den gesamten Prozess der „systematischen und strukturierten Verwaltung von Inhalten in Form von modularisierten elektronischen Inhalten“ [Nix05, S. 25]. Dieser Prozess, auch Content-Lifecycle genannt, umfasst die Erstellung, Verwaltung und Freigabe von Inhalten sowie alle Aspekte der Verwaltung in Übereinstimmung mit der Corporate Identity ([Nix05, S. 15, 19]; [Spö09, S. 6, 8]). Auf den Content-Lifecycle wird in Kapitel 2.1.4 genauer eingegangen.

Eine wichtige Anforderung an ein erfolgreiches Content Management ist eine explizite Trennung der Elemente Inhalt, Layout (Design) und Struktur ([Spö09, S. 6]; [Nix05, S. 25]; [Loh01, S. 4]; [Jab02, S. 102]). Die Möglichkeit zur Speicherung und Verarbeitung von Inhalten durch Content Management Systeme wird durch diese getrennte Speicherung und Verarbeitung ermöglicht [Loh01, S. 5].

Der Inhalt selbst sind die Rohdaten, die getrennt von Struktur und Darstellung gespeichert werden [Spö09, S. 6]. Diese Rohdaten sind Zeichenketten und setzen sich aus verschiedenen Teilmengen zusammen, die durch Daten repräsentiert werden und aus Text, Bildern, Sprache, Musik, Filmen und Programmen bestehen können [Loh01, S. 4]. Das Layout oder Design ist in sogenannten Stylesheet- oder Templatedateien gespeichert, die die Form definieren und Anweisungen liefern, wie der Inhalt präsentiert werden soll [Loh01, S. 4]; [Spö09, S. 6]; [Böh14, S. 23]. Die Struktur hingegen ist die Art und Weise, in der Informationen organisiert sind [Boi05, S. 21]. Sie umfasst die Gliederung und Klassifikation des Inhalts sowie Ergänzungen durch Links und Beziehungen [Loh01, S. 4]. Sie definiert Einzelinformationen, wie die Definition des Dokumenttyps, das Datenbankschema, die Reihenfolge oder die Verschachtelung ([Spö09, S. 6]; [Jab02, S. 107]). Die Struktur bestimmt also die Organisation und Präsentation der Daten. Abbildung 2.2 zeigt die Aufteilung von Content. Ein Beispiel für die Trennung von Inhalt, Layout und Struktur ist in Abbildung 2.3 anhand eines Bildes mit Beschreibung dargestellt.

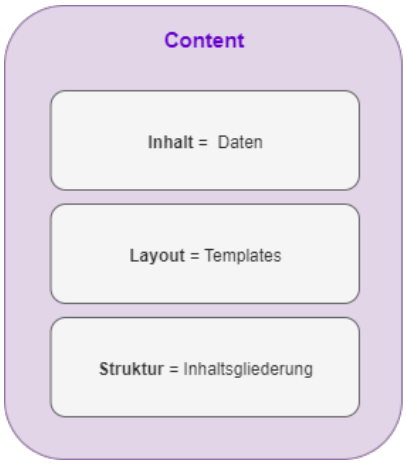


Abbildung 2.2: Content,
Quelle: Eigener Entwurf

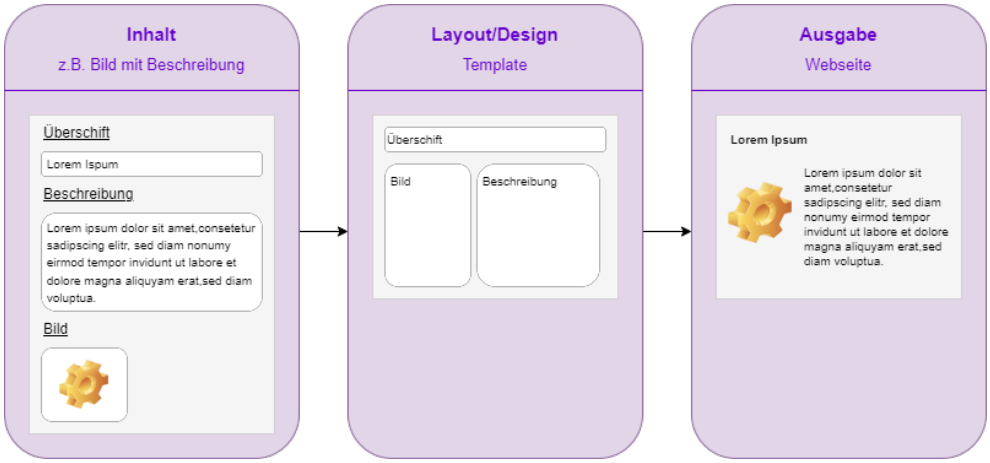


Abbildung 2.3: Content-Darstellung,
Quelle: Eigener Entwurf

Es gibt es zwei Arten von Strukturen: statische und dynamische ([Böh14, S. 21]; [Boi05, S. 75]). Eine Möglichkeit, Informationen zu präsentieren, ist eine statische Webseite (siehe Abbildung 2.4), die aus vorab erstellten HTML-Seiten und zugehörigen Dateien besteht, die auf einem Webserver gespeichert sind [Boi05, S. 75]. Dabei wird der Inhalt in HTML-Dateien gespeichert [Böh14, S. 21]. Sie bestehen aus einigen HTML-Dateien, wobei für jede Seite eine eigene HTML-Datei benötigt wird, einer CSS-Datei sowie einem Ordner mit Bildern [Böh14, S. 21]. Der Inhalt kann aktualisiert werden, indem die alten Seiten durch neue ersetzt werden [Boi05, S. 75], wodurch die Aktualisierung des Inhalts aufwendig ist [Böh14, S. 22]. Diese Art von Webseite eignet sich deshalb gut für kleine Webseiten, wie Vereine, Grundschulen oder Ladengeschäfte, bei denen sich Inhalte nicht oft ändern und keine Personalisierung erforderlich ist ([Boi05, S. 75]; [Böh14, S. 21]). Insgesamt sind statische Webseiten schnell, bieten jedoch weniger Flexibilität und Skalierbarkeit [Boi05, S. 75].

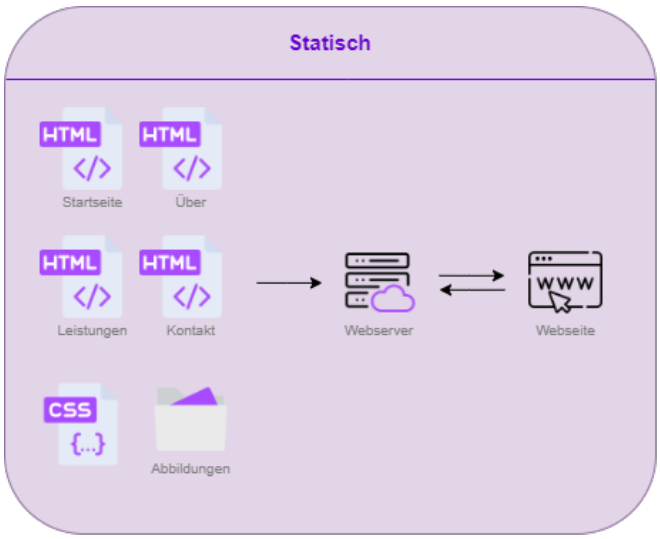


Abbildung 2.4: Statische Struktur,
Quelle: Eigener Entwurf

Im Gegensatz dazu steht die dynamische Webseite (siehe Abbildung 2.5), auch datenbankgesteuerte Webseite genannt [Boi05, S. 75]. Sie setzt darauf, den Content vollständig von der Umsetzung zu trennen [Böh14, S. 22]. Die dynamische Struktur generiert Seiten auf Anfrage des Benutzers durch etwa eine Suche oder einen Buttonklick ([Boi05, S. 75]; [Böh14, S. 22]). Die Datenquelle für Inhalte ist dabei eine Datenbank oder eine XML-Struktur ([Boi05, S. 75]; [Böh14, S. 22]), wodurch eine personalisierte Darstellung von Inhalten basierend auf Benutzeranfragen ermöglicht wird [Boi05, S. 75]. Diese Methode ist flexibler und eignet sich für größere Webauftritte [Böh14, S. 23].

Beide Strukturen erlauben die Trennung von Inhalt und Design und die unabhängige Änderung beider Bereiche [Böh14, S. 22].

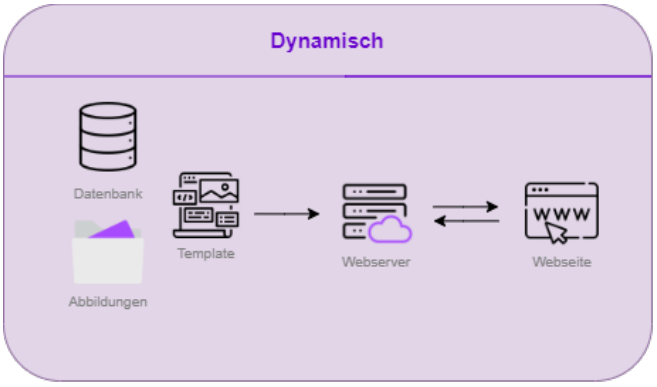


Abbildung 2.5: Dynamische Struktur,
Quelle: Eigener Entwurf

Modellierung von Inhalten

Die Inhaltsmodellierung ist der Prozess, bei dem die Struktur und Attribute von Inhalten definiert werden, die von einem Content Management System (CMS) verwaltet werden sollen. Hierbei geht es darum, das logische Konzept von Inhalten in eine spezifische Form zu übersetzen, die von einem CMS verarbeitet werden kann [Bar16, S. 76]. Um etwa eine Pressemitteilung zu definieren, müssen Komponenten wie Überschrift, Text, Veröffentlichungsdatum und Autor angegeben und Regeln und Einschränkungen festgelegt werden. Das Ergebnis ist eine Beschreibung aller vom CMS zu verwaltenden Inhalte, die als Inhaltsmodell bekannt ist [Bar16, S. 76]. Ein Inhaltsmodell, auch Content-Modell, wird durch Elemente wie Content-Typen und Attribute definiert [Bar16, S. 85]. Content-Typen stellen eine logische Gruppierung von Inhalten basierend auf ihrer Struktur und ihrem Zweck dar, und jeder Typ erfüllt eine andere Rolle im Modell und besteht aus unterschiedlichen Informationen [Bar16, S. 85]. Dabei ist es wichtig, eine klare Grenze zwischen Inhaltstyp und Inhaltsobjekt zu ziehen. Ein Inhaltstyp dient als Muster für ein Inhaltsobjekt, das möglicherweise tausendfach erstellt werden kann [Bar16, S. 86]. Ein Beispiel hierfür ist ein Ausstecher, mit dem viele Plätzchen erstellt werden können. Der Ausstecher ist der Inhaltstyp, während die Plätzchen die einzelnen Inhaltsobjekte darstellen. Ein Inhaltstyp kann als „Muster“ für einen Inhalt oder als Definition der Informationen betrachtet werden, die dieser benötigt, um als gültig betrachtet zu werden [Bar16, S. 86]. Die Organisation von Inhalten in Typen bietet mehrere Vorteile [Bar16, S. 87]. Sie sorgt für eine klare Struktur, da verschiedene Inhaltstypen unterschiedliche Informationen benötigen, um als gültig zu gelten. Eine Person benötigt so einen Vornamen, für eine Seite ist dieser aber nicht sinnvoll. Auch die Usability wird verbessert, da die meisten CMS automatisch eine passende Bearbeitungsoberfläche erstellen [Bar16, S. 87]. Ein weiterer Vorteil ist die einfache Suche nach spezifischen Inhalten, wenn alle Inhalte des gleichen Typs beisammen sind, ist es leicht diese zu finden. Das Templating, wird ebenfalls vereinfacht, da verschiedene Inhaltstypen unterschiedliche Methoden benötigen, um ihre Informationen auszugeben. Schließlich können Berechtigungen je nach Typ eingeschränkt werden, etwa so, dass nur

die Personalabteilung Mitarbeiterbiografien bearbeiten darf [Bar16, S. 87]. Attribute sind die kleinsten Einheiten von Informationen in einem Inhaltsmodell und beschreiben individuelle Informationsstücke eines Objekts [Bar16, S. 88]. Jedes Attribut ist einem spezifischen Datentyp zugeordnet, der die Art der gespeicherten Informationen festlegt. Typische grundlegende Datentypen sind Text (variabler Länge), Zahl, Datum, Bild, Datei oder Verweis auf andere Inhalte [Bar16, S. 88]. Integrierte Attribute sind bei den meisten CMS automatisch im Inhaltstyp vorhanden und müssen nicht separat hinzugefügt werden [Bar16, S. 90]. Dazu gehören die Identifikationsnummer (ID), der Titel, der Inhalt (Body) des Typs und eine Zusammenfassung [Bar16, S. 90-91].

Das Ziel des Content Managements besteht darin, alle Inhalte unabhängig von ihrem Format oder Typ in einem klar definierten Inventar zu verarbeiten und zu verwalten [Spö09, S. 6]. Dieser Prozess erleichtert die Strukturierung und Verwaltung von Inhalten auf Webseiten, da die Inhalte zu jedem Zeitpunkt in einer Form bereitliegen, in der sie verwendet werden können [Jab02, S. 105] und gewährleistet eine konsistente Darstellung.

2.1.3 Definition von Content Management Systemen

Content Management Systeme sind häufig bei Blogs, Nachrichtenseiten und Einkaufsportalen in Verwendung, aber was ist ein Content Management System? Ein Content Management System (CMS), auf Deutsch Inhaltsverwaltungssystem, ist eine Software, die es Nutzern ermöglicht, digitale Inhalte auf einer Webseite dynamisch zu erstellen, zu bearbeiten, zu verwalten und zu veröffentlichen, ohne technische Vorkenntnisse in HTML oder anderen Programmiersprachen zu besitzen ([Nix05, S. 15-16]; [Str22, S. 208]; [Bar16, S. 5]; [Böh14, S. 23]). Dies wird durch automatisierte Schnittstellen erreicht, die auch die Indexierung und den Zugriff auf Inhalte ermöglichen [Str22, S. 208]. Die serverbasierte Software ermöglicht es mehreren Benutzern, mit diesen digitalen Inhalten umzugehen, da sie an einem zentralen Ort, dem Repository gespeichert sind [Bar16, S. 6]. Das Repository ist eine der Komponenten, aus denen sich ein CMS zusammensetzt. Es besteht außerdem noch aus einer Bearbeitungsoberfläche, auch Interface genannt, und Veröffentlichungsmechanismen [Bar16, S. 6].

Ein CMS besteht größer gesehen aus Frontend und Backend [Böh14, S. 327]. Das Frontend ist der öffentlich zugängliche Teil der Webseite, während interne Bereiche eine Anmeldung im CMS erfordern. Die Rechte von angemeldeten Nutzern sind dabei in der Nutzerverwaltung festgelegt. Das Backend hingegen ist den Administratoren der Webseite vorbehalten und beinhaltet grundlegende Einstellungen wie die Menüstruktur, Templates und die Nutzerverwaltung [Böh14, S. 327].

Das CMS trennt Inhalte, Layout (Design) und Struktur und ermöglicht nur wenigen autorisierten Personen den Zugriff auf diesen Bereich ([Nix05, S. 16]; [Böh14, S. 23]).

Durch die Trennung können Änderungen an Texten, Grafiken oder Bildern vorgenommen werden, ohne das Design der Webseite zu ändern. Durch Verwendung von Templates kann so das Corporate Design erhalten werden [Nix05, S. 16]. Ursprünglich hatten CMS das Ziel, die Codierung zu vereinfachen. Heute verfügen sie jedoch über einen großen Funktionsumfang und eine Backend-Schnittstelle ermöglicht die Bearbeitung, Veröffentlichung und Änderung von großen Datenmengen und stellt gleichzeitig eine zentrale Verwaltung sicher [Str22, S. 208].

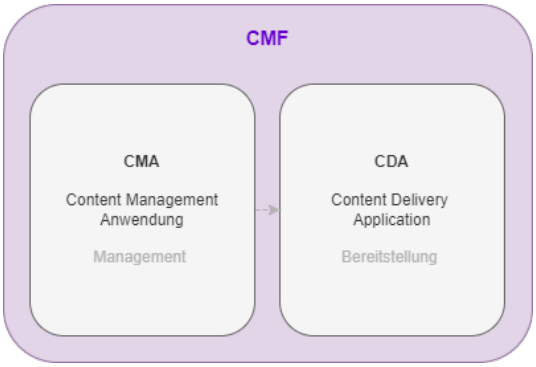


Abbildung 2.6: CMF,
Quelle: Eigener Entwurf

Ein Content Management Framework (CMF) besteht aus zwei Elementen: einer Content Management Application (CMA), die als Tool für den Inhaltsverwalter dient, und einer Content Delivery Application (CDA), die Informationen zusammenstellt und online veröffentlicht [Str22, S. 209] (siehe Abbildung 2.6). Sie bilden eine zentrale Schnittstelle, über die nicht technisch-versierte Benutzer Texte hinzufügen und bearbeiten, Revisionen verwalten, Daten indizieren und Inhalte verwalten können. Ein CMS ermöglicht Nicht-IT- oder Webentwicklungsexperten die Bearbeitung und Gestaltung nach dem Motto „What you see is what you get“ (WYSIWYG-Prinzip) [Str22, S. 209].

CMS-Funktionen wiederum können laut Strekalova in drei Hauptanwendungsbereichen genutzt werden: Datenpflege, Benutzerzugriffsverwaltung und Systemmodifikation [Str22, S. 209]. Bei der Datenpflege sorgt Standardisierung für einen gleichen Aufbau der gemeinsam genutzten Inhalte, ermöglicht eine robustere Analyse der Daten und erleichtert deren Nutzung. Beispiele sind die Automatisierung von Vorlagen, die Daten-Versionierung und die allgemeine Datenverwaltung [Str22, S. 209]. Die Benutzerzugriffsverwaltung umfasst die Berechtigungskontrolle und die Zusammenarbeitsverwaltung [Str22, S. 210]. Dabei können Benutzergruppen erstellt und Zugriffsprivilegien zugewiesen werden. Es ist auch möglich festzulegen, ob und wie Benutzer Inhalte für die Webseite bereitstellen können. Die Zusammenarbeitsverwaltung erlaubt mehreren Benutzern, an Inhalten zu arbeiten, und Administratoren können Workflows einrichten [Str22, S. 210]. Die Systemmodifikation hingegen umfasst die Skalierbarkeit und Upgrades [Str22, S. 210]. Durch Skalierbarkeit ist es möglich, Microsites zu erstellen und den Funktionsumfang des CMS zu erweitern, indem Module und Plug-ins hinzugefügt werden - diese können

Suchoptionen oder Inhalte bereitstellen sowie geteilte Inhalte moderieren. Upgrades hingegen beinhalten Upgrade-Optionen, um das CMS aktuell zu halten [Str22, S. 210].

Ein CMS ist also eine Software, die es Benutzern ermöglicht, digitale Inhalte auf einer Webseite zu erstellen, zu bearbeiten, zu verwalten und zu veröffentlichen, ohne technische Programmierkenntnisse zu benötigen. Es trennt Inhalt, Layout und Struktur, ermöglicht die Zusammenarbeit mehrerer Benutzer und bietet zusätzliche Funktionen.

2.1.4 Content-Lifecycle

Der Content-Lifecycle ist der Lebenszyklus von Informationen auf Webseiten und beschreibt den Prozess, den ein Content von seiner Erstellung bis zu seiner Löschung durchläuft ([Spö09, S. 46]; [Nix05, S. 33]; [Bar16, S. 136]). Dieser Lebenszyklus besteht aus verschiedenen Phasen [Nix05, S. 33] (siehe Abbildung 2.7). Es gibt keine allgemein anerkannte Definition der exakten Stadien und deren Reihenfolge, aber im Allgemeinen können die Phasen in Erstellung, Bearbeitung, Kontrolle, Freigabe, Veröffentlichung, Archivierung und Löschung zusammengefasst werden ([Bar16, S. 136-137]; [Böh14, S. 327]; [Spö09, S. 46]). Dabei werden alle Inhalte wie Texte, Bilder und Medien in der Datenbank des CMS gespeichert und erst beim Aufruf der Seite durch den Nutzer dynamisch konfiguriert und erstellt. Durch diese Methode ist es möglich, die Erstellung von Inhalten von der Veröffentlichung zu trennen [Böh14, S. 327].

In der Erstellungsphase erstellen Autoren die Inhalte als Inhaltsobjekte im CMS, also als digitale Assets wie Texte, Grafiken und andere Medienformate ([Bar16, S. 137]; [Nix05, S. 33]; [Spö09, S. 46]). Diese sind zu dem Zeitpunkt für den externen Nutzer noch nicht sichtbar [Bar16, S. 137]. Während der Bearbeitungsphase werden die Inhalte von einem oder mehreren Redakteuren bearbeitet, sind jedoch ebenfalls noch nicht sichtbar. In der Kontroll- und Freigabephase werden die Inhalte von autorisierten Mitarbeitern auf inhaltliche und gestalterische Korrektheit überprüft und für eine oder mehrere Genehmigungen eingereicht, sind aber noch immer nicht sichtbar ([Bar16, S. 137]; [Nix05, S. 33]). Sobald die Inhalte genehmigt werden, also die gewünschte Qualität erreicht haben, werden sie freigegeben und in die Publikationsphase überführt, wo sie im Internet[1], Intranet[2] oder Extranet[3] veröffentlicht werden – ab diesem Zeitpunkt sind sie für Nutzer sichtbar. Andernfalls werden die Inhalte zurück an die Autoren

1 "Weltweiter Zusammenschluss TCP/IP-basierter Netze, es dient dem weltweiten und grenzenlosen Austausch von Informationen."[Deu98, S. 17]

2 "[...] innerhalb eines Unternehmens oder einer Institution; es dient dem Informationsaustausch zwischen verschiedenen Abteilungen oder dezentral organisierten Mitarbeitern."[Deu98, S. 17]

3 "[...] innerhalb einer fest definierten geschlossenen Gruppe von Unternehmen oder Institutionen; es dient dem gezielten Austausch von Informationen zweier oder mehrerer in Geschäftsbeziehungen stehenden Unternehmen oder zwischen Unternehmen und Behörden."[Deu98, S. 17]

gereicht, die diese Inhalte entsprechend korrigieren ([Nix05, S. 33]; [Spö09, S. 46]). Der Übergang von der Freigabe zur Publikation markiert den Schritt von interner Verarbeitung zu externer Veröffentlichung [Nix05, S. 33]. Nach einer bestimmten Zeit kann der Inhalt aus dem öffentlichen Zugriff entfernt und archiviert werden, ohne jedoch gelöscht zu werden ([Bar16, S. 137]; [Nix05, S. 33]). Dies ist unter anderem nützlich, um Backups zu erstellen, ältere Inhalte wiederzuverwenden oder ältere Inhalte über das Archiv zugänglich zu machen. Die Archivierung kann sowohl intern als auch öffentlich erfolgen [Nix05, S. 33]. Schließlich kann der Inhalt endgültig aus dem CMS gelöscht werden [Bar16, S. 137]. Einige dieser Phasen können iterativ und gleichzeitig für verschiedene Versionen desselben Inhalts gelten [Bar16, S. 137]. Ein Inhalt kann für eine Weile veröffentlicht und schließlich geändert werden. In diesem Fall wird eine neue Version erstellt, die die verschiedenen Phasen durchläuft, während die vorherige Version archiviert wird [Bar16, S. 137].

Insgesamt ist der Content-Lifecycle also wichtig, um sicherzustellen, dass die Inhalte effektiv erstellt, bearbeitet, genehmigt, veröffentlicht, archiviert und gelöscht werden. Die Automatisierung des Content-Lifecycle in CMS kann dabei helfen die Verarbeitung von Content zu verbessern [Spö09, S. 46].

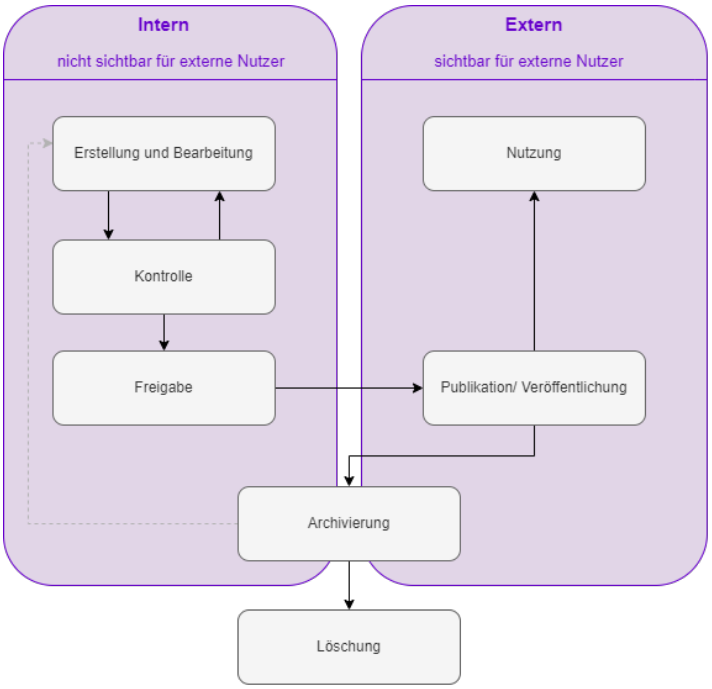


Abbildung 2.7: Content-Lifecycle,
Quelle: In Anlehnung an Spörrer, 2009, S.128

2.2 Allgemeines zu CMS

In diesem Kapitel wird ein allgemeiner Überblick über CMS, einschließlich verschiedener Arten, Vorteile und Grenzen gegeben. Dies hilft, realistische Erwartungen an die Leistung von CMS zu entwickeln und die Vorteile und Einschränkungen zu verstehen.

2.2.1 Arten von CMS

CMS können in verschiedene Kategorien eingeteilt werden. Je nach Quelle können unterschiedliche Kategorien und Unterscheidungen gefunden werden. Im Folgenden soll sich auf die Unterscheidungen der Lizenz, der Speicherform, der Architektur, sowie der gängigen Arten von CMS konzentriert werden (siehe Abbildung 2.8).

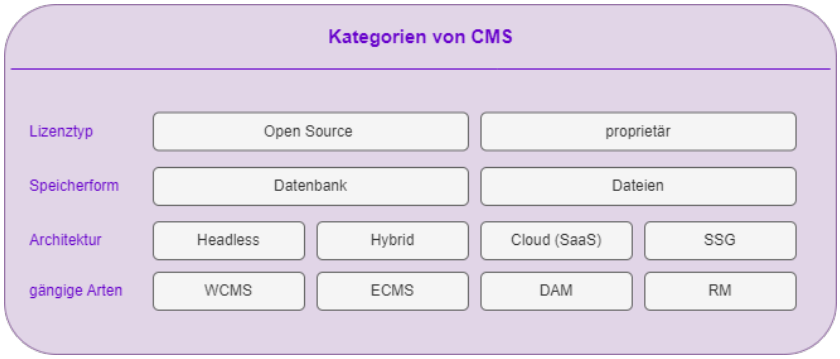


Abbildung 2.8: Kategorien von CMS,
Quelle: Eigener Entwurf

Bei der Kategorie des Lizenztyps wird zwischen Open-Source und proprietären Lizenzen unterschieden ([Sch21c]; [Str22, S.210]). Open-Source-Lizenzen sind offen für alle, können frei genutzt werden und ermöglichen es, den Code anzupassen, um das CMS an spezifische Anforderungen anzupassen ([Sch21c]; [Str22, S.210]). Ein Nachteil ist jedoch, dass diese CMS oft keine strenge Standardisierung haben [Str22, S.210]. Proprietäre Lizenzen hingegen werden von einem Entwickler erstellt und sind mit einem bestimmten Unternehmen verbunden ([Sch21c]; [Str22, S.210]). Sie definieren üblicherweise Arbeitsabläufe für die Erstellung, Überprüfung und Veröffentlichung von Inhalten [Str22, S.210].

Eine weitere Kategorie betrifft die Unterscheidung der Speicherform der Inhalte [Sch21c], auf welche sich im späteren Verlauf dieser Bachelorarbeit näher bezogen wird. Textinhalte können entweder in einer klassischen relationalen Datenbank wie MySQL, MariaDB oder PostgreSQL gespeichert werden, oder in Dateien mit Formaten wie Markdown, TXT, XML, JSON oder HTML [Sch21c]. Bilder, Dokumente oder andere Dateien werden als Assets auf dem Server-File-System abgelegt und einzeln verwaltet [Sch21c].

Des Weiteren kann zwischen verschiedenen Architekturen unterschieden werden [Sch22c, S. 7]. Zu diesen Architekturen gehören zum einen Headless CMS, die eine neue Form von CMS sind. Sie ermöglichen es Autoren, Inhalte zu erstellen und verwalten, ohne dabei eine Webseite zu erstellen [Sch22c, S. 7]. Die Inhalte werden dann über ein Application Programming Interface (API) abgerufen und können für die Webseite genutzt werden ([Sch22c, S. 7], [Sch21b]). Außerdem gibt es auch Hybride CMS. Diese können gleichzeitig als Headless und traditionelles CMS genutzt werden [Sch22c, S. 7]. Cloud CMS wiederum sind als Software-as-a-Service (SaaS) verfügbar und werden nicht lokal gehostet [Sch22c, S. 7]. Die letzte Architektur ist die der Static Site Generator (SSG), die Programme sind, die Markdown in statische Seiten übersetzten [Sch22c, S. 7].

Zuletzt gibt es verschiedene Arten von CMS. Darunter fällt vor allem Web Content Management, aber auch andere geläufige Arten wie Enterprise Content Management, Digital Asset Management und Records Management [Bar16, S. 7]. Das Web Content Management System (WCMS) ermöglicht hauptsächlich die Verwaltung, aber auch die Präsentation und Archivierung von großen Mengen von HTML-Inhalten und Mediendaten wie Texten, Bildern und Filmen im World Wide Web (WWW), die für große Masse bestimmt sind ([Nix05, S. 20]; [Spö09, S. 23-25]; [Bar16, S. 7]). Dieser Prozess wird häufig von zusätzlichen Funktionen für die Bearbeitung im Internet, Intranet oder Extranet begleitet [Spö09, S. 15]. Ein WCMS bildet die Grundlage für alle professionellen Webauftritte [Böh14, S. 324], ermöglicht die Inhaltserstellung und bietet Funktionen zum Überprüfen, Modifizieren und Überwachen von Inhalten über das WWW [Spö09, S. 25]. Sogenannte „Authoring Tools“ [Spö09, S. 25] werden bei der Entwicklung von Inhalten verwendet, um Autoren zu helfen, denen es an Programmier- oder Skriptkenntnissen fehlt. Ein WCMS umfasst die angepasste Planung, Steuerung, Kontrolle und Ausführung aller unverzichtbaren Schritte zur Erstellung, Verwaltung und Verteilung von Inhalten über eine Webseite [Spö09, S. 25]. Da in dieser Bachelorarbeit die Verwaltung von Webseiten durch CMS im Fokus steht, werden die Begriffe Content Management System und Web Content Management System synonym verwendet. Ein WCMS hat jedoch einen engeren Fokus, insbesondere auf die Verwaltung von Inhalten für Internet, Intranet und Extranet [Spö09, S. 25]. Ein Enterprise Content Management System (ECMS) wird von Unternehmen genutzt, um Inhalte aus verschiedenen Unternehmensbereichen zu erfassen, bereitzustellen und zu archivieren [Böh14, S. 324] und dient der Verwaltung von Unternehmensinhalten, die nicht für die Öffentlichkeit bestimmt sind, wie interne Berichte oder Mitarbeiterdaten [Bar16, S. 7]. Digital Asset Management (DAM) wird genutzt, um umfangreiche digitale Assets wie Bilder, Audio und Video zu verwalten [Bar16, S. 7]. Records Management (RM) wiederum ist für die Verwaltung von Transaktionsinformationen und anderen Aufzeichnungen zuständig [Bar16, S. 7].

2.2.2 Vor- und Nachteile von CMS

CMS bieten eine Vielzahl von Vorteilen. Sie ermöglichen es, Inhalte ohne Programmierkenntnisse zu aktualisieren, auch von mehreren Redakteuren [Böh14, S. 326]. Sie sind einfach zu bedienen und eignen sich für nicht technisch-versierte Inhaltsersteller [Str22, S. 208]. Kurzfristige Aktualisierungen können problemlos vorgenommen werden, und klare Rechtezuweisungen gewährleisten eine geordnete Zusammenarbeit [Böh14, S. 326]. Die Trennung von Frontend und Backend bietet Datensicherheit sowie eine plattformunabhängige und sichere Administration im Browser. Durch Anbindung an eine Datenbank ermöglichen CMS eine Suchfunktion [Böh14, S. 326]. Die Trennung von Inhalt, Layout und Struktur erleichtert Gestaltungsänderungen und ermöglicht mehrsprachige Seiten [Böh14, S. 326]. CMS können durch zusätzliche Module und Plugins erweitert werden und bieten eine einfache und zeitliche Steuerung der Publikation [Böh14, S. 326]. Freigabe- und Sperrmechanismen ermöglichen die Kontrolle von Inhalten durch autorisierte Personen [Böh14, S. 326]. CMS bieten zudem geringe Kosten, Anpassungsmöglichkeiten und Upgrades sowie Flexibilität bei der Inhaltsnutzung [Str22, S. 208]. Auch kann Zeitersparnis erreicht werden, da die Bedienung bei guter Usability intuitiv erfolgen kann und es nach langen Pausen leichter ist, in das System zurückzufinden [Nix05, S. 51-52]. Außerdem bieten sie eine hohe Nutzervertrautheit, wenn sich CMS wie andere Systeme oder Applikationen verhalten und Konsistenz, wenn gleiche Aktionen gleiche Auswirkungen haben oder ein konsistentes Aussehen und Interface ausweisen [Nix05, S. 53]. Jedoch gibt es auch Nachteile. Der Einsatz großer Datenmengen erfordert umfassende strategische Planung [Str22, S. 208]. Auf Unternehmensebene erfordert die Verwendung detaillierte Schulungen, Investitionen in Hardware und regelmäßige Wartung [Str22, S. 208]. Weiterhin stellen CMS ein potenzielles Ziel für Cyberangriffe dar und sind Sicherheitsbedrohungen ausgesetzt [Str22, S. 208].

2.2.3 Was ein CMS nicht kann

In diesem Kapitel soll eine Abgrenzung dessen erfolgen, was CMS nicht können. Barker hat dazu Funktionen identifiziert, die allgemein als von CMS erfüllbar angesehen werden, tatsächlich aber nicht zutreffen. Diese sind in Abbildung 2.9 zusammengefasst dargestellt.

So ist ein CMS weder in der Lage, Inhalte zu erstellen oder zu generieren, noch kann es garantieren, dass die verwalteten Inhalte von Qualität sind. Vielmehr liegt es beim Redaktionsteam, die Inhalte zu produzieren und zu pflegen [Bar16, S. 12]. Während ein CMS bei der Umsetzung von Marketingplänen helfen kann, ist es nicht in der Lage, diese von Grund auf zu erstellen. Die Entwicklung, Umsetzung und Bewertung von Marketingplänen erfordern nach wie vor das Fachwissen qualifizierter Fachkräfte, wie Barker feststellt [Bar16, S. 13].

Ebenso kann ein CMS Inhalte strukturieren und automatisch formatieren, aber es gibt immer noch Raum für menschliche Fehler. Viele Systeme verfügen über Bearbeitungsoptionen, die dazu führen können, dass Inhalte unästhetisch oder inkonsistent formatiert werden [Bar16, S. 13]. So schreibt Barker: „Editors have never seen a button on an editing interface that they didn't want to press." [Bar16, S. 13]. Eine Möglichkeit, solche Formatierungsfehler zu minimieren, besteht darin, die Anzahl der Bearbeitungsoptionen im CMS zu begrenzen und die nötigsten zu behalten [Bar16, S. 13].

Es ist außerdem wichtig zu beachten, dass ein CMS nicht in der Lage ist, die Governance zu verwalten. Obwohl es ermöglicht, bestimmte Aktionen eines Benutzers einzuschränken, müssen diese im Voraus festgelegt werden [Bar16, S. 14]. Das CMS führt ausschließlich die Aufgaben aus, die ihm zugewiesen werden, aber Pläne müssen durch menschliche Interaktion und Beurteilung erstellt und in Berechtigungen und Zugriffsbeschränkungen umgewandelt werden, die das CMS durchsetzen kann [Bar16, S. 14].

