

Hybrides Rendering, Headless CMS & Progressive Web Apps

Analyse der Technologieansätze und Kombination in
einer prototypischen Anwendung

Bachelor Thesis

im Studiengang Medien und Informationswesen

Vorgelegt von

Tim Peters | 185056

Sommersemester 2022

an der Hochschule Offenburg - Fakultät Medien

Erstprüfer Prof. Dr. Dipl.-Ing. Roland Riempp

Zweitprüfer Prof. Dr. rer. nat. Katharina Mehner-Heindl

Abstract

Diese Thesis umfasst mit dem hybriden Rendering, Headless CMS und Progressive Web Apps drei moderne technologische Ansätze aus dem Bereich der Webentwicklung. Ziel der vorliegenden Arbeit ist es dabei, auf Basis eines Einblicks in die Grundlagen des modernen Web Developments zunächst die drei Konzepte einzeln zu betrachten, zu analysieren, Vor- und Nachteile herauszustellen und deren technologische Möglichkeiten zu erklären, um schließlich auch das Zusammenspiel dieser, derzeit relevanten Entwicklungsansätze im Hinblick auf die zugrunde liegende Forschungsfrage zu untersuchen:

Ist es möglich verschiedene, aktuelle Trends der Webentwicklung sinnvoll miteinander zu kombinieren und so auf Basis eines Headless CMS, einem Frontendframework für verschiedene Renderingmethoden und der Idee der Progressive Web Apps eine performante und benutzerfreundliche prototypische Webapplikation zu implementieren?

Um dies zu überprüfen, wurde auf den zuvor erarbeiteten theoretischen Grundlagen mit *MovieRec* eine WebApp als interaktives Filmportal konzipiert und schlussendlich mit einem entsprechenden Technologiestack implementiert. Diese Schritte sowie die Umsetzung der Kernfunktionen der Anwendung wurden dabei anhand von Codeausschnitten erklärt.

Schlagworte

Webentwicklung - Headless CMS - Server Side Rendering - Static Site Generation - Client Side Rendering - Progressive Web Apps - PWA - Strapi - Next.js - React- JavaScript - SEO - JAMStack

Gender Disclaimer

Das in dieser Arbeit gewählte generische Maskulinum bezieht sich zugleich auf die männliche, die weibliche und andere Geschlechteridentitäten. Zur besseren Lesbarkeit wird auf die Verwendung männlicher und weiblicher Sprachformen verzichtet. Alle Geschlechteridentitäten werden ausdrücklich mitgemeint, soweit die Aussagen dies erfordern.

Inhaltsverzeichnis

Abstract	i
Inhaltsverzeichnis	ii
Einleitung	1
1 Grundlagen der Webentwicklung	4
1.1 Aufbau und Kommunikation	5
1.1.1 Client-Server-Architektur	5
1.1.2 APIs	6
1.1.3 JSON.....	7
1.2 Frontend	9
1.2.1 HTML	9
1.2.2 CSS	10
1.2.3 JavaScript	11
1.2.4 JavaScript-Frameworks	13
1.3 Backend	19
1.3.1 Serverseitige Programmierung	19
1.3.2 Datenbanken	22
1.4 Technologiestacks	25
1.4.1 LAMP-Stack	25
1.4.2 Fullstack JavaScript	25
1.4.3 JAMStack	26
1.5 Suchmaschinenoptimierung	27
2 Rendering von Webanwendungen	32
2.1 Clientseitiges Rendering	33
2.2 Serverseitiges Rendering	34
2.3 Statische Generierung	36
2.4 Inkrementell Statische Regenerierung	38

2.5	Vergleichender Überblick	40
2.5.1	Datenaktualität	41
2.5.2	Suchmaschinenoptimierung	42
2.5.3	Performance	43
2.5.4	Build-Time	44
2.6	Technologieansätze	45
2.6.1	Next.js	46
2.6.2	Gatsby	47
2.6.3	Nuxt.js	49
2.6.4	Hugo	49
3	Content Management Systeme	52
3.1	Traditionelle CMS	53
3.2	Headless CMS	55
3.3	Vergleichender Überblick	59
3.4	Technologieansätze	60
3.4.1	Traditionelle CMS	61
3.4.2	Headless CMS	63
3.4.3	Hybride CMS	68
4	Progressive Web Apps	70
4.1	Abgrenzung zu nativen Apps	71
4.2	Entstehung	73
4.3	Eigenschaften	74
4.4	Technologischer Hintergrund	77
4.4.1	Service Worker	78
4.4.2	Web App Manifest	78
4.5	Status Quo und Zukunft von PWAs	80
4.5.1	Praxisbeispiele	80
4.5.2	Chancen	81
4.5.3	Einschränkungen	82

5 Konzeption der prototypischen Anwendung	84
5.1 Motivation und Idee	85
5.2 Anforderungsanalyse	85
5.2.1 Recherche	86
5.2.2 User Stories	87
5.3 Technologiestack	89
5.3.1 Frontend	89
5.3.2 Backend	93
5.4 User Interface	93
5.5 Datenmodell	96
5.6 Nutzerinteraktionen	98
6 Implementierung	102
6.1 Backend	103
6.1.1 Installation und Einrichtung	103
6.1.2 Erstellen der Content Types	104
6.1.3 API	107
6.2 Frontend	110
6.2.1 React State und Hooks	110
6.2.2 Installation und Einrichtung	111
6.2.3 Serverseitig gerenderte Filmübersicht	114
6.2.4 Filtern und Suchen	116
6.2.5 Statisch generierte Einzelseiten	117
6.2.6 Registrierung und Login	119
6.2.7 Filmlisten und Reviews	121
6.2.8 Clientseitig gerenderte Ratinginformationen	123
6.2.9 Serverseitig gerenderter Feed	124
6.2.10 Suchmaschinenoptimierung	125
6.2.11 Upgrade zur PWA	126
7 Fazit und Ausblick	128
7.1 Fazit	129
7.2 Ausblick	131

Literaturverzeichnis	135
Abbildungsverzeichnis	143
Codesnippets	147
Abkürzungsverzeichnis	149
Eidesstattliche Erklärung	150

Einleitung

„In the era of global digitalization, the winner is the one who is faster than others mastering new channels of interaction with the audience and offering its customers more convenient digital solutions.“

Um in der heutigen digitalen Welt wahrgenommen zu werden und erfolgreich zu sein, gestalten Unternehmen Internetauftritte, entwickeln mobile Apps, positionieren sich in sozialen Netzwerken oder nutzen moderne Gadgets wie VR-Brillen, Smartwatches und Internet-of-Things-Anwendungen. So reicht es mittlerweile nicht mehr aus, die vor fünf Jahren eingerichtete Baukasten-Webseite mit neuen Inhalten zu pflegen, sondern es wird, wie auch Informatiker Mikhail Raevskiy zitiert (*siehe oben*), durch gestiegene Anforderungen von Nutzern und Kunden, sowie die kontinuierliche und rapide Weiterentwicklung von Technologien, immer schwerer am Markt zu bestehen, ohne sich diesen anzunehmen.

Gerade auch durch die mittlerweile enorme Anzahl und Vielfalt an mobilen Endgeräten hat das Feld des Web Developments seit der Einführung des Internet im Jahre 1983 eine wichtigere Rolle denn je, was auf der anderen Seite in einer rasanten Weiterentwicklung an Technologieansätzen und immer neuen Trends resultiert, die sich oftmals schneller ändern, als sie eingesetzt werden können. Für Webentwickler kann es dabei schnell schwierig werden, in einem Meer von Begriffen wie Single Page Applications, Headless CMS, Static Site Generators, Micro Frontends, Serverless Architecture, Low-Code-Development oder PWAs den Überblick zu behalten und je nach Anwendungszweck geeignete Technologien zu finden oder diese passend miteinander zu kombinieren.

„Lass uns für die Entwicklung unserer neuen Webseite ein Headless CMS verwenden, diese sind gerade modern.“ Sätze wie diese lassen sich dabei oftmals von Firmen, CEOs oder Projektleitern hören, die solche Technologietrends aufgeschnappt haben und sich ohne Hintergrundwissen dafür entscheiden diese zu nutzen. Zwar ist es durchaus richtig, sich nicht vor neuen Möglichkeiten zu verschließen, es bedarf jedoch deutlich mehr Fachkenntnis in diesen Ansätzen, um Neuerungen sinnvoll einzusetzen, da sich mit Sicherheit nicht jede Technologie oder jedes Framework für jeden Anwendungsfall eignet und häufig Web-Trends nur aufgrund deren aktueller Popularität genutzt werden wollen.

Mit dem hybriden Rendering von Webanwendungen, Headless CMS und Progressive Web Apps habe ich in dieser Thesis drei wichtige und in den letzten Jahren weit verbreitete Technologietrends aus dem Feld des Web Developments herausgegriffen und diese zunächst einzeln analysiert, deren technologischen Hintergrund, Stärken und Schwächen sowie Einsatzgebiete herausgearbeitet. Um die drei Ansätze zu vertiefen, habe ich mir zum Ziel gesetzt, diese in einer prototypischen Anwendung miteinander zu vereinen, die Konzeption als auch die Implementierung der Web App zu beschreiben und mich dabei an den in der Praxis gängigen Standards und Vorgehensweisen bei der Umsetzung eines Softwareprojekts zu orientieren, um schließlich die einzelnen Technologien und vor allem deren Kombination miteinander zu evaluieren.

So soll auf der einen Seite ein Leitfaden für die Nutzung der oben genannten Ansätze geschaffen werden und die Leser bei der Entscheidung, wann und ob der Einsatz einer oder mehrerer dieser Technologietrends in ihrem eigenen möglichen Praxisprojekt Sinn macht, unterstützen. Zugleich bietet diese Thesis für mich selbst die Möglichkeit, mein Interesse im breiten Feld der Webentwicklung zu vertiefen und neue Technologien und Frameworks durch die Implementierung der prototypischen App zu erlernen.

Zusammen mit meinem Erstbetreuer Prof. Dr. Dipl.-Ing. Roland Riempp, der mich in ersten Besprechungen zu dem Thema der Thesis auf die Idee brachte, nicht nur eine sondern auch das Zusammenspiel mehrerer Technologien zu betrachten, wurde mit Hilfe dessen Einblicke und tiefgreifenden Kenntnisse im Bereich der Produktion und Konzeption multimedialer Medien und interaktiver Webanwendungen zunächst eine Gliederung erstellt und erste Ideen zur prototypischen Anwendung besprochen. Der Aufbau der Thesis wurde nun durch Zweitbetreuerin Prof. Dr. rer. nat. Katharina Mehner-Heindl durch ihr Fachwissen im Gebiet der Informatik und der Software-Entwicklung sinnvoll ergänzt. So konnte nach einer Recherche und dem Zusammentragen verschiedener Quellen wie Fachliteratur oder Blog-Artikeln mit dem Schreibprozess begonnen werden. Zudem arbeitete ich mich anhand von Tutorials, Videos und Dokumentationen in die Implementierung der Technologien ein und verfeinerte parallel zum Schreiben die Idee der Applikation und beschäftigte mich anschließend mit deren Umsetzung.

Ich wünsche viel Freude und neue Erkenntnisse beim Lesen dieser Bachelor-Thesis!

Grundlagen der Webent- wicklung

Bevor in den folgenden Kapiteln das Augenmerk auf neue Ansätze in der Webentwicklung gelegt wird, sollen hier zunächst die grundlegenden Methoden dargestellt und die elementaren Bestandteile betrachtet werden.

So bietet dieser Abschnitt eine Übersicht über die Architektur und den Aufbau von Webseiten sowie eine Erläuterung zu einigen wichtigen Grundbegriffe und für das spätere Verständnis relevanten Technologien von HTML über JavaScript-Frameworks bis hin zu verbreiteten Technologiestacks.