Abbildung 2.9: Was ein CMS nicht kann,
Quelle: Eigener Entwurf

Barker hat dazu in seinem Buch eine Analogie zum Hausbau gezogen [Bar16, S. 14]: Das Haus ist eine Kombination aus Rohbaustoffen und Werkzeugen. Die menschliche Kraft hält dabei alles zusammen und erweckt es zum Leben. Holz und Nägel allein können kein Haus bauen - es ist der Mensch, der das Holz bearbeitet, die Nägel einschlägt und das Haus errichtet. Gleiches gilt beim Content Management. Die Inhalte sind Rohbaustoffe, während das CMS das Werkzeug ist, das zur Inhaltsverwaltung benötigt wird. Nur durch menschliche Kraft können das CMS zum Leben erweckt und Inhalte verwaltet werden. Ohne Inhalte ist das CMS nutzlos und ohne das CMS werden Inhalte nicht verwaltet. Schlussendlich ist der Mensch der Schlüssel zur Verwaltung der Inhalte und zum Erfolg beim Content Management. Das CMS ist nur ein Werkzeug, das dabei hilft.

Zusammenfassend kann ein CMS weder Marketingpläne oder Inhalte erstellen, noch ist es in der Lage Inhalte zu formatieren oder Governance bereitzustellen. Stattdessen ist es nur ein Werkzeug, das seine zugeteilten Aufgaben erledigt und der Mensch muss dieses Werkzeug einsetzen.

3 Traditionelle und Flat-File CMS

3.1 Definition und Funktionsweise von traditionellen CMS

In diesem Teil soll eine Antwort auf die Frage „Welche Merkmale und Arbeitsweisen kennzeichnen traditionelle CMS?“ [F2] gegeben werden.

Traditionelle Content Management Systeme wie WordPress sind darauf ausgelegt, einen Web-Auftritt zu erstellen und zu pflegen [Sch21b, S. 51]. Diese Systeme sind „monolithisch“ oder „gekoppelt“ ([Sch21b, S. 51]; [Bar16, S. 26]), was bedeutet, dass das Backend, also die Software, die für die Erstellung und Verwaltung von Inhalten verantwortlich ist, eng mit dem Frontend, also der Benutzeroberfläche der Webseite, und der Datenbank verbunden ist. Das bedeutet, dass sowohl die Erstellung als auch die Bereitstellung von Inhalten innerhalb desselben Systems erfolgen [Bar16, S. 26]. Wenn ein Benutzer eine Änderung an einer Seite vornimmt, wird diese Änderung im Backend des CMS gespeichert und das Frontend der Webseite wird automatisch aktualisiert.

Traditionelle CMS speichern Inhalte für eine strukturierte Datenhaltung in relationalen Datenbanken, worunter am häufigsten Datenbanken wie MySQL, MariaDB oder PostgreSQL gehören ([Spö09, S. 51]; [Sch21c, S. 51]). Die jeweils unterstützten Datenbanken hängen vom CMS ab und sind in den Systemvoraussetzungen zu finden [Sch21c, S. 51]. Die verwendeten Datenbanken bestehen aus Tabellen mit verschiedenen Strukturen von Daten, die vom Inhaltsaufbau anhängig sind. Jede Inhaltsart kann in der Datenbank eine eigene Tabelle haben [Spö09, S. 51].

3.2 Definition und Funktionsweise von Flat-File CMS

In diesem Teil soll eine Antwort auf die Frage „Was sind Flat-File CMS und wie ist deren Funktionsweise?“ [F3] gegeben werden.

Flat-File CMS sind vor allem bei kleineren Projekten vertreten, da sie sich schnell und einfach umsetzten lassen [Sch22c, S. 9]. Die Systeme werden komplett ohne Datenbank verwaltet und speichern Einstellungen und Inhalte auf dem Server in einer dateibasierten Struktur in „Flat-Files“, also „flachen“ Dateien ab ([All12]; [Sch22c, S. 9]).

Bei Text-Dateien hat sich in den vergangenen Jahren Markdown durchgesetzt, aber gelegentlich sind auch andere Formate zu finden [Sch22c, S. 9]. Bei Markdown handelt es sich um eine vereinfachte Text-zu-HTML-Sprache, die 2004 von John Gruber und Aaron Swartz entworfen wurde [Sch21a] und bereits vor der Konvertierung eine leicht zu lesende und zu schreibende Ausgangsform im „Nur-Text-Format“ bietet [Gru]. Dabei können Dokumente über spezielle Sonderzeichen formatiert und anschließend in gültiges HTML umgewandelt werden [Gru]. Markdown kann aus Readme-Dateien bekannt sein. Ein Beispiel zu einer solchen Datei und wie sie nach der Konvertierung aussieht, wurde in Abbildung 3.1 mit Dillinger [Dil] erstellt.

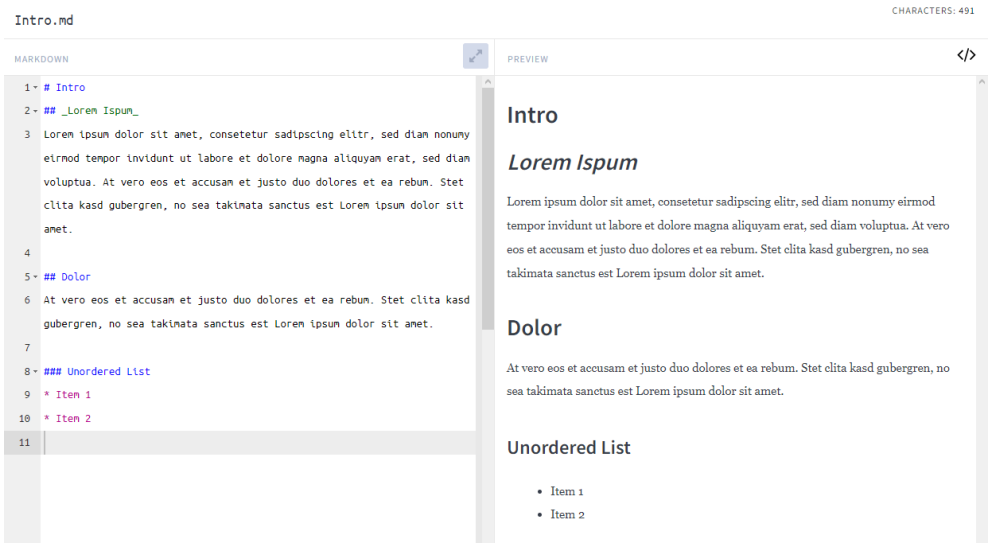


Abbildung 3.1: Markdown Beispiel,
Quelle: Erstellt mit Dillinger

Obwohl sie über keine Datenbank verfügen, gehören Flat-File-Systeme zu den CMS, da sie über eine Schnittstelle zum Erstellen und Verwalten sowie zum Generieren einer dynamischen Webseite beim Aufruf einer Uniform Resource Locator (URL) verfügen [Sch22c, S. 9].

3.3 Unterschiede der Speicherarten

Datenbankbasierte und dateibasierte (Flat-File) CMS sind zwei verschiedene Ansätze im Bereich des CMS, mit jeweils eigenen Vor- und Nachteilen. Die Tabelle in Anhang C fasst diese Punkte übersichtlich zusammen.

Datenbankbasierte CMS zeichnen sich durch ihre Skalierbarkeit und Effizienz bei der Verwaltung großer Datenmengen aus [Spö09, S. 54]. Sie dienen als Grundlage für die Verwaltung von Assets und ermöglichen eine flexible und offene Gestaltung sowie die Unterstützung einer sinnvollen Strukturierung und Dynamisierung [Spö09, S. 54].

Allerdings erfordert die Bearbeitung der Inhalte dieser CMS spezielle Programme und zusätzliche Administration für das Datenbanksystem im Backend [Spö09, S. 54]. Auch bei komplexen Abfragen und großen Datenmengen kann die Leistung datenbankbasierter CMS beeinträchtigt werden, da die Daten in den Datenbanktabellen lokalisiert und für die Weiterverarbeitung aufbereitet werden müssen, was zeitaufwendig sein kann [Spö09, S. 54]. Ebenso sind die Möglichkeiten der Volltextsuche dieser CMS eingeschränkt [Spö09, S. 54].

Flat-File CMS, auch dateibasierte CMS genannt, sind für ihre überlegene Leistung bekannt, da sie Seiten in Webbrowsern schnell laden können [Spö09, S. 54]. Diese CMS sind aufgrund ihres schlanken Designs, der einfachen Installation und der einfachen Verwaltung besonders für kleinere Projekte von Vorteil [Sch22c, S. 9]. Inhalte können als Markdown-Dateien heruntergeladen und auf verschiedene Arten genutzt werden [Sch22c, S. 10]. Darüber hinaus ist die Durchführung von Backups und Serverumzügen mit Dateikopien simpel [Sch22c, S. 10]. Ohne zusätzliche Optimierung bieten Flat-File CMS eine hohe Performance einen geringen Ressourcenverbrauch [Sch22c, S. 10]. Darüber hinaus wird durch das Fehlen einer Datenbank eine häufig von Hackern ausgenutzte Schwachstelle beseitigt, was zu einer erhöhten Sicherheit führt [Sch22c, S. 10]. Obwohl Flat-File CMS mehrere Vorteile bieten, bringen sie auch einige Nachteile mit sich. Autoren, die mit Markdown nicht vertraut sind, müssen sich möglicherweise mit der Sprache vertraut machen, bevor sie sie effizient nutzen können [Sch22c, S. 10].

Da Flat-File CMS einen Nischenmarkt besetzen, gibt es im Vergleich zu populäreren CMS-Plattformen wie WordPress nur wenige Themes und Plugins [Sch22c, S. 10]. Für individuelle Anpassungen kann die Hilfe eines Entwicklers erforderlich sein [Sch22c, S. 10]. Flat-File CMS eignen sich nicht für die Verwaltung umfangreicher, stark strukturierter Daten wie Produktinformationen [Sch22c, S. 10]. Ebenso kann es beim Umgang mit großen Inhaltsmengen zu Performance-Problemen kommen [Sch22c, S. 10]. Im E-Commerce steht für Flat-File CMS eine begrenzte Auswahl an Plugins und Erweiterungen zur Verfügung [Sch22c, S. 10].

Insgesamt bieten datenbankbasierte CMS Skalierbarkeit und effiziente Datenverwaltung, während dateibasierte CMS eine bessere Performance, einfache Installation und Verwaltung sowie eine erhöhte Sicherheit bieten. Die Wahl zwischen beiden Ansätzen hängt von den spezifischen Anforderungen des Projekts, der Größe der Inhalte und der Flexibilität bei individuellen Anpassungen ab.

3.4 Überblick über bekannte traditionelle CMS und Flat-File CMS

Wenn es um CMS geht, gibt es ein großes, unübersichtliches Angebot an Systemen [Sch22c, S. 6]. Es gibt kein System, das alle Anforderungen perfekt erfüllt [Ber16, S. 123]. Verschiedene Content Management Systeme sind für unterschiedliche Arten von Webseiten geeignet. Die Bandbreite der möglichen Endergebnisse ist riesig und kann laut Barker von einer kleinen Webseite für eine Zahnarztpraxis bis zu einer globalen Zeitung reichen, die mehrere Artikel pro Tag veröffentlicht [Bar16, S. 16]. Oftmals bleiben bei der Auswahl die eigenen Anforderungen an ein CMS unklar, und es wird sich an den Meinungen von Freunden, Kollegen oder Bekannten orientiert [Sch22c, S. 6]. Manchmal wird auch das System verwendet, das die eigene Web-Agentur bevorzugt, oder es wird sich auf die Suchergebnisse bezogen [Sch22c, S. 6]. Diese Vorgehensweise ist jedoch ungeeignet. Im ersten Schritt sollten die eigenen Anforderungen und das Einsatzszenario des CMS sowie Ziele für die Webseite klar und präzise formuliert werden können [Ber16, S. 123]. Außerdem ist es wichtig, einen umfassenden Marktüberblick zu haben, um Suchergebnisse und Empfehlungen sinnvoll sortieren zu können [Sch22c, S. 6]. Indem diese Schritte befolgt werden, wird die Wahrscheinlichkeit erhöht, das richtige CMS für die eigenen Bedürfnisse zu finden und eine erfolgreiche Webseite aufzubauen. Um einen Überblick über Flat-File CMS oder CMS allgemein bekommen zu können, wird für den Marktüberblick im Folgenden unter anderem CMSstash verwendet, welches eine web-basierte Fachpublikation zu CMS ist und ausführliche Markt-Übersichten zu verschiedenen CMS-Kategorien, sowie Reviews zu ausgewählten Systemen publiziert.

3.4.1 Traditionelle CMS

WordPress

WordPress (siehe Anhang B) wurde im Jahr 2003 von Matthew Mullenweg und Mike Little als eine Abspaltung von b2/cafelog geschaffen. Es wurde ursprünglich als Redaktionssystem für Blogs entworfen ([Sch22a]; [Sch22b]) und ist heute das am häufigsten genutzte CMS [W3T23f]. Die zentralen Aspekte konzentrieren sich auf Performance, Barrierefreiheit, Sicherheit und Nutzerfreundlichkeit [Worb]. Das System baut auf PHP und MySQL auf und unter ist unter der General Public License (GPL) lizenziert [Worb]. WordPress wirbt damit, dass nicht technisch-versierte Menschen das System ohne Codekenntnisse benutzen und es technisch-versierte wiederum anpassen können [Worb]. Es ist also flexibel und anpassbar und dennoch einfach zu bedienen. Bekannte Seiten, die WordPress nutzen, sind beispielsweise „Sony Music Entertainment“ [Son], der „Facebook Newsroom“ [Fac], „Flickr“ [Fli], „Nginx“ [NGI] oder auch „The Walt Disney Company“ [The] [Worb]. Das System kann selbst heruntergeladen und installiert oder über einen Hosting-Provider aufgesetzt werden [Worb]. Die derzeitige Version namens „Dolphy“

ist am 29. März 2023 erschienen. Die Versionsnamen bei WordPress sind besonders, da das Core-Team eine Vorliebe für Jazzmusik hat. Alle Hauptversionen werden gemäß WordPress „zu Ehren von Jazzmusikern benannt, die sie persönlich bewundern“ [Worb]. Mit dem Gutenberg-Editor (siehe Abbildung 3.2), der seit Version 5 im Dezember 2018 hinzugekommen ist, können einzelne Seiten durch Verwendung von individuellen, flexiblen Content-Blöcken frei zusammengestellt werden ([Sch22a]; [Sch22b]).

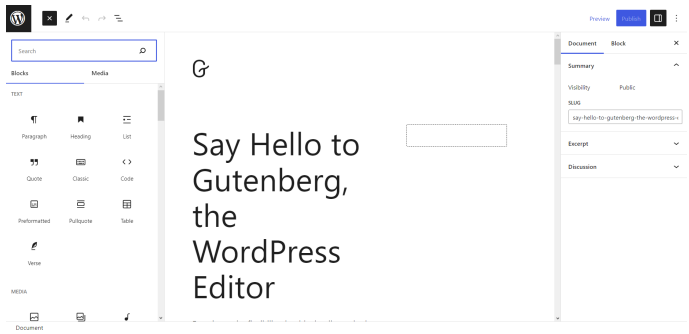


Abbildung 3.2: Gutenberg-Editor,
Quelle: WordPress Webseite

Drupal

Drupal (siehe Anhang B) ist seit 2001 ein Open-Source CMS, das im Jahr 2000 von Dries Buytaert entwickelt wurde [Dru]. Das System unterstützt unter anderem die Mehrsprachigkeit [Dru]. Das CMS kann mit verschiedenen Datenbanken wie MySQL, PostgreSQL und SQLite arbeiten und erfordert PHP ab Version 8.1 [Dru]. Es kann auf Webservern wie Apache, Nginx und Internet Information Services (IIS) betrieben werden [Dru]. Drupal verwendet die Template-Sprache Twig und ist unter der General Public License (GNU) lizenziert. Es basiert auf Prinzipien wie Zusammenarbeit, Globalismus und Innovation und zeichnet sich durch seine Flexibilität aus [Dru]. Das System ist laut Schürmanns ein nützliches und anspruchsvolles System für Einsteiger in die Programmierung [Sch22a].

Joomla

Joomla (siehe Anhang B) ist ein Open-Source CMS, das auf PHP und MySQL basiert und ursprünglich von Mambo abgezweigt wurde ([Sch22a]; [Joo]). Es verwendet ein Model-View-Controller-Webanwendungs-Framework und eignet sich für eine Vielzahl von Anwendungsfällen, darunter Unternehmenswebseiten, Portale, Webseiten für kleine Unternehmen, Online-Magazine, E-Commerce-Plattformen, Behördenwebseiten und persönliche Homepages [Joo]. Joomla hat seit seiner Gründung im Jahr 2005 viele Auszeichnungen erhalten, darunter den Best Open Source CMS Award 2021 und den Best Free CMS Award 2023 bei den CMS Critic People's Choice Awards [Joo]. Das System hat keinen speziellen Gründer, sondern wird von der durch Mambo entstandenen Community „Open Source Matters Inc.“ entwickelt [Sch22a]. Das System ist unter

der GNU lizenziert [Joo]. Im Jahr 2021 wurde Joomla! 4 veröffentlicht, eine stark überarbeitete Version des CMS, die nach knapp 10 Jahren Entwicklungsarbeit erschienen ist [Sch22a]. Joomla bietet eine solide Basis für die Erstellung verschiedener Arten von Webseiten und wird von einer engagierten Community unterstützt [Joo]. Das System ist vor allem für mittel-komplexe und komplexe Business-Seiten mit Systemflexibilität geeignet [Sch22a].

TYPO3
TYPO3 (siehe Anhang B) ist ein CMS, das seit Oktober 2012 offiziell als TYPO3 CMS bekannt ist [Typc]. Es wurde ursprünglich 1997 von Kasper Skårhøj entwickelt und veröffentlichte seine erste Open-Source-Version im Jahr 2001 [Sch22a]. Obwohl Kasper nicht mehr aktiv an der Entwicklung von TYPO3 beteiligt ist, trifft er sich weiterhin mit der Community und inspiriert sie bei einigen Veranstaltungen [Typc]. Das System ist ein leistungsstarkes CMS und eine geeignete Lösung für den Enterprise-Bereich und komplexe Corporate Websites [Sch22a].

Eine Übersicht aller vorgestellten CMS und zugehörigen spezifischen Daten in einer Tabelle aufgeführt, ist in Anhang D zu finden.

3.4.2 Flat-File CMS

Pico
Pico (siehe Anhang B) ist ein schlankes PHP-basiertes Mikro Flat-File CMS und wurde Mitte der 2000er-Jahre von einem schottischen Programmierer namens Gilbert Pellegrom entwickelt ([Spi14, S. 92]; [Pic]; [Sch22c, S. 21]). Es ist laut CMSstash eines der erfahrensten, noch aktiven Flat-File CMS [Sch22c, S. 21]. Im Gegensatz zu anderen CMS gibt es bei Pico kein Backend zur Administration oder Datenbanken, mit denen sich befasst werden müsste [Pic]. Das System bietet eine Sammlung von Plugins, funktioniert aber dennoch unabhängig von Frameworks und sonstigen Bibliotheken. Pico wird derzeit von Daniel Rudolf verwaltet [Sch22c, S. 21]. Pellegrom hat im Jahr 2020 „Saaze“ eingeführt, ein neues Flat-File CMS, das jedoch noch nicht viel Aufmerksamkeit erregt hat [Sch22c, S. 21].

Kirby
Kirby (siehe Anhang B) ist ein junges Flat-File CMS aus Deutschland [All12], das sich an Freelancer, Webagenturen und Unternehmen richtet, die eine optimierte Lösung für ihre Webseite suchen [Sch22c, S. 54]. Es wurde 2011 von Designer und Entwickler Bastian Allgeier entwickelt und ist PHP-basiert [Spi14, S. 92]. Kirby hat den Flat-File CMS-Markt revolutioniert und ist ein kommerzielles System, das von einem kleinen Team unter der Leitung des Gründers kontinuierlich verbessert wird [Sch22c, S. 14]. Das System hat den Slogan „Kirby is the CMS that adapts to you“ [Kirb], es kann

sich also an den Benutzer anpassen. Die Inhalte einer Webseite können im Browser, auf Smartphones, Tablets, Smartwatches oder auch auf dem Kühlschrank bereitgestellt werden – das heißt, das CMS kann je nach Belieben auch Headless verwendet werden [Kirb]. Die Verwaltung findet bei Kirby über File Transfer Protocol (FTP), WebDav oder die Dropbox statt. Außerdem ist es Nutzern möglich, die komplette Seite über Git oder ein anderes Version-Control-System zu verwalten [All12]. Inhaltsdateien verwenden Kirbytext, ein Markdown in einem von Allgeier geschriebenen Framework [Spi14, S. 92]. Beispielseiten, die mit Kirby erstellt wurden [Kirb] sind „Bauhaus / Richtig gut“ [Bau] oder „Fisherman's Friend“ [Fis]. Weitere Beispiele sind auf der Webseite von Kirby oder unter Kirbysites [Kirc] zu finden. Ebenfalls online ist die erste Webseite, die mit Kirby erstellt wurde. Das Kreativstudio „Deutsche und Japaner“ nutzt bis heute Kirby [Deu].

Kirby kann frei nach dem Motto „Try > Love > Buy“ ausprobiert werden. Dabei gibt es zwei Möglichkeiten, Kirby zu testen [Kirb]. Die erste Möglichkeit ist die der Online-Demo, bei der sich mit dem Panel vertraut gemacht und Kirby durch sechs zur Verfügung stehende Beispielprojekte kennengelernt werden kann. Diese Online-Demo wird nach einer Stunde gelöscht, allerdings besteht die Möglichkeit eine neue Demo zu starten, wenn mehr Zeit benötigt wird. Die zweite Möglichkeit ist die der Installation auf dem eigenen Computer oder einem Testserver [Kirb]. Diese Variante läuft nicht ab und sie kann so lange wie nötig getestet werden. Die zweite Variante bietet zwei Möglichkeiten: das Starterkit oder das Plainkit. Das Starterkit ist eine vollständig kommentierte Beispielseite, wenn das Interesse besteht, mehr über die Fähigkeiten von Kirby zu erfahren. Das Plainkit hingegen hat, wie der Name schon vermuten lässt, weder Templates noch Inhalte oder Designs. Es ist also mit dieser Variante möglich, von Grund auf zu starten.

Für Webprojekte kleiner oder mittlerer Größe eignet sich Kirby. Das CMS bietet sich auch für Projekte an, die Benutzerfreundlichkeit, Einfachheit und Vielseitigkeit erfordern [Sch22c, S. 61]. Mit der Veröffentlichung von Version 3 und der REST-API ist es nun möglich, Kirby in Verbindung mit Frontend-Frameworks wie Vue, React und Svelte sowie für Single Page Applications (SPAs) und Integration SSGs zu verwenden [Sch22c, S. 61]. Es ist jedoch erwähnenswert, dass Kirby nicht darauf ausgelegt ist, das Theme zu wechseln, da es tief in das System verwoben ist. Wenn das Design der Webseite häufig geändert werden können soll, ist die Verwendung eines anderen Systems möglicherweise besser geeignet [Sch22c, S. 61].

Statamic

Statamic (siehe Anhang B) ist ein modernes und hochanpassungsfähiges Enterprise Flat-File CMS, das vor allem für Freelancer und Agenturen gedacht ist ([Sch22c, S. 14-15]; [Stab]). Das System kann als Full-Stack, Headless, Flat-File oder SSG betrieben werden, ist laut CMSstash sehr feature-mächtig, aber trotzdem benutzerfreundlich ([Sch22c, S. 14-15]; [Stab]). Der Spiegel Verlag ist der bekannteste Kunde des Systems und ist

seit 2020 komplett auf Statamic gewechselt [Sch22c, S. 14-15]. Es wurde von Jack McDade, einem Designer und einer bekannten Persönlichkeit in der Laravel-Community, entwickelt und gemeinsam mit Mubashar Iqbal gegründet [Stab]. Statamic gilt als eines der am stärksten kollaborativen CMS-Projekte auf GitHub [Stab]. Der Name „Statamic“ ist eine Kombination der Wörter „static“ und „dynamic“ und repräsentiert die Fähigkeit der Plattform, statischen Dateien mit dynamischen Inhalten zu füllen [Stab]. Es wurde erstmals 2012 auf SlimPHP aufgebaut und hat seitdem verschiedene Updates durchlaufen, wobei die neueste Version im Mai 2023 als Laravel-Composer-Paket veröffentlicht wurde ([Sch22c, S. 15]; [Stab]). Im Jahr 2021 gewann Statamic den CMS Critic's Choice Award in der Kategorie „Flat File CMS“. Statamic ist aufgrund seiner Flexibilität und Usability insbesondere für kleinere Unternehmen und mittelständische Betriebe geeignet [Sch22c, S. 62]. Es ist auch bei E-Commerce-Projekten aufgrund ausgereifter Erweiterungen die erste Wahl [Sch22c, S. 72].

Grav

Grav (siehe Anhang B) ist ein bekanntes, schnelles, einfaches und flexibles Open-Source Flat-File CMS, das im Jahr 2014 unter der Massachusetts Institute of Technology (MIT)-Lizenz veröffentlicht wurde, und in PHP geschrieben ist ([Graa]; [Sch22c, S. 16]). Das Team hinter Grav besteht aus Lead-Entwickler Andy Miller sowie den Entwicklern Djamil Legato und Matias Griese [Grab]. Der Name Grav ist eine einfache Abkürzung des Wortes „Gravity“ [Grab]. Im Sommer 2014 gewann Grav an Popularität, da es die erste Open-Source-Option war, die voll funktionsfähig und mit kommerziellen Systemen wie Statamic oder Kirby vergleichbar war [Sch22c, S. 41]. Das System wurde 2016 von CMS Critic als „Bestes Open-Source-CMS“ ausgezeichnet und in den Jahren 2017, 2019, 2020 und 2021 als „Bestes Flat File CMS“ [Grab]. Die Entwickler-Community „GitHub-Grav“ des Flat-File CMS ist laut CMSstash zwar nicht so aktiv in der Entwicklung im Gegensatz zu anderen Systemen, aber dennoch sehr engagiert auf GitHub [Sch22c, S. 16]. Grav bietet eine umfangreiche Sammlung an Themes und Plugins, und seit der Einführung von Grav Premium 2020 kommen weitere kostenpflichtige Plugins und Themes hinzu [Sch22c, S. 16]. Eines der besonderen Merkmale von Grav ist laut CMSstash sein Seitenaufbau, der dem Bauen mit Legosteinen ähnelt - mehrere Unterseiten werden verwendet, um One-Pager zu erstellen [Sch22c, S. 16]. Grav eignet sich für kleine Webseiten aller Art, einschließlich, aber nicht beschränkt auf Landingpages, Portfolioseiten und Webseiten kleiner Unternehmen. Seine Grundgerüste machen es auch zu einer Alternative für verschiedene andere Zwecke wie Verzeichnisse, Dokumentation und kleine E-Commerce-Seiten [Sch22c, S. 47]. Das Flat-File CMS mag für Headless-Konzepte nicht so vorteilhaft sein, da es keine REST-API besitzt, aber Benutzern werden dennoch umfangreiche Konfigurationsmöglichkeiten und Anpassungsfähigkeiten geboten [Sch22c, S. 47]. Wenn also nach einem Open-Source CMS gesucht wird, das Flat-Files verwendet und eine Fülle von Funktionen und Flexibilität bietet, ist Grav eine Option.

Bludit

Bludit (siehe Anhang B) ist ein einfaches, schnelles und flexibles Open-Source Flat-File CMS, das von Djego Najar entwickelt wurde ([Bluc]; [Sch22c, S. 17, 35]). Bludit hat 2015 das alte XML-System „Nibbleblog“ ersetzt und hat von den Erfahrungen des vorherigen Systems profitiert [Sch22c, S. 17]. Die Installation ist einfach und leicht zu handhaben. Das System bietet auch eine PRO-Version, die spezielle, zusätzliche Plugins enthält [Blub] (siehe Anhang E.1). Das System ist für einfache Projekte, wie simple Blogs und Standard-Webseiten geeignet und bietet eine kleine Auswahl an Themes und Plugins [Sch22c, S. 18, 40]. Die API ermöglicht es Autoren, moderne frontend-getriebene Webseiten zu erstellen und mit einem HTML-Editor zu arbeiten [Sch22c, S. 18]. Seit 2015 wird Bludit ständig verbessert und eignet sich für simple Seiten [Sch22c, S. 18]. Benötigte Funktionen wie Kommentar-Systeme benötigen allerdings externe Dienste [Sch22c, S. 40]. Es ist nicht möglich, mehrere Sprachen zu verwenden, es sei denn, es werden zwei verschiedene Bludit-Installationen verwendet oder etwas kompliziertere Techniken mit verschiedenen Content-Ordnern angewendet [Sch22c, S. 40].

Typemill

Typemill (siehe Anhang B) ist ein Flat-File CMS, das speziell für Verlage und Autoren mit einem Veröffentlichungsprojekt entwickelt wurde [Typa], aber auch von Unternehmen für Dokumentationen verwendet wird [Sch22c, S. 18]. Es wurde 2017 von Sebastian Schürmanns veröffentlicht [Typa]. Mit Typemill's eigenem eBook-Plugin können Publisher eBooks im PDF- und EPUB-Format direkt von der Webseite erstellen und es kann sich auf Dokumentationen, Handbücher, Web-Novellen und ähnliche Veröffentlichungen konzentriert werden ([Typa]; [Sch22c, S. 18]). Außerdem ermöglicht der blockbasierte visuelle Editor die Erstellung von Markdown-Inhalten in einem WYSIWYG-Stil [Sch22c, S. 18]. Typemill verwendet das Mikro-Framework Slim-PHP und das Frontend-Framework Vue.js und bietet viele kostenlose Plugins, Themes und eBook-Layouts an [Sch22c, S. 18]. Das System eignet sich für kleine Unternehmen, die simples Print- und Web-Publishing für Dokumente oder Corporate Publishing suchen, sowie kleine Verlage und einzelne Autoren, die eine einfache, effektive Möglichkeit suchen, ihre Werke zu veröffentlichen und hilft ihnen dabei, eine Webseite zu erstellen und diese in buchähnliche Publikationen umzuwandeln ([Typa]; [Sch22c, S. 73]).

Automad

Automad (siehe Anhang B) ist ein Open-Source Flat-File CMS und eine Template-Engine [Aut]. Das System wurde 2013 eingeführt und war jahrelang inaktiv, bevor es im Juli 2018 überarbeitet wurde. Die neue Version ist nun moderner und organisierter [Sch22c, S. 19]. Das System bietet Autoren dank des Block-Editors ein großes Maß an kreativer Freiheit [Sch22c, S. 19]. Wenn der Wunsch besteht Automad zu unterstützen, gibt es dazu mehrere Möglichkeiten, ein Sponsor zu werden (siehe Anhang E.1). Automad eignet sich für Webseiten mit hohem Designanspruch und komplexen Designan-

forderungen, insbesondere im Kreativbereich [Sch22c, S. 20, 34]. Auch für freiberufliche Designer, Künstler und andere Kreativschaffende sowie kleinere Kulturinstitutionen oder Kreativunternehmen, die komplexe Designelemente integrieren möchten, ist Automad eine Option [Sch22c, S. 34].

Flextype

Flextype (siehe Anhang B) ist ein Open-Source Hybrid CMS, das die Freiheit eines Headless CMS mit der vollen Funktionalität eines traditionellen CMS bietet [Fle]. Es ist aus dem XML-basierten System Monstra hervorgegangen, hat sich im Laufe der Entwicklung dem Headless-Trend angeschlossen und wird heute als Hybrid CMS bezeichnet [Sch22c, S. 22]. Die Grundlage des Systems ist eine API, die Inhalte steuern und bereitstellen kann. Um die API-Nutzung zu erleichtern, wird dabei ein Token generiert, das externe Systeme autorisiert [Sch22c, S. 23]. Flextype bietet zwei Optionen zum Herunterladen der Software: eine normale Version, die die neuste Version des CMS enthält, und eine erweiterte Version, die Benutzern erlaubt, die Kernfunktionalitäten des Flat-File CMS durch Plugins und andere Pakete zu erweitern und anzupassen [Fle]. Des Weiteren werden zwei Starterkits angeboten, mit denen zum einen eine Dokumentationsseite (Starterkit namens Simpledoc) und zum anderen ein Blog oder eine persönliche Seite erstellt werden können (Starterkit namens Clean) [Fle]. Dennoch ist die Vielfalt der verfügbaren Themes begrenzt, da die Hauptzielgruppe Entwickler sind, die ein minimalistisches Headless CMS suchen, das keine Datenbank verwendet [Sch22c, S. 23].

HTMLy

HTMLy (siehe Anhang B) ist eine Flat-File Open-Source PHP-Blogging-Plattform, die keine Datenbank benötigt, um zu funktionieren [HTMb]. Danang Probo Sayekti führte diese Plattform im Jahr 2013 als Ersatz für WordPress ein. Die Entwicklung wurde jedoch von 2017 bis 2020 pausiert, seitdem wird HTMLy aber wieder weiterentwickelt [Sch22c, S. 21]. Mit HTMLy können Webseiten oder Blogs schnell, einfach, sicher und leistungsstark in Sekundenschnelle erstellt werden [HTMb]. Das System bietet alle notwendigen Funktionen zum Erstellen eines einfachen Blogs, einschließlich verschiedener Inhaltstypen für Beiträge, RSS-Feeds und die Möglichkeit, externe Kommentarsysteme zu integrieren [Sch22c, S. 22]. Obwohl sich die Themenauswahl auf wenige Variationen beschränkt, ist für jeden etwas dabei. Auch ohne Programmierkenntnisse lässt sich HTMLy schnell und einfach installieren und nutzen [Sch22c, S. 22]. Es ist eine Möglichkeit, ein erstes Konzept zu testen, das später im Erfolgsfall auf ein komplexeres System übertragen werden kann [Sch22c, S. 53].

Eine Übersicht aller vorgestellten Falt-File CMS mit ihren spezifischen Daten ist in Anhang D in einer Tabelle aufgeführt zu finden.

4 Analyse und Vergleich

In diesem Kapitel sollen verschiedene Kriterien, die durch den Marktüberblick herausgestochen sind, aufgezeigt und verglichen werden. Dazu werden die im Überblick vorgestellten Systeme genutzt, um Forschungsfrage [F4] „Welches sind mögliche Kriterien für einen Vergleich von traditionellen CMS und Flat-File CMS?“ beantworten zu können. Außerdem werden die letzten beiden Kriterien auf einer Wertungsskala bewertet, um am Ende die „besten“ CMS herausfinden zu können. Die Wertungsskala hat dabei die Punkte 0 bis 4 (siehe Anhang E.4). Bei Kriterien, bei denen keine Daten bekannt sind, bleibt die Wertung aus und ist mit null Punkten gleichzusetzen. Die Kriterien Kosten und Lizenzierung sowie Datenstruktur und Speicherarten werden nicht bewertet, da die Art der Lizenz oder die Speicherart von dem Anwendungsfall des CMS abhängen - So beispielsweise, ob mit dem System ein One-Pager oder eine große, mehrsprachige Webseite umgesetzt werden soll. Deshalb werden diese Kriterien ausgelassen.

4.1 Datenstruktur und Speicherarten

Ein wichtiger Aspekt bei der Wahl eines CMS ist die Datenstruktur des Systems, da diese bestimmt, wie Inhalte organisiert und gespeichert werden. Die Wahl der Datenstruktur und Speicherart beeinflusst Leistung, Flexibilität und Skalierbarkeit des Systems. So bieten traditionelle CMS oft umfangreichere Möglichkeiten zur Strukturierung und Verwaltung von Inhalten, während Flat-File CMS aufgrund ihrer einfachen Dateistruktur oft schneller und leichter zu handhaben sind. In diesem Abschnitt werden die verschiedenen Speicherarten und Datenstrukturen aufgezeigt und verglichen.

Wie bereits in Kapitel 3.1 und 3.2 erklärt, haben die CMS unterschiedliche Arten der Speicherung von Inhalten und Daten. Zu den CMS, die Inhalte in Datenbanken speichern, gehören Systeme wie WordPress, Drupal, Joomla! und TYPO3. Im Gegensatz dazu können Inhalte aber auch in Dateien gespeichert werden. Dies ist der Fall bei den Flat-File Systemen Kirby, Statamic, Pico, HTMLy, Bludit, Grav, Automad, Flextype und Typemill. Das CMS Kirby ist beispielsweise sogar kombinierbar mit Datenbanken. Das heißt, dass Inhalte anstelle von der Speicherung in Textdateien auch in einer Datenbank gespeichert oder Daten aus einer Datenbank für die Webseite exportiert werden können [Kirb]. Dazu hat Kirby eine eigene DB-Klasse, die mit MySQL oder SQLite funktioniert

[Kirb]. Auch im Punkt der Datenbankstruktur unterscheiden sich die Systeme wie bei der Speicherart. So haben die traditionellen CMS eine Datenbankstruktur mit festgelegten Tabellen und Relationen [Spö09, S. 51], Flat-File Systeme hingegen haben eine simple, dateibasierte Struktur ([All12]; [Sch22c, S. 9]).

4.2 Kosten und Lizenzierung

Ein weiterer wichtiger Aspekt bei der Auswahl eines CMS ist der Kostenfaktor und die Lizenzierung da diese Auswirkungen auf das Budget und die langfristige Wirtschaftlichkeit des Projekts hat. Der Kostenfaktor bezieht sich dabei auf die verschiedenen Ausgaben, die mit dem CMS verbunden sein können, die Lizenzierung hingegen betrifft die Art der Lizenz, unter der das CMS verfügbar ist. Hier werden die Kostenmodelle und Lizenzierungsbedingungen von traditionellen CMS und Flat-File CMS verglichen.

Alle im Marktüberblick vorgestellten Systeme sind mit Ausnahme von Statamic Open Source. Wenn es um den Kostenfaktor geht, sind alle Systeme, außer Kirby und Statamic frei verfügbar und nicht kommerziell. Kirby ist zwar für Studierende, Bildungsprojekte, Sozial-, Umwelt-, Wohltätigkeits- oder Non-Profit-Organisationen kostenlos erhältlich, eine Lizenz, bei der ein eigenes Hosting und eine eigene Domain mitzubringen sind, kostet aber einmalig 99 € [Kirb] (siehe Anhang E.1). Statamic hingegen bietet zwar eine kostenlose Variante an, diese darf aber nur für persönliche Zwecke, einen Freund oder ein Hobby verwendet werden, andernfalls ist mit einem Preis von $ 259 zu rechnen [Stab] (siehe Anhang E.1). Die anderen Systeme können durch Spenden unterstützt werden. So kann Automad beispielsweise durch einmalige oder monatliche Spenden [Aut] oder Bludit durch ein bis zehn Euro pro Monat unterstützt werden [Blub] (siehe Anhang E.1). Des Weiteren treten bei allen Systemen Kosten für Plugins, Themes oder Ähnlichem auf.

Die traditionellen CMS WordPress, Drupal, Joomla und TYPO3, sowie das Flat-File CMS HTMLy sind alle unter der GPL, oder auch GNU lizenziert – also der General Public License [Worb, Joo, Dru, Typc, HTMb]. Die aktuelle Version der Lizenz ist Version 3, auch als GNUv3 zu finden [GNU]. Diese Richtlinie ist eine Copyleft-Lizenz der Free Software Foundation. Bei Copyleft-Lizenzen müssen abgeleitete Werke nach Veränderungen, Erweiterungen oder Verbreitungen wieder unter der zuvor verwendeten Lizenz, also hier der GPL stehen (Freie Software) [GNU]. Da die Lizenz eine Freie Software ist, bietet sie Nutzern die vier Freiheiten die Software auszuführen (1), den Quellcode zu untersuchen und zu ändern (2), Kopien zu erstellen (3) und Modifikationen vorzunehmen (4) [GNU]. Die restlichen Flat-File CMS (Pico, Bludit, Grav, Automad, Flextype und Typemill) hingegen sind unter der MIT lizenziert [Pic, Bluc, Graa, Aut, Fle, Typa]. Dies ist eine freizügige Lizenz, die kein Copyleft besitzt. Das heißt, abgeleitete Werke

können unter anderen Lizenzen stehen. Diese Lizenz stammt vom Massachusetts Institute of Technology und ist eine Open-Source Lizenz, die die Verwendung, Modifikation, Vervielfältigung und Verbreitung von lizenzierten Inhalten erlaubt, solange der Urheberrechtsvermerk und der Haftungsausschluss beibehalten werden [MIT]. Eine Tabelle mit einer Übersicht der Systeme, deren Kostenmodelle und Lizenzen ist im Anhang E.1 zu finden.

4.3 Entwicklerfreundlichkeit und Support

Ein weiterer wichtiger Aspekt ist die Entwicklerfreundlichkeit, also die Einschätzung der Eignung des CMS für Entwickler und der Support, da die Verfügbarkeit des Supports und Hilfsressourcen wie Foren, Tutorials oder offiziellen Dokumentationen den Entwicklern helfen können. Hier werden die Dokumentationen und deren Umfang und die Unterstützung gängiger Entwicklungspraktiken wie der Versionierung von traditionellen CMS und Flat-File CMS verglichen. Außerdem werden die verschiedenen Punkte Webserver, Datenbanken, Dokumentation, Unterstützung und Multilanguage, sowie Formatierung (Formatting) anhand einer Wertungsskala bewertet. Bei den Flat-File Systemen ist jedoch zu beachten, dass nur bei Kirby und Grav etwas über Multilanguage bekannt ist und diese somit in dieser Kategorie mehr Punkte erreichen können. Des Weiteren ist bei Typemill nichts über die PHP-Version zu finden, weshalb hier keine Punkte vergeben werden können - gleiches ist bei Statamics Webservern der Fall. Der Punkt Technologie wird in die Bewertung nicht einbezogen, da alle Systeme auf PHP setzen und somit die gleiche Punktzahl erreichen würden. Gleiches ist der Fall bei der Versionierung. Außerdem wird auch die unterstützte PHP-Version nicht bewertet.

Alle Systeme sind PHP-CMS. Kirby, Typemill und Statamic nutzen außerdem Vue.js, und Statamic setzt Laravel als Framework ein, Typemill hingegen Slim [Kirb, Typa, Stab]. Für Konfigurationen verwenden Systeme, wie Grav oder Flextype YAML-Dateien [Graa, Fle]. Des Weiteren unterstützen alle Systeme gängige Webserver wie Apache und Nginx. Vereinzelt können auch LiteSpeed, Lighttpd oder Microsoft IIS verwendet werden. Dabei können PHP-Versionen ab 5.3 (HTMLy, Pico und Bludit) [HTMa, Blua, Pic], 7.4 (Statamic, Automad, WordPress) [Staa, Aut, Worb] oder 8.0/8.1/8.2 (TYPO3, Joomla!, Drupal, Flextype, Kirby) [Typb, Joo, Dru, Fle, Kirb] genutzt werden. Die traditionellen CMS unterstützen außerdem gängige Datenbanken wie etwa MySQL, MariaDB oder PostgreSQL. Beim Formatting setzten alle Flat-File Systeme auf Markdown-Dateien und vereinzelt auch auf HTML. Bei der Wertung erhalten deshalb alle Systeme 3 Punkte, da sie gängige Webserver (außer Statamic), Datenbanken und Formatting-Dateien unterstützen.

Bei dem Umfang, der Detailliertheit und der Übersichtlichkeit der Dokumentationen fällt auf, dass alle „großen“ CMS, wie WordPress, Joomla!, Drupal, TYPO3, Statamic, Grav oder Kirby und das kleine CMS Typemill sehr umfangreiche und detaillierte Dokumentationen (4 Punkte) und weniger verbreitete Systeme vergleichsweise kleinere Dokumentationen haben (2 Punkte für Flextype, Automad und HTMLy) oder zwar umfangreiche aber nicht so detaillierte (3 Punkte für Pico und Bludit).

Bei allen Flat-File CMS ist eine Versionskontrolle über Git möglich. Systeme wie Flextype oder Grav unterstützen aber auch andere Versionskontrollsysteme. Traditionelle CMS hingegen bieten vereinzelt Mechanismen zur Versionskontrolle. Statamic bietet sogar eine automatisierte Versionskontrolle mit Git. Dabei kann es Inhalte bei Änderung committen und pushen, Commits planen oder nicht Git-affine Nutzer können trotzdem Änderungen über das Control Panel committen und pushen [Staa].

Support wird bei allen Flat-File CMS über GitHub Issues gegeben. CMS wie Kirby, Statamic oder TYPO3 besitzen einen eigenen YouTube Kanal, auf dem Tutorials angeboten werden [Kira, Typc, Staa]. Die meisten Systeme haben außerdem einen Discord-Channel (Grav, Bludit, Kirby, Statamic), auf dem Fragen gestellt werden können oder besitzen ein eigenes Forum (Grav, Bludit, Automad). Systeme, wie Kirby, Statamic und TYPO3 erhalten wegen der großen Bandbreite an Unterstützungsmöglichkeiten 4 Punkte, alle anderen Systeme 3 Punkte, da diese Support über Foren oder Discord geben und Systeme wie Pico, HTMLy, Flextype und Typemill erhalten 2 Punkte, da diese nur über GitHub Issues Support anbieten.

Im Bereich der Erweiterbarkeit, kann Kirby beispielsweise mit Datenbanken, APIs oder Daten aus Excel-Tabellen kombiniert oder als Headless CMS in Kombination mit SSG oder einer mobilen App genutzt werden [Kirb]. Statamic, Automad und Flextype können ebenfalls als Headless CMS genutzt werden - Flextype sogar als Hybrid CMS [Stab, Aut, Fle]. Außerdem kann das traditionelle CMS Drupal als „Decoupled Drupal“ verwendet werden [Dru]. Systeme wie Kirby, Grav, Drupal, Joomla! Und TYPO3 bieten schließlich auch Multilanguage an (3 Punkte), ohne dies über Plugins wie bei WordPress hinzufügen zu müssen (1 Punkt).

Im Bereich der Entwicklerfreundlichkeit und des Supports treten besonders die Systeme TYPO3 und Kirby mit 17 Punkten hervor, dicht gefolgt von Grav, Drupal und Joomla mit 16 Punkten. Die niedrigste Punktzahl verzeichnen mit 10 Punkten HTMLy und Flextype.

Eine Tabelle mit einer Übersicht der Systeme, deren Dokumentationsumfang, Support und Weiterem, ist in Anhang E.2 sowie der Tabellen mit den Punkten der Wertungsskala in Anhang E.4 zu finden.

4.4 Zukünftige Entwicklung und Aktualisierung

Die zukünftige Entwicklung und Aktualisierung eines CMS ist ein wichtiger Faktor, da die kontinuierliche Entwicklung und Aktualisierung entscheidend ist, um mit den sich ständig ändernden Anforderungen und technologischen Entwicklungen Schritt halten zu können. In diesem Abschnitt werden die Aktualitäten der Systeme eingeschätzt und die Häufigkeit der Updates verglichen. Nachfolgend werden verschiedene Aspekte wie der letzte Release, seit wie vielen Jahren die Systeme auf dem Markt sind, die Aktualität und die Updatehäufigkeit anhand der Wertungsskala bewertet.

Die traditionellen CMS hatten alle von 2001 bis 2005 ihren ersten Release und sind somit um die 20 Jahre auf dem Markt[Worb, Joo, Dru, Typc] - deshalb erhalten all diese CMS 4 Punkte. Alle CMS zwischen 10 und 15 Jahren Marktzeit erhalten 3 Punkte, Systeme zwischen 5 und 9 Jahren erhalten 2 Punkte und jene, die unter einem Jahr verfügbar sind, 0 Punkte. Außerdem sind diese traditionellen CMS sehr aktuell, da deren letztes Update im März 2023 (WordPress) [Worb] oder im Mai (Drupal, Joomla!, TYPO3) war [Gitf, Dru, Typc] - wegen hoher Aktualität und ihrem letzten Release im Jahr 2023 erhalten hier auch alle 4 Punkte. Beim Thema der Updatehäufigkeit unterscheiden sich die Systeme jedoch. So kommen bei WordPress zwei bis drei Unterversionen im Jahr und größere Updates, wie neue Versionen circa alle vier bis fünf Jahre (siehe Anhang E.3). WordPress' aktuellste Version (V) ist derzeit V6.2 „Eric Dolphy“ vom 29.03.2023 [Worb]. Drupal veröffentlicht mehrere Unterversionen im Jahr und große Updates wie neue Versionen kommen unregelmäßig (siehe Anhang E.3). Die derzeitige Version von Drupal ist V10.0.9, die am 03.05.2023 veröffentlicht wurde [Dru]. Bei Joomla! erscheinen ebenfalls mehrere Unterversionen im Jahr und neue Versionen kommen unregelmäßig (siehe Anhang E.3). Joomlas derzeitig aktuellste Version vom 29.05.2023 ist V4.3.2 [Gitf]. Das CMS TYPO3 veröffentlicht ebenfalls mehrere Unterversionen im Jahr, aber große Updates wie neue Versionen kommen seit Version 6.0 alle ein bis zwei Jahre [Typc]. Des Weiteren bietet TYPO3 LTS Versionen an. Diese erscheinen circa ein Jahr nach Veröffentlichung der normalen Version (siehe Anhang E.3). Am 09.05.2023 ist bei dem System die aktuellste Version 12.4.1 herausgekommen. LTS steht für Long Term Support, was bedeutet, dass diese Version einen langfristigen Support erhält [Typc]. Es wird garantiert, dass sie drei Jahre lang unterstützt wird [Sch22a]. Im Bereich der Updatehäufigkeit erhalten WordPress, Joomla! und Drupal 3 Punkte, da diese Systeme große Updates unregelmäßig veröffentlichen oder in einer Zeitspanne von vier bis fünf Jahren. TYPO3 hingegen erhält 4 Punkte, da dieses System regelmäßig in einem Abstand von ein bis zwei Jahren neue Versionen veröffentlicht.

Bei den Flat-File Systemen hatten die ersten Systeme, wie Kirby, Statamic und Pico ihren ersten Release im Jahr 2012 [Gith, Kirb, Stab], Automad ein Jahr später [Gita] und HTMLy und Grav im Jahr 2014 [Gite, Gitd]. Bludit ging 2016 an den Start [Gitb],

ein Jahr später folgte Typemill [Gitj] und 2018 kam Flextype auf den Markt [Gitc]. Insgesamt sind die Systeme also bereits zwischen fünf und elf Jahren auf dem Markt (teilweise mit Pausen) (siehe Anhang E.3). Da Flat-File CMS noch nicht so lange auf dem Markt sind, wie traditionelle CMS wurde bei der Punktevergabe ab einer Zeit von 10 Jahren auf dem Markt eine Punktzahl von 4 Punkten verteilt, da die Systeme Erfahrung aufweisen können. Alle Systeme, die jünger als 10 Jahre sind, bekommen 3 Punkte und jene, die seit drei bis sechs Jahren verfügbar sind, bekommen 2 Punkte. Systeme, die weniger als ein Jahr auf dem Markt sind, erhalten 0 Punkte.

Auch im Bereich der Aktualität unterscheiden sich die Systeme. Kirby, Statamic und Grav sind mit letzten Updates von Anfang Juni 2023 sehr aktuell [Gitg, Gitd, Giti]. Andere Systeme wie HTMLy, Bludit, Automad oder Flextype sind, wegen eines letzten Updates im Jahr 2022, nicht so aktuell [Gite, Gitb, Gita, Gitc]. Pico hatte sein letztes Update im Jahr 2020 [Gith] und Typemill im Jahr 2018 [Gitj] (siehe Anhang E.3). Diese Aktualität spiegelt sich auch bei der Updatehäufigkeit der CMS wider. So kamen beispielsweise bei Typemill seit 2018 keine Updates mehr, es ist aber angeblich ein Alpha-Release von Version 2 für Juli 2023 geplant [Gitj]. Bei Flextype wurden seit 2022 keine Updates mehr veröffentlicht und die letzte Version ist V1.0.0 Alpha 3, die am 19.11.2022 herausgekommen ist [Gitc]. Auch große Updates, wie neue Unterversionen, kommen unregelmäßig (siehe Anhang E.3) [Gitc]. Bei Automad ist die Updatehäufigkeit ähnlich. Bei dem System sind ebenfalls seit 2022 keine Updates mehr herausgekommen und Unterversionen werden unregelmäßig veröffentlicht. Das CMS Grav hingegen hatte sein letztes Update am 01.06.2023 und ist somit aktuell [Gita]. Das System bekommt monatlich mehrere kleinere Updates, neue Unterversionen kommen aber unregelmäßig (siehe Anhang E.3) [Gita]. Bludit's Regelmäßigkeit der Updates ist wie bei Automad und Flextype, da bei dem System ebenfalls seit 2022 keine Updates kamen und neue Versionen unregelmäßig herauskommen [Gitb]. Das System hat am 07.09.2022 V3.14.1 veröffentlicht, Version 4.0 ist aber bereits seit November 2021 als Beta Pre-Release verfügbar (siehe Anhang E.3) [Gitb]. HTMLy hatte ebenfalls 2022 seinen letzten Release von V2.8.2. Seitdem kamen keine Updates mehr und neue Versionen erscheinen unregelmäßig (die letzte 2014) [Gite]. Außerdem hat das System von 2017 bis 2020 eine Pause eingelegt. Das bereits elf Jahre alte CMS Pico hat V2.1.4 am 29.08.2020 veröffentlicht, seitdem wurden aber ebenfalls keine Updates mehr veröffentlicht, bis auf einen Alpha Pre-Release für V3.0 im Dezember 2020 [Gith]. Auch bei dem System erscheinen neue Versionen unregelmäßig, bis zum Jahr 2020 sind diese aber alle zwei bis drei Jahre gelauncht worden [Gith]. Statamic und Kirby hingegen fallen, wie Grav, mit ihrer Updatehäufigkeit auf. So ist Statamic mit einem letzten Update für Version 4.6 am 07.06.2023 sehr aktuell [Giti]. Des Weiteren kommen kleine Update-Fixes fast wöchentlich heraus und große Updates wie neue Versionen kamen bisher in einem regelmäßigen Abstand von circa drei bis vier Jahren – Die letzte große Version erschien erst am 09.05.2023 (V4.0) (siehe Anhang E.3) [Giti]. Auch Kirby hatte sein letztes Update am 07.06.2023

(V3.9.5) [Gitg]. Bei dem System kommen kleine Update-Fixes fast monatlich, große Updates wie neue Unterversionen jedes halbe Jahr und neue Versionen alle zwei bis fünf Jahre. Version 3.0 kam im Februar 2019 heraus und Version 4.0 ist seit Mai bereits als Alpha-Release verfügbar, ab Juli soll die Version in die Beta-Phase übergehen und im späten Sommer 2023 schließlich veröffentlicht werden (siehe Anhang E.3) [Gitg]. Die Punktevergabe bei dem Punkt des letzten Releases wurde wie folgt festgelegt: Systeme, die ein letztes Update im Jahr 2023 hatten, bekommen 4 Punkte, da sie sehr aktuell sind, Systeme, deren letztes Update im Jahr 2022 war, erhalten 3 Punkte, 2 Punkte erhalten diejenigen mit einem Update im Jahr 2021 und einen Punkt gibt es für ein letztes Update im Jahr 2020. Gar keine Punkte bekommen alle Systeme, deren letztes Update mehr als 3 Jahre zurückliegt. Die Punkte der Aktualität sind mit denen des letzten Releases identisch.

Bei der Wertung der Updatehäufigkeit wurde darauf geachtet, ob regelmäßige Updates kommen und in welchem Abstand - So bekommen beispielsweise TYPO3, Kirby oder Statamic 4 Punkte, da sie wöchentliche oder monatliche Update-Fixes und regelmäßig große Updates wie neue Versionen oder Unterversionen bekommen. Drei Punkte hingegen wurde an die Systeme verteilt, die zwar mehrere kleine Updates im Jahr oder Monat aufweisen können und bei denen große Updates unregelmäßig kommen. Zwei Punkte hingegen bekommen all jene Systeme, die seit einem Jahr keine Updates mehr bekommen haben und bei denen ebenfalls (früher) unregelmäßig große Updates kamen, einen Punkt solche, bei denen seit zwei oder mehr Jahren keine Updates veröffentlicht wurden und gar keinen Punkt verzeichnen die Systeme, die seit fünf Jahren keine Updates mehr hatten.

Insgesamt stechen WordPress, Joomla!, Drupal, TYPO3, Kirby, Statamic und Grav mit einer Punktzahl von 14 bis 16 Punkten hervor. Gar nicht überzeugen hingegen kann Typemill mit einer Punktzahl von 2 Punkten.

Eine Tabelle mit einer Übersicht der Systeme, deren Entwicklung, Aktualisierung und Häufigkeit der Updates, ist in Anhang E.3, sowie der Punktevergabe in Anhang E.4 zu finden. Außerdem sind dort auch Tabellen der Versionen und Unterversionen der Systeme und wann diese jeweils veröffentlicht wurden zu finden.

4.5 Wertung

Insgesamt konnten Systeme wie Kirby, Grav, Drupal, Joomla! und TYPO3 mit einer sehr guten Punktzahl von mindestens 30 Punkten bei der Bewertung abschließen - auch Statamic und WordPress können dabei mit 27 und 29 Punkten mithalten. Im Gegensatz

dazu fällt Typemill leider auf, da es insgesamt nur 14 Punkte erzielen konnte und somit weniger als die Hälfte der insgesamt 36 erreichbaren Punkte bekommen hat.

Infolge der Punkte sollte deshalb bei Nutzung eines Flat-File CMS auf Systeme wie Kirby, Statamic oder Grav zurückgegriffen werden, da diese Systeme aktuell sind und Beratung in Form von Dokumentation und Support verfügbar ist. Im Bereich der traditionellen CMS hingegen ist es ganz gleich, welches System genutzt werden sollte, da alle Systeme ungefähr die gleichen Punkte verzeichnen können.

Gesamtwertung

CMS	Entwicklerfr.	Aktualisierung	Gesamtwertung
Kirby	17	16	33
Statamic	11	16	27
Pico	11	7	18
HTMLy	10	11	21
Bludit	12	11	23
Grav	16	14	30
Automad	11	12	23
Flextype	10	10	20
Typemill	12	2	14
WordPress	14	15	29
Drupal	16	15	31
Joomla!	16	15	31
TYPO3	17	16	33
max. erreichbare Punktzahl	20	16	36

Legende

Entwicklerfreundlichkeit	Aktualisierung	Gesamtwertung
> 16	> 13	> 29
11-15	11-13	25-29
06-10	10-12	20-24
< 06	07-09	15-19
	< 06	< 15

5 Implementierung eines Beispielszenarios

In diesem Kapitel soll ein Beispielszenario implementiert und umgesetzt werden. Dabei wurde sich für die Erstellung und Programmierung eines Post Types entschieden. In Abschnitt 5.1 werden die dazu ausgewählten Systeme vorgestellt und in Kapitel 5.2 die Installation und Konfiguration der Systeme beschrieben, erklärt und aufgezeigt. Schlussendlich wird in Abschnitt 5.3 beschrieben, wie das Beispielszenario des Post Types umgesetzt wurde.

5.1 Auswahl der Systeme

Um traditionelle und Flat-File CMS miteinander vergleichen zu können, wurde sich im Folgenden auf jeweils ein System begrenzt.

Dabei ist die Wahl bei den traditionellen CMS auf WordPress gefallen, da mit diesem System immer mehr Webseiten erstellt werden – So beispielsweise, wie schon in Kapitel 1.1 erwähnt, ein Drittel mehr Webseiten im Jahr 2023 gegenüber dem Jahr 2018 [W3T23f, Nie18]. Des Weiteren hat WordPress mit 64 % den höchsten Marktanteil unter den vorgestellten traditionellen CMS [W3T23e] und wird auch mit rund 43 % am meisten für Webseiten genutzt [W3T23f]. In Kapitel 4 hat WordPress zwar die niedrigste Punktzahl der traditionellen CMS erhalten, wegen der oben genannten Gründe wurde sich aber dennoch für die Umsetzung des Beispielszenarios mit diesem System entschieden.

Bei den Flat-File CMS wurde sich für Kirby entschieden, da es zwar im Gegensatz zu anderen Flat-File CMS wie Statamic oder Grav nur circa ein Drittel mal so häufig genutzt wird [W3T23b], die Nutzung des Systems selbst aber im vergangenen Jahr bereits um 31 % angestiegen ist [W3T23g]. Außerdem hat Kirby, wie schon in 1.1 erwähnt, eine treue Benutzerbasis, da nur circa 3 % von Kirby auf ein anderes CMS wie WordPress wechseln und umgekehrt [W3T23a]. Des Weiteren hat Kirby in Kapitel 4 die höchste Gesamtwertung im Bereich der Flat-File CMS erhalten.

5.2 Installation und Konfiguration der Systeme

Im folgenden Abschnitt werden die jeweilige Installation der Systeme WordPress und Kirby und alle benötigten Programmen beschrieben und weiterhin auch die Konfiguration zur erfolgreichen Umsetzung des Beispielszenarios. Für beide CMS wird dafür eine lokale Installation erstellt, da die Webseite nicht live gehen soll und nur zu Testzwecken gedacht ist.

5.2.1 Tools

Um WordPress erfolgreich herunterzuladen und installieren zu können, wird serverseitig PHP Version 7.4 oder höher benötigt, sowie MySQL V5.7 oder MariaDB V10.3 oder höher [Wora]. Des Weiteren wird ein unterstützter Webserver wie Apache oder Nginx sowie HTTPS-Support benötigt [Wora]. Lokal hingegen sind ein Texteditor zum Bearbeiten der Hauptkonfigurations-Datei, sowie FTP und ein Webbrowser notwendig [Wora]. Für Kirby ist serverseitig PHP V8.0, V8.1 oder V8.2 und ein unterstützter Webserver wie Apache oder Nginx erforderlich und lokal ebenfalls FTP und ein Webbrowser [Kirb].

Um all diese Anforderungen umsetzen zu können, wird ein LAMP-Stack benötigt. Tools, die solche LAMP-Umgebungen aufsetzen, sind XAMPP oder MAMP. In diesem Fall wurde sich für XAMPP entschieden. XAMPP ist „eine vollständig kostenlose, leicht zu installierende Apache-Distribution, die MySQL, PHP und Perl enthält“ [XAM]. Die aktuellste Version von XAMPP wird unter https://www.apachefriends.org/de/index.html heruntergeladen - in diesem Fall Version 8.2.4. Nach einem Klick auf die fertig installierte exe, öffnet sich der Setup Wizard (siehe Anhang F.1). Danach können die zu installierenden Komponenten ausgewählt werden. XAMPP enthält den Apache-Webserver und PHP, außerdem werden MySQL, FileZilla FTP Server, Perl und PhpMyAdmin als Komponenten mitinstalliert. Außerdem können Webalizer, Fake Sendmail, Tomcat und Mercury Mail Server mitinstalliert werden (siehe Anhang F.1.1). Danach wird der Ordner ausgewählt, in dem XAMPP installiert werden soll (siehe Anhang F.1). Anschließend ist es möglich, die Sprache zu wechseln (siehe Anhang F.1.1). Schlussendlich startet die Installation, nachdem zuvor ein „Ready to Install“-Fenster erscheint (siehe Anhang F.1.1, F.1). Nach erfolgreicher Installation öffnet sich das Control Panel (siehe Anhang F.1.1), in welchem Apache, MySQL und FileZilla gestartet werden.

Da bei WordPress außerdem eine Datenbank benötigt wird, wird auch diese aufgesetzt. Dazu wird PhpMyAdmin verwendet, das unter http://localhost/phpmyadmin/ aufgerufen werden kann. Hier wird eine neue Datenbank erstellt. Dazu wird in der linken Spalte auf „Neu“ geklickt (siehe Anhang F.1.2). Für die neue Datenbank wird in diesem Fall der Datenbankname „wordpressdb“ vergeben, außerdem wird die Kollation

„utf8_general_ci“ ausgewählt und letztlich auf „Anlegen“ geklickt (siehe Anhang F.1.2). Die erstellte Datenbank erscheint nun in der linken Spalte bei den anderen Datenbanken. Danach wird im Reiter „Rechte“ ein neuer Datenbanknutzer hinzugefügt, indem der Benutzername „db-admin“ gewählt, der Hostname im Dropdown auf lokal geändert und ein Passwort vergeben wird - hier „db-admin@wordpress“. Nach dem Klick auf „OK“ ist der Nutzer bei den anderen zu finden.

5.2.2 WordPress

Um WordPress installieren und aufsetzten zu können, wird die aktuellste Version 6.2.2 von https://wordpress.org/download/ heruntergeladen (siehe Anhang F.2). Danach werden die Dateien entpackt und in den „htdocs“-Ordner des XAMPP-Ordners kopiert (siehe Anhang F.2). Nun wird die `wp-config.php`-Datei geändert. Dazu wird diese von `wp-config-sample.php` zu `wp-config.php` umbenannt und in einem Texteditor geöffnet. Nun werden die Informationen der erstellten Datenbank wie in F.2 eingetragen. Außerdem werden „secret key values“ generiert und eingetragen, wie in F.2 zu sehen. Nach erfolgreicher Speicherung der `wp-config.php`-Datei kann nun WordPress installiert werden. Dazu wird im Browser die Seite http://localhost/wordpress/wp-admin/install.php aufgerufen (WordPress deshalb, weil der Ordner so heißt, in dem WordPress in diesem Beispielszenario liegt). Auf dieser Seite wird nun die Sprache ausgewählt, sowie im folgenden Fenster Daten zur Webseite, wie der Titel, der Benutzername, das Passwort und die E-Mail (siehe Anhang F.2). Außerdem wird ein Häkchen bei „Suchmaschinen davon abhalten, diese Webseite zu indexieren“ gesetzt. Schließlich erscheint ein „Installation erfolgreich“-Screen (siehe Anhang F.2) und es ist möglich, sich nun unter http://localhost/wordpress/wp-login.php anzumelden. Danach wird auf das Dashboard weitergeleitet (siehe Anhang F.2), wobei dieses per URL+/wp-admin erreichbar ist.

5.2.3 Kirby

Auf der Seite https://getkirby.com/try kann Kirby entweder als Starterkit oder Plainkit herunterladen werden (siehe Anhang F.3) – Für das Beispielszenario wird sich für das Plainkit entschieden. Nach dem Download werden die Dateien des ZIP-Ordners wieder entpackt und in den htdocs-Ordner von XAMPP kopiert. Der neu erstellte Ordner dafür heißt hier „kirby“. Nun wird Kirby installiert, indem die Seite http://localhost/kirby/ im Browser aufgerufen wird. Beim ersten Aufruf wird auf die Seite URL+/panel/installation weitergeleitet, auf der die E-Mail, ein Passwort und die Sprache eingegeben werden (siehe Anhang F.3). Nach der Installation wird das Panel automatisch geladen, welches unter der URL+/panel verfügbar (siehe Anhang F.3).

5.3 Erstellung und Verwaltung von Inhalten

Das Beispielszenario soll in diesem Abschnitt in Form eines Post Types umgesetzt werden. Dazu wird in den jeweiligen Unterkapiteln beschrieben, wie ein solcher Post Type programmiert und umgesetzt werden kann.

5.3.1 WordPress

Um einen Post Type in WordPress umsetzten zu können, ist es wichtig zu verstehen, was ein Post Type überhaupt ist. Ein Post Type ist ein Beitragstyp, der unterschiedlicher Art sein kann, und bezeichnet, wie Inhalte der Webseite dargestellt werden ([Worb]). WordPress verfügt über mehrere Standard Post Types, wie Beiträge (Posts), Seiten (Pages), Anhänge (Attachments) und Überarbeitungen (Revisions) ([Worb]), die gängigsten sind jedoch Seiten und Beiträge.

Ein Custom Post Type (CPT) hingegen ist ein benutzerdefinierter Beitragstyp, der es ermöglicht, neue Inhaltsarten (neben den standardmäßigen Beiträgen und Seiten) zu erstellen ([Worb]). Ein CPT kann individuell angepasst werden und hat seine eigenen Eigenschaften, Felder und Taxonomien. Dadurch ist es möglich, Post Types zu erstellen, die speziell auf die Bedürfnisse und Anforderungen der Webseite zugeschnitten sind.

Um einen CPT in WordPress selbst zu erstellen, wird als erstes der Ordner, in dem die WordPress-Installation liegt in VS Code geöffnet. Nun ist es möglich, den CPT in die `functions.php` des aktuellen Themes zu schreiben. Da beim Wechseln oder aktualisieren des Themes die `functions.php` geändert und der selbst hinzugefügte Code gelöscht wird, bietet es sich an, CPT durch ein selbst geschriebenes Plugin zu installieren. Dabei können Plugins aber leicht aktiviert und deaktiviert werden, was bei CPT nicht sinnvoll ist, da diese vielleicht auf Seiten genutzt werden und bei Deaktivierung des Plugins Fehlermeldungen verursachen. Stattdessen können sogenannte „must use Plugins“ erstellt werden.

Dazu wird in `/wp-content` ein neuer Ordner namens „mu-plugins“ angelegt (mu steht für must use). Alle sich in diesem Ordner befindenden php-Dateien werden automatisch von WordPress geladen und sind immer aktiviert. Sie können nur deaktiviert werden, indem die jeweilige Datei gelöscht wird. In diesen Ordner wird nun eine Datei namens „`wppt-book.php`“ erstellt. Um Informationen über das Plugin anzeigen zu können, wird an den Anfang der Datei der folgende Code geschrieben, der den Namen des Plugins definiert, sowie eine Beschreibung, den Autor und die Version:

Listing 5.1: MU-Plugin Informationen

```
/**
 * Plugin Name: Books
 * Description: This plugin will add a Custom Post Type for Books
 * Author: Chiara Funke
 * Version: 1
 */
```

Darunter folgt die Funktion `create_posttype()`:

Listing 5.2: MU-Plugin Funktion create_posttype()

```
function create_posttype() {
    register_post_type( 'wppt_book', array(
        'labels' => array(
          'name' => 'Bücher',
          'singular_name' => 'Buch',
          'add_new_item' => 'Add New Book',
          'edit_item' => 'Edit Book',
          'all_items' => 'All Books',
          'item_published' => 'Book published',
          'item_updated' => 'Book updated',
        ),
        'public' => true,
        'has_archive' => true,
        'rewrite' => array('slug' => 'books'),
        'menu_icon' => 'dashicons-book',
        'supports' => array(
            'title', 'thumbnail',
        ),
        'capability_type' => 'post',
        'show_ui' => true,
      )
    );
  }
add_action( 'init', 'create_posttype' );
add_theme_support('post-thumbnails');
```

Dieser Code registriert einen CPT namens 'wppt_book'. Dabei wird die Funktion `register_post_type()` aufgerufen, um diesen CPT zu registrieren. Sie erhält zwei Parameter: den Namen des CPT und ein Array mit den Einstellungen. In diesem gibt es verschiedene Schlüssel-Wert-Paare, um die Konfiguration des CPT anzugeben: '`labels`' definiert Beschriftungen für den Post Type, wie den Plural- und Singularnamen und verschiedene Aktionen wie das Hinzufügen und Bearbeiten von Büchern. '`public`' legt fest, dass der Post Type öffentlich zugänglich ist. '`has_archive`' ermöglicht das Archivieren der Bücher, sodass sie auf einer Archivseite angezeigt werden können. '`rewrite`'

definiert die Permalinks-Struktur. In diesem Fall wird der Slug „books“ verwendet, was bedeutet, dass die Bücher-Posts unter der URL+/books/ verfügbar sein werden. 'menu_icon' legt das Symbol fest, das in der Admin-Oberfläche für den Post Type angezeigt werden soll. In diesem Fall wird das Symbol 'dashicons-book' [1] verwendet. 'supports' gibt an, welche Funktionen für den Post Type unterstützt werden. In diesem Fall werden 'title' (Titel) und 'thumbnail' (Beitragsbild) unterstützt. 'capability_type' legt den Fähigkeitstyp für diesen Beitragstyp fest. In diesem Fall wird 'post' verwendet, was bedeutet, dass die Berechtigungen dem Standardbeitrag entsprechen. 'show_ui' legt schließlich fest, dass das Benutzerinterface für den Post-Typ angezeigt wird. Die letzte Zeile `add_action('init', 'create_posttype');` fügt eine Aktion hinzu, um die Funktion `create_posttype()` während des Initialisierungsvorgangs von WordPress aufzurufen. Dadurch wird der benutzerdefinierte Post-Typ registriert, wenn WordPress initialisiert wird.

Der Code `add_theme_support('post-thumbnails');` fügt das Feature der „Beitragsbilder“ (Post Thumbnails) zu dem Theme hinzu, welches ermöglicht, für jeden Beitrag ein ausgewähltes Bild festzulegen, das dann auf verschiedenen Teilen der Webseite angezeigt werden kann. Dazu wird die Funktion `add_theme_support()` aufgerufen, um das Feature „Beitragsbilder“ zu aktivieren, wobei der Parameter 'post-thumbnails' übergeben wird, um anzugeben, welches Feature hinzugefügt werden soll. Nach Speichern der Datei, ist das Plugin aktiviert und der erstellte CPT ist auf der Admin-Oberfläche in der linken Spalte zu finden (siehe G.2). Das MU-Plugin ist in dem Reiter Plugins unter Organisatorisches zu finden (siehe G.2).