1.1 Aufbau und Kommunikation

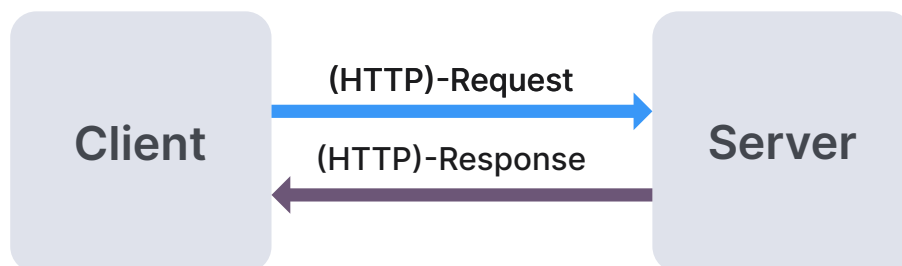
Ein elementares Prinzip der Funktionsweise von Webseiten oder Anwendungen im World Wide Web ist die Übertragung und der Austausch von Daten und Informationen. Da diese Konzepte nicht immer einfach zu verstehen sind und recht komplex sein können, werden im Folgenden die wichtigsten Modelle und Ansätze vorgestellt, die auch für die spätere Entwicklung der prototypischen Anwendung von Bedeutung sind.

1.1.1 Client-Server-Architektur

Klassischerweise setzen sich Webseiten und auch Webanwendungen aus einem Part, der auf der Clientseite ausgeführt wird, oftmals auch Frontend genannt, und einem Gegenpart, dem Backend, das serverseitig läuft, zusammen. Clients und Server teilen sich in dieser Architektur Aufgaben und Funktionen, die das Gesamtsystem erledigen muss. Dabei stellt auf der Serverseite ein Webserver die Webseite zur Verfügung, auf der Clientseite greifen ein oder mehrere Clients, typischerweise in Form eines Browsers, auf diese Inhalte zu.¹

Die Kommunikation beim Surfen im World Wide Web findet über das Protokoll HTTP (*Hypertext Transfer Protocol*) bzw. HTTPS statt. Hierbei stellt der Browser einen HTTP-Request an den entsprechenden Webserver, erreichbar über seine IP-Adresse. Dieser nimmt die Anfrage entgegen und generiert die zugehörige Antwort, die als HTTP-Response in Form der gewünschten Website zurück an den anfragenden Client gesendet wird und deren Quelltext vom Browser interpretiert und dem Nutzer grafisch dargestellt wird.²

Weitere verbreitete Anwendungen dieser Architektur sind neben dem Zugriff auf Internetseiten die Abwicklung des Nachrichtenaustauschs über E-Mails, der Transfer von Dateien oder auch das Auflösen von Domainnamen in IP-Adressen.²



◀ Abbildung 1

Kommunikation zwischen Client und Server

¹ (Ackermann, 2021)

² (Luber & Donner, 2020)

1.1.2 APIs

Application Programming Interfaces, kurz APIs, sind Programmierschnittstellen, die sich grundsätzlich als Vermittler zwischen Nutzern oder Clients und den von ihnen benötigten Ressourcen oder Diensten verstehen lassen. Möchten Nutzer mit einer Plattform oder einem System interagieren und dabei Informationen abrufen oder Funktionen ausführen, stellt ein API sicher, dass diese mit dem System so kommunizieren, dass entsprechende Anforderungen verstanden und erfüllt werden können.³

Im Hinblick auf die Entwicklung von Webanwendungen fällt in den letzten Jahren häufig der Begriff der RESTful-API bzw. der REST-API. Dieser steht kurz für *Representational State Transfer* und bezeichnet einen Architekturstil für Schnittstellen, der den Austausch von Informationen, verteilt auf verschiedenen Systemen, ermöglicht und somit die Kommunikation von web-basierten Applikationen unterstützt.⁴

Damit ein Application Programming Interface als *RESTful* gilt, müssen folgende Bedingungen erfüllt sein:

- Einheitliche Schnittstelle, d.h. der REST-Client kann den Server immer auf die gleiche Weise aufrufen und auf dessen Ressourcen zugreifen, unabhängig von Anwendung oder Endgerät.⁵
- Client-Server-Architektur, d.h. Client und Server sind klar voneinander abgegrenzt und unabhängig voneinander.³
- Zustandslos, d.h. der Client stellt eine Anfrage, die API führt die in der Anfrage definierte Aktion durch und antwortet. Nachdem die API geantwortet hat, gibt sie die Verbindung auf und behält keine Informationen mehr über den Client im aktiven Speicher.³
- Caching, d.h. es kann angegeben werden, ob bestimmte Ressourcen zwischengespeichert werden sollen.⁵
- Mehrschichtiges System, d.h. wenn ein Client eine Anfrage auf einen URL-Endpunkt stellt, muss er nicht genau wissen, welcher Server seine Anfrage bearbeitet. Ein mehrschichtiges System macht eine API sicherer und skalierbarer.⁵

³ (RedHat, 2020)

⁴ (Talend, kein Datum)

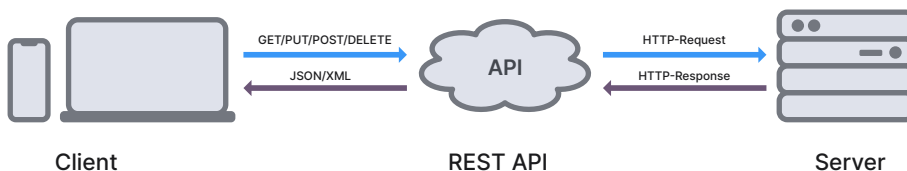
⁵ (Uptrends, kein Datum)

Für die Umsetzung dieser Prinzipien wird das zustandslose Client-Server-Protokoll HTTP eingesetzt, um mit jeweiligen Requests und Responses Daten zwischen Nutzern oder Apps auszutauschen. Die Ressourcen selbst sind dabei über eine URL eindeutig identifiziert. Für die verschiedenen Schreib- oder Leseaktionen werden nun HTTP-Methoden genutzt.

Die vier bedeutendsten Methoden, um die sogenannten *CRUD*-Operationen, d.h. Create, Read, Update und Delete, auf Ressourcen durchzuführen, sind dabei:¹

- GET, um Ressourcen abzurufen.
- POST, um neue Inhalte zu erstellen.
- PUT, um Ressourcen zu aktualisieren.
- DELETE, um Inhalte zu löschen.

Stellt ein Client nun beispielsweise eine gültige HTTP-GET-Anfrage an die URL `http://myshop.com/api/smartphones`, bekommt dieser in der Antwort alle Ressourcen zurückgegeben. Typischerweise überträgt REST die Daten heutzutage als JSON- (siehe 1.1.3) oder XML-Syntax. Seltener kommen auch andere Formate oder Protokolle zum Einsatz.¹



◀ Abbildung 2

Austausch von Daten mittels REST

1.1.3 JSON

JSON steht für *JavaScript Object Notation* und bezeichnet ein standardisiertes Format, um Daten zu übertragen oder zu speichern. Wie der Name schon andeutet, lässt sich eine Verbindung zwischen der Skriptsprache und JSON erkennen. Zwar ist das Darstellungsformat von der Definition der JavaScript-Objekte abgeleitet, es ist aber bei weitem nicht nur auf die Programmiersprache beschränkt, sondern mittlerweile ein weit verbreiteter Standard, um beispielsweise Informationen über eine API wie REST auszutauschen.⁶

¹ (Ackermann, 2021)

⁶ (Augusten, 2018)

Ein Anwendungsfall für die Nutzung dieses Datenformats wäre beispielsweise der Aufruf eines Produkts in einem Webshop, bei dem die Artikelinformationen über eine API aus der Datenbank abgerufen, in JSON konvertiert und anschließend zur Darstellung auf der Seite an den Client gesendet werden.

Codesnippet 1 ►

Darstellung von Daten im JSON-Format

```

1 {
2   "name": "Galaxy S 22",
3   "marke": "Samsung",
4   "preis": 849,
5   "rabatt": false,
6   "features": ["256GB Speicher", "Octacore Prozessor", "8GB RAM"],
7   "display": {
8     "größe": 6.1,
9     "auflösung": "2341x1080 Pixel",
10    "touchscreen": true,
11    "bildwiederholrate": 120
12  },
13  "verfügbar": 73
14 }
```

Wie sich in dem Beispiel erkennen lässt, bestehen JSON- Objekte, hier die Informationen über ein Produkt, aus Key-Value-Paaren. Die Keys auf der linken Seite stehen immer in Anführungszeichen, während dies bei den zugehörigen Werten nur bei Zeichenketten der Fall ist. In diesem Beispiel wäre zum Key *name* der entsprechende Value der String *Galaxy S22*.

Key-Value Paare können folgenden Datentypen enthalten:⁷

- String
- Number (keine Unterscheidung zwischen Ganz- u. Gleitkommazahlen)
- Boolean
- Null

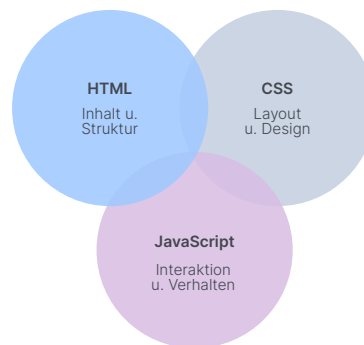
Neben den Schlüssel-Wert-Kombinationen ist es zudem möglich, diese wie *feautres* mit eckigen Klammern in Listen bzw. Arrays anzuordnen oder über geschweifte Klammern ein weiteres JSON-Objekt, in diesem Beispiel *display*, darzustellen. Diese drei Optionen lassen sich auch beliebig ineinander schachteln, sodass unter anderem Arrays Objekte und umgekehrt Objekte mehrere Listen oder andere Objekte enthalten können.⁷

⁷ (Sverkis, Percival & van Putten, 2021)

1.2 Frontend

Unter dem Begriff Frontend lässt sich die Präsentationsschicht einer Website oder Anwendung beschreiben. Das Frontend umfasst dabei den Teil der Applikation, den der Nutzer sehen und mit dem er interagieren kann.⁸ Das Frontend befindet sich daher grundsätzlich auf der Clientseite, wobei die verschiedenen eingesetzten Technologien vom Interpreter des Webbrowsers übersetzt und dem User in Form einer graphischen Benutzeroberfläche dargestellt werden. Daten und Prozesse des Backends werden durch die UI benutzerfreundlich visualisiert.⁹

Die drei wichtigsten Technologien der clientseitigen Webentwicklung sind dabei HTML, CSS und JavaScript.