Um dem CPT nun benutzerdefinierte Felder hinzufügen zu können, wird ein zweites Plugin erstellt, das kein mu-plugin ist, damit es aktiviert und deaktiviert werden kann, falls die Felder durch Plugins wie Advanced Custom Fields erstellt werden sollen. Dazu wird in `/wp-content/plugins` ein Ordner namens „books-plugin“ erstellt, in dem eine Datei namens „books-plugin-php“ angelegt wird. In dieser Datei folgt ein Kommentar, das den Namen des Plugins angibt, sowie die Beschreibung, den Autor und die Version (siehe G.1.2). Danach folgt die Funktion `custom_register_buch_meta()`, um eine benutzerdefinierte Metabox für ein Buch zu registrieren:

Listing 5.3: Plugin Metabox registrieren

```
1 function custom_register_buch_meta() {
2     add_meta_box( 'buch_details', 'Buchdetails', '
          custom_buch_details_callback', 'wppt_book', 'normal', 'default' )
          ;
3 }
4 add_action( 'add_meta_boxes', 'custom_register_buch_meta' );
```

1 Von https://developer.wordpress.org/resource/dashicons

Innerhalb dieser wird die Funktion `add_meta_box()` aufgerufen, welche für das Erstellen und Registrieren verantwortlich ist. Dazu werden verschiedene Parameter übergeben: '`buch_details`' ist der Name der Metabox, der später verwendet werden kann, um die Box anzusprechen oder anzupassen. '`Buchdetails`' ist der Titel der Metabox, welche in der Admin-Oberfläche angezeigt wird. '`custom_buch_details_callback`' ist der Name der Funktion, die den Inhalt der Metabox generiert und anzeigt. Diese Funktion erzeugt den HTML-Code für die Metabox-Felder. '`wppt_book`' ist der Post Type, für den die Box angezeigt werden soll. Danach folgt die Position der Box im Editor-Fenster, die hier als normal definiert ist. Zuletzt ist '`default`' die Priorität, welche die Reihenfolge bestimmt, in der die Metaboxen im Editor-Fenster angezeigt werden. Nachfolgend wird die Funktion `custom_register_buch_meta()` an die Aktion '`add_meta_boxes`' angehängt, wodurch die Funktion aufgerufen wird, wenn die Metabox registriert werden soll.

Danach wird die Callback-Funktion `custom_buch_details_callback()` definiert, die verwendet wird, um den Inhalt der zuvor registrierten benutzerdefinierten Metabox anzuzeigen und zu bearbeiten (siehe G.1.1). Dazu wird zuerst ein `<style>`-Block definiert, der CSS-Regeln enthält, die das Aussehen der Metabox-Elemente definieren. Daraufhin werden einige Metadaten des Beitrags abgerufen, die zuvor in der Datenbank gespeichert wurden. Dafür wird die Funktion `get_post_meta()` verwendet, um den Wert der Metadaten anhand von Schlüsseln wie '`autor`' zu erhalten. Für alle Metadaten folgt jeweils HTML-Code, um das entsprechende Eingabefeld anzuzeigen, wobei echo verwendet wird, um den HTML-Code auszugeben.

Um die Inhalte der Felder der Metabox in der Datenbank speichern zu können, wird die Funktion `custom_save_buch_meta()` verwendet (siehe G.1.1). Dazu wird die Funktion definiert und erhält den Parameter `$post_id`, der die ID des Beitrags enthält. Innerhalb der Funktion wird überprüft, ob bestimmte POST-Variablen vorhanden sind, die die Werte der Metafelder enthalten, die in `custom_buch_details_callback()` definiert wurden. Wenn eine Variable gesetzt ist, wird die Funktion `update_post_meta()` aufgerufen, um den Wert des Felds in der Datenbank zu aktualisieren. Dabei werden drei Parameter erwartet: die Post-ID, der Metadaten-Schlüssel und der neue Wert. Der neue Wert wird mit `sanitize_text_field()` bereinigt, um potenziell gefährlichen Code zu entfernen und den Text sicherzumachen. Die Funktion überprüft und aktualisiert die Metafelder separat für jede POST-Variablen, sodass nur die tatsächlich gesendeten Felder aktualisiert werden. Schließlich wird die Funktion `custom_save_buch_meta()` an die Aktion '`save_post_buch`' angehängt. Dadurch wird die Funktion aufgerufen, wenn ein Beitrag des Typs '`buch`' gespeichert wird. Danach werden auf der Admin-Oberfläche Bücher erstellt und die jeweiligen Felder ausgefüllt (siehe G.2). Eine Übersicht der erstellten Bücher ist ebenfalls in G.2 zu finden.

Um die Inhalte der benutzerdefinierten Felder auf der Webseite anzeigen zu können, wurde sich für Shortcodes [1] entschieden. Der Code für die Shortcodes wird im Folgenden anhand der Funktion für das Feld „Reihenname und Band“ erklärt. Zuerst wird eine Funktion `custom_buch_reihenname_und_band_shortcode()` definiert. Innerhalb dieser wird `get_post_meta()` verwendet, um den Wert des Metafelds 'reihenname_und_band' für den Beitrag abzurufen, wobei `get_the_ID()` die ID des aktuellen Beitrags zurückgibt. Wenn ein Wert für das Metafeld vorhanden ist, das heißt `$reihenname_und_band` wahr ist, wird der Wert zurückgegeben. Schlussendlich wird der Shortcode mit der Funktion `add_shortcode()` registriert, wobei der erste Parameter der Name des Shortcodes ist und der zweite Parameter der Name der Funktion, die den Shortcode behandelt und den Inhalt generiert.

Listing 5.4: Plugin Shortcode Reihenname und Band

```
1 function custom_buch_reihenname_und_band_shortcode() {
2     $reihenname_und_band = get_post_meta( get_the_ID(), '
          reihenname_und_band', true );
3     if ( $reihenname_und_band ) {
4         return $reihenname_und_band;
5     }
6 }
7 add_shortcode('buch-reihenname_und_band', '
      custom_buch_reihenname_und_band_shortcode');
```

Um den Inhalt der Metafelder nun auf der Webseite anzeigen zu können, wird zusätzlich zu dem CPT und den benutzerdefinierten Feldern ein Theme erstellt. Dazu wird in `/themes` ein neuer Ordner namens „books“ erstellt, in den folgende Dateien kommen: Eine `style.css`, welche die Haupt-Stylesheet-Datei ist und das Aussehen des Themes bestimmt; eine `index.php`, die zusammen mit der style.css die Basis für das Template bildet; eine `front-page.php`, welche die Template-Datei für die Startseite ist; `die header.php`, welche den Kopfbereich der Webseite anzeigt, inklusive des Menüs und dem Titel; eine `footer.php`, die den Fußbereich der Webseite anzeigt; eine page.php, welche die Datei für alle Seiten, also hier die Buchübersichtseite, ist, sowie die `single.php`, die die Detailseiten beschreibt. Des Weiteren ist es möglich ein Vorschaubild durch eine Datei namens `screenshot.png` zu setzen, welches in der Theme-Übersicht angezeigt wird. Nach Erstellen dieser Dateien kann das Theme auf der Admin-Oberfläche gefunden und aktiviert werden (siehe G.2). Danach wird zurück in VS Code gewechselt. Zuerst wird in die `front-page.php` ein HTML-Grundgerüst geschrieben und verschiedene Teile wie der Titel dynamisch erzeugt. Dazu werden innerhalb von PHP-Syntax[2] Funktionen aufgerufen, wie `wp_title()`, welche den

1 Ein Shortcode ermöglicht es, benutzerdefinierten Code in Beiträge, Seiten oder Widgets einzufügen und dynamische Inhalte zu generieren.
2 erkennbar an <?= und ?>

Titel der Webseite abruft und zurückgibt. Genauso werden auch Stylesheet-Dateien dynamisch eingebunden. Dazu wird `bloginfo('stylesheet_url')` genutzt, was auf die Stylesheet-Datei des Themes verweist und den jeweiligen Pfad generiert. Im Header-Bereich wird ein Abschnitt für das Logo hinzugefügt, der nur den Namen der Webseite ausgeben soll, was durch `bloginfo('name')` erreicht werden kann, sowie ein Bereich, der das Menü zeigen soll. Dazu wird ein Array definiert, das Argumente enthält, um das Hauptmenü anzuzeigen. Hier wird das Hauptmenü anhand des 'primary-menu'-Standortes und ohne Container angezeigt. Schließlich wird die Funktion `wp_nav_menu()` aufgerufen, die das Hauptmenü anhand der definierten Argumente des Arrays anzeigt. Der Main-Bereich enthält ausschließlich den Titel der jeweiligen Seite. Wie die Startseite aussieht, ist in G.2 zu sehen.

Da der Header und der Footer auf jeder Seite gleich aussehen und dieselben Dinge enthalten sollen, können diese Teile ausgelagert werden. Dazu wird in die `header.php` der Anfang des Grundgerüsts und der Header-Bereich ausgelagert (siehe G.1.2) und in die `footer.php` nur die schließenden Tags für body und html (siehe G.1.2). Um in Dateien diese ausgelagerten Teile nutzen zu können, werden die Funktionen `get_header()` und `get_footer()` genutzt. Die `front-page.php` enthält schließlich folgendes (dasselbe wird außerdem in die `index.php` kopiert (siehe G.1.2 und G.1.2)):

Listing 5.5: Theme front-page.php und index.php Inhalt

```
1 <?php get_header(); ?>
2 <main class="main">
3     <h1><?php the_title(); ?></h1>
4 </main>
5 <?php get_footer(); ?>
```

Um den Inhalt der Buchübersichtsseite anders aussehen zu lassen, wird dazu in der `page.php` eine benutzerdefinierte Abfrage (`WP_Query`) definiert, damit Beiträge vom Post Type „`wppt_book`“ abgerufen werden können. Dabei werden alle Beiträge in aufsteigender Reihenfolge nach dem Datum sortiert angezeigt.

Listing 5.6: Theme benutzerdefinierte Abfrage

```
1 $args = array(
2             'post_type' => 'wppt_book',
3             'posts_per_page' => -1,
4             'order' => 'ASC',
5             'orderby' => 'date',
6         );
7 $books_query = new WP_Query($args);
```

Danach wird durch `if $books_query->have_posts()` überprüft, ob die benutzerdefinierte Abfrage Beiträge zurückgibt (siehe G.1.2). Wenn dies nicht der Fall ist, wird `'Keine Bücher gefunden.' ausgegeben.` Andernfalls wird für jeden gefundenen Beitrag das Beitragsbild angezeigt, sowie der Titel als Bildunterschrift und ein Link zu der jeweiligen Buchunterseite. Nachdem die benutzerdefinierte Abfrage durchlaufen wurde, wird die Funktion `wp_reset_postdata()` aufgerufen, die die ursprünglichen Daten wieder herstellt und den Post-Daten-Zeiger zurücksetzt. Die jeweiligen Design-Einstellungen sind in der `style.css` (siehe G.1.2) zu finden. Außerdem wird in dieser Datei angegeben, welchen Namen das Theme haben soll (siehe G.1.2). Wie die Buchübersichtsseite aussieht, ist in G.2 zu sehen.

Um die jeweiligen Details zu den Büchern anzeigen zu können, wird das Aussehen und der Aufbau dieser in der `single.php` definiert. Dabei stehen die Anweisungen für das Aussehen inline am Anfang. Danach folgt der Main-Bereich, der nun die vorher definierten Shortcodes nutzt, um die Felder auszugeben. Der Reihenname, der Titel des Buches, der Untertitel und der Autor füllen dabei den ersten Teil der Seite, gefolgt von einem zweispaltigen Layout, das auf der linken Seite das Cover des Buches zeigt, also das Beitragsbild, und auf der rechten Seite Informationen wie den Klappentext, den Preis oder die EAN. Ein Beispiel für einen PHP-Syntax, der einen Shortcode für das Feld des Buchautors aufruft, ist `<?php echo do_shortcode("[buch-autor]") ?>`. Die ganze Datei ist in G.1.2 zu finden und die Buchdetailseite in G.2.

Schließlich ist es möglich, auf der Webseite durch das Menü durch die unterschiedlichen Seiten zu navigieren und Details zu den jeweiligen Büchern einzusehen. Außerdem können auf der Admin-Oberfläche Bücher erstellt werden, die alle sowohl dieselben Felder besitzen als auch auf der Webseite dieselbe Darstellung. Alle Dateien sind im Anhang G.1.2 zu finden.

5.3.2 Kirby

Um den Post Type in Kirby umsetzten zu können, muss als Erstes die Ordnerstruktur verstanden werden, die in vier grundlegende Teile gegliedert ist: „`/content`“, „`/site`“, „`/kirby`“ und „`/media`“.

Der Ordner „`/content`“ enthält die Seiten der Webseite, wobei jeder Unterordner eine Seite repräsentiert. Innerhalb der Unterordner liegt der Inhalt der Unterseite, der aus einer TXT-Datei, sowie Ressourcen wie Bilder oder anderen Dokumenten besteht. Die „`site.txt`“-Datei auf der obersten Ebene des „`/content`“-Ordners enthält allgemeine Informationen zur gesamten Webseite. Der „`/site`“-Ordner ist der Projektordner und enthält Templates, Konfigurationen, Plugins und Blueprints für das Panel. Templates sind eine Kombination aus HTML und der PHP API von Kirby, während das Panel

die Benutzeroberfläche darstellt und an jedes Projekt angepasst werden kann. Dafür werden Blueprints verwendet, die in YAML geschriebene Konfigurationsdateien sind, die das Panel-Layout, Felder und Einstellungen definieren. Dabei helfen fünf verschiedene sections, die Inhalte zu organisieren und zusätzliche Informationen anzuzeigen: die fields section, die files section, die info section (Hilfetext und Informationsboxen), die pages section und die stats section (Statistiken für die Webseite). Der „`/kirby`“-Ordner enthält die Kirby-App und sollte nicht verändert werden, da er regelmäßig bei neuen Versionen aktualisiert wird. Schließlich gibt es den „`/media`“-Ordner, der von Kirby verwaltet wird und öffentliche Bilder sowie Plugin- und Panel-Asset-Dateien enthält. Dieser Ordner und seine Dateien werden automatisch generiert.

Bei Kirby gibt es viele Dateien mit dem gleichen Dateinamen, aber unterschiedlichen Endungen, z. B. `book.txt` und `/site/blueprints/pages/book.yml` und `/site/templates/book.php`. Diese sind durch ihre Dateinamen miteinander verbunden. Der Inhalt in der TXT-Datei wird mit dem Template aus der PHP-Datei gerendert. Wenn die Seite im Panel besucht wird, kümmert sich der Blueprint, also die YAML-Datei, um die Anzeige der Formularfelder für die Felder in der Textdatei.

Um nun einen Post Type für Bücher erstellen zu können, wird als Erstes eine neue Seite angelegt, auf der später eine Übersicht der Bücher angezeigt werden kann. Dafür wird in dem Programm VS Code gearbeitet und erst später im Panel – auch wenn Seiten anlegen usw. bereits im Panel möglich ist. Außerdem werden zur Umsetzung YouTube-Videos von Kirbycast genutzt – Kirbys eigenem YouTube Kanal [Kira]. Eine Übersicht der Dateien ist in Anhang H.1.2, H.1.2 und H.1.2 zu finden.

Um eine neue Seite erstellen zu können, wird in `/content` ein neuer Ordner namens „books“ angelegt, in welchem eine „`books.txt`“ erstellt wird, in der Felder wie der Titel eingetragen werden, wodurch Kirby nicht den Namen des Ordners als Seitenname nimmt, sondern den eingetragenen Titel. Die neue Seite kann nun durch den Namen des Ordners, also `/books` erreicht werden. Da es mehrere Bücher geben soll, die auf der books-Seite angezeigt werden, werden Unterseiten benötigt. Dazu wird in `/content/books` ein neuer Ordner erstellt, mit dem Namen eines Buches, hier „`TheFinePrint`“. Dieser Ordner wird wieder mit einer txt-Datei gefüllt, die „`book.txt`“ heißt, da es sich hier um ein einzelnes Buch handelt. Diese Unterseite ist dann wieder erreichbar durch die Ordnernamen, also hier `/books/TheFinePrint`, da das Buch in dem Ordner books liegt. In der `book.txt`-Datei können nun ebenfalls Felder wie der Titel oder Text hinzugefügt werden. Dabei ist es simpel neue Felder zu erstellen, da der Name des Feldes vorn steht, gefolgt von einem Doppelpunkt und dem jeweiligen Inhalt (siehe folgendes Listing). Um Felder voneinander zu trennen, werden vier aufeinanderfolgende Bindestriche verwendet, worauf in der nächsten Zeile das nächste Feld kommt. Zu den jeweiligen Büchern können auch Bilder in die Buchordner hinzugefügt werden.

Listing 5.7: Buchfelder in Kirbytext-Syntax

```
1 Title: The Fine Print
2 ----
3 Text: This is awesome
```

Um den Namen der gesamten Seite ändern zu können, wird in `/content/site.txt` navigiert und wie bei den Büchern der gewünschte Titel der Webseite eingetragen.

Um später im Navigationsmenü eine geordnete Struktur erreichen zu können, ist es bei Kirby möglich vor den jeweiligen Ordnernamen Sortiernummern zu schreiben, worauf ein Unterstich folgt, wie „`1_TheFinePrint`“ oder „`1_home`“ „`2_books`“. Dadurch weiß Kirby automatisch, in welcher Reihenfolge die Seiten im Menü angezeigt werden sollen. Außerdem erkennt Kirby durch die vergebenen Nummern, welche Seiten im Menü auftauchen sollen und welche nicht. Um in Frontend die passende Ansicht erzielen zu können, wird die `default.php-Datei` in `/site/templates` benötigt. Diese ist wie der Name schon vermuten lässt die Standarddatei, die für jede Seite geladen wird. Dabei steht am Anfang die dynamische Codezeile `<h1><?= $page->title() ?></h1>` in der Datei. Diese ist eine Überschrift in der Größe h1 und der PHP-Syntax zeigt von der jeweiligen Seite (`$page`) den Titel (`title()`) als diese Überschrift an.

Um diesen Inhalt zu verbessern und zu einer „richtigen“ HTML-Seite machen zu können, wird ein Grundgerüst hinzugefügt (siehe folgendes Listing). Außerdem wird im head der Titel ebenfalls dynamisch erzeugt durch den PHP-Syntax, der sich den Titel der gesamten Webseite holt, die in `/content/site.txt` steht.

Listing 5.8: Grundgerüst

```
1 <html lang="en">
2 <head>
3     <meta charset="UTF-8">
4     <meta name="viewport" content="width=device-width,initial-scale=1.0">
5     <title><?= $site->title() ?></title>
6 </head>
7 <body>
8     <h1><?= $page->title() ?></h1>
9 </body>
10 </html>
```

Um das ganze Layout anzupassen, wird ein neuer Ordner auf oberster Ebene angelegt, der „`assets/css`“ heißt, in den eine `index.css` eingefügt wird. In diese Datei wird der Inhalt von https://github.com/getkirby/kirbycasts/blob/main/Basics/index.css kopiert (aus dem Tutorial von Kirbycasts übernommen [Kira]). Um diese verknüpfen zu können, hat Kirby den CSS-Helfer `<?= css('assets/css/index.css')?>`, der im head eingebunden wird. Darin kann nun eine CSS-Datei verlinkt werden.

Um nun ein Navigationsmenü hinzufügen zu können, wird ein Header erstellt. In diesen kommt auf der linken Seite der Seitenname, der auf die Home-Seite verlinkt, und auf der rechten Seite ein Menü, das automatisch alle Seiten anzeigen soll, die gelistet sind:

Listing 5.9: Dynamisches Menü

```
1 <nav class="menu">
2       <ul>
3         <?php foreach ($site->children()->listed() as $item): ?>
4         <li><a href="<?= $item->url()?>"><?= $item->title()?></a></li>
5         <?php endforeach ?>
6       </ul>
7 </nav>
```

Hier werden alle Kinder der Seite auf oberster Ebene in `/content` genommen (also alle Seiten), die gelistet sind. Für jede dieser Seiten wird ein li-Element erzeugt, bei dem der Titel angezeigt und die jeweilige Seite verlinkt wird. Durch „`listed()`“ werden alle Seiten gefiltert und nur die ausgegeben, die eine Nummer vor dem Ordnernamen stehen haben, alle anderen können durch „`unlisted()`“ erreicht werden – um alle Seiten zu bekommen, kann `listed()` usw. weggelassen werden.

Da nun die `default.php` sehr unübersichtlich und voll ist, können bestimmte Teile, die auf jeder Seite vorhanden sein sollen, als sogenannte „Snippets“[1] ausgelagert werden, um den Code übersichtlicher zu machen – wie Header oder Footer. Dazu werden in `site/snippets` eine `header.php` und eine `footer.php`-Datei erstellt, für die der Header-Teil in die `header.php` und der Footer-Teil in die `footer.php` ausgelagert wird. Um diese Snippets aus der `default.php`-Datei aufrufen zu können, muss in dieser Datei `<?php snippet('header') ?>` und `<?php snippet('footer') ?>` eingefügt werden, wobei in die Klammern von Snippet genau der Name geschrieben werden muss, den die Datei im Snippets-Ordner hat: Der Zwischenstand sieht dann wie in H.2 aus.

Da auf der Bücherübersicht der body anders sein soll, als auf den restlichen Seiten, wird der Inhalt der `default.php` kopiert und in eine neue Datei in `/site/templates` namens „`books.php`“ eingefügt. In dieser Datei kann nun das Layout für die Bücherübersicht verändert werden - wobei alle Bücher angezeigt werden sollen. Dafür wird der Code aus dem Header, der die Menüpunkte anzeigt, kopiert und in main der „`books.php`“ eingefügt. Da nun aber nicht alle gelisteten Seiten der Webseite angezeigt werden sollen, sondern alle Unterseiten der aktuellen Books-Seite wird hier `$site` durch `$page` ersetzt – Außerdem wird `$item` zu `$book` geändert:

1 kleinere Codeblöcke, die in verschiedenen Vorlagen wiederverwendet werden können

Listing 5.10: Dynamischer Code für alle Bücher

```
<ul class="books">
    <?php foreach ($page->children()->listed() as $book): ?>
    <li>
<a href="<?= $book->url() ?>">
     <?= $book->title() ?>
</a>
    </li>
    <?php endforeach ?>
</ul>
```

Um Bilder als Buchcover mit anzeigen zu können, muss zwischen den <a>-Tags folgendes eingefügt werden, wodurch das erste Bild in dem jeweiligen Buchordner genommen und auf 200 × 200 px zugeschnitten wird. Außerdem wird der Buchtitel verschoben und als Caption des Covers angezeigt (siehe folgendes Listing). Der Zwischenstand sieht dann wie in H.2 aus.

Listing 5.11: Buchübersichtseite Bücher anzeigen

```
<figure>
     <?= $book->image()->crop(200) ?>
     <figcaption>
          <?= $book->title() ?>
     </figcaption>
</figure>
```

Um auf den einzelnen Unterseiten, also den Buchdetailseiten, alle benötigten Informationen anzeigen zu können, wird eine neue „`book.php`“ Datei in `/site/templates` erstellt, in die der Inhalt der `default.php`-Datei kopiert wird. Danach wird der main-Teil erweitert, wie in H.1.1 zu sehen. Es wird in einem `article`-Element dynamisch, jeweils falls vorhanden, der Reihenname angezeigt, sowie der Titel, der Untertitel und der Autor. Darauf folgt auf der linken Seite das Buchcover und auf der rechten Seite, jeweils wieder falls vorhanden, Informationen wie der Klappentext, der Preis, das Erscheinungsdatum, der Verlag, die Seitenzahl, sowie die EAN und die Dateigröße. Diese Informationen werden alle in einer Definitionlist geladen und die Definitiontitle sollen fett angezeigt werden. Dazu wird in `assets/css` ein neuer templates-Ordner erstellt, in den eine neue css-Datei namens „`book.css`“ erstellt wird, mit dem Inhalt, der in H.1.2 zu sehen ist. Um diese CSS-Datei laden zu können, wird der Code `<?= css('@auto') ?>` in den header-Snippet eingefügt, der automatisch schaut, ob in `assets/css/templates` eine CSS-Datei vorhanden ist, die durch den Namen dem Elternelement zugeordnet werden kann – hier `book.css`. Das Ganze sieht dann im Frontend so aus, wie in H.2 zu sehen.

Um nun im Panel die passenden Felder angezeigt zu bekommen, wird ein neuer Blueprint erstellt, indem in `/site/blueprints/pages` eine neue Datei namens „`books.yml`“ erstellt wird. Diese hat als Titel Books und besitzt außerdem sogenannte sections. In diesem Fall nur eine Section namens books, weil auf der Bücherseite nur Unterseiten verlinkt werden können sollen. Um Unterseiten verlinken zu können, wird die books-section vom Typ pages sein. Im Panel sieht das Ganze dann so aus, wie in H.2 zu sehen.

Wenn die `books`-section in dem Blueprint um ein Feld namens `layout: cards` erweitert wird (siehe folgendes Listing), ist das Layout der Unterseiten in Karten wie in H.2 zu sehen. Um die Größe der Bilder im Panel anzupassen, wird ein `image`-Feld hinzugefügt, bei dem eine Größe angegeben werden kann, sowie `cover: true`, wodurch der schwarze Hintergrund verschwindet und das Bild den ganzen verfügbaren Platz einnimmt, wie in H.2 zu sehen.

Listing 5.12: Blueprint Books

```
1 title: Books
2
3 sections:
4   books:
5     type: pages
6     layout: cards
7     image:
8       ratio: 2/3
9       cover: true
```

Um die Ansicht der einzelnen Bücher im Panel ändern zu können, muss der Blueprint für ein Buch angepasst werden. Dazu wird eine neue Datei in `/site/blueprints/pages` namens „`book.yml`“ erstellt. Für diesen Blueprint soll nun im Panel dieselbe Ansicht vorliegen wie auch im Frontend, also werden zwei Spalten benötigt – auf der linken das Bild und auf der rechten Seite der Klappentext, der Preis usw. In H.2 ist die Ansicht vor Anpassung des Panels zu sehen.

Im ersten Schritt wird sich um die Felder gekümmert. Hier muss es einerseits ein Buchcover geben, das wie in der Buchübersicht als Karte angezeigt werden soll und vom Typ `files` sein muss, damit Coverbilder hinzugefügt werden können. Außerdem werden Felder für den Autor, den Untertitel und die Reihe benötigt. Diese Section wird hier `main` genannt und ist vom Typ `fields`, da Felder zum Eintragen der Daten erscheinen sollen. Kirby bietet verschiedene Arten von Feldern an – von Checkboxen und Radio-Buttons, über Date- und E-Mail-Felder, hin zu Slidern oder Tags. Bei dem Post Type der Bücher werden aber nur die Felder Text, Number, Date und Writer benötigt. Außerdem wird das Feld label mit angegeben, welches den Namen repräsentiert, der im Panel angezeigt wird, wie in H.2 zu sehen. Des Weiteren wird eine Info-Section erstellt,

die Felder enthalten soll, wie den Klappentext oder den Preis. Der Klappentext ist ein Writer-Feld, damit bei diesem auch Teile des Textes fett oder kursiv geschrieben, sowie Absätze gemacht werden können. Außerdem wird bei dem Preis ein Schritt angegeben, damit nicht nur volle Preise wie 15 €, sondern auch Preise wie 14,99 € angegeben werden kann und bei dem Erscheinungsdatum wird hinzugefügt, in welchem Format das Datum angezeigt werden soll. Da für die Feldnamen dieselben Namen genutzt wurden wie in `book.txt`, füllt Kirby im Panel die Felder automatisch mit dem Inhalt, der in der txt-Datei steht, wie in H.2 zu sehen. Wenn nun etwas im Panel in einem Textfeld geändert wird, wird dies automatisch in die jeweilige txt-Datei geschrieben.

Um das Layout anpassen zu können, gibt es das Feld `columns`, das auf der obersten Ebene liegt. Hier können Spalten eingerichtet werden, die durch einen Bindestrich definiert werden, worauf die Breite der Spalte folgt – Hier werden zwei Spalten benötigt, die jeweils die Hälfte der Seite einnehmen:

Listing 5.13: Columns

```
1 columns:
2   - width: 1/2
3   - width: 1/2
```

Danach wird der Inhalt der `sections` in die jeweilige `column` geschoben:

Listing 5.14: Column mit Section

```
1 - width: 1/2
2     sections:
3       bookcover:
4           type: files
5           layout: cards
6           image:
7             ratio: 2/3
8             cover: true
```

Das finale Layout einer Buchseite im Panel, sowie der zugehörige Blueprint sind in H.1.2 und H.2 zu sehen. Jetzt können im Panel durch den Hinzufügen-Button auf der „Bücher-Seite“ neue Bücher angelegt werden, die vom Typ book sind, indem als Vorlage Book gewählt wird (siehe H.2). So haben alle neu erstellten Bücher dieselben Felder und der Blueprint ist somit ein Post Type für Bücher.

6 Evaluation und Fazit

In diesem Kapitel werden die Ergebnisse evaluiert, alle Forschungsfragen beantwortet und ein Fazit gezogen.

6.1 Evaluation

In diesem Abschnitt werden die Ergebnisse der Arbeit zusammengefasst und die Vor- und Nachteile beider Content Management Systeme gegenübergestellt, um zu ermitteln, welches CMS im konkreten Anwendungsbeispiel besser geeignet gewesen ist.

6.1.1 Vergleich der Ergebnisse

In diesem Teil werden die Ergebnisse aus Kapitel 5 verglichen und aufgezeigt, um so in 6.1.2 besser auf die Vor- und Nachteile eingehen zu können.

Die Installation der benötigten Programme und Systeme hat insgesamt mindestens eine halbe Stunde in Anspruch genommen, da einige Schritte, wie das Finden und Umbenennen der wp-config-sample.php zu wp-config.php, zeitaufwendig waren. Die Installation war jedoch auf den jeweiligen Seiten gut dokumentiert.

Die Umsetzung des Beispielszenarios mit einem Custom Post Type in Kirby und WordPress hat deutliche Unterschiede gezeigt. Kirby besitzt eine umfangreiche und detaillierte Dokumentation sowie einen eigenen YouTube-Kanal, was die Erstellung und Verwendung eines benutzerdefinierten Post Types erleichtert hat. Im Gegensatz dazu konnte WordPress keine vergleichbare eigene Unterstützung bieten und erforderte das Ausprobieren verschiedener Ansätze, wie das Erstellten von Metaboxen, um benutzerdefinierte Felder erstellen zu können, sowie die Hilfe von Plugins. Die Verwendung von Plugins wie Advanced Custom Fields (ACF) und ACF Views hat zwar geholfen die Umsetzung zu beschleunigen, ist aber keine optimale Lösung für Entwickler, die ausschließlich mit Code im Backend arbeiten möchten und nicht unbedingt von der Admin-Oberfläche Gebrauch machen wollen. Denn durch diese Plugins wird ausschließlich auf der Admin-Oberfläche gearbeitet und nicht direkt mit Code, was für manche Entwickler suboptimal sein kann.

Schlussendlich wurde in WordPress ein Mittelweg zwischen den Metaboxen und den externen Plugins gefunden, indem die in einem Umsetzungsversuch zuvor erstellte Metabox mit Feldern wieder eingebaut und mit Shortcodes verknüpft wurde, um die Inhalte im Frontend anzuzeigen. Zusätzlich wurde ein eigenes Theme erstellt, da keine ausreichenden Anleitungen für die direkte Einbindung der Metabox-Felder im Frontend vorhanden waren.

Insgesamt war die Erstellung in Kirby deutlich einfacher, schneller und besser dokumentiert als in WordPress, da bei WordPress verschiedene Ansätze ausprobiert wurden. Dennoch haben beide Systeme nach einer Einarbeitungsphase gute und passende Ergebnisse erzielt. Die Verwendung derselben Seiten-Layouts und Stylesheet-Dateien in beiden Systemen hat zudem zusätzliche Arbeit erspart.

6.1.2 Vor- und Nachteile der Systeme

Kirby zeichnet sich durch eine detaillierte und umfangreiche Dokumentation aus, die Entwicklern eine Unterstützung bei der Erstellung von Webseiten und Blueprints bietet. Die bereitgestellten Videos ergänzen diese Dokumentation und ermöglichen es, anhand von praktischen Beispielen das gewünschte Szenario umzusetzen. Die Einfachheit bei der Erstellung von Blueprints ist ein weiterer Vorteil von Kirby, da Entwickler dadurch schnell und effizient benutzerdefinierte Inhalte für ihre Seiten erstellen können. Die kostenlose Verfügbarkeit für die lokale Installation ist besonders hilfreich, um das System ausgiebig zu testen und einen umfassenden Einblick in seine Funktionen zu erhalten. Für die Installation ist zudem keine Datenbank erforderlich, was den Prozess weniger zeitaufwendig gestaltet und den Einstieg in das System erleichtert. Die Arbeit mit Kirby erfordert grundlegende HTML- und CSS-Kenntnisse, was für Entwickler, die sich in diesen Bereichen bereits auskennen, einen reibungslosen Arbeitsablauf ermöglicht. Obwohl Kirby auch PHP-Elemente enthält, werden die benötigten Snippets und Befehle in der umfassenden Dokumentation ausführlich erklärt. Dies macht es auch für Entwickler ohne umfangreiche PHP-Kenntnisse zugänglich.

WordPress ist ebenfalls mit einer guten Dokumentation ausgestattet, die Entwicklern Unterstützung bei der Umsetzung bietet. Es gibt auch eine Vielzahl von Videos und Tutorials, die von verschiedenen Quellen, wie YouTube bereitgestellt werden und als Ressourcen zur Lösung spezifischer Herausforderungen dienen können, auch wenn für manche Lösungen mehrere unterschiedliche Videos zur Rate gezogen werden müssen. Ein großer Vorteil von WordPress gegenüber Kirby besteht darin, dass es eine breite Palette von Plugins gibt, die für Nicht-Entwickler eine effiziente Möglichkeit bieten, komplexe Funktionen und benutzerdefinierte Inhalte zu implementieren. So können es Plugins wie Advanced Custom Fields (ACF) und ACF Views ermöglichen, Felder einfach zu erstellen und sie über Shortcodes auf der Webseite anzuzeigen. Es gibt auch umfassende Lösungen

wie die GREYD.SUITE, die CPT-Erstellung, WYSIWYG-Editor und Theme in einem Paket vereinen, was die Entwicklungszeit erheblich verkürzen kann. Die kostenlose lokale Installation von WordPress ermöglicht ähnlich wie bei Kirby, das System in einer Testumgebung zu evaluieren und sich mit seinen Funktionen vertraut zu machen. Eine Herausforderung besteht in der Initialisierungsphase der WordPress-Installation. Vor der 5-Minuten-Installation von WordPress ist es notwendig, eine Datenbank einzurichten und andere vorbereitende Schritte zu durchlaufen, was zusätzliche Zeit in Anspruch nehmen kann.

Insgesamt hängt die Wahl zwischen Kirby und WordPress von den individuellen Anforderungen und dem Umfang des Projekts ab, sowie dem Kenntnisstand des Entwicklers und dem Aspekt, ob externe Plugins genutzt werden sollen oder nicht. Je nachdem eignet sich das eine System besser oder schlechter. In diesem Fall war jedoch dank der Umsetzung eines kleinen Beispiels einer Buchübersicht das System Kirby ausreichend, da lediglich die Top drei Bücher eingepflegt wurden. Wenn die Seite jedoch erweitert wird und an die hundert Bücher eingepflegt und angezeigt werden sollen, ist WordPress zu bevorzugen, da die verschiedenen Bücher so geordnet auf der Admin-Oberfläche angezeigt werden können und auf der linken Seite alle beisammen gefunden werden können. Bei Kirby hingegen würde für jedes Buch ein eigener Ordner kreiert werden, was die Ansicht im Code unübersichtlich machen würde. Bei Kirby können zwar auch alle Bücher geordnet auf dem Panel dargestellt werden, die Speicherung in separaten Ordnern jedoch ist unübersichtlich und könnte im späteren Verlauf zu Zeit- und Performance Problemen führen. Deshalb sollte die Wahl der Art des CMS, wie bereits erwähnt, von dem Umfang des Projekts anhängen.

6.2 Fazit

Content Management Systeme haben eine bedeutende Rolle in der heutigen Webentwicklung eingenommen, indem sie es Nutzern ermöglichen, Webseiten zu erstellen und zu verwalten, ohne umfassende technische Kenntnisse zu besitzen. Sie beeinflussen unseren Alltag, wenn regelmäßig aktualisierende Nachrichtenseiten oder andere Webseiten besucht werden. Während traditionelle CMS auf Datenbanken zur Speicherung von Inhalten und Konfigurationseinstellungen angewiesen sind, hat sich in den vergangenen Jahren das Flat-File CMS entwickelt, das Inhalte und Konfigurationen in simplen Textdateien verwaltet. In dieser Bachelorarbeit wurden die beiden CMS-Ansätze umfassend verglichen und untersucht.