◀ Abbildung 3

Grundlegende Bestandteile und Technologien des Frontends

1.2.1 HTML

Bei der *Hypertext Markup Language*, kurz HTML, handelt es sich um eine textbasierte Auszeichnungssprache zur Strukturierung von Webseiten. Die 1992 veröffentlichte und seither stetig weiterentwickelte Sprache bildet den Grundbaustein des Webs und wird von allen Browsern genutzt, um die vom Server abgerufenen Seiten für den Nutzer zu interpretieren. Im Gegensatz zu einer tatsächlichen Programmiersprache lässt sich damit jedoch keine Logik mit Schleifen, Bedingungen oder Variablen implementieren.⁹

Die einzelnen HTML-Elemente bestehen aus sogenannten Tags. Dabei umschließen in der Regel immer zwei Tags den Inhalt eines Elements, ein öffnender Tag `<` und ein schließender Tag `>`. Jeder Tag hat, abhängig vom Namen, eine andere Bedeutung, wodurch sich beispielsweise Überschriften (`<h>`), Bilder (`<img>`), Listen (`<li>`), Buttons (`<button>`), Hyperlinks (`<a>`) und viele weitere Bestandteile definieren lassen. Über Attribute können den jeweiligen Elementen zusätzliche Informationen übergeben werden. Diese stehen immer im öffnenden Tag eines Elements und sind aus einem spezifischen Namen und dem zugehörigen Wert zusammengesetzt, der in Anführungszeichen steht.¹

¹ (Ackermann, 2021)

⁸ (Frontend GmbH, kein Datum)

⁹ (Geißler & Ostler, 2018)

Abbildung 4 ►
Aufbau eines HTML-Elements



Ein HTML-Dokument weist eine hierarchische Baumstruktur auf und ist grundlegend in einen Head und einen Body aufgeteilt. Der erste Teil enthält Metainformationen für den Browser oder Suchmaschinen und ist nicht direkt für den User sichtbar, während der Body den auf dem Bildschirm sichtbaren Inhalt umfasst.¹⁰

1.2.2 CSS

Cascading Stylesheets, kurz CSS, ist eine Auszeichnungssprache, mit der sich das Aussehen der HTML-Dokumente beeinflussen lässt. Während mit Hilfe der HTML-Tags die einzelnen Inhalte auf einer Webseite bestimmt werden können, ermöglicht CSS, diese zu gestalten und auf der Präsentationsfläche anzuordnen. Um so das User Interface und die dargestellten Inhalte anzupassen und zu layouten, können verschiedene Regeln definiert werden. Über Selektoren lassen sich dann einzelne oder mehrere Elemente bestimmen, für die diese Gestaltungsregeln gelten sollen. Wie der Name der Stylesheet-Sprache schon erkennen lässt, sind die Selektoren hierarchisch aufgebaut und können auch ineinander verschachtelt werden.¹¹

Die einfachste Möglichkeit, ein Element in CSS anzusprechen, ist es, den Tag-Namen der jeweiligen HTML-Elemente zu nutzen. Zusätzlich können Elemente mit einem Punkt und der entsprechenden Bezeichnung (*.example*) über ihr Klassenattribut angesprochen werden. Da Elemente über mehr Klassen verfügen sowie eine Klasse zu verschiedenen Elementen gehören kann, ist es möglich, definierte Stile gezielt mehreren Komponenten auf der Webseite zuzuweisen. Der ID-Selektor (*#example*) gilt hingegen nur für ein einziges Element.¹²

¹⁰ (Mardan, 2018)

¹¹ (Northwood, 2018)

¹² (Dowden & Dowden, 2020)

Um schließlich die ausgewählten Elemente zu stylen oder auf dem Screen anzuordnen, werden in geschweiften Klammern entsprechende Regeln bzw. Deklarationen definiert, die aus mehreren Eigenschaften, auch *Properties* genannt, und zugehörigen Werten wie Farben, Einheiten oder Layoutbefehlen bestehen und durch ein Semikolon voneinander getrennt werden.¹²

In diesem Beispiel werden über einen verschachtelten Selektor alle *p*-Elemente, die sich innerhalb von Elementen mit der Klasse *textblock* befinden, ausgewählt und über zwei Deklarationen die Schriftgröße auf 12px festgelegt sowie der Hintergrund der Komponenten weiß eingefärbt.

```
.textblock p {
  font-size: 12px;
  background-color: white;
}
```

◀ Abbildung 5

Definition von CSS-Regeln

Bei kleineren Projekten können die Stylesheets über einen Tag direkt in den Head der HTML-Seite geschrieben werden. Gerade bei komplexen Webseiten macht es allerdings Sinn, die Sheets separat zu definieren und als externe CSS-Datei einzubinden.⁹

Um die Nutzung der Auszeichnungssprache für Entwickler einfacher und komfortabler zu machen, sind mittlerweile diverse CSS-Präprozessoren wie *SASS*, *LESS* oder *Stylus* zu nutzen. Diese bieten Möglichkeiten wie Variablen, Vererbung, mathematische Funktionen oder eine Verschachtlung der Selektoren und ermöglichen es so, den Code übersichtlicher zu halten, Fehler zu reduzieren und einzelne, wiederverwendbare Snippets zu erstellen. Für die Darstellung der Inhalte wird die jeweilige spezifische Syntax des Präprozessors wieder in herkömmliches CSS kompiliert, das dann vom Browser interpretiert werden kann.¹³

```
1 $base-color: #758c8b;
2
3 .textblock{
4   background-color: $base-color;
5   .paragraph{
6     font-size: 12px;
7   }
8 }
```

◀ Codesnippet 2

Syntax des CSS-Präprozessors SASS mit einer Variable

1.2.3 JavaScript

Da es sich bei HTML und CSS lediglich um Auszeichnungs- und keine Programmiersprachen handelt, wird JavaScript eingesetzt, um Anwendungslogik in der Webentwicklung zu implementieren und die zuvor statischen Seiten interaktiv zu machen.¹

¹ (Ackermann, 2021)

⁹ (Geißler & Ostler, 2018)

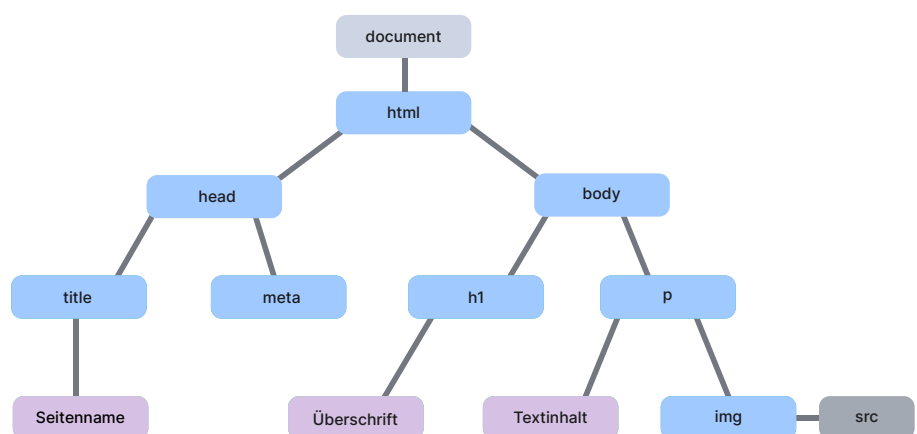
¹² (Dowden & Dowden, 2020)

¹³ (Monus, 2018)

Das heutige JavaScript wurde 1995 von dem damaligen Browserhersteller Netscape unter der ursprünglichen Bezeichnung *LiveScript* ins Leben gerufen, kurz darauf jedoch aus Vermarktungsgründen, angelehnt an die zur damaligen Zeit sehr populäre Programmiersprache Java, umbenannt. Bis auf die Ähnlichkeit der Namen haben diese beiden jedoch kaum etwas miteinander gemeinsam. So wird JavaScript von einer Browser-engine wie Google Chromes V8 interpretiert, Java hingegen von einer virtuellen Maschine kompiliert und ausgeführt und es gibt zudem deutliche Unterschiede in Syntax, Typisierung und Verwendungszweck. Heutzutage ist JavaScript unter dem Sprachkern *EcmaScript* standardisiert, weitverbreitet und wird unter anderem auch für die Entwicklung von Desktop-Applikationen, serverseitiger Programmierung oder Internet-of-Things-Anwendungen eingesetzt. Hauptanwendungsfall ist jedoch weiterhin die Möglichkeit, auf Benutzerevents und Eingaben zu reagieren und Inhalte im HTML dynamisch zu verändern.^{9 10}

Dabei dient das *Document Object Model*, kurz DOM, als Schnittstelle zwischen JavaScript und HTML. Zweck des DOM ist es, Zugriff auf die einzelnen Komponenten zu gewähren, um damit das Löschen, Hinzufügen oder Bearbeiten von Elementen, Attributen und Styles möglich zu machen. Um die hierarchische Struktur im Document Object Model korrekt wiederzugeben, greift die Schnittstelle dazu auf die bereits beschriebene Baumstruktur des HTML-Dokuments zurück, wodurch die auf dem Bildschirm gerenderten Objekte angesprochen werden können.¹⁴

Abbildung 6 ►
Beispielhafte Darstellung
der Knoten des Document
Object Models



9 (Geißler & Ostler, 2018))

10 (Mardan, 2018)

14 (IONOS, 2020)

In der Regel sind JavaScript Anwendungen im Browser eventbasiert und warten auf das Eintreten bestimmter externer Ereignisse, anstatt dauerhaft komplexe Berechnungen im Hintergrund auszuführen.¹⁰ Solche Events entstehen ab dem Laden der Seite und können vom Browser erfasst werden. Diese reichen vom Scrollen über Klicks, Mausbewegungen oder dem Drücken von Tasten bis hin zum Verlassen der Seite. Um solche Vorgänge und Nutzerinteraktionen zu erkennen und das anschließende Verhalten zu programmieren, gibt es in JavaScript vorgefertigte Event Handler. Die Reaktion auf ein Ereignis für bestimmte HTML-Elemente kann dann über spezielle Schlüsselwörter wie *onclick* für Mausklicks, *onscroll* für Scrollereignisse, *onmousemove* bei Mausbewegungen und vielen weiteren registriert werden.¹⁵

In dem unten gezeigten Code-Ausschnitt wird zunächst mit *document.getElementById* ein bestimmtes HTML-Element im DOM mit der ID *button* selektiert und ein Event Listener hinzugefügt. Das Schlüsselwort *click* gibt dabei an, dass bei einem Klick auf den Button eine übergebene Funktion ausgeführt wird, die eine Benachrichtigung mit dem Text *Hello World* öffnet.



```
1 document.getElementById("button").addEventListener("click",
2     function(){
3         alert("Hello World!");
4     }
5 );
6
```

◀ Codesnippet 3

Hinzufügen eines Event-Listeners mit einer Funktion in JavaScript

1.2.3 JavaScript-Frameworks

Nach dem Durchbruch von JavaScript gegen Ende der 90er Jahre verwandelte sich das Web zunehmend von einem Ort, um ausschließlich statische Informationen abzurufen, zu einem, um auch selbst Dinge zu tun. Die Bekanntheit von JavaScript nahm in den darauffolgenden Jahren stark zu und immer mehr Entwickler schrieben kleine, hilfreiche Tools in wiederverwendbaren Paketen.¹⁶

¹⁰ (Mardan, 2018)

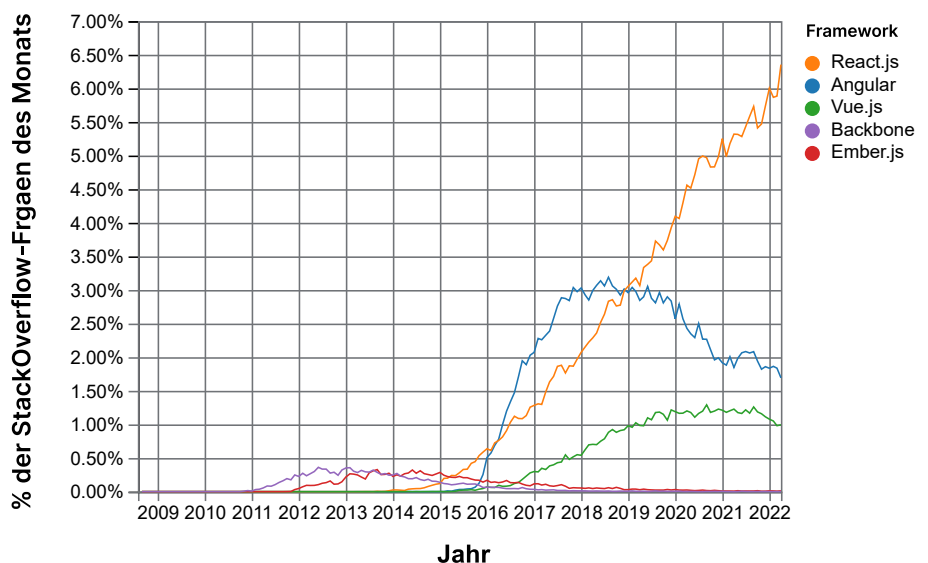
¹⁵ (Mediaevents, 2021)