Um sich dem Thema Content Management System und dessen Komponenten anzunähern, wurden zu Beginn in dieser Arbeit die grundlegenden Begriffe Content, Content Management, Content Management System und Content-Lifecycle definiert und in

Verbindung gebracht, um ein Grundverständnis bilden zu können. Content ist ein Begriff, der die Erstellung und Verwendung von Inhalten für spezifische Zwecke durch Informationssysteme zur effektiven Kommunikation mit dem Publikum umfasst, während Content Management darauf abzielt, Inhalte unabhängig von Format und Typ zu verarbeiten und zu verwalten, indem es den Content-Lifecycle unterstützt. Content Management Systeme dienen als Software, um digitale Inhalte auf Webseiten zu erstellen, zu bearbeiten, zu verwalten und zu veröffentlichen. CMS automatisieren den Content-Lifecycle und ermöglichen die Zusammenarbeit mehrerer Benutzer, sind jedoch begrenzt in ihren Fähigkeiten zur Erstellung von Marketingplänen, Formatierung von Inhalten und Bereitstellung von Governance. Auf Basis der Literaturrecherche wurden außerdem traditionelle und Flat-File CMS betrachtet, sowie die Unterschiede ihrer Speicherarten. Bei traditionellen CMS sind Backend, Frontend und Datenbank eng miteinander verbunden. Die Inhalte werden innerhalb desselben Systems erstellt, gespeichert und Änderungen automatisch aktualisiert. Flat-File CMS hingegen speichern Inhalte in dateibasierten „Flat-Files“, wobei häufig Markdown zum Einsatz kommt. Trotz Fehlen einer Datenbank verfügen sie über eine Schnittstelle zum Erstellen, Verwalten und Generieren einer dynamischen Webseite. Durch einen Überblick über bekannte CMS wurden im praktischen Teil verschiedene Systeme anhand unterschiedlicher Kriterien, wie Kosten, Lizenzierung, Entwicklerfreundlichkeit und Aktualisierung verglichen und bewertet. Daraufhin wurden die Systeme Kirby und WordPress anhand eines Beispielszenarios eines Custom Post Types umgesetzt und verglichen, wobei dieser durch Bücher repräsentiert wurde. Durch die Umsetzung wurden Ergebnisse, sowie Vor- und Nachteile der beiden Systeme herausgefiltert.

Die Ergebnisse zeigen, dass keine der beiden CMS-Arten eindeutig überlegen ist, sondern die Wahl des geeigneten Systems von mehreren Faktoren abhängt, wie dem spezifischen Einsatzszenario, der Datenmenge und dem Umfang des Projekts. Für einfache Webseiten wie One-Pager, Blogs und Magazine sowie kleine bis mittelgroße Web-Projekte empfiehlt sich die Nutzung beider Arten. Anspruchsvollere Business-Seiten können von der Verwendung von Joomla! profitieren, während Drupal und TYPO3 sich im Enterprise-Bereich bewährt haben. Flat-File CMS wie Kirby eignen sich außerdem für vielfältige Zwecke, wie E-Commerce-Projekte, kreative Webseiten ohne Entwickler, informationsorientierte Webseiten und kleine Headless-CMS-Anwendungen. Des Weiteren sollten bei großen Datenmengen traditionelle CMS oder Flat-File Systeme, die eine Datenbankverbindung ermöglichen, genutzt werden, um die Performance nicht zu beeinträchtigen. Die Untersuchung ergab, dass traditionelle CMS aufgrund ihrer etablierten Marktpräsenz und kontinuierlichen Aktualisierungen als zuverlässige Lösungen gelten. Einige der untersuchten Flat-File CMS weisen ebenfalls eine angemessene Marktpräsenz, sowie Aktualität auf, während andere weniger verbreitet und aktuell sind. Anhand der verglichenen Kriterien wurden die CMS Kirby, Grav, Drupal, Joomla, Typo3, Statamic und WordPress als die „besten“ identifiziert. Hinsichtlich der Umsetzung in den CMS Kirby und

WordPress fiel auf, dass die Installation des traditionellen Systems WordPress aufgrund der Einrichtung der Datenbank mehr Zeit in Anspruch genommen hat. Dennoch wiesen beide eine gute Dokumentation auf und ermöglichten eine erfolgreiche Umsetzung aus der Perspektive eines Entwicklers. Abschließend zeigt die Untersuchung, dass die Wahl zwischen traditionellen CMS und Flat-File CMS von den spezifischen Anforderungen des jeweiligen Projekts abhängt. Eine sorgfältige Abwägung der genannten Faktoren ist daher entscheidend für die Auswahl des optimalen CMS für eine gegebene Situation.

Die zugrundeliegenden Forschungsragen [F1] „Welche Komponenten bilden die Grundlage von CMS und wie sind sie definiert?“, [F2] „Welche Merkmale und Arbeitsweisen kennzeichnen traditionelle CMS?“, [F3] „Was sind Flat-File CMS und wie ist deren Funktionsweise?“ und [F4] „Welches sind mögliche Kriterien für einen Vergleich von traditionellen CMS und Flat-File CMS?“ wurden anhand der Literaturrecherche, sowie der vorgestellten Systeme beantwortet.

Die Analyse der Komponenten, die die Grundlage von CMS bilden, hat gezeigt, dass Content, Content Management, der Content-Lifecycle und Content Management Systeme selbst die zentralen Bausteine für die Struktur und Funktionsweise von CMS darstellen. Content umfasst sämtliche digitale Inhalte, die innerhalb des Systems verwaltet werden. Content Management beschreibt die Prozesse, die für die Erstellung, Organisation, Bearbeitung und Veröffentlichung von Inhalten verantwortlich sind. Der Content-Lifecycle befasst sich mit dem Lebenszyklus der Inhalte, von der Erstellung bis zur Archivierung oder Löschung. Content Management Systeme stellen die Software-Plattformen dar, die all diese Komponenten zusammenführen und die Verwaltung von Inhalten ermöglichen.

Die Untersuchung der Merkmale und Arbeitsweisen traditioneller CMS ergab, dass sie in der Regel auf relationalen Datenbanken basieren, um Inhalte zu speichern und zu verwalten. Sie bieten umfangreiche Funktionen für die Erstellung und Bearbeitung von Inhalten sowie Benutzer- und Zugriffsrechte. Diese CMS-Art ist besonders für große Webseiten und Unternehmensanwendungen geeignet, da sie eine umfassende Verwaltung und ein komplexes Publishing ermöglicht.

Die Funktionsweise von Flat-File CMS wurde eingehend analysiert und zeigt, dass diese Systeme einen alternativen Ansatz zur Speicherung von Inhalten verfolgen. Hierbei werden Inhalte nicht in zentralen Datenbanken, sondern in einfachen Dateien wie Markdown oder Textdateien gespeichert. Flat-File CMS sind häufig leichtgewichtiger und weisen geringere Ladezeiten auf, was sie für kleinere Webseiten und Blogs attraktiv macht. Sie bieten eine einfache Strukturierung der Inhalte und sind oft flexibel in der Handhabung, können jedoch bei sehr umfangreichen Webseiten an ihre Grenzen stoßen.

Die Identifizierung möglicher Kriterien für den Vergleich von traditionellen CMS und Flat-File CMS hat dazu beigetragen, objektive Bewertungsmethoden für beide Ansätze zu entwickeln. Die analysierten Kriterien umfassen Datenstruktur und Speicherarten, die Unterschiede zwischen zentralen Datenbanken und dateibasierten Ansätzen aufzeigen. Des Weiteren wurden Kosten und Lizenzierung untersucht, wobei diese vom jeweiligen Anwendungsfall abhängen und daher nicht direkt verglichen werden konnten. Die Entwicklerfreundlichkeit und der Support wurden als wichtige Faktoren betrachtet, die die Dokumentationen, Unterstützung gängiger Entwicklungspraktiken und andere Aspekte umfassen. Schließlich wurden die zukünftige Entwicklung und Aktualisierung als relevanter Kriterien identifiziert, da sich Technologien und Anforderungen im Laufe der Zeit ändern und ein nachhaltiges CMS langfristig von Vorteil ist.

Obwohl diese Arbeit einen gründlichen Vergleich zwischen traditionellen CMS und Flat-File CMS vorgenommen hat, eröffnet das Thema weiterhin interessante Forschungsbereiche. Ein Forschungsbereich könnte sich auf die Performance-Aspekte und mögliche Probleme bei der Verarbeitung großer Datenmengen konzentrieren, was dazu beitragen könnte, das Verständnis über die Leistungsfähigkeit beider CMS-Typen in Bezug auf umfangreiche Datenbestände zu erweitern. Die Erweiterbarkeit der Systeme im Hinblick auf Plugin- und Theme-Auswahl sowie ihre jeweiligen Anpassungsfähigkeiten sind weitere spannende Forschungsrichtungen. Eine umfassende Analyse dieser Aspekte würde Aufschluss darüber geben, welche CMS-Typen sich besser für individuelle Anpassungen und Erweiterungen eignen und wie flexibel sie auf unterschiedliche Anforderungen reagieren können. Darüber hinaus stellt die Sicherheit ein interessantes Thema dar, das weitere Untersuchungen erfordert. Die Tatsache, dass Flat-File CMS keine SQL-Injections zulassen, während traditionelle CMS zusätzliche Sicherheitsmaßnahmen benötigen, könnte eine vertiefte Analyse verdienen. Eine solche Untersuchung könnte dazu beitragen, die Sicherheitsstärken und -schwächen beider CMS-Typen zu identifizieren und potenzielle Sicherheitsrisiken zu minimieren. Insgesamt ermöglichen diese zukünftigen Forschungsbereiche, das Verständnis für die Vor- und Nachteile, sowie Stärken und Schwächen von traditionellen CMS und Flat-File CMS weiter zu vertiefen. Die gewonnenen Erkenntnisse könnten die Auswahl und Implementierung der geeigneten CMS-Lösungen für spezifische Anwendungsszenarien noch präziser und fundierter gestalten.

Literaturverzeichnis

[All12] Allgeier, Bastian: Kirby: CMS ohne Datenbank (2012), URL https://t3n.de/magazin/kirby-cms-ohne-datenbank-231172/ (10.04.2023)

[Aut] Automad: Automad Webseite, URL https://automad.org/ (13.06.2023)

[Bar16] Barker, Deane: *Web content management: Systems, features, and best practices*, O'Reilly, Beijing and Boston, first edition Aufl. (2016)

[Bau] Bauhaus: Bauhaus / Richtig gut, URL https://richtiggut.bauhaus.info/

[Ber16] Berger, Daniel, Schürmann, Tim, Violka, Karsten: Marktübersicht Content-Management-Systeme: Acht beliebte Content-Management-Systeme im Vergleich (2016): S. 122–131

[Blua] Bludit: Bludit Dokumentation, URL https://docs.bludit.com/ (13.06.2023)

[Blub] Bludit: Bludit PRO, URL https://pro.bludit.com/ (13.06.2023)

[Bluc] Bludit: Bludit Webseite, URL https://www.bludit.com/de/ (13.06.2023)

[Böh14] Böhringer, Bühler P. Schlaich P. Sinner D., J.: *Content-Management-System: Kompendium der Mediengestaltung*, X.media.press, Springer Vieweg, Berlin, 6 Aufl. (2014)

[Boi05] Boiko, Bob: *Content management bible*, Wiley, Indianapolis, Ind., 2. ed. Aufl. (2005), URL http://www.loc.gov/catdir/enhancements/fy0616/2004019371-b.html

[Bus23] BuschfeuerDesign: Warum wir standardmäßig Kirby einsetzen statt Wordpress (2023), URL https://buschfeuerdesign.de/blog/warum-wir-standardmaessig-kirby-einsetzen-statt-wordpress (13.04.2023)

[Deu] Deutsche und Japaner: Webseite, URL https://deutscheundjapaner.com/

[Deu98] Deutsch, Markus: *Electronic Commerce: Zwischenbetriebliche Geschäftsprozesse und neue Marktzugänge realisieren*, Springer eBook Collection Computer Science and Engineering, Vieweg+Teubner Verlag, Wiesbaden and s.l. (1998)

[Dil] Dillinger: Dillinger Webseite, URL https://dillinger.io/

[Dru] Drupal: Drupal Webseite, URL https://www.drupal.org/(13.06.2023)

[Fac] Facebook Newsroom: Webseite, URL https://about.fb.com/news/

[Fis] Fisherman's Friend: Webseite, URL https://fishermansfriend.com/de-de/

[Fle] Flextype: Flextype Webseite, URL https://awilum.github.io/flextype/(13.06.2023)

[Fli] Flickr: Webseite, URL https://blog.flickr.net

[Gita] GitHub: Automad, URL https://github.com/marcantondahmen/automad(14.06.2023)

[Gitb] GitHub: Bludit, URL https://github.com/bludit/bludit(14.06.2023)

[Gitc] GitHub: Flextype, URL https://github.com/flextype/flextype

[Gitd] GitHub: Grav, URL https://github.com/getgrav/grav(13.06.2023)

[Gite] GitHub: HTMLy, URL https://github.com/danpros/htmly(13.06.2023)

[Gitf] GitHub: Joomla, URL https://github.com/joomla/joomla-cms(13.06.2023)

[Gitg] GitHub: Kirby, URL https://github.com/getkirby/kirby/(13.06.2023)

[Gith] GitHub: Pico, URL https://github.com/picocms/Pico(13.06.2023)

[Giti] GitHub: Statamic, URL https://github.com/statamic/cms(13.06.2023)

[Gitj] GitHub: Typemill, URL https://github.com/getgrav/grav(13.06.2023)

[GNU] GNU: GNU General Public License, URL https://www.gnu.org/licenses/(06.06.2023)

[Graa] Grav: Grav Dokumentation, URL https://learn.getgrav.org(13.06.2023)

[Grab] Grav: Grav Webseite, URL https://getgrav.org/(13.06.2023)

[Gru] Gruber, John: Daring Fireball: Markdown, URL https://daringfireball.net/projects/markdown/(23.05.2023)

[HTMa] HTMLy: HTMLy Dokumentation, URL https://docs.htmly.com/(13.06.2023)

[HTMb] HTMLy: HTMLy Webseite, URL https://www.htmly.com/(13.06.2023)

[Jab02] Jablonski, Stefan und Meiler, Christian: Web-Content-Managementsysteme. *Informatik-Spektrum* (2002), Bd. 25(2): S. 101–119

[Joo] Joomla: Joomla Webseite, URL https://www.joomla.org/(13.06.2023)

[Kira] Kirby: Kirby Casts, URL https://www.youtube.com/kirbycasts(13.06.2023)

[Kirb] Kirby: Kirby Webseite, URL https://getkirby.com/(13.06.2023)

[Kirc] Kirby: Kirbysites, URL https://www.kirbysites.com/(13.06.2023)

[Kou] Kouba, Tomáš: Why we're switching from WordPress to Kirby CMS. URL https://www.netmagnet.cz/en/blog/why-were-switching-from-wordpress-to-kirby-cms/(13.04.2023)

[Loh01] Lohr, Jürgen und Deppe, Andreas: *Der CMS-Guide: Content Management-Systeme: Erfolgsfaktoren, Geschäftsmodelle, Produktübersicht*, XBusiness Computing, Vieweg+Teubner Verlag, Wiesbaden (2001)

[Mic16] Michael, Fabian: Why I Switched from WordPress to Kirby (2016), URL https://fabianmichael.de/blog/from-wordpress-to-kirby(13.04.2023)

[MIT] MIT: MIT License, URL https://opensource.org/license/mit/(06.06.2023)

[NGI] NGINX: Webseite, URL https://www.nginx.com/

[Nie18] Nier, H.: Wordpress ist am beliebtesten (2018), URL https://de.statista.com/infografik/12683/meistgenutzte-content-management-systemen-fuer-webseiten-weltweit/(02.05.2023)

[Nix05] Nix, Markus et al.: *Web Content Management: CMS verstehen und auswählen*, Software & Support Verlag GmbH, Frankfurt, 1 Aufl. (2005)

[Pic] Pico: Pico Webseite, URL https://picocms.org/(13.06.2023)

[Sch21a] Schürmanns, Sebastian: Markdown Einführung und Syntax (2021), URL https://cmsstash.de/website-praxis/markdown-fur-webseiten(23.05.2023)

[Sch21b] Schürmanns, Sebastian: Marktübersicht Headless CMS: Wie ein Headless CMS funktioniert (2021), URL https://cmsstash.de/empfehlungen/headless-cms(11.05.2023)

[Sch21c] Schürmanns, Sebastian: Wie ein CMS funktioniert (2021), URL https://cmsstash.de/content-management-system/wie-ein-cms-funktioniert(11.05.2023)

[Sch22a] Schürmanns, Sebastian: CMS Vergleich mit Testberichten (2022), URL https://cmsstash.de/cms-reviews(31.05.2023)

[Sch22b] Schürmanns, Sebastian: Content Management System: Diese CMS-Trends sind relevant (2022), URL https://cmsstash.de/content-manag

ement-system(31.05.2023)

[Sch22c] Schürmanns, Sebastian: Flat-File-CMS: CMS-Report (2022)

[Son] Sony Music Entertainment: Webseite, URL https://www.sonymusic.com/

[Spi14] Spiller, Jerry: Decoder Ring: The CMS is Flat. *Against the Grain* (2014), Bd. 26(2)

[Spö09] Spörrer, Stefan: *Content Management Systeme: Begriffsstruktur und Praxisbeispiel*, Springer eBooks Business and Economics, Springer Gabler, Wiesbaden (2009)

[Staa] Statamic: Statamic Dokumentation, URL https://statamic.dev/(13.06.2023)

[Stab] Statamic: Statamic Webseite, URL https://statamic.com/(13.06.2023)

[Str22] Strekalova, Y.A., Bouakkaz, M.: *Encyclopedia of Big Data: Content Management System (CMS)*, Springer International Publishing, Cham (2022)

[The] The Walt Disney Company: Webseite, URL https://thewaltdisneycompany.com/

[Typa] Typemill: Typemill Webseite, URL https://typemill.net/(13.06.2023)

[Typb] Typo3: TYPO3 Dokumentation, URL https://docs.typo3.org/(13.06.2023)

[Typc] Typo3: Typo3 Webseite, URL https://typo3.org/(13.06.2023)

[W3T23a] W3Techs: Changes in the usage of Kirby (2023), URL https://w3techs.com/technologies/changes/cm-kirby(04.05.2023)

[W3T23b] W3Techs: Comparison of the usage statistics of Grav vs. Statamic vs. Kirby vs. Bludit vs. HTMLy for websites: Historical trend and Market position (2023), URL https://w3techs.com/technologies/comparison/cm-bludit,cm-grav,cm-htmly,cm-kirby,cm-statamic(04.05.2023)

[W3T23c] W3Techs: Comparison of the usage statistics of Kirby vs. Pico for websites: Historical trend and Market position (2023), URL https://w3techs.com/technologies/comparison/cm-kirby,cm-pico(04.05.2023)

[W3T23d] W3Techs: Comparison of the usage statistics of WordPress vs. Kirby for websites (2023), URL https://w3techs.com/technologies/comparison/cm-kirby,cm-wordpress(04.05.2023)

[W3T23e] W3Techs: Ranking der Top 10 Content-Management-Systeme (CMS) weltweit nach Marktanteil im April 2023 (2023), URL https://de.statista.com/statistik/daten/studie/320670/umfrage/marktanteile-der-content-management-systeme-cms-weltweit/

(04.05.2023)

[W3T23f] W3Techs: Top 10 Content-Management-Systeme (CMS) weltweit nach Nutzung für Webseiten im April 2023 (2023), URL https://de.statista.com/statistik/daten/studie/320685/umfrage/nutzungsanteil-der-content-management-systeme-cms-weltweit/(04.05.2023)

[W3T23g] W3Techs: Usage statistics and market share of Kirby (2023), URL https://w3techs.com/technologies/details/cm-kirby(04.05.2023)

[Wora] Wordpress: Wordpress Developer Resources, URL https://developer.wordpress.org(10.07.2023)

[Worb] Wordpress: Wordpress Webseite, URL https://wordpress.org/(13.06.2023)

[XAM] XAMPP: XAMPP Download, URL https://www.apachefriends.org/de/index.html

Abkürzungsverzeichnis

API Application Programming Interface

CDA Content Delivery Application

CMA Content Management Application

CMF Content Management Framework

CMS Content Management System

CPT Custom Post Type

CSS Cascading Style Sheets

DAM Digital Asset Management

DB Datenbank

ECMS Enterprise Content Management System

EPUB Electronic publication

FTP File Transfer Protocol

GNU General Public License

GPL General Public License

HTML Hypertext Markup Language

ID Identifikationsnummer

IIS Internet Information Services

IT Information Technology

JSON JavaScript Object Notation

MIT Massachusetts Institute of Technology

PDF Portable Document Format

REST Representational State Transfer

RM Records Management

RSS Rich Site Summary

SaaS Software-as-a-Service

SSG Static Site Generator

SPA Single Page Application

SVN Apache Subversion

TXT Textdatei

URL Uniform Resource Locator

UV Unterversion

V Version

WCMS Web Content Management System

WWW World Wide Web

WYSIWYG What you see is what you get

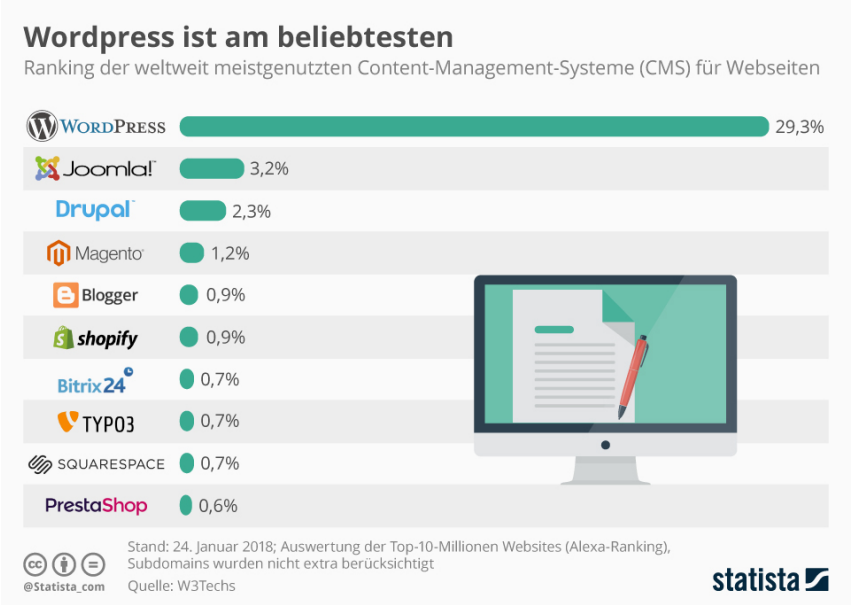
XML Extensible Markup Language

Abbildungsverzeichnis

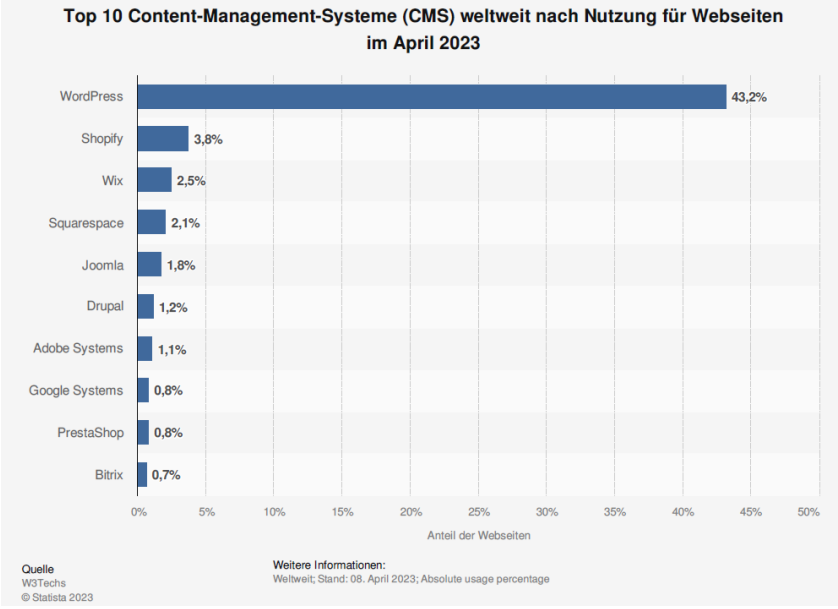
Listings

A Statistiken

Nutzung Top 10 CMS 2018,
Quelle: [Nie18]

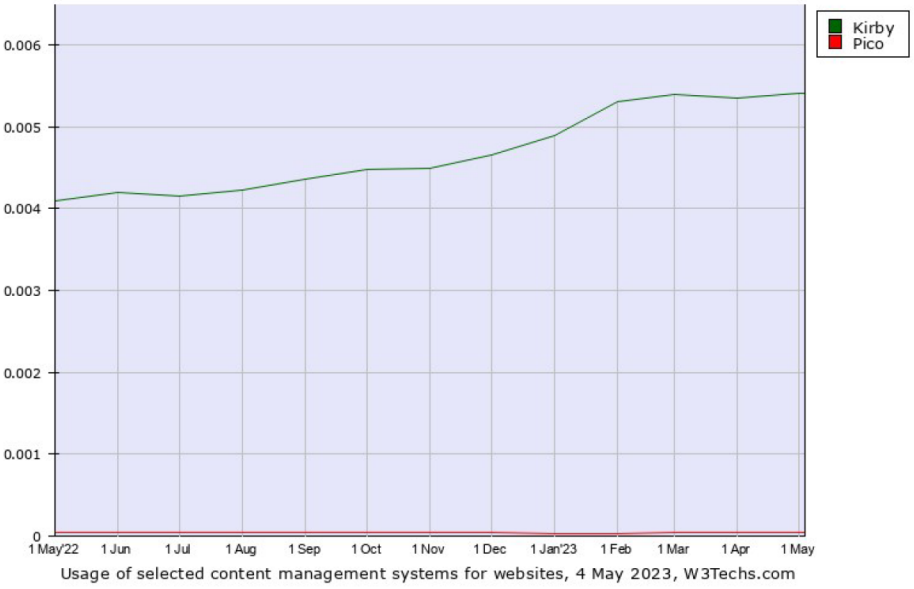


Nutzung Top 10 CMS 2023,
Quelle: [W3T23f]

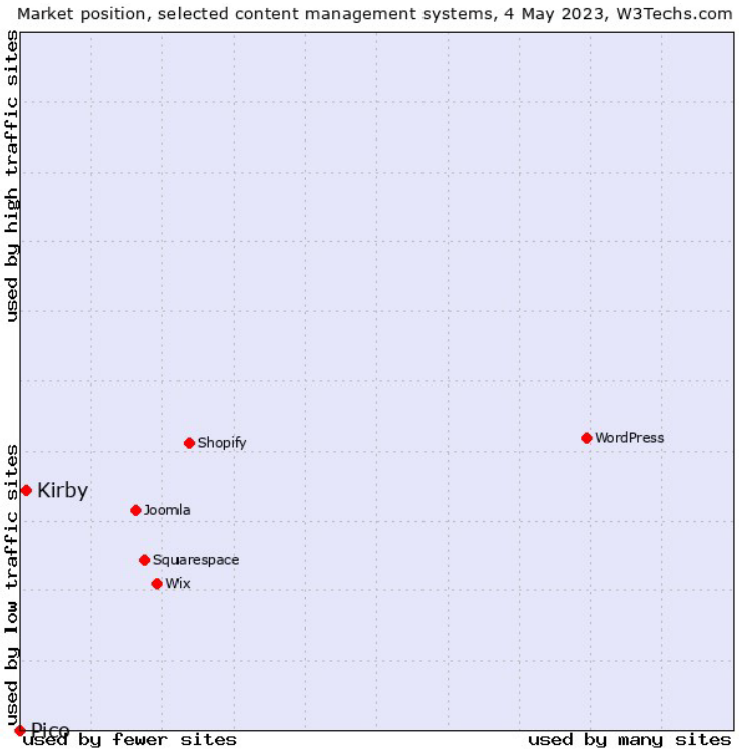


Kirby Nutzung (prozentual),
Quelle: [W3T23g]

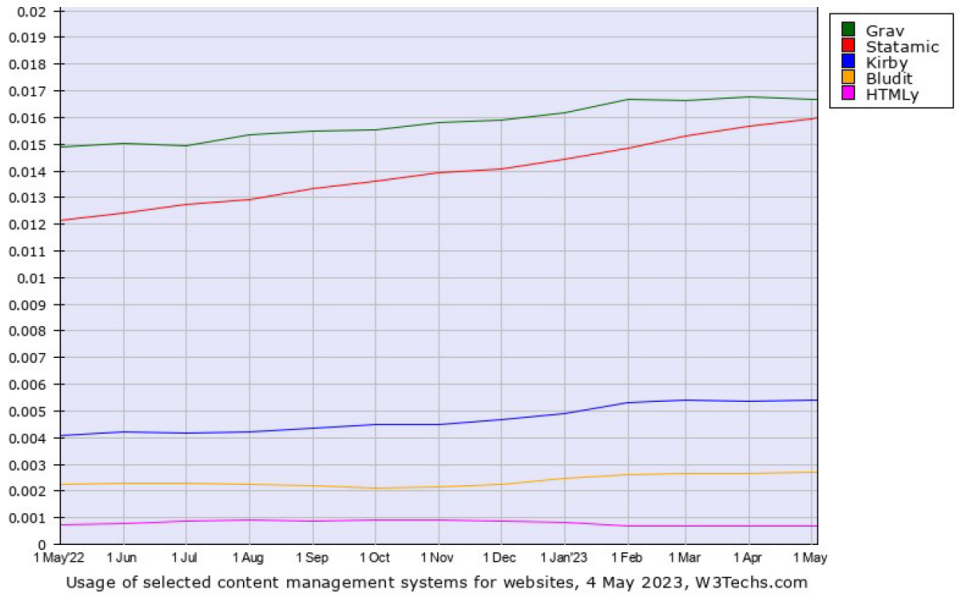
Kirby vs. Pico - Nutzung (prozentual),
Quelle: [W3T23c]



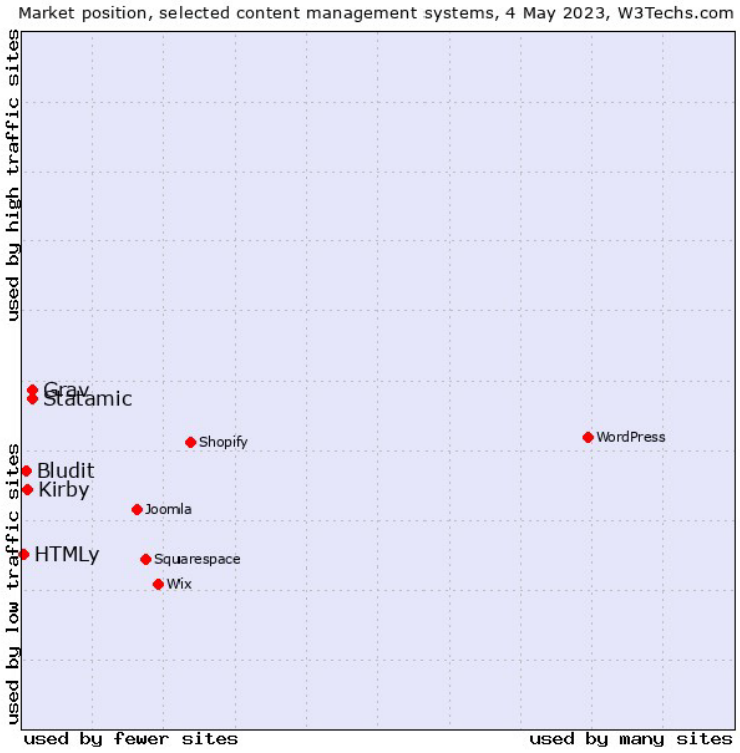
Kirby vs. Pico - Marktposition,
Quelle: [W3T23c]



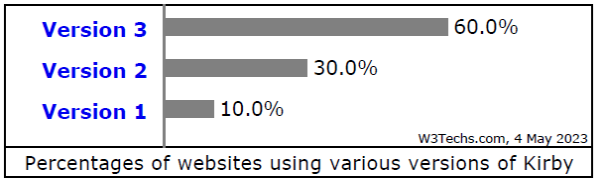
Grav vs. Statamic vs. Bludit vs. Kirby vs. HTMLy - Nutzung (prozentual),
Quelle: [W3T23b]



Grav vs. Statamic vs. Bludit vs. Kirby vs. HTMLy - Marktposition ,
Quelle: [W3T23b]

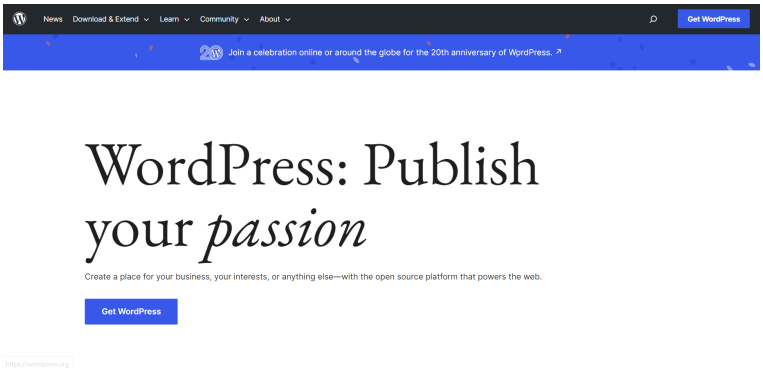


Kirby Versionen Nutzung,
Quelle: [W3T23g]

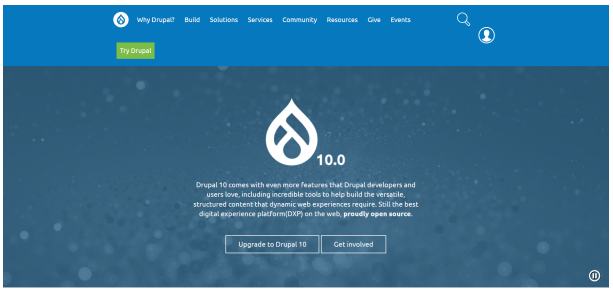


B Bilder

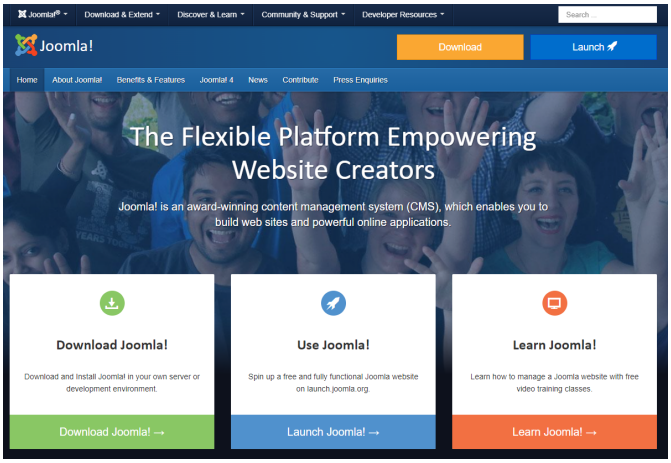
WordPress Webseite,
Quelle: [Worb]



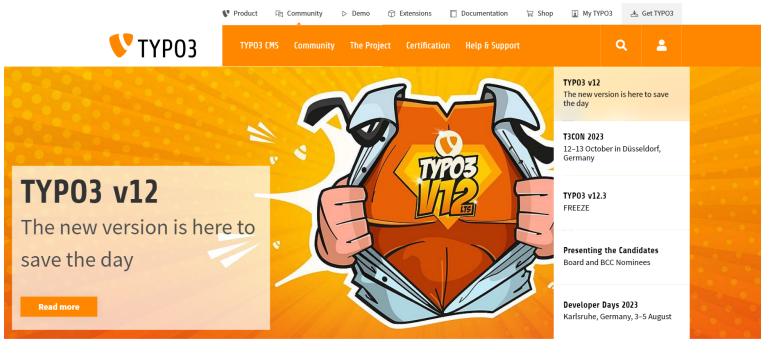
Drupal Webseite,
Quelle: [Dru]



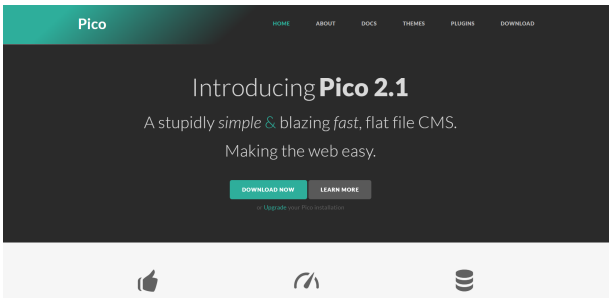
Joomla Webseite,
Quelle: [Joo]



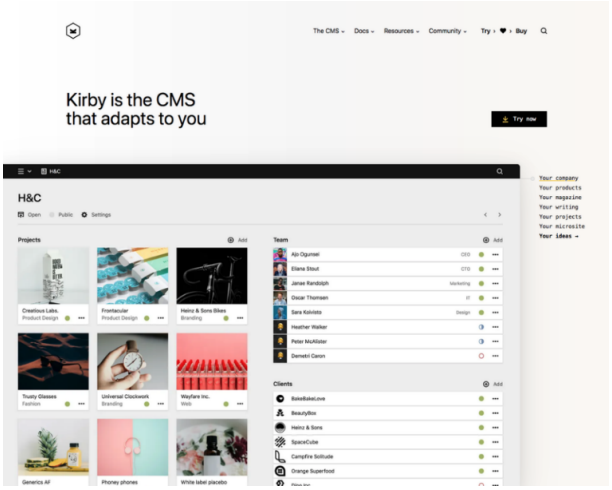
TYPO3 Webseite,
Quelle: [Typc]



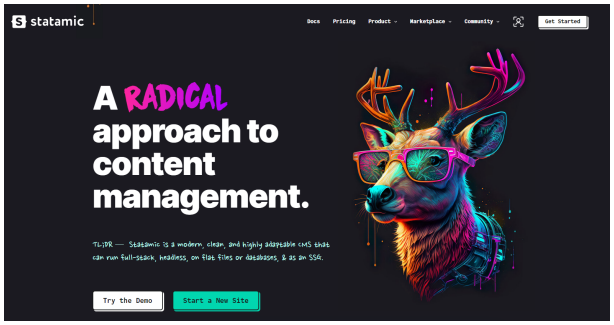
Pico Webseite,
Quelle: [Pic]



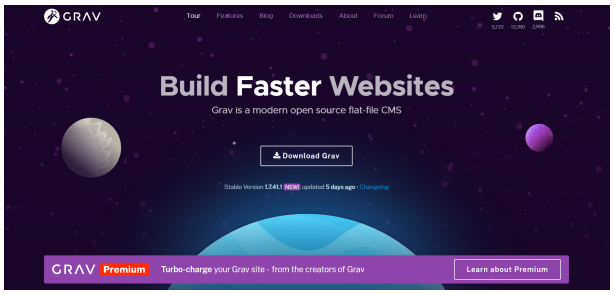
Kirby Webseite,
Quelle: [Kirb]



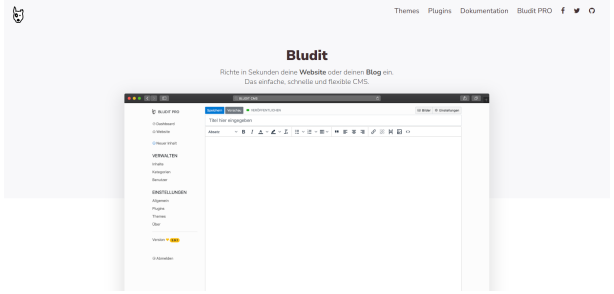
Statamic Webseite,
Quelle: [Stab]



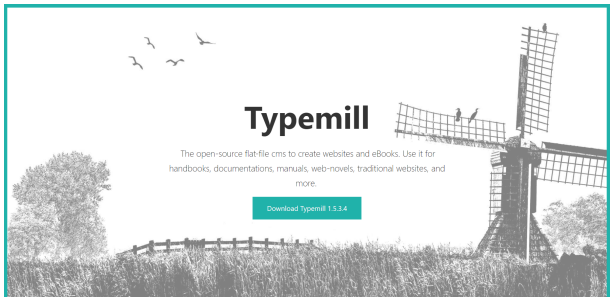
Grav Webseite,
Quelle: [Grab]



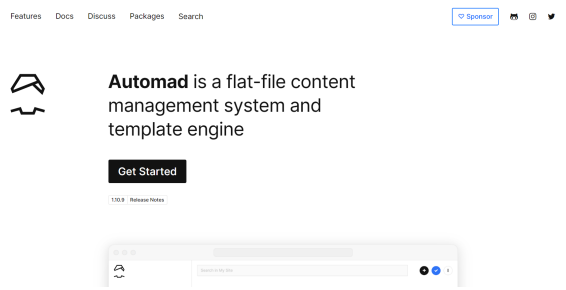
Bludit Webseite,
Quelle: [Bluc]



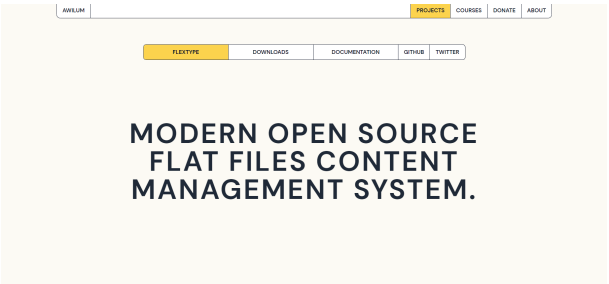
Typemill Webseite,
Quelle: [Typa]



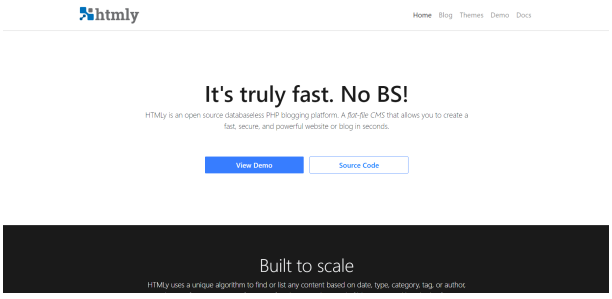
Automad Webseite,
Quelle: [Aut]



Flextype Webseite,
Quelle: [Fle]



HTMLy Webseite,
Quelle: [HTMb]



C Dateibasierte Systeme vs. Datenbankbasierte Systeme

Kategorie	**dateibasierte Systeme**	**datenbankbasierte Systeme**
Performance	Bessere Performance	Geringere Performance bei großen Datenmengen
Inhaltsbearbeitung	Einfache Bearbeitung der Inhalte mit Markdown-Dateien	Erfordert spezielle Programme zur Bearbeitung des Inhalts
Volltextsuche	Eingeschränkt	Umfangreich
Strukturierung	weniger strukturierte Informationen	flexible und offene Designs
Installation und Betrieb	Einfache Installation durch Hochladen des Quellcodes und Einrichten eines Benutzers	Zusätzliche Verwaltung des Datenbanksystems im Backend erforderlich
Erweiterbarkeit	Begrenzte Theme- und Plugin-Auswahl	Größere Auswahl an Themes und Plugins
Anpassungsfähigkeit	Für individuelle Anpassungen kann ein Entwickler erforderlich sein	Bietet Möglichkeiten für individuelle Anpassungen
Geeignet für	Kleine Projekte, weniger strukturierte Informationen	Große Datenmengen, stark strukturierte Inhalte wie Produktdaten und Asset-Verwaltung
Performance-Probleme	Weniger wahrscheinlich bei kleineren Inhaltsmengen	Möglich bei großen Inhaltsmengen
Sicherheit	Kein häufiges Angriffsziel, keine SQL-Injections möglich	Erfordert zusätzliche Sicherheitsmaßnahmen für das Datenbanksystem
Erweiterbarkeit im E-Commerce	Wenige Systeme bieten passende Plugins an	Größere Auswahl an Systemen mit passenden E-Commerce-Plugins
Speicherart	speichern Inhalte in Flat-Files in Ordnerstruktur	speichern Inhalte in Tabellen in Datenbank
Skalierbarkeit	Begrenzte Skalierbarkeit	Bessere Skalierbarkeit mit größeren Datenmengen

D CMS Übersicht

CMS	Entwickler	Geeignet für	Webseite
WordPress	Matthew Mullenweg, Mike Little	simple One-Pager, Shops, komplette Magazin-Projekte von etablierten Verlagen, Sinnvoll: Blogs und Magazine	wordpress.org
Drupal	Dries Buytaert	Mittelstand, Enterprise-Bereich, Business-Bereich, Handel, Barrierefreiheit, Online-Shops	drupal.org
Joomla!	Keinen dezidierten Gründer, „Open Source Matters Inc.“	mittelkomplexe, komplexe Business-Seiten mit Systemflexibilität	joomla.org
TYPO3	Kasper Skårhøj	Enterprise-Bereich, komplexe Corporate Websites	typo3.org

CMS	Entwickler	Geeignet für	Webseite
Kirby	Bastian Allgeier	schlanke Webseiten, kleine und etwas größere Web-Projekte, mit Einfachheit, Flexibilität und guter Usability für Autoren	getkirby.com
Statamic	Jack McDade	kleinere Unternehmen und Mittelstand, E-Commerce-Projekte	statamic.com
Pico	Gilbert Pellegrom, jetzt Daniel Rudolf	einfache, kleine Webauftritte	picocms.org
HTMLy	Danang Probo Sayekti	Blogs	demo.htmly.com
Bludit	Djego Najar	schlanke Webseiten, Blogs	bludit.com
Grav	Andy Miller	alle Arten von kleineren Webseiten: Blog, Corporate Website, One-Pager, Landing-Pages, Portfolio-Seiten, kleine Business-Auftritte, Verzeichnisse, Dokumentationen, kleine E-Commerce-Seiten	getgrav.org
Automad	Marc Anton Dahmen	Gestaltungsfreiheit, komplexe Layouts, gestalterisch anspruchsvolle Webseiten aus dem Kreativ-Bereich ohne Entwickler	automad.org
Flextype	Sergey Romanenko	Entwickler, mit kleinem Headless-CMS ohne Datenbank	awilum.github.io/flextype
Typemill	Sebastian Schürmanns	informationsorientierte Webseiten, strukturierte Inhalte, buchartige Publikationen	typemill.net

E Vergleich

E.1 Kosten und Lizenzierung

Kostenmodelle und Lizenzierung

CMS	Kosten	Lizenz	Open source
Automad	0 €	MIT	Ja
Bludit	0 €	MIT	Ja
Drupal	0 €	GPL	Ja
Flextype	0 €	MIT	Ja
Grav	0 €	MIT	Ja
HTMLy	0 €	GPL	Ja
Joomla!	0 €	GNU	Ja
Kirby	ab 99 €	-	Ja
Pico	0 €	MIT	Ja
Statamic	0 €/259 €	-	Nein
Typemill	0 €	MIT	Ja
TYPO3	0 €	GPL	Ja
WordPress	0 €	GPL	Ja

Kirby Preise

1 Lizenz	5 Lizenzen	10 Lizenzen	15 Lizenzen	Benutzerdefiniertes Paket	Für den guten Zweck
99 €	470 €	890 €	1260 €	nach Absprache	0 €
Eigenes Hosting und eine eigene Domain mitzubringen					

Statamic Preise

Solo	Pro	Master	Enterprise
$ 0	$ 259 pro Projekt	$ 1095	Nach Vereinbarung
Website für sich selbst, einen Freund oder ein Hobby	Ein Jahr Updates und $ 59/Jahr für weitere Updates	5 Pro-Lizenzen und 25 % Rabatt auf zusätzliche Lizenzen für ein Jahr	Pro-Lizenzen, Anzahl je nach Vereinbarung

Automad Spende

Ko-fi	GitHub Sponsor	PayPal
einmalige oder monatliche Spende	einmalige oder monatliche Spende	einmalige Spende
	Auflistung als Unterstützer auf der Sponsorenseite	

Bludit Spende

Unterstützer	Unterstützer	Gold Sponsor	Platin Sponsor
1 € / Monat	2 € / Monat	5 € / Monat	10 € / Monat
Alle Versionen enthalten die neuste Version von Bludit PRO und alle Updates			

E.2 Entwicklerfreundlichkeit und Support

CMS	WordPress	Drupal	Joomla!	TYPO3
PHP	ab 7.4	ab 8.1	ab 8.0	ab 8.1
Webserver	Apache, Nginx	Apache, Nginx, IIS	Apache, Nginx, IIS	Apache, Nginx, IIS
Technologie	PHP	PHP	PHP	PHP
Datenbanken	MySQL, MariaDB, PostgreSQL, Oracle	MySQL, PostgreSQL, SQLite	MySQL, PostgreSQL	MySQL, MariaDB, PostgreSQL, SQLite
Dokumentatio	umfangreich, detailliert	umfangreich, detailliert	umfangreich, detailliert	umfangreich, detailliert
Unterstützung	Forum, Tutorials	Foren	Forum, Tutorials	Slack, Forum, YouTube-Tutorials
Multilanguage	über Plugins	Ja	Ja	Ja
Sonstiges		decoupled Drupal		

CMS	Kirby	Statamic	Pico	HTMLy	Bludit	Grav	Automad	Flextype	Typemill
PHP	8.0, 8.1, 8.2	ab 7.4 (8.0+)	ab 5.3.6	ab 5.3	ab 5.3	ab 7.3.6	ab 7.4,	ab 8.1	
Webserver	Apache, Nginx, LiteSpeed, Caddy		Apache, Nginx, Lighttpd	Apache, Nginx, Lighttpd	Apache, Nginx, Lighttpd, H2O, Hiawatha	Apache, Nginx, LiteSpeed, IIS, Lightly	Apache, Nginx	Apache, Nginx, LiteSpeed, IIS, Lightly	Apache
Technologie	PHP, Vue	PHP, Laravel, Vue	PHP	PHP	PHP, JSON	PHP, YAML	PHP	PHP, YAML, JSON	PHP, Vue, Slim
Formatting	Markdown	Markdown, HTML	Markdown	Markdown	Markdown, HTML	Markdown	Markdown	Markdown, HTML	Markdown
Dokumentation	umfangreich, detailliert	umfangreich, detailliert	umfangreich	klein	umfangreich	umfangreich, detailliert	klein aber übersichtlich	klein aber übersichtlich	umfangreich, detailliert, mit Videos
Versionskontrolle	Git	Git	Git	Git	Git	Git, SVN, Dropbox, andere Systeme	Git	Git, andere Systeme	Git
Unterstützung	GitHub, Forum, Discord, YouTube-Tutorials	GitHub, Forum, Discord, Laracasts-Videos, SLA-Support	GitHub	GitHub	GitHub, Forum, Discord	GitHub, Forum, Discord	GitHub, Forum	GitHub	GitHub
Multilanguage	Ja	-	-	-	-	Ja	-	-	-
Sonstiges	Headless, mit SSG, mobiler App	Headless, mit DB, als SSG, statisch	-	-	-	-	Headless	Headless, Hybrid	-

E.3 Zukünftige Entwicklung und Aktualisierung

CMS	erster Release	letzter Release	auf dem Markt	Aktualität	Updatehäufigkeit
WordPress	2003	V6.2, 29.03.2023	20 Jahren	sehr aktuell	2 - 3 Unterversionen im Jahr, große Updates (neue Versionen) alle 4 - 5 Jahre
Drupal	2001	V10.0.9, 03.05.2023	22 Jahren	sehr aktuell	Mehrere Unterversionen im Jahr, große Updates (neue Versionen) unregelmäßig
Joomla!	2005	V4.3.2, 29.05.2023	18 Jahren	sehr aktuell	Mehrere Unterversionen im Jahr, große Updates (neue Versionen) unregelmäßig
TYPO3	2001	V12.4.1, 09.05.2023	22 Jahren	sehr aktuell	Mehrere Unterversionen im Jahr, große Updates (neue Versionen) alle 1 - 2 Jahre (seit V6.0), LTS ein Jahr später als normale Version

CMS	Erster Release	Letzter Release	Auf dem Markt	Aktualität	Updatehäufigkeit
Kirby	2012	V3.9.5, 07.06.2023	11 Jahren	sehr aktuell	Kleine Update-Fixes monatlich, neue Unterversionen circa jedes halbe Jahr, neue Versionen alle 2 -5 Jahre
Statamic	2012	V4.6, 07.06.2023	11 Jahren	sehr aktuell	Kleine Update-Fixes wöchentlich, neue Versionen bisher circa alle 3 - 4 Jahre
Pico	2012	V2.1.4, 29.08.2020	11 Jahren	nicht aktuell	Seit 2020 keine Updates mehr, neue Versionen unregelmäßig, bis 2020 alle 2 - 3 Jahre
HTMLy	2014	V2.8.2, 12.02.2022	9 Jahren	nicht aktuell	Seit 2022 keine Updates mehr, neue Versionen unregelmäßig, Pause von 2017 - 2020
Bludit	2016	V3.14.1, 07.09.2022	7 Jahren	nicht aktuell	Seit 2022 keine Updates mehr, neue Versionen unregelmäßig
Grav	2014	V1.7.41.2, 01.06.2023	9 Jahren	sehr aktuell	Monatlich mehrere kleine Updates, neue Unterversionen unregelmäßig
Automad	2013	V1.10.9, 19.05.2022	10 Jahre	nicht aktuell	Seit 2022 keine Updates mehr, neue Unterversionen unregelmäßig
Flextype	2018	V1.0.0 Alpha 3, 19.11.2022	5 Jahre	nicht aktuell	Seit 2022 keine Updates mehr, neue Unterversionen unregelmäßig
Typemill	2017	V1.1.6, 23.05.2018	6 Jahre	nicht aktuell	Seit 2018 keine Updates mehr

Drupal	
neue Versionen	Version 10
V01.0 - 15.01.2001 V02.0 - 15.03.2001 V03.0 - 15.09.2001 V04.0 - 07.03.2005 V05.0 - 15.01.2007 V06.0 - 13.02.2008 V07.0 - 05.01.2011 V08.0 - 19.11.2015 V09.0 - 03.06.2020 V10.0 - 15.12.2022	V10.0.1 - 04.01.2023 V10.0.2 - 18.01.2023 V10.0.3 - 01.02.2023 V10.0.4 - 01.03.2023 V10.0.5 - 15.03.2023 V10.0.6 - 24.03.2023 V10.0.7 - 24.03.2023 V10.0.8 - 19.04.2023 V10.0.9 - 03.05.2023

Joomla!	
neue Versionen	Version 4
V1.0 - 15.09.2005	V4.1 - 15.02.2022
V3.0 - 27.09.2012	V4.2 - 16.08.2022
V4.0 - 17.08.2021	V4.3 - 18.04.2023

WordPress	
neue Versionen	Version 6
V0.7 - 27.05.2003 V1.0 - 03.01.2004 “Miles Davis“ V2.0 - 26.12.2005 “Duke Ellington“ V3.0 - 17.06.2010 “Thelonious Monk“ V4.0 - 04.09.2014 “Benny Goodman“ V5.0 - 06.12.2018 “Bebo Valdés“ V6.0 - 24.05.2022 “Arturo O’Farrill“	V6.1 - 01.11.2022 “Mikhail Alperin“ V6.2 - 29.03.2023 “Eric Dolphy“ V6.3 - geplant August 2023 V6.4 - geplant für November 2023
Version 5	Version 4
V5.1 - 21.02.2019 “Betty Carter“ V5.2 - 07.05.2019 “Jaco Pastorius“ V5.3 - 12.11.2019 “Rahsaan Roland Kirk“ V5.4 - 31.03.2020 “Nat Adderley“ V5.5 - 11.08.2020 “Billy Eckstine“ V5.6 - 08.12.2020 “Nina Simone“ V5.7 - 09.03.2021 “Esperanza Spalding“ V5.8 - 20.07.2021 “Art Tatum“ V5.9 - 25.01.2022 “Joséphine Baker“	V4.1 - 18.12.2014 “Dinah Washington“ V4.2 - 23.04.2015 “Bud Powell“ V4.3 - 01.08.2015 “Billie Holiday“ V4.4 - 08.12.2015 “Clifford Brown“ V4.5 - 12.04.2016 “Coleman Hawkins“ V4.6 - 16.08.2016 “Pepper Adams“ V4.7 - 06.12.2016 “Sarah Vaughan“ V4.8 - 08.06.2017 “Bill Evans“ V4.9 - 15.11.2017 “Billy Tipton“
Version 3	Version 2
V3.1 - 23.02.2011 “Django Reinhardt“ V3.2 - 04.07.2011 “George Gershwin“ V3.3 - 12.12.2011 “Sonny Stitt“ V3.4 - 13.06.2012 “Grant Green“ V3.5 - 11.12.2012 “Elvin Jones“ V3.6 - 01.08.2013 “Oscar Peterson“ V3.7 - 24.10.2013 “Count Basie“ V3.8 - 12.12.2013 “Charlie Parker“ V3.9 - 16.04.2014 “Jimmy Smith“	V2.1 - 22.01.2007 “Ella Fitzgerald“ V2.2 - 16.05.2007 “Stan Getz“ V2.3 - 24.09.2007 “Dexter Gordon“ V2.5 - 29.03.2008 “Michael Brecker“ V2.6 - 15.07.2008 “McCoy Tyner“ V2.7 - 10.12.2008 “John Coltrane“ V2.8 - 11.06.2009 “Chet Baker“ V2.9 - 18.12.2009 “Carmen McRae“
Version 1	
V1.0 - 03.01.2004 “Miles Davis” V1.2 - 22.05.2004 “Charles Mingus” V1.5 - 17.02.2005 “Billy Strayhorn”	

TYPO3		
neue Versionen	LTS Versionen	Version 12
V03.0 - 2001 V04.0 - 07.04.2006 V06.0 - 27.11.2012 V07.0 - 02.12.2014 V08.0 - 22.03.2016 V09.0 - 12.12.2017 V10.0 - 23.07.2019 V11.0 - 22.12.2020 V12.0 - 04.10.2022	V08 LTS - 04.04.2017 V09 LTS - 30.10.2018 V10 LTS - 21.04.2020 V11 LTS - 05.10.2021 V12 LTS - 25.04.2023	V12.1 - 06.12.2022 V12.2 - 07.02.2023 V12.3 - 28.03.2023 V12.4 - 25.04.2023

Kirby	
neue Versionen	Version 3
V1.0 - 09.01.2012 V2.0 - 07.10.2014 V3.0 - 05.02.2019 V4.0 - später Sommer 2023	V3.1 - 19.03.2019 "Chamaeleo" V3.2 - 25.06.2019 Ärchaius" V3.3 - 05.11.2019 "Trioceros" V3.4 - 07.07.2020 "Furcifer" V3.5 - 15.12.2020 "Calumma" V3.6 - 16.11.2021 "Jungle Calumma" V3.7 - 27.06.2022 "Kinyongia" V3.8 - 06.10.2022 "Rhampholeon" V3.9 - 17.01.2023 "Brookesia"

Statamic		
neue Versionen	Version 4	Version 3
V1.0 - 19.06.2012 V2.0 - 31.03.2016 V3.0 - 19.08.2020 V4.0 - 09.05.2023	V4.1.0 - 11.05.2023 V4.2.0 - 19.05.2023 V4.3.0 - 24.05.2023 V4.4.0 - 30.05.2023 V4.5.0 - 02.06.2023 V4.6.0 - 07.06.2023	V3.1.0 - 24.03.2021 V3.2.0 - 24.08.2021 V3.3.0 - 15.03.2022 V3.4.0 - 27.01.2023

Pico		
neue Versionen	Version 2.1	Version 2.0
V1.0 - 24.12.2015 V2.0 - 01.07.2018 V3.0 - 24.12.2020 (Pre-Release Alpha)	V2.1.0 - 24.11.2019 V2.1.1 - 31.12.2019 V2.1.2 - 10.04.2020 V2.1.3 - 10.07.2020 V2.1.4 - 29.08.2020	V2.0.1 - 29.07.2018 V2.0.2 - 12.08.2018 V2.0.3 - 03.12.2018 V2.0.4 - 17.12.2018
Version 1	Version 0	
V1.0.1 - 27.02.2016 V1.0.2 - 16.03.2016 V1.0.3 - 11.05.2016 V1.0.4 - 04.10.2016 V1.0.5 - 02.05.2017 V1.0.6 - 25.07.2017	V0.1 - 04.04.2012 V0.2 - 26.04.2013 V0.3 - 27.04.2013 V0.4 - 01.05.2013 V0.5 - 03.05.2013 V0.6 - 06.05.2013 V0.7 - 04.09.2013 V0.8 - 23.10.2013 V0.9 - 28.04.2015	

HTMLy		
neue Versionen	Version 2	Version 1
V1.0 - 31.01.2014 V2.0 - 27.06.2014	V2.0.0 - 27.06.2014 V2.1.0 - 25.07.2014 V2.2.0 - 21.08.2014 V2.3.0 - 28.09.2014 V2.4.0 - 28.12.2014 V2.5.0 - 10.02.2015 V2.6.0 - 17.08.2015 V2.7.0 - 01.01.2016 V2.8.0 - 01.04.2021	V1.0 - 31.01.2014 V1.1 - 08.02.2014 V1.5 - 15.02.2014 V1.6 - 17.02.2014 V1.7 - 21.02.2014 V1.8 - 25.02.2014 V1.9 - 14.06.2014

Bludit		
neue Versionen	Version 3	Version 2
V1.0 - 21.01.2016 (Pre-Release Beta) V2.0 - 16.10.2017 V3.0 - 21.09.2018 V4.0 - 25.11.2021 (Pre-Release Beta)	V3.00 - 21.09.2018 "Hops" V3.01 - 08.10.2018 "Malt" V3.02 - 21.10.2018 "Yeast" V3.03 - 31.10.2018 "Water" V3.04 - 10.11.2018 "Beer" V3.05 - 01.12.2018 ÏPA" V3.06 - 15.01.2019 "Pilsner" V3.07 - 27.01.2019 "Bock" V3.08 - 23.02.2019 ÄPA" V3.09 - 27.05.2019 "Porter" V3.10 - 19.10.2019 "Mateo" V3.11 - 08.02.2020 "Pizza Time" V3.12 - 21.03.2020 "Handwashing" V3.13 - 29.07.2020 Ëdi" V3.14 - 05.09.2022 Öut of time"	V2.0 - 16.10.2017 V2.1 - 27.12.2017 "Rick" V2.2 - 23.01.2018 "Pepper" V2.3 - 08.03.2018 "1-Commit"

Grav	
neue Versionen	Version 1
V1.0 - 11.12.2015	V1.0 - 11.12.2015 V1.1 - 14.07.2016 V1.2 - 31.03.2017 V1.3 - 17.07.2017 V1.4 - 09.03.2018 V1.5 - 17.08.2018 V1.6 - 11.04.2019 V1.7 - 19.01.2021

Automad		
neue Versionen	Version 1	Version 0
V0.01 - 15.10.2013 V1.00 - 22.09.2018	V1.01 - 08.12.2018 V1.02 - 26.01.2019 V1.03 - 22.11.2019 V1.04 - 09.01.2020 V1.05 - 02.06.2020 V1.06 - 31.10.2020 V1.07 - 02.05.2021 V1.08 - 07.08.2021 V1.09 - 13.10.2021 V1.10 - 24.11.2021	V0.01 - 15.10.2013 V0.02 - 22.10.2013 V0.03 - 04.11.2013 V0.04 - 08.11.2013 V0.05 - 15.11.2013 V0.06 - 24.11.2013 V0.07 - 30.11.2013 V0.08 - 04.04.2014 V0.09 - 27.07.2014 V0.10 - 12.09.2015

Flextype		
neue Versionen	Version 0	Version 0.9
V1.0.0 - 19.11.2020 (Release Alpha)	V0.01 - 21.03.2018 V0.02 - 23.03.2018 V0.03 - 05.05.2018 V0.04 - 17.05.2018 V0.05 - 03.06.2018 V0.06 - 09.06.2018 V0.07 - 15.11.2018 V0.08 - 28.12.2018 V0.09 - 14.06.2019	V0.9.1 - 18.06.2019 V0.9.2 - 06.07.2019 V0.9.3 - 07.07.2019 V0.9.4 - 11.09.2019 V0.9.5 - 21.09.2019 V0.9.6 - 01.12.2019 V0.9.7 - 03.03.2020 V0.9.8 - 14.05.2020 V0.9.9 - 05.08.2020 V0.9.10 - 19.08.2020 V0.9.14 - 25.08.2020 V0.9.12 - 07.12.2020 V0.9.13 - 20.12.2020 V0.9.14 - 30.12.2020 V0.9.15 - 03.01.2021 V0.9.16 - 14.01.2021

Typemill	
neue Versionen	Version 1
V1 - 14.04.2017 V2 - Sommer 2023 (Alpha-Release geplant für Juli 2023)	V1.1.0 - 14.04.2017 V1.1.1 - 26.02.2018 V1.1.2 - 15.03.2018 V1.1.3 - 20.04.2018 V1.1.4 - 30.04.2018 V1.1.5 - 10.05.2018 V1.1.6 - 23.05.2018

E.4 Wertung

Wertungsskala

sehr gut	4
gut	3
okay	2
schlecht	1
nicht ausreichend	0

Entwicklerfreundlichkeit Traditionelle CMS

CMS	Webserver	Datenbanken	Dokumentation	Support	Multilanguage	Ergebnis
WordPress	3	3	4	3	1	**14**
Drupal	3	3	4	3	3	**16**
Joomla!	3	3	4	3	3	**16**
TYPO3	3	3	4	4	3	**17**

Entwicklerfreundlichkeit Flat-File CMS

CMS	Webserver	Formatting	Dokumentation	Support	Multilanguage	Ergebnis
Kirby	3	3	4	4	3	**17**
Statamic		3	4	4		**11**
Pico	3	3	3	2		**11**
HTMLy	3	3	2	2		**10**
Bludit	3	3	3	3		**12**
Grav	3	3	4	3	3	**16**
Automad	3	3	2	3		**11**
Flextype	3	3	2	2		**10**
Typemill	3	3	4	2		**12**

Entwicklung Traditionelle CMS

CMS	letzter Release	Auf dem Markt seit	Aktualität	Updatehäufigkeit	Ergebnis
WordPress	4	4	4	3	**15**
Drupal	4	4	4	3	**15**
Joomla!	4	4	4	3	**15**
TYPO3	4	4	4	4	**16**

Entwicklung Flat-File CMS

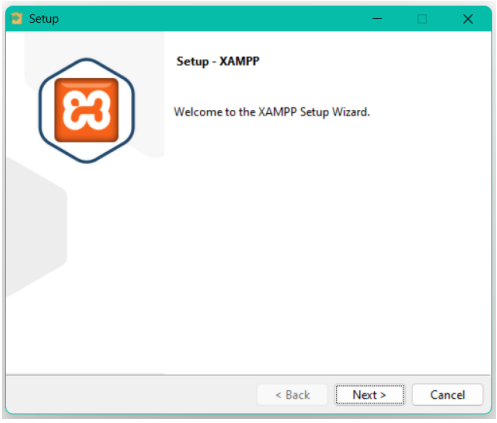
CMS	letzter Release	Auf dem Markt seit	Aktualität	Updatehäufigkeit	Ergebnis
Kirby	4	4	4	4	**16**
Statamic	4	4	4	4	**16**
Pico	1	4	1	1	**7**
HTMLy	3	3	3	2	**11**
Bludit	3	3	3	2	**11**
Grav	4	3	4	3	**14**
Automad	3	4	3	2	**12**
Flextype	3	2	3	2	**10**
Typemill	0	2	0	0	**2**