¹⁶ (MDN Web Docs, 2021)

Diese sogenannten Bibliotheken dienten dazu, Lösungen zu bestimmten Problemen mit anderen Entwicklern zu teilen, wodurch sich im Laufe der Jahre ein großes Ökosystem entwickelte, das die Entwicklung des Webs maßgeblich beeinflusste. Aus den losen Ansammlungen von verschiedenen Funktionen in Bibliotheken entstanden schließlich feste Programmiergerüste, sogenannte Frameworks, die es den Entwicklern erleichtern sollten, moderne und interaktive Webanwendungen zu erstellen.¹⁶

Mittlerweile gibt es eine Vielzahl verschiedener Frameworks für die clientseitige Entwicklung von Webapplikationen, die teilweise von großen Unternehmen oder Einzelpersonen entwickelt wurden und auch durch den regen Austausch in den immer größer werdenden Communitys kontinuierlich optimiert und erweitert werden. Da die jeweiligen Frameworks unterschiedliche Stärken, Schwächen und Features haben und sich auch teilweise in ihrer grundlegenden Architektur unterscheiden, muss der Entwickler hierbei je nach Anwendungszweck entschieden werden, auf welcher Basis das Frontend aufgebaut werden soll. Zu den mittlerweile beliebtesten Vertretern zählen React, Angular und Vue.

Abbildung 7 ►
Vergleich verschiedener
Frontend-Frameworks in
den Stack Overflow
Trends



React

React, manchmal auch als *React.js* oder *ReactJS* bezeichnet, wurde von Facebook entwickelt und bereits einige Jahre intern eingesetzt, bevor das Framework dann im Jahr 2013 veröffentlicht wurde. Es gibt viele Diskussionen darüber, ob React als Framework oder als Library bezeichnet werden sollte.¹⁵

¹⁵ (Mediaevents, 2021)

¹⁶ (MDN Web Docs, 2021)

Technisch gesehen handelt es sich bei React nur um eine Bibliothek, diese kann jedoch auch mit zusätzlichen Erweiterungen als Framework zur Erstellung von User Interfaces der Single-Page-Apps genutzt werden.¹⁵

Im Gegensatz zu anderen Frameworks, die das MVC bzw. MVVM-Muster nutzen (*Model-View-Controller* bzw. *Model-View-Viewmodel*), deckt React nur die View-Komponente, d.h. die für den User sichtbare Präsentationsebene des Models, ab. Dazu setzt die Library auf das Konzept eines virtuellen DOM als eine Art abstrakte, schlankere Kopie des im vorherigen Kapitel beschriebenen realen DOM. Da Veränderungen zunächst an dem Virtual-DOM vorgenommen werden und dieser schließlich mit dem realen abgeglichen und Unterschiede angepasst werden, muss nicht mehr jeder Knoten einzeln überprüft und neu gerendert werden, was selbst bei komplexen Anwendungen für eine optimale Performance sorgt.¹⁷

Eine weitere Besonderheit von React ist der Einsatz von JavaScript XML, kurz *JSX*. Hierbei handelt es sich um eine Syntaxerweiterung, die erlaubt, auf einfache Weise HTML-Elemente direkt in JavaScript einzubetten und so den Code kompakt und übersichtlich zu halten. Damit die Anwendung letztendlich im Browser funktioniert, ist ein Präprozessor notwendig, der die JSX-Syntax in simples JavaScript umwandelt.

Wie viele andere Frameworks ist die Library komponentenbasiert. Das bedeutet, dass die Präsentationsebene aus einzelnen Blöcken wie beispielsweise Buttons, Dropdown-Menüs oder Card-Elementen zusammengesetzt wird. React erlaubt es dabei, die einzelnen Komponenten isoliert und unabhängig zu betrachten ist und diese daher auch problemlos wiederzuverwenden.¹⁸

Einzelne Komponenten werden in React als JavaScript-Funktionen geschrieben, die sowohl die Logik, als auch die UI enthalten, die im *Return*-Statement als JSX geschrieben wird.

```
1 function Button () {  
2  
3   return(  
4     <button className="bt">  
5       Hier klicken  
6     </button>  
7   )  
8 }  
9
```

◀ Codesnippet 4

React-Komponente als
JavaScript Funktion

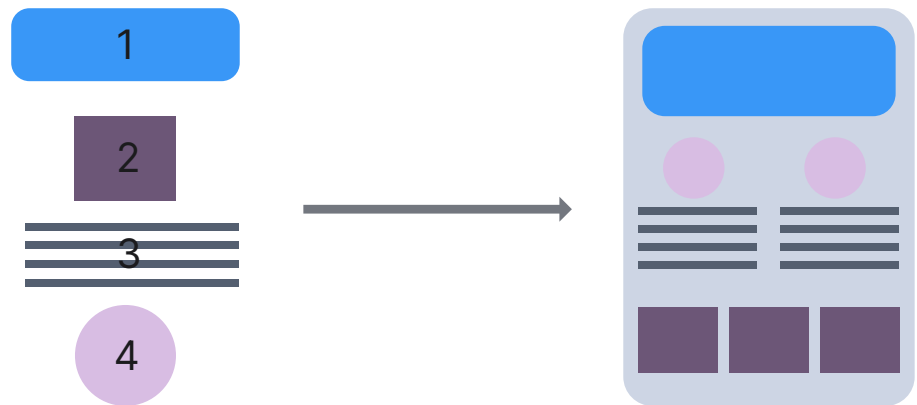
¹⁵ (Mediaevents, 2021)

¹⁷ (Tillmann, 2017)

¹⁸ (Sturm, 2019)

React zielt auf einen minimalen Ansatz ab und bietet den Entwicklern viele Freiheiten und Möglichkeiten, bei der Konzeption einer eigenen flexiblen Architektur für individuelle Probleme die richtige Lösung zu finden. Bei Facebooks Library ist man daher eher gefordert, auf die vielfältigen, von der Community gepflegten Module zu setzen, was darin resultiert, dass oftmals mehrere Wege zum Ziel führen.¹⁹

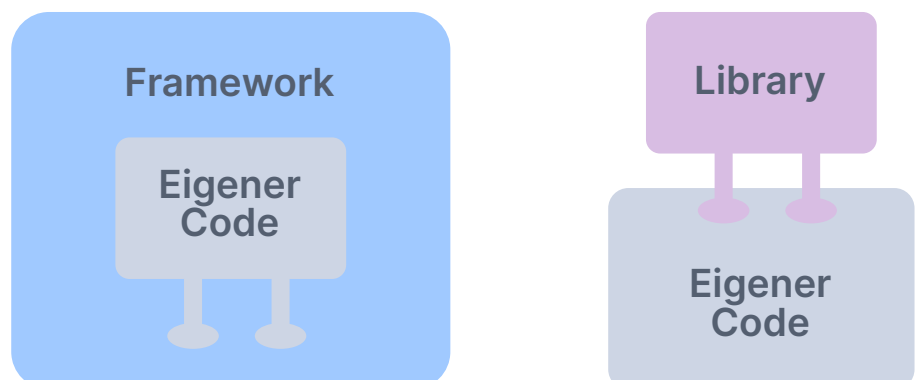
Abbildung 8 ▶
Beispiel eines komponentenbasierten Aufbaus einer React-App



Angular

Neben React zählt auch Angular zu den populärsten Frameworks für die Erstellung von Single-Page-Applications. Das Framework wurde Anfang der 2010er Jahre mit der Unterstützung von Google entwickelt und im Herbst 2016 schließlich in einer grundlegend überarbeiteten zweiten Version als Angular veröffentlicht. Im Gegensatz zur Library React, handelt sich bei Angular um ein tatsächliches Framework, welches über die Funktionen einer Bibliothek hinausgeht. Das Programmiergerüst gibt dabei eine bestimmte Struktur und Architektur vor, in dessen Rahmen die eigene Anwendung integriert wird.¹⁶

Abbildung 9 ▶
Vergleich Framework und Library



¹⁶ (MDN Web Docs, 2021)

¹⁹ (Tappert, 2020)

Googles Angular setzt auf weitgehend auf Konzepte des MVVM-Architekturmodells (*Model-View-ViewModel*) zur Erstellung der Benutzeroberflächen. Das Viewmodel dient als Bindeglied für das Model mit der Anwendungslogik und der View mit der grafischen Benutzeroberfläche und enthält die UI-Logik.¹⁶

Wie auch in React sind die wichtigsten Bausteine einer Angular-Anwendung sogenannte Komponenten. Diese bestehen immer aus einem HTML-Template, einer Klasse, die die View-Logik zur Verfügung stellt, und einem Stylesheet. Abhängig von der definierten Anzeige-Logik und den aktuellen Daten stellen sie den Zustand der Anwendung dar. Übergreifende Daten und Programmlogik, die nicht zwingend an eine Komponente gekoppelt sind, lagern die Entwickler in sogenannten Services aus.²⁰ Der Code kann so in der Regel recht kompakt gehalten werden, da im Grunde nur die HTML-Elemente mit Angular-Syntax erweitert werden müssen, um der Anwendung Interaktivität zu verleihen.¹⁶

Dafür bietet sich die Programmiersprache TypeScript an, die auch von Google genutzt wird, um das Framework zu entwickeln. Diese lässt sich als Erweiterung von JavaScript vorstellen und bietet zusätzliche Möglichkeiten wie objektorientierte Konzepte und statische Typisierung, die dafür sorgen, gerade größere, komplexe Projekte besser zu strukturieren und Fehler zu reduzieren.¹⁸

Angular zielt klar auf das Einsatzgebiet der professionellen Entwicklung in Enterprise Projekten ab. Durch die strengereren Vorgaben hinsichtlich der Struktur und Architektur lassen sich gut wartbare und skalierbare Softwarelösungen umsetzen. Aufgrund des massiven Grundgerüsts hat Angular mitunter den Ruf, etwas überladen und kompliziert zu sein. Auf der anderen Seite bietet es allerdings auch ohne externe Ressourcen einen Funktionsumfang, der bei Frontend Frameworks seinesgleichen sucht.¹⁹

Vue

Anders als bei Angular und React steht hinter diesem Framework kein großer Global Player, sondern mit Evan You eine einzelne Person als Initiator und Schöpfer. Der Chinese, der in den Anfangsjahren Mitglied in dem Entwicklerteam hinter Angular war, beschrieb die Vision des 2014 veröffentlichten Open-Source Frameworks wie folgt: „*I figured, what if I could just extract the part that I really liked about Angular and build something really lightweight.*“¹⁸

¹⁶ (MDN Web Docs, 2021)

¹⁸ (Sturm, 2019)

¹⁹ (Tappert, 2020)

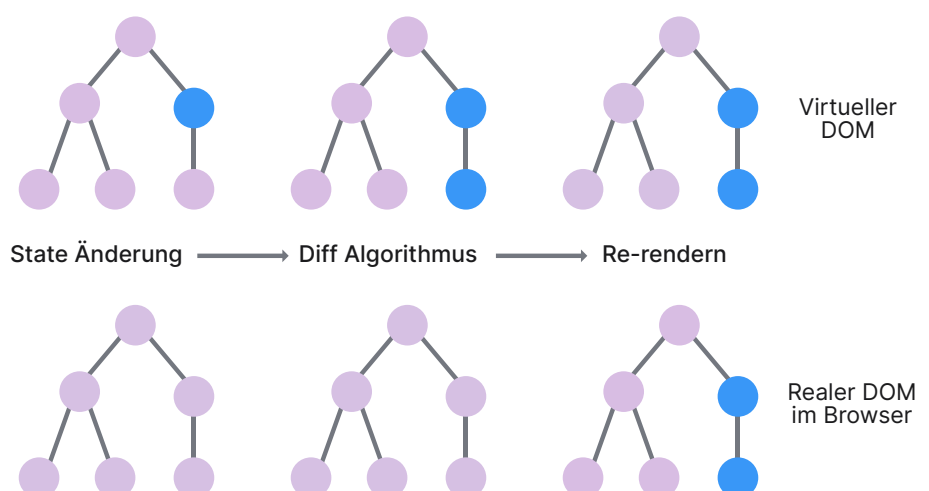
²⁰ (Böhm, 2020)

Vue.js bietet daher im Gegensatz zu Angular kein mächtiges, umfassendes Gesamtpaket. Stattdessen beschränkt sich das Framework in seinem Kern bewusst nur auf den View-Layer des MVC-Modells, wodurch auch der Name, der im Englischen *vju* ausgesprochen wird, abgeleitet wurde. Durch die schlanke und überschaubare Core-Library kann Vue ähnlich wie React problemlos in bestehende Projekte integriert werden. Zudem bietet das umfangreiche Ökosystem auch Bibliotheken, mit denen Vue bei Bedarf beispielsweise um serverseitiges Rendering oder Routing erweitert werden kann. Damit ist Vue in seinem Ansatz nicht ganz unähnlich zu React, mit dem Unterschied, dass die Zusatzmodule nicht aus der Community, sondern direkt aus der Hand von Evan You und seinem Team stammen.²¹

Bei Vue muss zudem neben HTML, CSS und JavaScript vorerst keine weitere Sprache oder Syntaxerweiterung erlernt werden, sondern die HTML-Datei kann einfach mit dem Vue-Markup angereichert werden kann – anders als bei React mit JSX und Angular mit TypeScript.²²

Neben vielen Gemeinsamkeiten zu Angular weist das von Evan You konzipierte Framework auch Parallelen zu React auf und setzt so ebenfalls ein virtuelles DOM ein, um über einen Algorithmus Änderungen im realen HTML-Baum zu überprüfen und so das Rendering bei Änderungen effizienter zu gestalten.²³

Abbildung 10 ►
Rendern der UI-Elemente
mit einem Virtual DOM



²¹ (Peterke, 2019)

²² (Tillmann, 2018)

²³ (Lottana, 2020)

Vue hat sich in den letzten Jahren zu einer echten Alternative zu den Frameworks von Google und Facebook entwickelt und bedient dabei die Anforderungen zwischen den beiden großen Konkurrenten, indem es zum einen klare Strukturen bietet, aber auf der anderen Seite weiterhin versucht, leichtgewichtig zu bleiben und in seinem Kern mit wenig Overhead auszukommen. Zwar ist die Community um Vue noch nicht so groß wie bei Angular oder React, jedoch gibt es eine gute Dokumentation. So lässt sich das Framework ohne Bedenken gerade für kleinere bis mittelgroße Anwendungen einsetzen, stößt bei komplexen Enterprise-Projekten allerdings an seine Grenzen.¹⁸

1.3 Backend

Während das Frontend auf der Clientseite Komponenten enthält, die für die Darstellung der Benutzeroberfläche zuständig sind, bezeichnet das Backend den für den Nutzer nicht sichtbaren Teil einer Anwendung. Dies umfasst unter anderem Anwendungslogik zum Abrufen angeforderter Informationen, den Zugriff auf Datenbanken sowie die Bereitstellung von weiteren Funktionen und Diensten für das Frontend.²⁴

Backend und Frontend sind dabei über eine Programmierschnittstelle miteinander verknüpft. Die Kommunikation zwischen den beiden Bestandteilen findet häufig über eine REST-API (*siehe* 1.1.2) und HTTP-Requests statt.²⁴

1.3.1 Serverseitige Programmierung

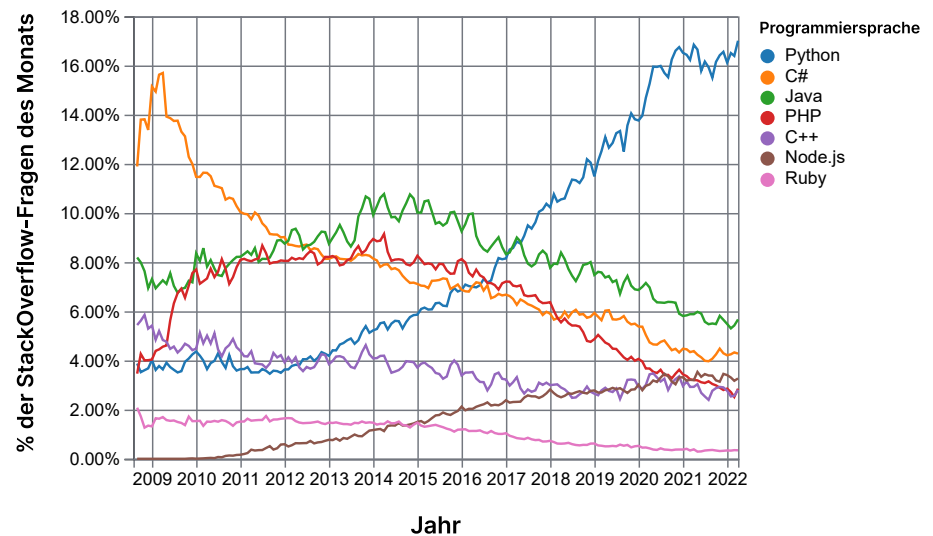
Im vorherigen Kapitel wurde bereits die Programmiersprache JavaScript und einige, darauf basierende Frameworks vorgestellt. Während diese Sprache im Hinblick auf die Entwicklung im Frontend nahezu konkurrenzlos ist, gibt es auf der Serverseite eine deutlich größere Auswahl an verschiedenen Möglichkeiten und Ansätzen.¹

1 (Ackermann, 2021)

18 (Sturm, 2019)

24 (Development Support, kein Datum)

Abbildung 11 ►
Vergleich verschiedener
Programmiersprachen in
den Stack Overflow
Trends



Java

Bei der objektorientierten, klassenbasierten Sprache handelt es sich um eine *General Purpose* Language. Dies ist eine Programmiersprache für allgemeine Anwendungen, die neben der Webentwicklung auch recht populär für die Erstellung von Desktopanwendungen und weiterer Software ist.²⁵

Zur Ausführung der Programme wird der Code durch den Java Compiler in eine zunächst nicht lauffähige Zwischenstufe übersetzt, die schließlich mittels der *Java Virtual Machine*, kurz JVM, in den Maschinencode der zugrunde liegenden Hardware übertragen wird. Vorteil dieses Verfahrens ist es, dass die Anwendungen ohne Neu-Kompilierung auf allen Systemen laufen können, sofern dort eine JVM vorhanden ist.²⁶

Python

Python ist eine interpretierte Programmiersprache, was bedeutet, dass das Programm vor dem Start nicht erst in Maschinencode kompiliert werden muss, sondern von einem Python-Interpreter direkt ausgeführt werden kann.¹ Der Code ist in der Regel recht kompakt und übersichtlich, die Syntax kann jedoch gerade für Java oder JavaScript-Entwickler zu Beginn etwas gewöhnungsbedürftig sein. Zudem wird oftmals die geringe Ausführungsgeschwindigkeit und Performance zur Laufzeit kritisiert.²⁷

1 (Ackermann, 2021)

25 (Thattil, 2019)

26 (Augusten, 2020)

27 (IONOS, 2022)

Ruby

Die Programmiersprache Ruby wurde von dem Japaner Yakuhiro Matsumoto Mitte der 90er Jahre entwickelt. Die Sprache ist von Grund auf objektorientiert und hat seine Vorteile neben einer dynamischen Typisierung und automatischen Speicherbereinigung auch in einer angenehmen Syntax und guten Lesbarkeit.²⁷

Ruby ist für alle gängigen Betriebssysteme verfügbar und wird mit Unterstützung des Frameworks Ruby on Rails gerne für die serverseitige Programmierung eingesetzt. Es gibt nach wie vor eine aktive Community um die Sprache, allerdings ist die Beliebtheit von Ruby in den letzten Jahren durch das Aufkommen von Frontend-Frameworks, mit denen viele Funktionalitäten bereits auf der Client-Seite umgesetzt werden können, etwas zurückgegangen.¹

PHP

Im Gegensatz zu anderen Programmiersprachen, die für die serverseitige Programmierung genutzt werden, jedoch häufig auch noch weitere Anwendungsgebiete haben, wird PHP ausschließlich für die Webentwicklung genutzt. Entwickler können Befehle dabei einfach in den HTML-Code einbetten. Diese werden dann von einem entsprechenden Interpreter auf der Serverseite ausgeführt.¹

Der *Hypertext Preprocessor* zeichnet sich durch eine einfache Syntax aus und ist gerade für Einsteiger auf diesem Gebiet relativ einfach zu erlernen. Obwohl die Sprache mittlerweile als etwas veraltet gilt, erfreut sie sich doch nach wie vor großer Beliebtheit und wird weiterhin von einigen großen Websites oder Content Management Systemen eingesetzt, um dynamische HTML-Seiten zu generieren.²⁵

Node.js

Bei Node.js handelt es sich nicht um eine Programmiersprache, sondern lediglich um eine Laufzeitumgebung für JavaScript, die es ermöglicht, die typischerweise im Frontend eingesetzte Sprache auch außerhalb des Browsers zu nutzen.²⁸

1 (Ackermann, 2021)

25 (Thattil, 2019)

27 (IONOS, 2022)

28 (Springer, 2016)

Da Node.js auf Chromes V8-Engine zur Ausführung von JavaScript Code basiert, stehen sämtliche Sprachfeatures zur Verfügung. Entwickler aus dem Frontend, die bereits mit JavaScript vertraut sind, benötigen kaum Eingewöhnungszeit in die neue Plattform.²⁸

Zusätzliche Module, die Teil der Node-API sind, ermöglichen beispielsweise einen Zugriff auf das Dateisystem oder bieten die Funktion mit nur wenigen Zeilen Code einen Webserver zu erstellen.¹

1.3.2 Datenbanken

Auch die einfachsten Webseiten, bis hin zu komplexen Anwendungen, benötigen alle einen Ort, an dem Daten dauerhaft gespeichert werden und bei Bedarf abgerufen werden können. Dies können beispielsweise einfache HTML-Dateien, Produktinformationen eines Online-Shops, die registrierten Nutzer in einem sozialen Netzwerk oder Medieninhalte einer Streaming-Plattform sein.¹

Grundsätzlich lässt sich dabei zwischen den relationalen Datenbanken, die lange Zeit den Markt dominierten, und No-SQL Datenbanken unterscheiden, die gerade in den letzten Jahren im Zeitalter von *Big Data* an Relevanz gewonnen.