F Installation

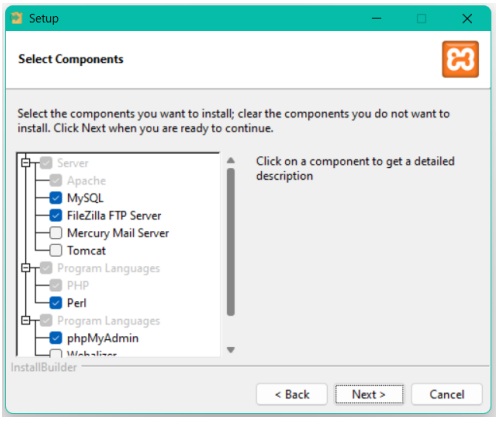
F.1 Tools

F.1.1 XAMPP

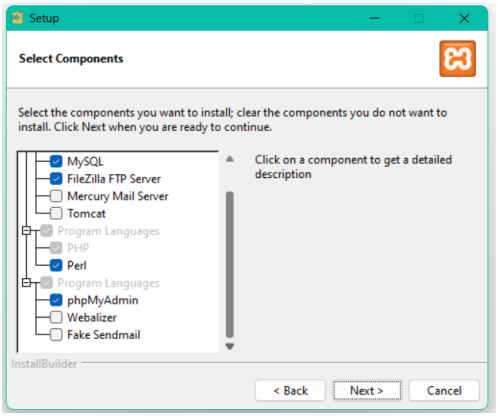
Setup Wizard



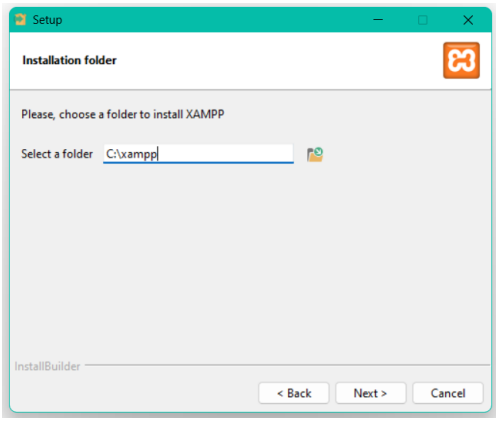
Komponentenauswahl 1



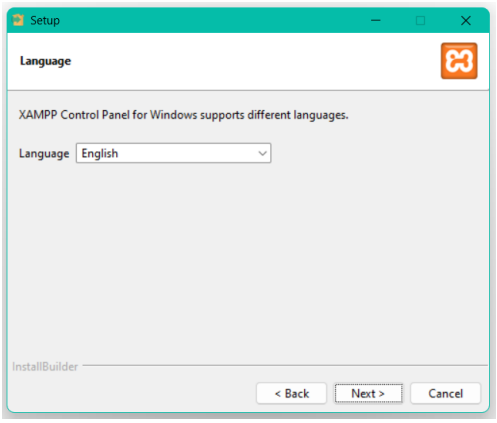
Komponentenauswahl 2



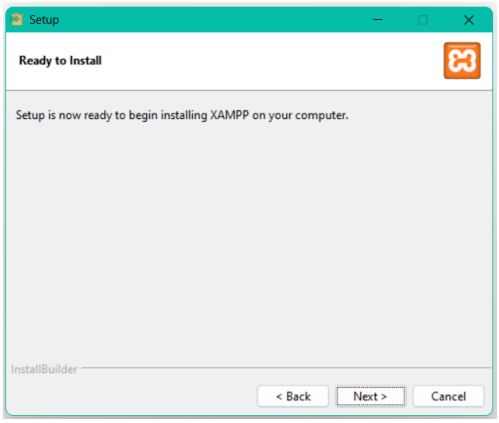
Installationsordner



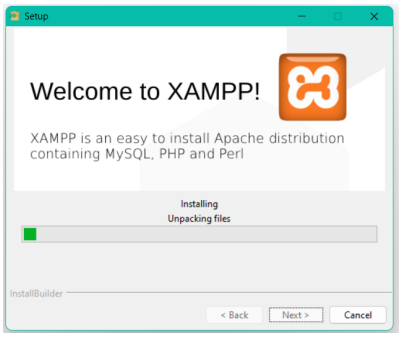
Sprachauswahl



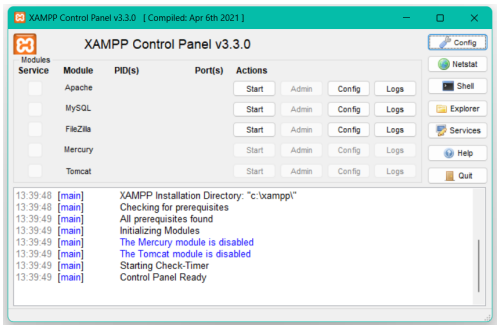
Ready to install



Installation

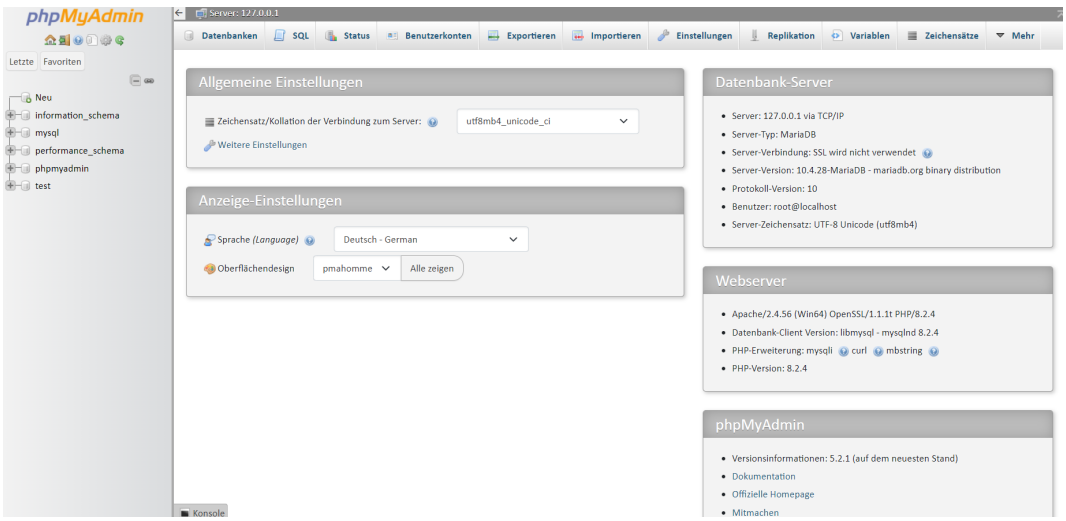


Control Panel

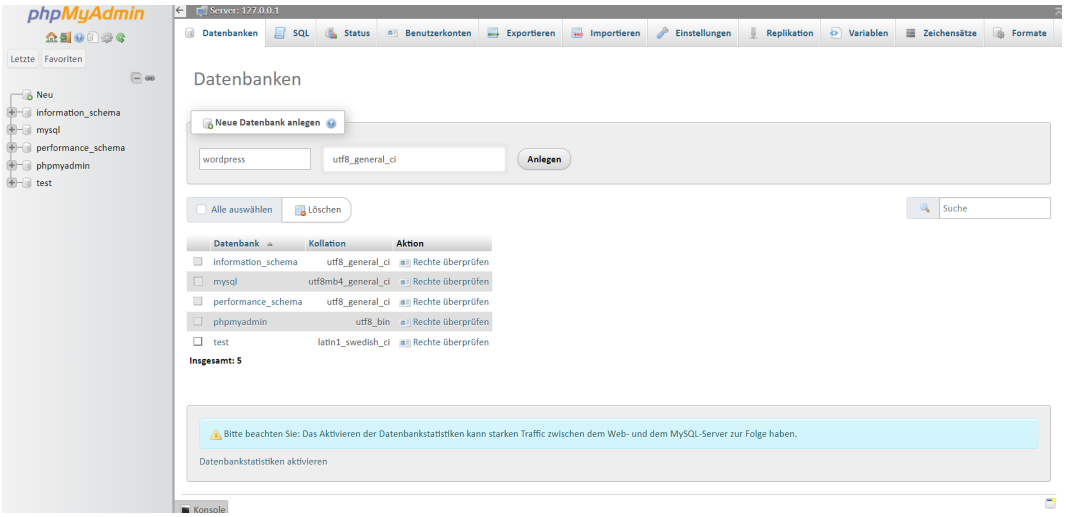


F.1.2 PhpMyAdmin

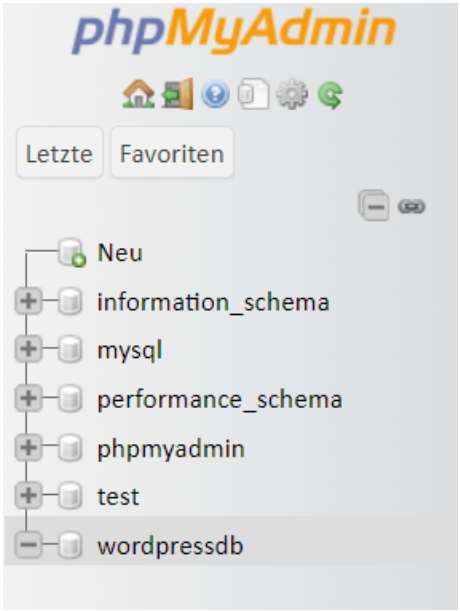
Übersicht



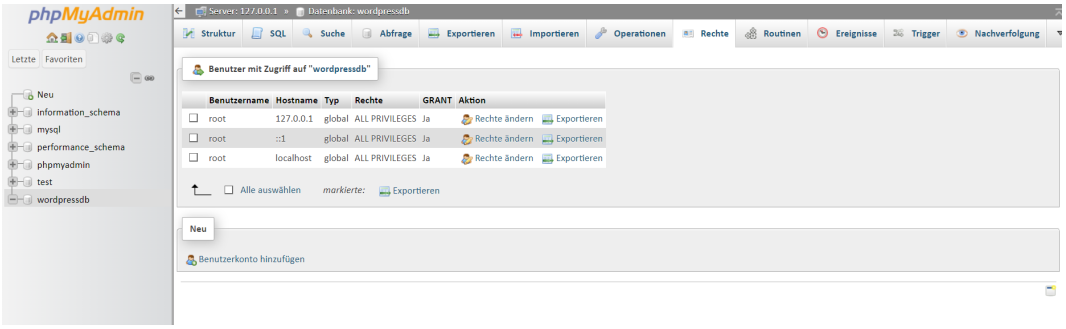
neue Datenbank anlegen



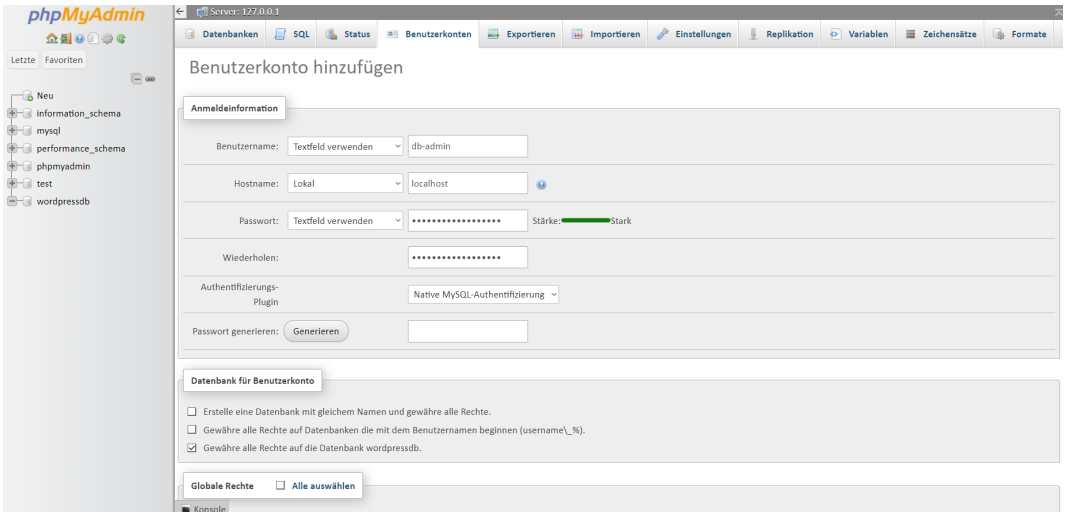
neue Datenbank



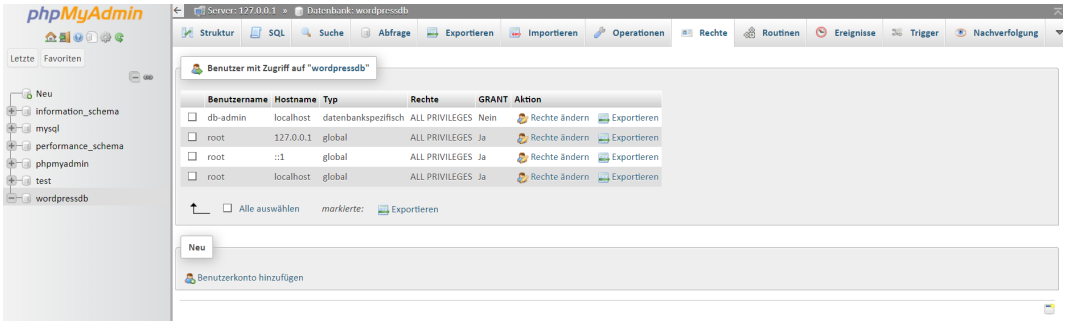
Rechte Reiter



Benutzer anlegen

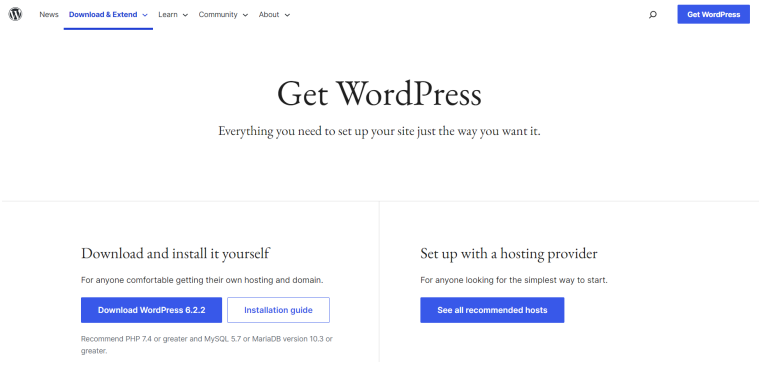


Rechte Reiter mit neuem Benutzer

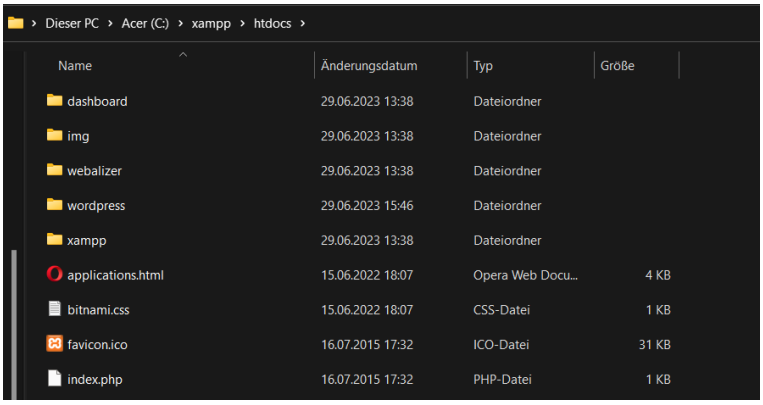


F.2 WordPress

Download WordPress



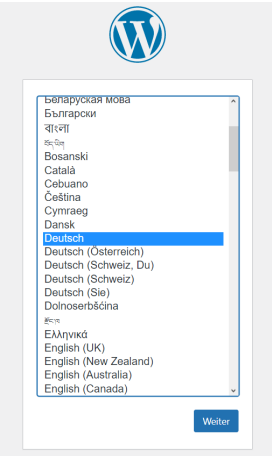
htdocs-Ordner



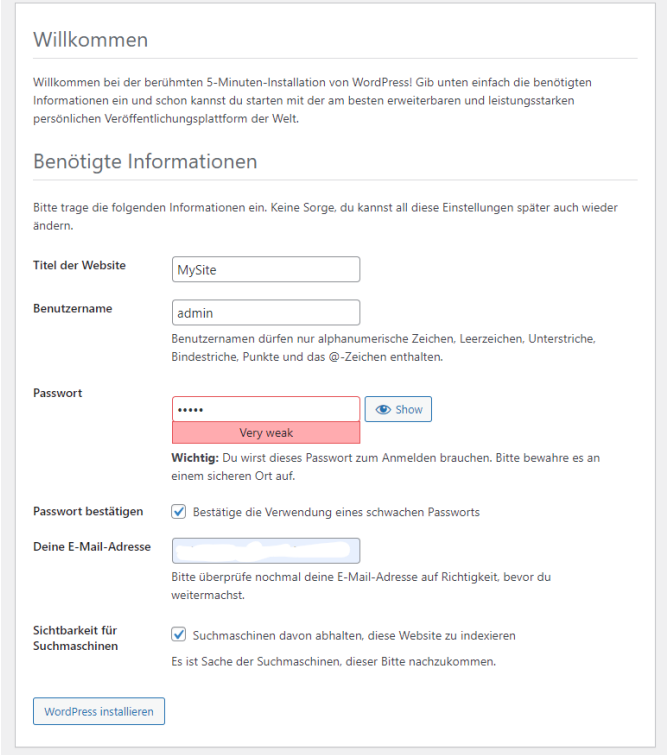

```
1  // ** Database settings - You can get this info from your web host ** //
2  /** The name of the database for WordPress */
3  define( 'DB_NAME', 'wordpressdb' );
4
5  /** Database username */
6  define( 'DB_USER', 'db-admin' );
7
8  /** Database password */
9  define( 'DB_PASSWORD', 'db-admin@wordpress' );
10
11 /** Database hostname */
12 define( 'DB_HOST', 'localhost' );
13
14 /** Database charset to use in creating database tables. */
15 define( 'DB_CHARSET', 'utf8' );
16
17 /** The database collate type. Don't change this if in doubt. */
18 define( 'DB_COLLATE', '' );
```

```
1  /**#@+
2  * Authentication unique keys and salts.
3  *
4  * Change these to different unique phrases! You can generate these using
5  * the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org
       secret-key service}.
6  *
7  * You can change these at any point in time to invalidate all existing
       cookies.
8  * This will force all users to have to log in again.
9  *
10 * @since 2.6.0
11 */
12 define('AUTH_KEY',         's!un!.)?^)hNd.$!wp(ge SZ-t,g;c+F?!6~?dmtT+!Q
       }9Lq73u!(M)Hh<%LZ[J0');
13 define('SECURE_AUTH_KEY',  '|{r5>-Xc>7*{rk5EUC?k-<4%*cC6kHs;N=_k|
       v99n$@7Du[fsw6Sh@7+Ye7 }R|=');
14 define('LOGGED_IN_KEY',    '.)T3oAuIw7z.30^KD}xq5kBs-@_-}i/!+~22?FF5{/ZYx
       #GokMJCp0#;-!,:zkC+');
15 define('NONCE_KEY',        '(D]qhw5`Zq^3),~t:Sy5J+L|?9%e)in^`v3r%#Jp{
       V_Qp@u4$P<p-VN8j!s9e4+Z');
16 define('AUTH_SALT',        'GWnn40;o7TIo0A+~0-;_T{HB+<;@Ygn-)9kv6xmT aw{;
       u#J&%V}L-F?pII!+ ~T');
17 define('SECURE_AUTH_SALT', '$wk`-)[1n~7E]{*qEeXtm:Z!L ]DNWt4<--=AE: e(]_g
        (=C>B%*WU.+P~zcCot');
18 define('LOGGED_IN_SALT',   '9-b) /<TwXA2Y3~5M|STjC75JDvv7q)Uhp|_Z}5 b||-
       Vb{*8V=xJu|R4EXFyDrd');
19 define('NONCE_SALT',       'k-b11V+EwG3S<ZcE;gvwlY=a3I+.6~uI-ULN:- ^$!)J:
       @BDF+J|JXqn/UdM%v_y');
```

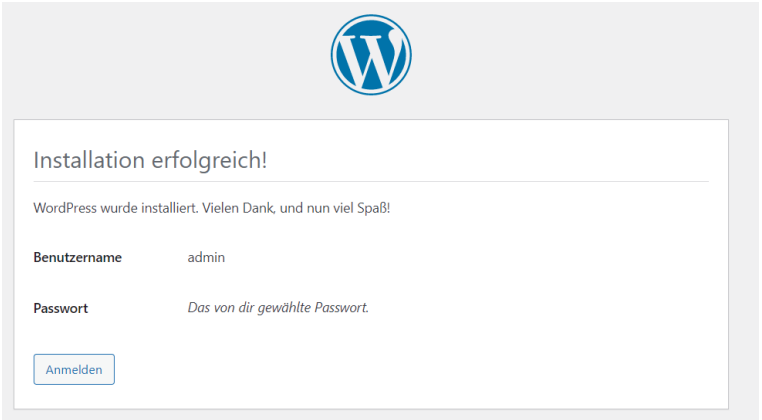
Sprachauswahl



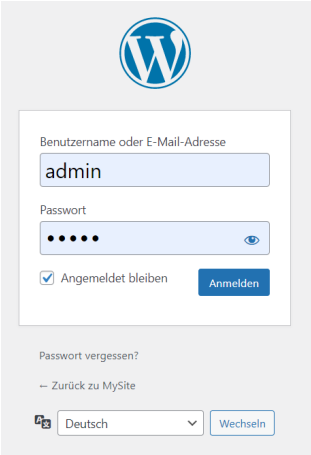
Persönliche Angaben



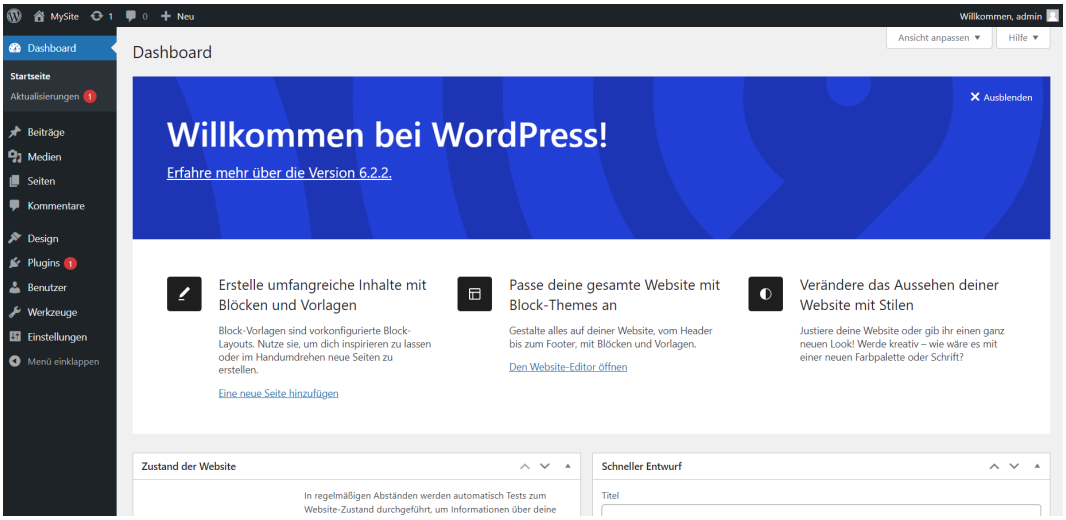
Installation erfolgreich



Anmelden

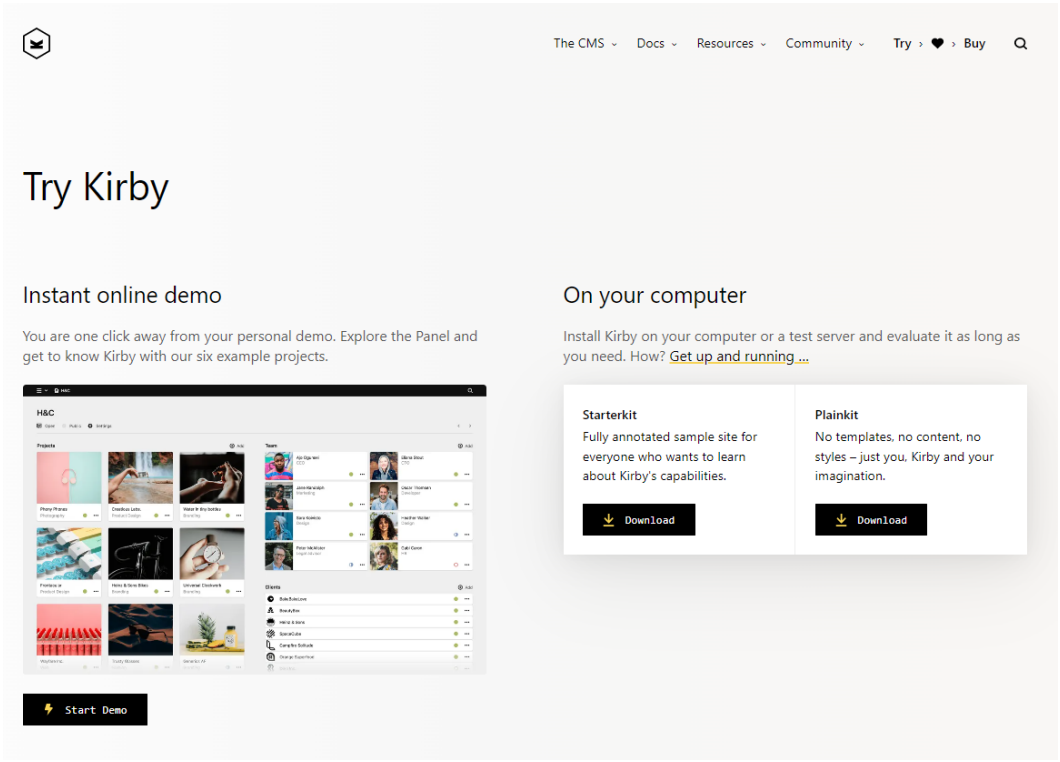


Dashboard

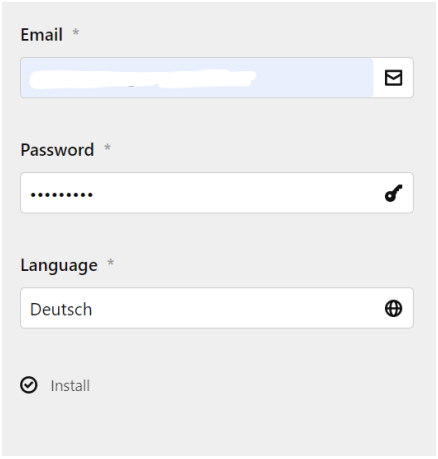


F.3 Kirby

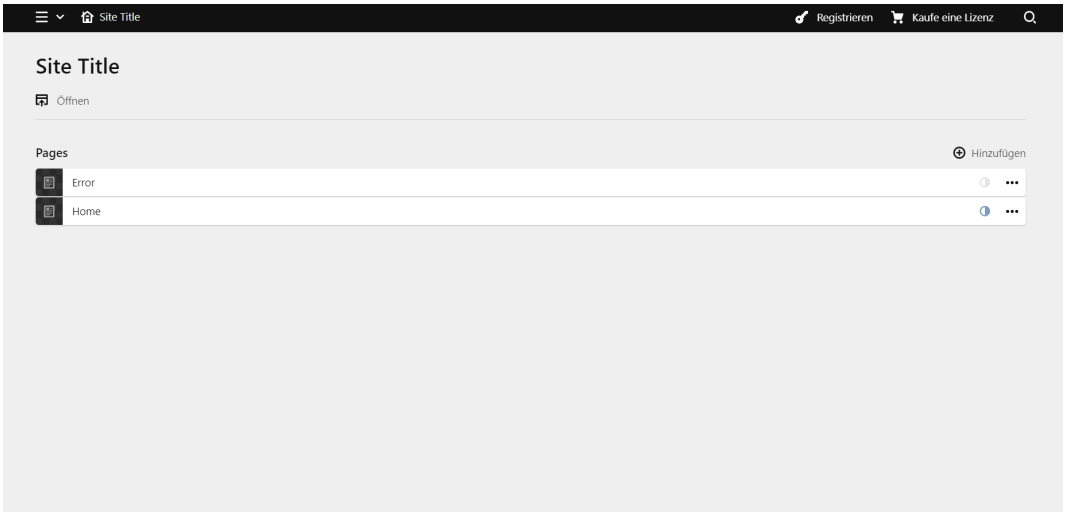
Download Kirby



Installation



Panel


Site Title
Registrieren
Kaufe eine Lizenz
Site Title
Öffnen
Pages
Hinzufügen
Error
Home

G Erstellung PostType WordPress

G.1 Listings

G.1.1 Code snippets

MU-Plugin

```
function create_posttype() {
    register_post_type( 'wppt_book', array(
        'labels' => array(
          'name' => 'Bücher',
          'singular_name' => 'Buch',
          'add_new_item' => 'Add New Book',
          'edit_item' => 'Edit Book',
          'all_items' => 'All Books',
          'item_published' => 'Book published',
          'item_updated' => 'Book updated',
        ),
        'public' => true,
        'has_archive' => true,
        'rewrite' => array('slug' => 'books'),
        'menu_icon' => 'dashicons-book',
        'supports' => array(
            'title', 'thumbnail',
        ),
        'capability_type' => 'post',
        'show_ui' => true,
      )
    );
  }
add_action( 'init', 'create_posttype' );
```

Plugin

```
function custom_buch_details_callback( $book ) {
    ?>
    <style>
        .book-element {
            margin-top: 25px; }

        label {
            font-weight: 600;
            margin-bottom: 5px;
            display:inline-block;
        }
        input {
            width: 100%;
        }
        textarea {
            width: 100%;
        }
    </style>
    <?php
    // Reihenname und Band
    $reihenname_und_band = get_post_meta( $book->ID, 'reihenname_und_band', true );
    echo '<div class="book-element"><label for="reihenname_und_band">Reihenname und Band </label><br>';
    echo '<input type="text" id="reihenname_und_band" name="reihenname_und_band" value="' . esc_attr( $reihenname_und_band ) . '" /><br></div>';

    // Untertitel/Genre
    $untertitel = get_post_meta( $book->ID, 'untertitel', true );
    echo '<div class="book-element"><label for="untertitel">Untertitel/Genre </label><br>';
    echo '<input type="text" id="untertitel" name="untertitel" value="' . esc_attr( $untertitel ) . '" /><br></div>';

    // Autor
    $autor = get_post_meta( $book->ID, 'autor', true );
    echo '<div class="book-element"><label for="autor">Autor </label><br>';
    echo '<input type="text" id="autor" name="autor" value="' . esc_attr( $autor ) . '" /><br></div>';

    // Klappentext
    $klappentext = get_post_meta( $book->ID, 'klappentext', true );
    echo '<div class="book-element"><label for="klappentext">Klappentext</label><br>';
```

```
38      echo '<textarea  type="text" id="klappentext " name="klappentext"
            rows="10" cols="103">'. esc_attr( $klappentext ) .'</textarea><br
            ></div>';
39
40      // Preis
41      $preis = get_post_meta( $book->ID, 'preis', true );
42      echo '<div class="book-element"><label for="preis">Preis in    </
            label><br>';
43      echo '<input type="number" id="preis" name="preis" min="0" step
            ="0.01" value="' . esc_attr( $preis ) . '" /><br></div>';
44
45      // Erscheinungsdatum
46      $erscheinungsdatum = get_post_meta( get_the_ID(), 'erscheinungsdatum'
            , true );
47      $convertedDate = date('Y-m-d', strtotime($erscheinungsdatum));
48      echo '<div class="book-element"><label for="erscheinungsdatum">
            Erscheinungsdatum </label><br>';
49      echo '<input type="date" id="erscheinungsdatum" name="
            erscheinungsdatum" value="' . esc_attr( $convertedDate ) . '" /><
            br></div>';
50
51      // Verlag
52      $verlag = get_post_meta( $book->ID, 'verlag', true );
53      echo '<div class="book-element"><label for="verlag">Verlag </label><
            br>';
54      echo '<input type="text" id="verlag" name="verlag" value="' .
            esc_attr( $verlag ) . '" /><br></div>';
55
56      // Seitenzahl
57      $seitenzahl = get_post_meta( $book->ID, 'seitenzahl', true );
58      echo '<div class="book-element"><label for="seitenzahl">Seitenzahl </
            label><br>';
59      echo '<input type="number" id="seitenzahl" name="seitenzahl" min="0"
            value="' . esc_attr( $seitenzahl ) . '" /><br></div>';
60
61      // EAN
62      $ean = get_post_meta( $book->ID, 'ean', true );
63      echo '<div class="book-element"><label for="ean">EAN </label><br>';
64      echo '<input type="number" id="ean" name="ean" min="0" value="' .
            esc_attr( $ean ) . '" /><br></div>';
65
66      // Dateigröße
67      $dateigrose = get_post_meta( $book->ID, 'dateigrose', true );
68      echo '<div class="book-element"><label for="dateigrose">Dateigröße in
             KB </label><br>';
69      echo '<input type="number" id="dateigrose" name="dateigrose" min="0"
            value="' . esc_attr( $dateigrose ) . '" /><br></div>';
70  }
```

```
1 function custom_save_buch_meta( $book_id ) {
2     if ( isset( $_POST['reihenname_und_band'] ) ) {
3         update_post_meta( $book_id, 'reihenname_und_band',
              sanitize_text_field( $_POST['reihenname_und_band'] ) );
4     }
5     if ( isset( $_POST['untertitel'] ) ) {
6         update_post_meta( $book_id, 'untertitel', sanitize_text_field(
              $_POST['untertitel'] ) );
7     }
8     if ( isset( $_POST['autor'] ) ) {
9         update_post_meta( $book_id, 'autor', sanitize_text_field( $_POST[
              'autor'] ) );
10    }
11    if ( isset( $_POST['klappentext'] ) ) {
12        update_post_meta( $book_id, 'klappentext', sanitize_text_field(
              $_POST['klappentext'] ) );
13    }
14    if ( isset( $_POST['preis'] ) ) {
15        update_post_meta( $book_id, 'preis', sanitize_text_field( $_POST[
              'preis'] ) );
16    }
17    if ( isset( $_POST['erscheinungsdatum'] ) ) {
18        update_post_meta( $book_id, 'erscheinungsdatum',
              sanitize_text_field( $_POST['erscheinungsdatum'] ) );
19    }
20    if ( isset( $_POST['verlag'] ) ) {
21        update_post_meta( $book_id, 'verlag', sanitize_text_field( $_POST
              ['verlag'] ) );
22    }
23    if ( isset( $_POST['seitenzahl'] ) ) {
24        update_post_meta( $book_id, 'seitenzahl', sanitize_text_field(
              $_POST['seitenzahl'] ) );
25    }
26    if ( isset( $_POST['ean'] ) ) {
27        update_post_meta( $book_id, 'ean', sanitize_text_field( $_POST['
              ean'] ) );
28    }
29    if ( isset( $_POST['dateigrose'] ) ) {
30        update_post_meta( $book_id, 'dateigrose', sanitize_text_field(
              $_POST['dateigrose'] ) );
31    }
32 }
33 add_action( 'save_post_buch', 'custom_save_buch_meta' );
```

Theme

```
if ($books_query->have_posts()) {
            while ($books_query->have_posts()) {
                $books_query->the_post();?>
                <li>
                <a href="<?= get_the_permalink(); ?>">
                    <figure>
                        <?php $thumbnail = wp_get_attachment_image_src(
                            get_post_thumbnail_id( $post->ID ) ); ?>
                        <img src="<?= $thumbnail[0]; ?>">
                        <figcaption>
                            <?= the_title(); ?>
                        </figcaption>
                    </figure>
                </a>
            </li>
                <?php
            }
            wp_reset_postdata();
        } else {
            echo 'Keine Bücher gefunden.';
        }
```

G.1.2 Dateien

wppt-book.php

```
<?php
/**
 * Plugin Name: Books
 * Description: This plugin will add a Custom Post Type for Books
 * Author: Chiara Funke
 * Version: 1
 */

 /*----------------------------------------*\
       Create Custom Post Type
 \*----------------------------------------*/
function create_posttype() {
  register_post_type( 'wppt_book', array(
    'labels' => array(
      'name' => 'Bücher',
      'singular_name' => 'Buch',
      'add_new_item' => 'Add New Book',
      'edit_item' => 'Edit Book',
      'all_items' => 'All Books',
```

```
20        'item_published' => 'Book published',
21        'item_updated' => 'Book updated',
22      ),
23      'public' => true,
24      'has_archive' => true,
25      'rewrite' => array('slug' => 'books'),
26      'menu_icon' => 'dashicons-book',
27      'supports' => array(
28        'title', 'thumbnail',
29      ),
30      'capability_type' => 'post',
31      'show_ui' => true,
32      )
33    );
34    }
35 add_action( 'init', 'create_posttype' );
36 add_theme_support('post-thumbnails');
```

books-plugin.php

```
1  <?php
2  /*
3  Plugin Name: Buch Plugin
4  Description: Ein Plugin für Feldern und Shortcodes für Bücher.
5  Author: Chiara Funke
6  Version: 1.0.0
7  */
8
9  /**
10  * Register meta boxes.
11  */
12 function custom_register_buch_meta() {
13     add_meta_box( 'buch_details', 'Buchdetails', '
           custom_buch_details_callback', 'wppt_book', 'normal', 'default' )
           ;
14 }
15 add_action( 'add_meta_boxes', 'custom_register_buch_meta' );
16
17 /**
18  * Meta box display callback.
19  *
20  * @param WP_Post $book Current book object.
21  */
22 function custom_buch_details_callback( $book ) {
23     ?>
24     <style>
25         .book-element {
26             margin-top: 25px; }
```

```
27
28          label {
29              font-weight: 600;
30              margin-bottom: 5px;
31              display:inline-block;
32          }
33          input {
34              width: 100%;
35          }
36          textarea {
37              width: 100%;
38          }
39      </style>
40      <?php
41      // Reihenname und Band
42      $reihenname_und_band = get_post_meta( $book->ID, 'reihenname_und_band
            ', true );
43      echo '<div class="book-element"><label for="reihenname_und_band">
            Reihenname und Band </label><br>';
44      echo '<input type="text" id="reihenname_und_band" name="
            reihenname_und_band" value="' . esc_attr( $reihenname_und_band )
            . '" /><br></div>';
45
46      // Untertitel/Genre
47      $untertitel = get_post_meta( $book->ID, 'untertitel', true );
48      echo '<div class="book-element"><label for="untertitel">Untertitel/
            Genre </label><br>';
49      echo '<input type="text" id="untertitel" name="untertitel" value="' .
             esc_attr( $untertitel ) . '" /><br></div>';
50
51      // Autor
52      $autor = get_post_meta( $book->ID, 'autor', true );
53      echo '<div class="book-element"><label for="autor">Autor </label><br>
            ';
54      echo '<input type="text" id="autor" name="autor" value="' . esc_attr(
             $autor ) . '" /><br></div>';
55
56      // Klappentext
57      $klappentext = get_post_meta( $book->ID, 'klappentext', true );
58      echo '<div class="book-element"><label for="klappentext">Klappentext
            </label><br>';
59      echo '<textarea  type="text" id="klappentext " name="klappentext"
            rows="10" cols="103">'. esc_attr( $klappentext ) .'</textarea><br
            ></div>';
60
61      // Preis
62      $preis = get_post_meta( $book->ID, 'preis', true );
63      echo '<div class="book-element"><label for="preis">Preis in    </
            label><br>';
```

```
64      echo '<input type="number" id="preis" name="preis" min="0" step
            ="0.01" value="' . esc_attr( $preis ) . '" /><br></div>';
65
66      // Erscheinungsdatum
67      $erscheinungsdatum = get_post_meta( get_the_ID(), 'erscheinungsdatum'
            , true );
68      $convertedDate = date('Y-m-d', strtotime($erscheinungsdatum));
69      echo '<div class="book-element"><label for="erscheinungsdatum">
            Erscheinungsdatum </label><br>';
70      echo '<input type="date" id="erscheinungsdatum" name="
            erscheinungsdatum" value="' . esc_attr( $convertedDate ) . '" /><
            br></div>';
71
72      // Verlag
73      $verlag = get_post_meta( $book->ID, 'verlag', true );
74      echo '<div class="book-element"><label for="verlag">Verlag </label><
            br>';
75      echo '<input type="text" id="verlag" name="verlag" value="' .
            esc_attr( $verlag ) . '" /><br></div>';
76
77      // Seitenzahl
78      $seitenzahl = get_post_meta( $book->ID, 'seitenzahl', true );
79      echo '<div class="book-element"><label for="seitenzahl">Seitenzahl </
            label><br>';
80      echo '<input type="number" id="seitenzahl" name="seitenzahl" min="0"
            value="' . esc_attr( $seitenzahl ) . '" /><br></div>';
81
82      // EAN
83      $ean = get_post_meta( $book->ID, 'ean', true );
84      echo '<div class="book-element"><label for="ean">EAN </label><br>';
85      echo '<input type="number" id="ean" name="ean" min="0" value="' .
            esc_attr( $ean ) . '" /><br></div>';
86
87      // Dateigröße
88      $dateigrose = get_post_meta( $book->ID, 'dateigrose', true );
89      echo '<div class="book-element"><label for="dateigrose">Dateigröße in
             KB </label><br>';
90      echo '<input type="number" id="dateigrose" name="dateigrose" min="0"
            value="' . esc_attr( $dateigrose ) . '" /><br></div>';
91  }
92
93  /**
94   * Save meta box content for each field.
95   *
96   * @param int $book_id Post ID (Book ID)
97   */
98  function custom_save_buch_meta( $book_id ) {
99      if ( isset( $_POST['reihenname_und_band'] ) ) {
100         update_post_meta( $book_id, 'reihenname_und_band',
                sanitize_text_field( $_POST['reihenname_und_band'] ) );
```

```
101     }
102     if ( isset( $_POST['untertitel'] ) ) {
103         update_post_meta( $book_id, 'untertitel', sanitize_text_field(
                $_POST['untertitel'] ) );
104     }
105     if ( isset( $_POST['autor'] ) ) {
106         update_post_meta( $book_id, 'autor', sanitize_text_field( $_POST[
                'autor'] ) );
107     }
108     if ( isset( $_POST['klappentext'] ) ) {
109         update_post_meta( $book_id, 'klappentext', sanitize_text_field(
                $_POST['klappentext'] ) );
110     }
111     if ( isset( $_POST['preis'] ) ) {
112         update_post_meta( $book_id, 'preis', sanitize_text_field( $_POST[
                'preis'] ) );
113     }
114     if ( isset( $_POST['erscheinungsdatum'] ) ) {
115         update_post_meta( $book_id, 'erscheinungsdatum',
                sanitize_text_field( $_POST['erscheinungsdatum'] ) );
116     }
117     if ( isset( $_POST['verlag'] ) ) {
118         update_post_meta( $book_id, 'verlag', sanitize_text_field( $_POST
                ['verlag'] ) );
119     }
120     if ( isset( $_POST['seitenzahl'] ) ) {
121         update_post_meta( $book_id, 'seitenzahl', sanitize_text_field(
                $_POST['seitenzahl'] ) );
122     }
123     if ( isset( $_POST['ean'] ) ) {
124         update_post_meta( $book_id, 'ean', sanitize_text_field( $_POST['
                ean'] ) );
125     }
126     if ( isset( $_POST['dateigrose'] ) ) {
127         update_post_meta( $book_id, 'dateigrose', sanitize_text_field(
                $_POST['dateigrose'] ) );
128     }
129 }
130 add_action( 'save_post_buch', 'custom_save_buch_meta' );
131
132  /**
133   * Register shortcode for the field reihenname_und_band
134   */
135 function custom_buch_reihenname_und_band_shortcode() {
136     $reihenname_und_band = get_post_meta( get_the_ID(), '
            reihenname_und_band', true );
137     if ( $reihenname_und_band ) {
138         return $reihenname_und_band;
139     }
140 }
```

```
141 add_shortcode( 'buch-reihenname_und_band', '
        custom_buch_reihenname_und_band_shortcode' );
142
143 /**
144  * Register shortcode for the field untertitel
145  */
146 function custom_buch_untertitel_shortcode() {
147     $untertitel = get_post_meta( get_the_ID(), 'untertitel', true );
148     if ( $untertitel ) {
149         return $untertitel;
150     }
151 }
152 add_shortcode( 'buch-untertitel', 'custom_buch_untertitel_shortcode' );
153
154 /**
155  * Register shortcode for the field autor
156  */
157 function custom_buch_autor_shortcode() {
158     $autor = get_post_meta( get_the_ID(), 'autor', true );
159     if ( $autor ) {
160         return $autor;
161     }
162 }
163 add_shortcode( 'buch-autor', 'custom_buch_autor_shortcode' );
164
165 /**
166  * Register shortcode for the field klappentext
167  */
168 function custom_buch_klappentext_shortcode() {
169     $klappentext = get_post_meta( get_the_ID(), 'klappentext', true );
170     if ( $klappentext ) {
171         return $klappentext;
172     }
173 }
174 add_shortcode( 'buch-klappentext', 'custom_buch_klappentext_shortcode' );
175
176 /**
177  * Register shortcode for the field preis
178  */
179 function custom_buch_preis_shortcode() {
180     $preis = get_post_meta( get_the_ID(), 'preis', true );
181     if ( $preis ) {
182         return $preis;
183     }
184 }
185 add_shortcode( 'buch-preis', 'custom_buch_preis_shortcode' );
186
187 /**
188  * Register shortcode for the field erscheinungsdatum (converted to d.m.Y
        )
```

```
189  */
190 function custom_buch_erscheinungsdatum_shortcode() {
191     $erscheinungsdatum = get_post_meta( get_the_ID(), 'erscheinungsdatum'
            , true );
192     $convertedDate = date('d.m.Y', strtotime($erscheinungsdatum));
193 
194     if ( $erscheinungsdatum ) {
195         return $convertedDate;
196     }
197 }
198 add_shortcode( 'buch-erscheinungsdatum', '
        custom_buch_erscheinungsdatum_shortcode' );
199 
200 /**
201  * Register shortcode for the field verlag
202  */
203 function custom_buch_verlag_shortcode() {
204     $verlag = get_post_meta( get_the_ID(), 'verlag', true );
205     if ( $verlag ) {
206         return $verlag;
207     }
208 }
209 add_shortcode( 'buch-verlag', 'custom_buch_verlag_shortcode' );
210 
211 /**
212  * Register shortcode for the field seitenzahl
213  */
214 function custom_buch_seitenzahl_shortcode() {
215     $seitenzahl = get_post_meta( get_the_ID(), 'seitenzahl', true );
216     if ( $seitenzahl ) {
217         return $seitenzahl;
218     }
219 }
220 add_shortcode( 'buch-seitenzahl', 'custom_buch_seitenzahl_shortcode' );
221 
222 /**
223  * Register shortcode for the field ean
224  */
225 function custom_buch_ean_shortcode() {
226     $ean = get_post_meta( get_the_ID(), 'ean', true );
227     if ( $ean ) {
228         return $ean;
229     }
230 }
231 add_shortcode( 'buch-ean', 'custom_buch_ean_shortcode' );
232 
233 /**
234  * Register shortcode for the field dateigrose
235  */
236 function custom_buch_dateigrose_shortcode() {
```

```
237     $dateigrose = get_post_meta( get_the_ID(), 'dateigrose', true );
238     if ( $dateigrose ) {
239         return $dateigrose;
240     }
241 }
242 add_shortcode( 'buch-dateigrose', 'custom_buch_dateigrose_shortcode' );
243 ?>
```