Relationale Datenbanken

Relationale Datenbanken drücken Beziehungen zwischen den Daten aus. Sie sind in Tabellen strukturiert, wobei sich jeder einzelne Datensatz innerhalb einer Zeile, auch Tupel genannt, befindet. Die einzelnen Spalten innerhalb der Tabelle werden als Attribute bezeichnet und stellen die Eigenschaften der vorhandenen Datensätze dar. Normalerweise hat jedes Tupel einen sogenannten Primärschlüssel, der dies innerhalb einer Tabelle eindeutig identifiziert und daher nur einmal vorhanden sein darf.¹¹

Beziehungen zwischen den Daten werden realisiert, in dem bestimmte Attribute einer Zeile der Tabelle als Fremdschlüssel oder auch Foreign Key dienen und auf den Primärschlüssel eines Datensatzes in einer anderen Tabelle verweisen.¹¹

1 (Ackermann, 2021)

11 (Northwood, 2018)

28 (Springer, 2016)

Tabelle: Filme				Tabelle: Regisseure		
FilmID	Name	Regisseur	Jahr	RegID	Name	Alter
1	Inception	5	2010	1	Tarantino	59
2	Pulp Fiction	1	1994	2	Scott	84
3	Alien	2	1979	3	Scorsese	79

◀ Abbildung 12

Tabellen einer relationalen Datenbank

In diesem Beispiel handelt es sich bei dem Attribut *Regisseur* aus der Tabelle *Filme* um einen Fremdschlüssel, der den Primärschlüssel *RegID* aus der Tabelle *Regisseure* referenziert.

Es gibt mittlerweile sehr viele verschiedene Datenbanken, die alle auf dem relationalen Modell basieren. Zu den bekanntesten zählen hierbei:¹

- MySQL
- PostgreSQL
- SQLite
- MariaDB

Um die entsprechenden Tabellen innerhalb einer Datenbank anzulegen sowie Daten zu speichern, abzurufen oder zu manipulieren, wird die Sprache SQL (*Structured Query Language*) eingesetzt, über die es möglich ist, auf die jeweiligen relationalen Datenbanken zuzugreifen.¹

Nicht-relationale Datenbanken

Nicht-relationale Datenbanken, oftmals auch als NoSQL-DBs (*Not only SQL*) bezeichnet, haben erst in den letzten Jahren stark an Bedeutung gewonnen, nachdem die klassischen relationalen Datenbanken nicht mehr alle Anforderungen der heutigen Zeit optimal erfüllen konnten. So müssen im heutigen World Wide Web große unstrukturierte Datenmengen gespeichert und eine Vielzahl an zeitgleichen Anfragen innerhalb kürzester Zeit beantwortet werden.¹ Daher sind die deutlich flexibleren NoSQL-Datenbanken inzwischen eine unverzichtbare Alternative, wenn die starren, tabellenbasierten Datenbanken an ihre Grenzen stoßen. Doch nicht jeder Typ eignet sich für jeden Anwendungszweck, weshalb viele nicht-relationale Ansätze sogar sehr spezialisiert sind.²⁹

¹ (Ackermann, 2021)

²⁹ (Matzer & Litzl, 2018)

Die große Auswahl, die inzwischen in dem Anwendungsfeld der NoSQL-Datenbanken existiert, lässt sich dabei grob in vier Kategorien klassifizieren:

- *Key-Value-Datenbanken* wie Amazons DynamoDB oder Redis speichern die Daten als Schlüssel-Wert-Paare, wobei der Schlüssel als Index genutzt wird, um den Datenbestand durchsuchen zu können. Aufgrund ihres Aufbaus sind diese Datenbanken auch bei riesigen Datenmengen sehr effizient.¹
- In *Graph-Datenbanken* wie Neo4js oder SAP Hybris liegt der Schwerpunkt auf der Abbildung von Beziehungen zwischen den Daten. Häufiger Anwendungsfall dieser Gattung sind soziale Netzwerke, bei denen die Relationen zwischen Followern, Posts und Likes gespeichert und stark vernetzte Datenmengen performant verarbeitet werden können.¹
- Bei *spaltenorientierten Datenbanken* wie Facebooks Cassandra oder Big Table von Google werden die Datensätze, wie der Name schon andeutet, in Spalten oder Spaltenfamilien organisiert. Im Gegensatz zu den relationalen Tabellen sind Zugriffe auf die Attribute versioniert und erfolgen ohne den Umweg über Tabelle, ID und Spalte. Dies hat deutliche Vorteile auf die Lese-Performance, wenn nur auf bestimmte Eigenschaften bzw. Spalten zugegriffen werden soll.
- *Dokumentorientierte Datenbanken* wie MongoDB oder CouchDB kommen meistens zum Einsatz, wenn es um die Speicherung von unstrukturierten Daten geht. So lassen sich Daten unterschiedlichster Formate und ohne ein durchgängiges Schema in diesen Datenbanken verwalten. Einzelne Datensätze werden dafür als Key-Value-Paare in den Dokumenten gespeichert, die wiederum in sogenannten Collections zusammengefasst werden können. Typische Formate für die Dokumente sind JSON bzw. BSON (*Binary JSON*) oder XML.¹

1 (Ackermann, 2021)

1.4 Technologiestacks

Jede Webseite oder Webapplikation setzt sich aus einer Kombination an Technologien zusammen, die auch als *Stack* bezeichnet wird. Dieser umfasst grundsätzlich die Auswahl an bestimmten Komponenten, die genutzt werden, um die entsprechende Applikation zu entwickeln und zu betreiben.¹

Inzwischen ist das Gebiet der Entwicklung von Webanwendungen nicht mehr dasselbe wie noch vor zehn oder zwanzig Jahren. Zahlreiche neu entstandene Frameworks, Bibliotheken, Tools oder Technologieansätze lassen einen schnell den Überblick verlieren und machen es gerade Einsteigern in diesem Bereich schwer, die passenden Optionen für die Anforderungen ihres Projekts auszuwählen. So soll dieses Kapitel einen kurzen Überblick über die bedeutendsten und beliebtesten Technologie-Stacks geben.³⁰

1.4.1 LAMP-Stack

LAMP ist ein Akronym und bezieht sich auf die eingesetzten Komponenten Linux als zugrunde liegendes Betriebssystem, Apache als Webserver, die relationale Datenbank MySQL sowie die serverseitige Programmiersprache PHP, um die entsprechenden Inhalte aus der Datenbank abzufragen und an das Frontend auszuliefern.³¹

Der LAMP-Stack gilt als der älteste und für lange Zeit der am weitesten verbreitete Technologiestack im Webdevelopment. Inzwischen wird der Ansatz häufig aufgrund seiner recht schwierigen Lernkurve für unerfahrene Entwickler kritisiert und als nicht mehr zeitgemäß abgetan. Nichtsdestotrotz hat sich der Stack für viele Jahre in der Branche als zuverlässig bewährt und ist zudem vollständig Open-Source.³¹

1.4.2 Fullstack JavaScript

Im Gegensatz zum klassischen LAMP-Stack, bei dem das Wissen verschiedener Programmiersprachen gefragt ist, basiert bei diesem Konzept die komplette Applikation auf JavaScript, sodass Frontend-Developer, die bereits mit der entsprechenden Technologie vertraut sind, in der Regel kaum Schwierigkeiten haben, für voll funktionsfähige Webanwendungen auch das Backend in derselben Sprache zu entwickeln.³²

1 (Ackermann, 2021)

30 (Subramanian, 2019)

31 (Rogojan, 2019)

32 (Holmes & Harber, 2019)

Dies ist mitunter auch ein Grund, warum Full Stack JavaScript besonders bei kleineren Start Ups recht beliebt ist.³²

Mit dem Aufkommen der Laufzeitumgebung Node.js (siehe 1.3.1) war der MEAN-Stack (*MongoDB, Express, Angular, Node*) einer der ersten Open-Source-Ansätze, der auf das Konzept der Nutzung von JavaScript sowohl auf der Client- als auch auf der Serverseite setzte. Im Backend wird Node dabei vom Framework Express unterstützt, welches das Routing und die Verarbeitung von HTTP-Requests vereinfacht und mit dies der beliebten NoSQL-DB MongoDB verknüpft.³¹ Die Informationen werden flexibel im JSON-Format gespeichert werden und müssen daher zum Abrufen und zur Darstellung im Frontend nicht umgewandelt werden. Somit eignet sich die dokumentenbasierte Datenbank sehr gut für den JavaScript-zentrierten Stack.³³

Auf der Client-Seite kommt das Googles Angular (*siehe 1.2.3*) zum Einsatz. Eine Alternative ist der MERN-Stack, bei dem Angular durch die etwas schlankere, jedoch nicht ganz so umfangreiche Library React ersetzt wird und sich sehr gut für Singe-Page-Apps und kleinere Projekte anbietet.³¹

1.4.3 JAMStack

Mathias Biilmann, CEO der Firma Netlify und Mitbegründer des JAMStack (*JavaScript, API, Markup*) beschreibt seinen Ansatz folgendermaßen: „*A modern web development architecture based on client-side JavaScript, reusable APIs, and prebuild markup.*“ So ist dieses Konzept mehr als eine grundlegende Philosophie und Methode zur Webentwicklung zu sehen und gibt dem Entwickler im Vergleich zu dem LAMP oder MERN-Stack keine Verwendung eines bestimmten Servers, einer Datenbank, Frameworks oder weiteren Tools vor.³³

Fordert ein Nutzer im herkömmlichen Workflow eine Webseite an, so zieht dies eine Reihe von Interaktionen zwischen Backend-Code, Datenbank und Server mit sich, bevor die dynamisch erstellte Seite schließlich an den Browser ausgeliefert wird. Im Gegensatz dazu liegt das HTML-Markup bei dem JAMStack-Ansatz bereits fertig gerendert vor und wird von einem *Content Delivery Network* (CDN) als verteiltes Netzwerk direkt an den Browser ausgeliefert.³⁴

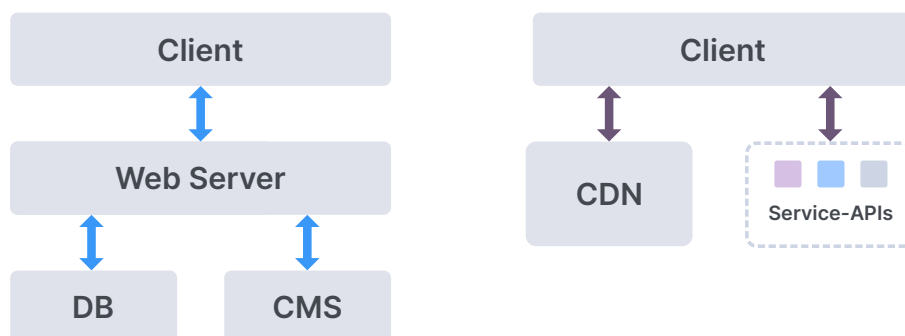
³¹ (Rogojan, 2019)

³² (Holmes & Harber, 2019)

³³ (Zammetti, 2020)

³⁴ (Biilmann & Hawkswort, 2019)

Typischerweise werden zur Erstellung *Static Site Generators* genutzt. Dies muss jedoch nicht bedeuten, dass es sich um rein statische Webseiten handelt. Dynamische Interaktionen oder zusätzliche Funktionalitäten werden dem User im JAMStack über JavaScript im Browser und eigens entwickelte oder externe APIs ermöglicht. Diese können beispielweise Authentifizierungs- oder Bezahlvorgänge abwickeln, einen Warenkorb für E-Commerce-Anwendungen bereitstellen oder Inhalte aus einem HeadlessCMS (siehe Kapitel 3) abrufen.³⁴



◀ Abbildung 13

Auslieferung einer Website im herkömmlichen Ansatz im Vergleich zum JAMStack

1.5 Suchmaschinenoptimierung

Die Suchmaschinenoptimierung, auf Englisch als *Search Engine Optimization*, kurz SEO, bezeichnet, ist inzwischen ein eigenes weitreichendes Fachgebiet, wenn auch nach wie vor eng mit dem der Webentwicklung verwandt. Während die Entwickler in der Regel damit beschäftigt sind, Webseiten technisch mit allen erforderlichen Funktionen umzusetzen, gibt es mittlerweile gerade in größeren Unternehmen eigene SEO-Spezialisten. Deren Aufgabe ist es, dass die entsprechende Seite oder Anwendung später von den Suchmaschinen, insbesondere von Google, möglichst gut gefunden wird.³⁵

Dabei gibt es eine Vielzahl von Aspekten, die bei der Implementierung einer Webseite oder Anwendung berücksichtigt werden müssen, um den Usern zum einen eine optimale Nutzererfahrung zu bieten und damit die Seite zum anderen von den Crawlern der Suchmaschinen gut erfasst werden kann.³⁵

³⁴ (Biilmann & Hawksworth, 2019)

³⁵ (Prerender.io, 2022)

Typischerweise unterscheidet man im Bereich der SEO zwischen der On-Site- und der Off-Site-Optimierung. Als On-Site-Optimierung werden alle Maßnahmen bezeichnet, die auf der zu optimierenden Website selbst durchgeführt werden und somit auch teilweise im Aufgabenfeld der Webdeveloper liegen. Dies sind beispielsweise Verbesserungen des Textes, der Struktur und des Programmcodes der Website. Als Off-Site-Optimierung werden Maßnahmen bezeichnet, die nicht auf der eigenen, sondern auf dritten Websites initiiert werden. Hintergrund einer solchen Optimierungsarbeit ist die Tatsache, dass Suchmaschinen Seiten in der Regel höher bewerten, wenn diese häufiger verlinkt sind.³⁶

Suchmaschinen wie Google nutzen dabei bestimmte Rankingfaktoren, die jedoch nicht veröffentlicht sind und in bestimmten Abständen immer wieder angepasst oder verändert werden. Zwar ist man sich unter Experten einig, dass Google ungefähr 200 bis 400 verschiedene Faktoren zur Bewertung einer Webseite im Hinblick auf einen Suchbegriff heranzieht, welcher dieser Punkte jedoch wie stark gewichtet wird, weiß jedoch nur das Unternehmen selbst.³⁶

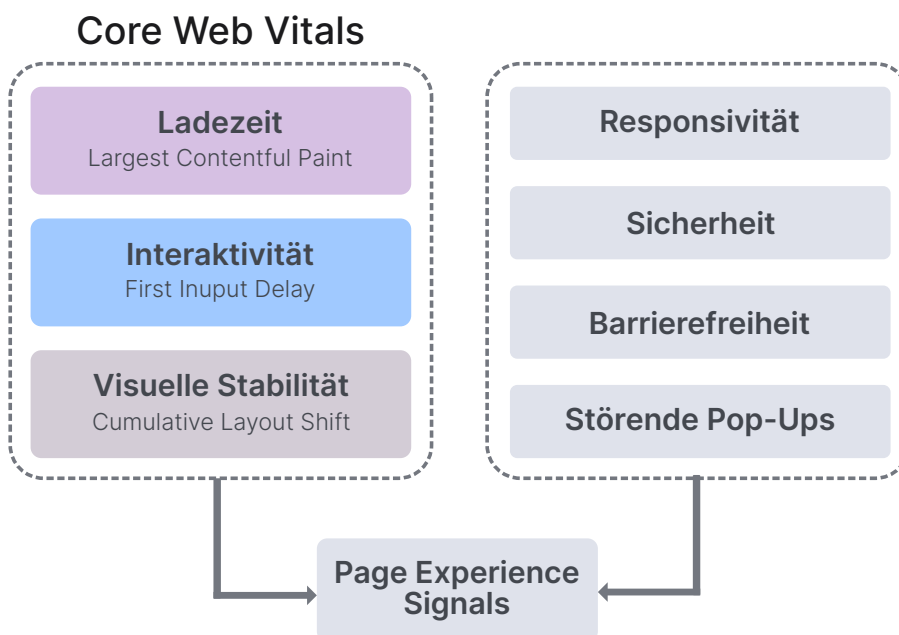
Nachgewiesen bedeutsame Rankingfaktoren sind unter anderem:

- **Text und Keywords:** Die Crawler arbeiten nach wie vor textorientiert und können Grafiken, Bilder oder Videos nur bedingt indizieren. Da die Suchmaschinen mittlerweile deutlich intelligenter sind als früher, reicht es nicht mehr aus, eine möglichst hohe Anzahl an Key-Words in den Text und Überschriften einzubauen. Nichtsdestotrotz sollte die Keyword-Density als Verhältnis eines bestimmten Begriffs zur Gesamtanzahl der Wörter nicht außer Acht gelassen werden.³⁵
- **Struktur und interne Verlinkung:** Gut organisierte Websites mit hochwertigen Inhalten, die den Nutzern einen Mehrwert bieten, werden von Google bevorzugt. Neben einer strukturierten Aufbereitung der Inhalte sind dabei auch eine geplante, sinnvolle Navigation und Verlinkung sowie eine Sitemap hilfreich.³⁶

³⁵ (Prerender.io, 2022)

³⁶ (Lammenett, 2021)

- **Ladezeiten:** Auch die Seitengeschwindigkeit wird von Google seit einigen Jahren berücksichtigt. Um diese zu verbessern, können beispielsweise Bilder und weitere Assets komprimiert, überflüssiger Code entfernt, die Serverantwortzeit verbessert oder bestimmte Renderingmethoden (*siehe Kapitel 2*) genutzt werden. Eine gute Geschwindigkeit hat zudem doppelten Wert, da sie zusätzlich nachgewiesen die User Experience und Conversions erhöht.³⁶
- **Page Experience Signals:** Dieser relativ neu eingeführte Rankingfaktor berücksichtigt die Nutzerfreundlichkeit der Seiten. Neben der Ladezeiten zählen hierzu auch die Sicherheit über HTTPS, Barrierefreiheit und insbesondere die Optimierung für mobile Endgeräte, sodass Seiten ohne responsive Ansicht deutlich schlechtere Chancen auf die vorderen Plätze in den Suchergebnissen haben. Faktoren wie diese wirken sich außerdem auf die Verweildauer oder die Absprungrate aus, was ebenfalls Einfluss auf die Bewertung hat.³⁵



◀ **Abbildung 14**

Einflussfaktoren auf die Page Experience

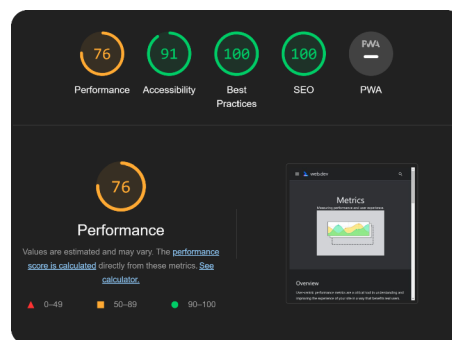
³⁵ (Prerender.io, 2022)

³⁶ (Lammenett, 2021)

Google Lighthouse und Core Web Vitals

Aufgrund der gestiegenen Bedeutung der Suchmaschinen-optimierung stehen mittlerweile immer mehr Tools und Hilfsmittel für Entwickler oder SEO-Experten zur Verfügung. Eine nützliche Open-Source-Anwendung ist Google Lighthouse, das sogar direkt in die Entwicklertools des Chrome-Browsers integriert ist, und dabei hilft die Qualität einer Webseite oder Progressive Web App (siehe Kapitel 4) hinsichtlich Performance, Barrierefreiheit, Sicherheit sowie Benutzerfreundlichkeit zu messen und die Page Experience Signals verbessern zu können.

Abbildung 15 ►
Google Lighthouse Report einer Webseite



Als wichtige und einheitliche Qualitätssignale für die Nutzererfahrung einer Seite stellt Google die sogenannten *Core Web Vitals* bereit, die als messbare Kennzahlen für Seitenbetreiber ermöglichen, die Page Experience ihrer Seite einzustufen.³⁷

Dazu zählen folgende Parameter:

- **Largest Contentful Paint (LCP)** misst die Dauer des Vorgangs bis das Hauptelement wie z.B. ein großes Bild einer Seite geladen ist. Für eine gute Bewertung sollte dieser Wert weniger als 2,5s betragen.³⁷
- **First Input Delay (FID)** gilt als Kennzahl für den Zeitpunkt, ab dem der User mit der Seite interagieren kann und diese auf Klicks oder Eingaben reagiert. Für einen guten FID-Score gibt Google eine Zeit von 100ms an.³⁷
- **Cumulative Layout Shift (CLS)** beschreibt, wie stark sich das Layout einer Webseite während dem Ladevorgang und der Nutzung verändert und sich einzelne Elemente unerwartet verschieben.³⁷

³⁷ (Kessler, 2021)

Rendering von Weban- wendungen

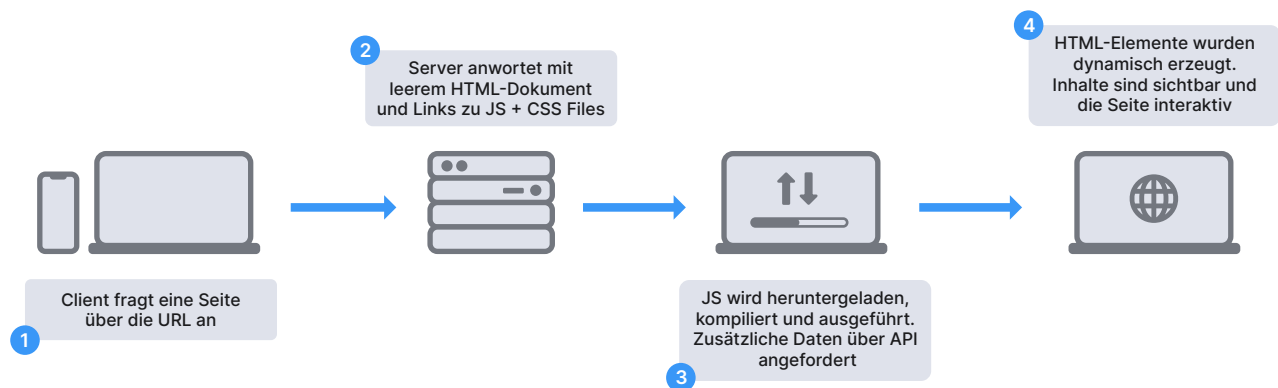
Während zu Beginn des Webs den Nutzern statische HTML-Dokumente auf Servern bereitgestellt und von Browsern abgerufen und dargestellt werden konnten, wandelte sich die reine Informationsplattform im Laufe der Jahre immer mehr zu einem lebendigen Ort, in dem die User mittlerweile vielseitige Serviceleistungen in Anspruch nehmen, Daten in Echtzeit austauschen oder dynamische, personalisierte Webseiten nutzen.

Bei jedem Seitenaufruf müssen dazu die entsprechenden Informationen aus Datenbanken, Content Management Systemen oder über andere Schnittstellen abgerufen und letztendlich gerendert werden, um diese in ansprechender Weise auf dem Bildschirm darstellen zu können. Je nach Programmieransatz und genutzter Technologie können diese Vorgänge auf verschiedene Arten implementiert werden, die unter anderem Ladezeiten, Suchmaschinenoptimierung und User Experience beeinflussen. So sollen in diesem Kapitel mit dem Client-Side-Rendering, dem Server-Side-Rendering, der Static-Site-Generation und der Incremental Static Regeneration die vier grundlegenden Konzepte betrachtet, erklärt und deren Vor- sowie Nachteile und Anwendungszweck herausgestellt werden.