index.php

```
1 <?php get_header(); ?>
2 <main class="main">
3     <h1><?php the_title(); ?></h1>
4 </main>
5 <?php get_footer(); ?>
```

style.css

```
1 /*
2 Theme Name: Books
3 Description: Custom theme for displaying books.
4 Version: 1.0
5 Author: Chiara Funke
6 */
7
8 *,
9 *:after,
10 *:before {
11   margin: 0;
12   padding: 0;
13   box-sizing: border-box;
14 }
15
16 html {
17   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto,
         Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-
         serif
18 }
19
20 a {
21   color: currentColor;
22 }
23
24 body {
25   padding: 3rem;
26   display: flex;
```

```
27   flex-direction: column;
28 }
29
30 .header {
31   display: flex;
32   justify-content: space-between;
33   align-items: center;
34   margin-bottom: 3rem;
35 }
36
37 .header a {
38   text-decoration: none;
39 }
40
41 .logo {
42   font-weight: bold;
43 }
44 .menu ul {
45   list-style: none;
46   display: flex;
47 }
48
49 .menu ul li + li {
50   margin-left: 1.5rem;
51 }
52
53 .main {
54   flex-grow: 1;
55   margin-bottom: 3rem;
56 }
57
58 .main h1 {
59   margin-bottom: 1.5rem;
60 }
61
62 .books {
63   display: grid;
64   grid-template-columns: repeat(4, 1fr);
65   list-style: none;
66   grid-column-gap: 1.5rem;
67   grid-row-gap: 3rem;
68 }
69 .books img {
70   width: 100%;
71   margin-bottom: .5rem;
72 }
73
74 .pagination {
75   display: flex;
76   justify-content: space-between;
```

```
77   padding-top: 3rem;
78 }
79 .pagination span {
80   color: #999;
81 }
82
83 .filter {
84   display: flex;
85   align-items: center;
86   margin-bottom: 1.5rem;
87 }
88 .filter a {
89   padding: .5rem 1rem;
90   background: #000;
91   color: #fff;
92   margin-right: .5rem;
93   border-radius: 3px;
94   text-decoration: none;
95 }
96 .filter a[aria-current] {
97   background: blue;
98 }
```

header.php

```
1 <!doctype html>
2 <html <?php language_attributes(); ?>>
3 <head>
4   <meta charset="<?php bloginfo( 'charset' ); ?>">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title><?php wp_title(); ?></title>
7     <link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>">
8   <?php wp_head(); ?>
9 </head>
10 <body <?php body_class(); ?>>
11 <header class="header">
12     <div class="logo">
13         <span><?php bloginfo( 'name' ); ?></span>
14     </div>
15         <?php
16             $args = array(
17                 'theme_location' => 'primary-menu',
18                 'container' => false,
19             );
20             wp_nav_menu($args);
21         ?>
22 </header>
```

footer.php

```
<?php wp_footer(); ?>
</body>
</html>
```

front-page.php

```
<?php get_header(); ?>

<main class="main">
    <h1><?php the_title(); ?></h1>
</main>

<?php get_footer(); ?>
```

page.php

```
<?php get_header(); ?>

<main class="main">
    <h1><?php the_title(); ?></h1>

    <ul class="books">
        <?php
        $args = array(
            'post_type' => 'wppt_book', // Post Type "Bücher"
            'posts_per_page' => -1, // Alle Beiträge anzeigen
            'order' => 'ASC', // Sortierreihenfolge: Aufsteigend (älteste
                zuerst)
            'orderby' => 'date', // Nach Datum sortieren
        );

        $books_query = new WP_Query($args);

        if ($books_query->have_posts()) {
            while ($books_query->have_posts()) {
                $books_query->the_post();?>
                <li>
                <a href="<?= get_the_permalink(); ?>">
                    <figure>
                        <?php $thumbnail = wp_get_attachment_image_src(
                            get_post_thumbnail_id( $post->ID ) ); ?>
                        <img src="<?= $thumbnail[0]; ?>">
                        <figcaption>
                            <?= the_title(); ?>
```

```
                        </figcaption>
                    </figure>
                </a>
            </li>
                <?php
            }
            wp_reset_postdata();
        } else {
            echo 'Keine Bücher gefunden.';
        }
        ?>
    </ul>
</main>

<?php get_footer(); ?>
```

single.php

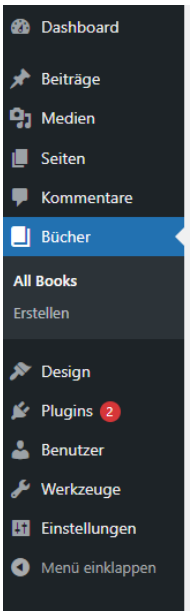
```
<?php get_header(); ?>

<style>
.main h1, h4 {
    margin-bottom: 0.5rem;
}
.main-info {
    margin-bottom: 10px;
}
.book-layout {
    display: grid;
    grid-template-columns: 1fr 2fr;
    grid-gap: 6rem;
}
.book-info {
    line-height: 1.5rem;
}
.book-info dt{
    font-weight: bold;
}
.book-text {
    margin-bottom: 1.5rem;
}
.book-cover ul{
    list-style: none; /* remove list bullet points */
}
.book-cover img{
    width: 100%; /*makes sure that the images don#t overflow*/
}
</style>
```

```
31 <main class="main">
32     <article>
33         <div class="main-info">
34             <?php echo do_shortcode("[buch-reihenname_und_band]") ?>
35         </div>
36
37         <h1><?= get_the_title(); ?></h1>
38
39         <h4>
40             <?php echo do_shortcode("[buch-untertitel]"); ?>
41         </h4>
42
43         <div class="main-info">
44             <?php echo do_shortcode("[buch-autor]") ?>
45         </div>
46
47         <div class="book-layout">
48             <div class="book-cover">
49                 <ul>
50                     <li>
51                     <?php $thumbnail = wp_get_attachment_image_src(
                           get_post_thumbnail_id( $post->ID ) ); ?>
52                         <img src="<?= $thumbnail[0]; ?>">
53                     </li>
54                 </ul>
55             </div>
56             <div class="book-info">
57                 <dl>
58                     <div class="book-text">
59                         <dt>Klappentext</dt>
60                         <dd><?php echo do_shortcode("[buch-klappentext]")
                               ?></dd>
61                     </div>
62
63                     <dt>Preis</dt>
64                     <dd><?php echo do_shortcode("[buch-preis]") ?>    </
                           dd>
65
66                     <dt>Erscheinungsdatum</dt>
67                     <dd><?php echo do_shortcode("[buch-erscheinungsdatum]
                           ") ?></dd>
68
69                     <dt>Verlag</dt>
70                     <dd><?php echo do_shortcode("[buch-verlag]") ?></dd>
71
72                     <dt>Seitenzahl</dt>
73                     <dd><?php echo do_shortcode("[buch-autor]") ?></dd>
74
75                     <dt>EAN</dt>
76                     <dd><?php echo do_shortcode("[buch-ean]") ?></dd>
```

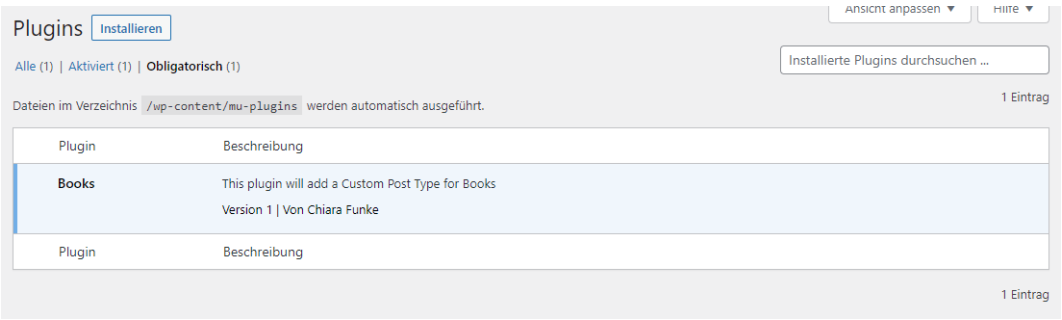
```
77
78                          <dt>Dateigröße</dt>
79                          <dd><?php echo do_shortcode("[buch-dateigrose]") ?>
                                KB</dd>
80                      </dl>
81                  </div>
82              </div>
83          </article>
84  </main>
85
86  <?php get_footer(); ?>
```

G.2 Bilder

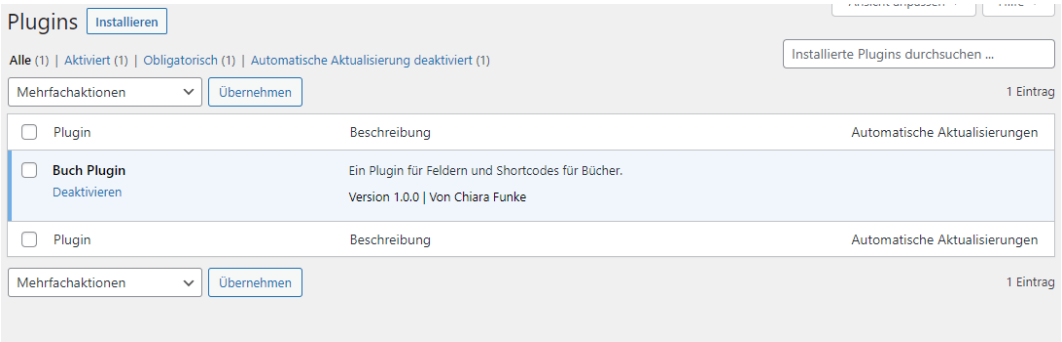
Admin-Oberfläche Bücher Post Type



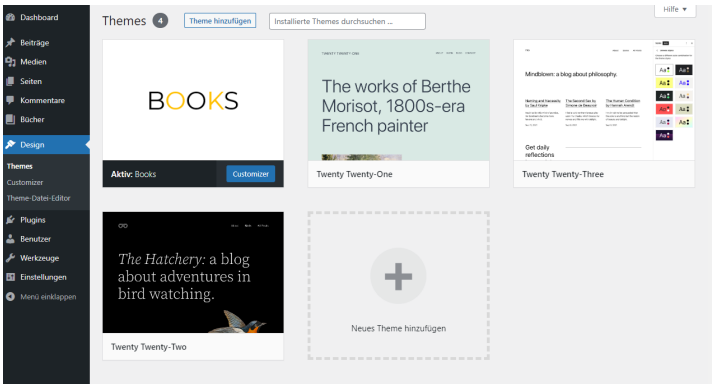
MustUse-Plugin Books



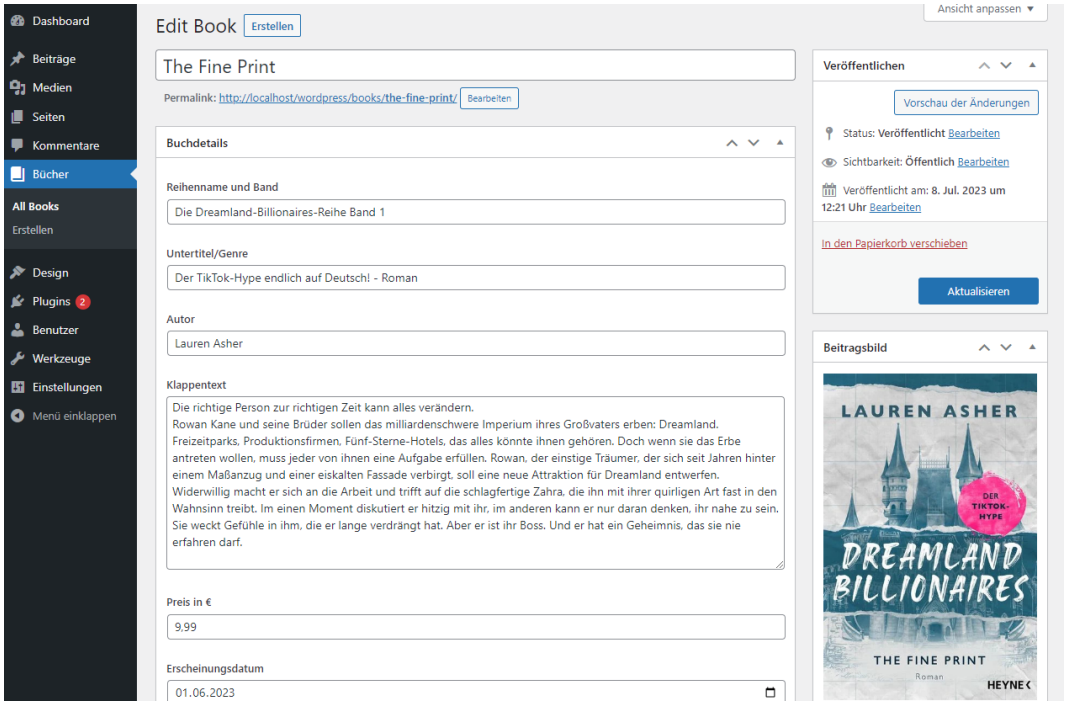
Plugin Buch Plugin



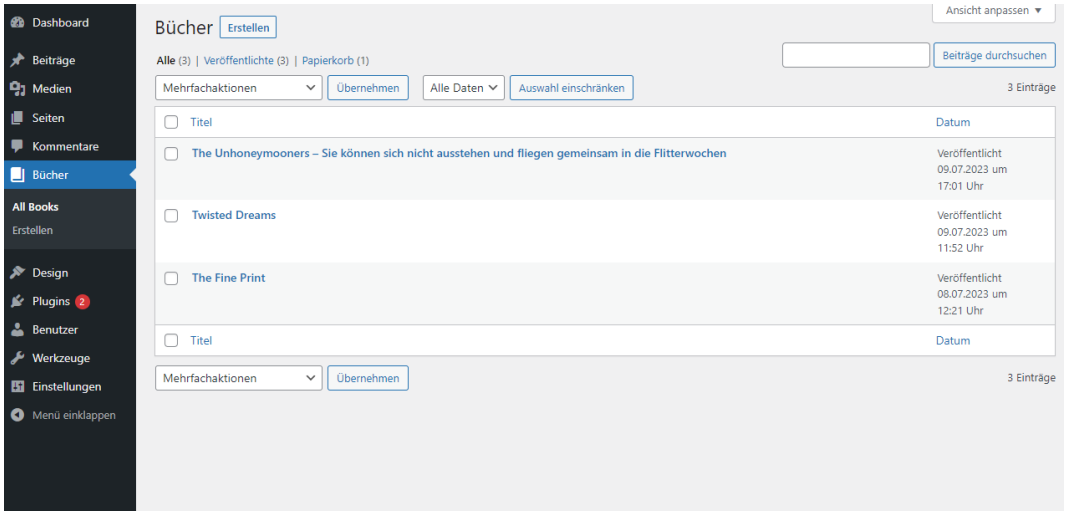
Theme



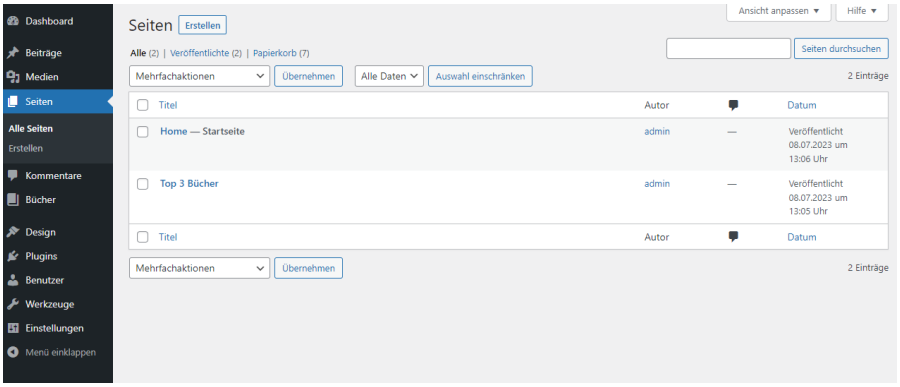
Felder ausfüllen Buch



Übersicht Bücher

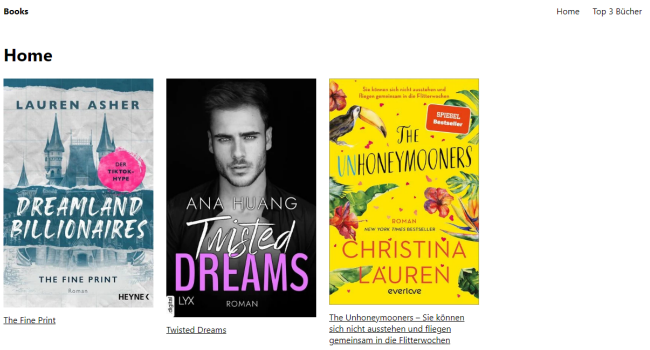


Seitenübersicht

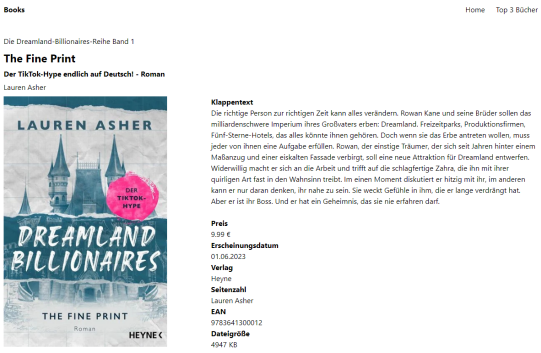


Home Seite

Buchübersichtseite



Buchdetailseite



H Erstellung PostType Kirby

H.1 Listings

H.1.1 Code snippets

```
<main class="main">
    <article>
        <?php if($page->reihe()->isNotEmpty()): ?>
        <div class="main-info">
            <?= $page->reihe() ?>
        </div>
        <?php endif ?>

        <h1><?= $page->title() ?></h1>

        <?php if($page->untertitel()->isNotEmpty()): ?>
            <h4><?= $page->untertitel() ?></h4>
        <?php endif ?>

        <?php if($page->autor()->isNotEmpty()): ?>
        <div class="main-info">
            <?= $page->autor() ?>
        </div>
        <?php endif ?>

        <div class="book-layout">
            <div class="book-cover">
                <ul>
                    <li>
                        <a href="<?= $page->image()->url()?>">
                            <?= $page->image()->resize(300, 400) ?>
                        </a>
                    </li>
                </ul>
            </div>
            <div class="book-info">
                <dl>
                    <?php if($page->klappentext()->isNotEmpty()): ?>
                    <div class="book-text">
                        <dt>Klappentext</dt>
```

```
36                          <dd><?= $page->klappentext() ?></dd>
37                      </div>
38                      <?php endif ?>
39
40                      <?php if($page->preis()->isNotEmpty()): ?>
41                      <dt>Preis</dt>
42                      <dd><?= $page->preis() ?>    </dd>
43                      <?php endif ?>
44
45                      <?php if($page->erscheinungsdatum()->isNotEmpty()):
                            ?>
46                      <dt>Erscheinungsdatum</dt>
47                      <dd><?= $page->erscheinungsdatum() ?></dd>
48                      <?php endif ?>
49
50                      <?php if($page->verlag()->isNotEmpty()): ?>
51                      <dt>Verlag</dt>
52                      <dd><?= $page->verlag() ?></dd>
53                      <?php endif ?>
54
55                      <?php if($page->seitenzahl()->isNotEmpty()): ?>
56                      <dt>Seitenzahl</dt>
57                      <dd><?= $page->seitenzahl() ?></dd>
58                      <?php endif ?>
59
60                      <?php if($page->ean()->isNotEmpty()): ?>
61                          <dt>EAN</dt>
62                          <dd><?= $page->ean() ?></dd>
63                      <?php endif ?>
64
65                      <?php if($page->dateigrosse()->isNotEmpty()): ?>
66                          <dt>Dateigröße</dt>
67                           <dd><?= $page->dateigrosse() ?> KB</dd>
68                      <?php endif ?>
69                  </dl>
70              </div>
71          </div>
72      </article>
73  </main>
```

```
1  main:
2      type: fields
3      fields:
4        reihe:
5          label: Reihe und Band
6          type: text
7        untertitel:
8          label: Untertitel/ Genre
9          type: text
10       autor:
```

```
11          label: Autor/-in
12          type: text
```

```
1 info:
2     type: fields
3     fields:
4       klappentext:
5         label: Klappentext
6         type: writer
7       preis:
8         label: Preis
9         type: number
10        step: .01
11      erscheinungsdatum:
12        label: Erscheinungsdatum
13        type: date
14        display: DD.MM.YYYY
15      verlag:
16        label: Verlag
17        type: text
18      seitenzahl:
19        label: Seitenzahl
20        type: number
21      ean:
22        label: EAN
23        type: number
24      dateigrosse:
25        label: Dateigröße
26        type: number
```

H.1.2 Dateien

header.php

```
1 <html lang="en">
2 <head>
3     <meta charset="UTF-8">
4     <meta name="viewport" content="width=device-width, initial-scale=1.0"
          >
5     <title><?= $site->title() ?></title>
6
7     <?= css('assets/css/index.css')?>
8 </head>
9 <body>
10
11 <header class="header">
12     <a class="logo" href="<?= $site->url() ?>"><?= $site->title() ?></a>
```

```
13
14      <nav class="menu">
15          <ul>
16          <?php foreach ($site->children()->listed() as $item): ?>
17          <li><a href="<?= $item->url()?>"><?= $item->title()?></a></li>
18          <?php endforeach ?>
19          </ul>
20      </nav>
21  </header>
```

footer.php

```
1  </body>
2  </html>
```

default.php

```
1  <?php snippet('header') ?>
2
3  <main class="main">
4      <h1><?= $page->title() ?></h1>
5  </main>
6
7  <?php snippet('footer') ?>
```

/assets/css/templates/book.css

```
1  .main h1, h4 {
2      margin-bottom: 0.5rem;
3  }
4
5  .main-info {
6      margin-bottom: 10px;
7  }
8
9  .book-layout {
10     display: grid;
11     grid-template-columns: 1fr 2fr;
12     grid-gap: 6rem;
13 }
14
15 .book-info {
16     line-height: 1.5rem;
17 }
```

```
18
19 .book-info dt{
20     font-weight: bold;
21 }
22
23 .book-text {
24     margin-bottom: 1.5rem;
25 }
26 .book-cover ul{
27     list-style: none; /* remove list bullet points */
28 }
29
30 .book-cover img{
31     width: 100%; /*makes sure that the images  d o n t  overflow*/
32 }
```

book.yml (Blueprint)

```
1 title: Book
2
3 columns:
4   - width: 1/1
5     sections:
6       main:
7           type: fields
8           fields:
9             reihe:
10              label: Reihe und Band
11              type: text
12            untertitel:
13              label: Untertitel/ Genre
14              type: text
15            autor:
16              label: Autor/-in
17              type: text
18   - width: 1/2
19     sections:
20       bookcover:
21           type: files
22           layout: cards
23           image:
24             ratio: 2/3
25             cover: true
26   - width: 1/2
27     sections:
28       info:
29         type: fields
30         fields:
```

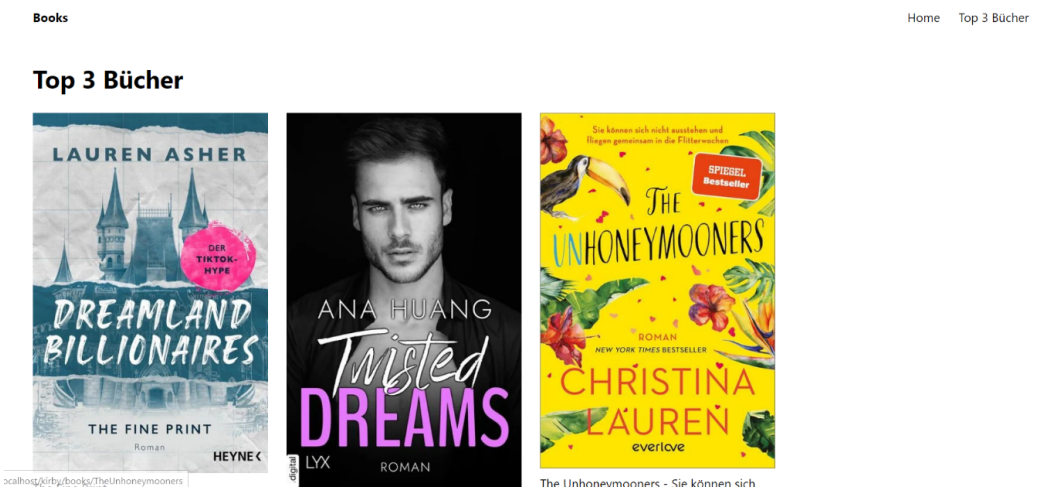
```
31            klappentext:
32              label: Klappentext
33              type: writer
34            preis:
35              label: Preis
36              type: number
37              step: .01
38            erscheinungsdatum:
39              label: Erscheinungsdatum
40              type: date
41              display: DD.MM.YYYY
42            verlag:
43              label: Verlag
44              type: text
45            seitenzahl:
46              label: Seitenzahl
47              type: number
48            ean:
49              label: EAN
50              type: number
51            dateigrosse:
52              label: Dateigröße
53              type: number
```

H.2 Bilder

Zwischenstand 1

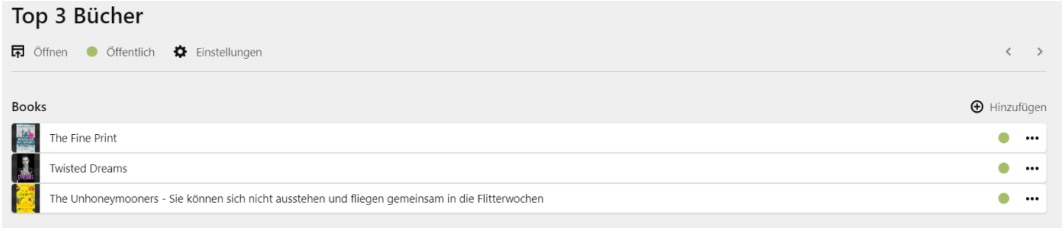
Home

Zwischenstand 2

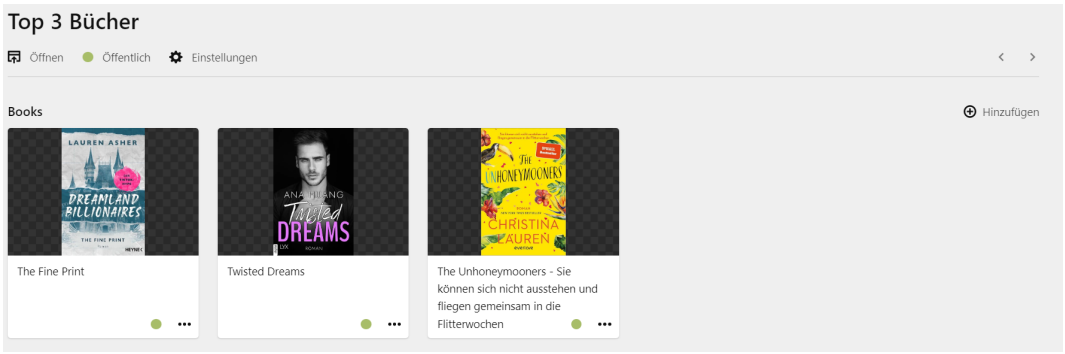


Zwischenstand Buch

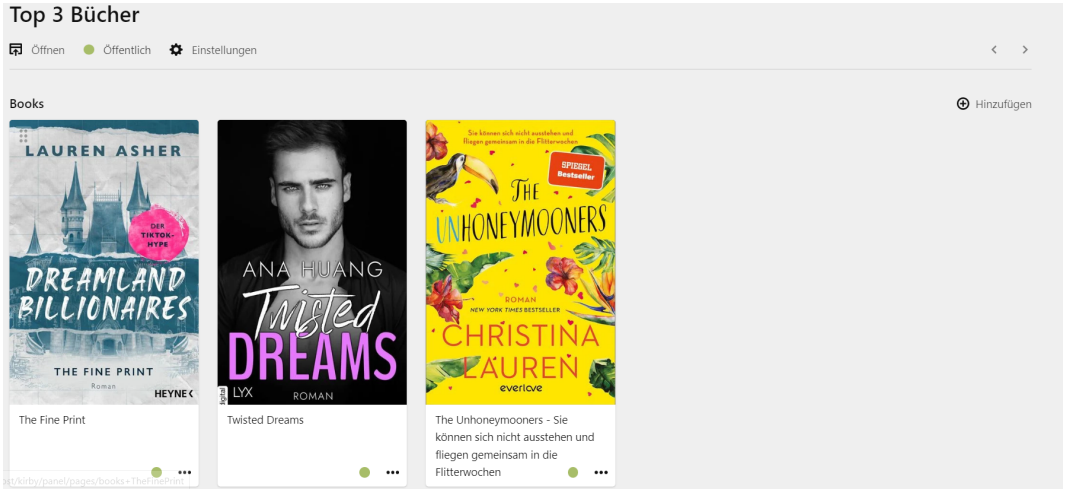
Zwischenstand Panel Übersicht 1



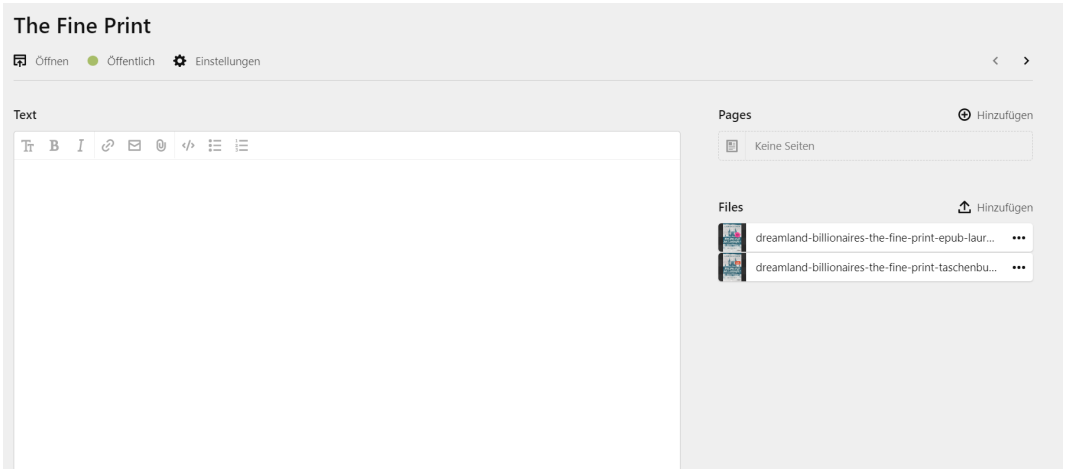
Zwischenstand Panel Übersicht 2



Zwischenstand Panel Übersicht 3

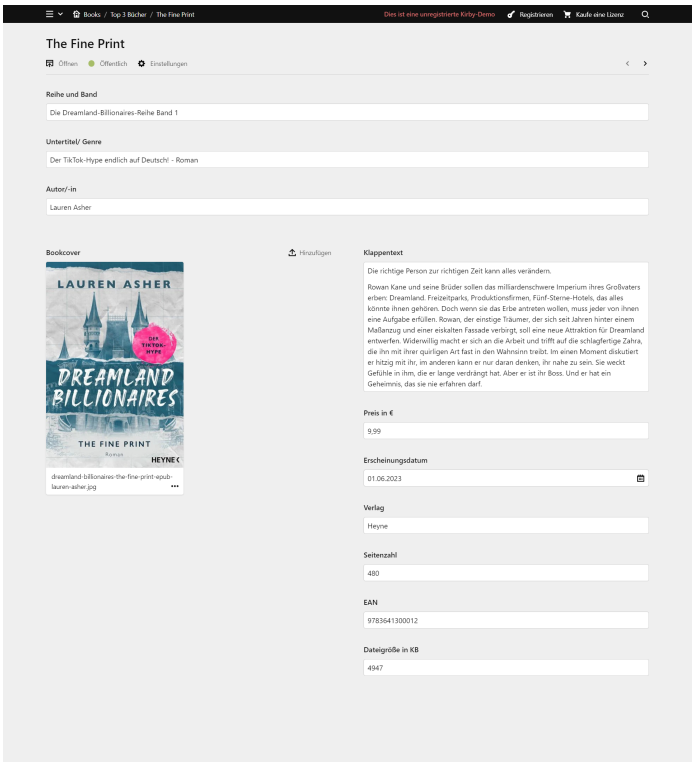


Vorher



Zwischenstand 3

Nachher



neues Buch anlegen