2.1 Clientseitiges Rendering

Seit dem Aufkommen der vielen JavaScript-Frameworks und Bibliotheken wie Vue oder React setzen viele Entwickler auf das Konzept, Webseiten oder Single Page Apps im Browser auf der Clientseite zur Darstellung für den Nutzer zu rendern. Ruft ein User eine Seite über die entsprechende URL auf, so antwortet der Server mit einer einfachen HTML-Datei, die in der Regel nur ein leeres Root-Element sowie Links zu den entsprechenden JavaScript-Files oder Stylesheets enthält, woraufhin der Browser weitere Requests sendet, um diese Dateien zu laden.³⁸

Sind schließlich alle Ressourcen verfügbar, wird der JavaScript-Code kompiliert, zusätzliche Daten, beispielsweise aus einer Datenbank, über REST-API-Aufrufe angefordert, die HTML-Elemente dynamisch erzeugt und auf dem Display ausgegeben, sodass die Seite schließlich für den Nutzer verfügbar ist.³⁸



▲ **Abbildung 16** Vorgang des Client Side Renderings vom Seitenaufruf bis zur Darstellung im Browser

+ Vorteile

- Single Page Applications oder Webseiten, die auf clientseitiges Rendering setzen, sind nach dem initialen Ladevorgang sehr flüssig und performant und fühlen sich für den User oftmals wie eine native App an. Ist einmal der komplette JavaScript-Code geladen und kompiliert, so steht jede Ansicht der Webapp bereits zur Verfügung und kann dynamisch je nach Nutzerinteraktion geändert werden, ohne wie bei herkömmlichen URL-Wechseln immer wieder neue Inhalte vom Server nachzuladen.³⁹

³⁸ (Thakkar, 2020)

³⁹ (Riva, 2022)

- Da nahezu das komplette HTML-Markup dynamisch im Browser generiert und die komplette Seite erst dort gerendert wird, kann der Server deutlich entlastet werden. Dieser sendet auf die Anfragen einfach ein simples HTML-Dokument, sodass in der Regel für diese Art von Anwendungen auf leistungsstarke Server verzichtet wird. Der Server dient nur als reine Datenschnittstelle, um den JavaScript-Code zum Browser zu senden und ist nicht für aufwendige Logikoperationen zuständig.³⁹



Nachteile

- Da bei erstmaligem Aufruf der Seite zunächst die zusätzlichen JavaScript- und CSS-Dateien heruntergeladen und verarbeitet werden müssen, sieht der Nutzer bei dem initialen Ladevorgang für einige Zeit nur einen leeren Screen. Besonders problematisch kann dies sein, wenn der User zu diesem Zeitpunkt nur über eine schlechte Internetverbindung verfügt.⁴⁰
- Aufgrund dessen, dass die ungerenderte HTML-Seite nur ein leeres `<Div>`-Element enthält und die eigentlichen Inhalte noch nicht erzeugt wurden, haben die Crawler der Suchmaschinen Probleme, den Content der Seite richtig zu erfassen oder müssen zuerst lange auf die Ausführung des Programmcodes warten, was sich negativ auf die SEO auswirkt.⁴¹

2.2 Serverseitiges Rendering

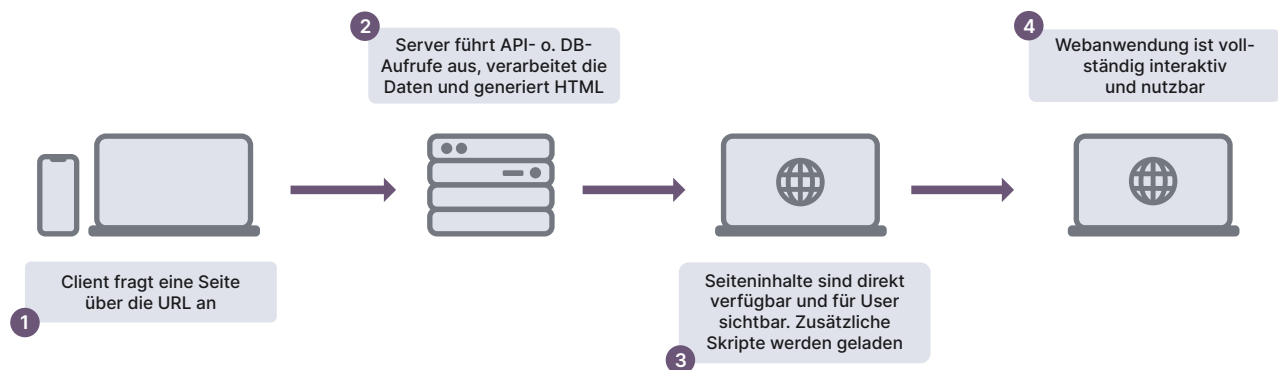
Obwohl sich die Bezeichnung Server Side Rendering wie ein neuer Begriff im Vokabular von Webentwicklern liest, so handelt es sich im Grunde genommen um die ursprüngliche Art wie mit Programmiersprachen wie PHP oder Python im Backend dynamische Webseiten erzeugt, gerendert und im Browser dargestellt werden. Während diese Methode mit dem Aufkommen leichtgewichtiger Frameworks wie React, Vue oder Angular etwas in Vergessenheit geriet, erfreut sich das Konzept gerade in den letzten Jahren mit dem Aufkommen neuer Technologien (*siehe* 2.6), auch im Hinblick auf die zuvor beschriebenen Probleme des Renderings im Browser enormer Beliebtheit.³⁹

³⁹ (Riva, 2022)

⁴⁰ (Moore, 2021)

⁴¹ (Farmer, 2021)

Wird hier eine Seite vom Client aufgerufen, so ist es, im Gegensatz zum davor beschriebenen Client-Side-Rendering, bei diesem Ansatz die Aufgabe des Servers, nach Eingang des Requests externe Daten aus einer Datenbank oder Schnittstelle abzurufen, den Code und die Styles zu kompilieren und als bereits fertige HTML-Seite dem Browser zur Verfügung zu stellen. Bei Interaktion mit der Seite, beispielsweise bei URL-Wechsel oder API-Aufrufen, wird dieser Prozess auf der Serverseite wiederholt.⁴¹



▲ **Abbildung 17** Vorgang des Server Side Renderings vom Seitenaufruf bis zur Darstellung im Browser

+ Vorteile

- Im Gegensatz zum clientseitigen Rendering haben die Bots der Suchmaschinen hier keine Probleme den Content zu erfassen, da dieser bereits bei der Antwort des Servers fertig generiert vorliegt und nicht erst von der Ausführung des JavaScript-Codes im Browser abhängt. Serverseitig gerenderte Seiten bieten daher Vorteile bei der SEO und erzielen dadurch in der Regel bessere Positionen in den Google Rankings.⁴¹
- Ein weiterer Vorteil, gerade bei wenig interaktiven Seiten, sind die meist schnelleren Ladezeiten und die bessere User Experience, da der Nutzer bereits direkt nach der Response die fertig gerenderten Inhalte zu sehen bekommt und nicht erst vor einem leeren Bildschirm oder einer Ladeanimation warten muss, bis das JavaScript kompiliert und ausgeführt ist. Zudem ist ein Großteil des Contents auch verfügbar, wenn der Nutzer JavaScript im Browser deaktiviert hat.⁴¹

⁴¹ (Farmer, 2021)

- Da Aktivitäten wie das Management von Cookies, Aufrufe von privaten APIs oder die Validierung von Daten auf dem Server abgewickelt werden, bietet dieser Ansatz mehr Sicherheit und Kontrolle und es wird verhindert, dass private Daten zu einfach freigegeben werden.³⁹



Nachteile

- Viele Aufgaben werden hier wieder auf der Server- statt auf der Clientseite ausgeführt. Insbesondere bei fortwährenden oder vielen gleichzeitigen Anfragen kann es zu einer hohen Auslastung des Servers und gegebenenfalls zu Verzögerungen kommen. Im Gegensatz zum CSR oder SSG können die Seiten nicht einfach auf Cloud-Anbietern gehostet werden. Es wird mehr Rechenleistung benötigt, wodurch auch mehr Unterhaltungskosten anfallen.³⁹
- Ein weiterer Punkt, der beachtet werden sollte, sind zusätzliche Latenzen, die bei jedem Request anfallen. Gerade die Navigation zwischen verschiedenen Unterseiten, die bei jeder Anfrage zunächst gerendert werden müssen, ist deutlich langsamer im Vergleich zu clientseitigen Anwendungen oder statisch ausgelieferten Seiten.³⁹

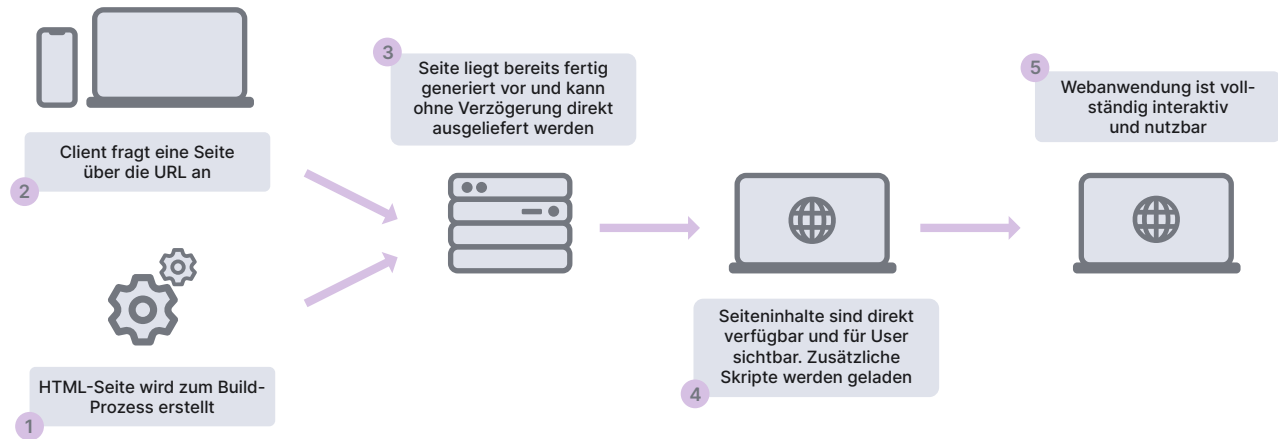
2.3 Statische Generierung

Die Idee der Static Site Generation, welche bereits in 1.4.3 als Teil des JAMStacks angedeutet wurde, geht auf die Ursprünge des Webs zurück, wo statische HTML-Seiten auf den Servern gehostet wurden und nach einer Anfrage direkt ausgeliefert werden konnten.⁴⁰

Auf den ersten Blick ähnelt dieser Ansatz dem des Server Side Renderings, sodass entsprechende API-Requests oder Datenbankinteraktionen auf der Serverseite ausgeführt und anschließend ein HTML-Template generiert wird. Der entscheidende Unterschied dabei ist jedoch der Zeitpunkt der Erstellung des HTML-Codes. So werden diese Schritte bereits im Voraus bei der Erstellung im sogenannten *Build-Prozess* beim Hosten auf dem Server oder einem Cloud-Anbieter durchgeführt.⁴⁰

³⁹ (Riva, 2012)
⁴⁰ (Moore, 2021)

Wird nun eine Seite durch den User aufgerufen, liegt diese bereits fertig generiert auf dem Server vor, sodass dieser entsprechend der angefragten URL ohne Verzögerung antworten kann.⁴⁰



▲ **Abbildung 18** Vorgang der Static Site Generation vom Seitenaufruf bis zur Darstellung im Browser

+ Vorteile

- Da das HTML bereits zur Build-Time und nicht erst auf Anfrage eines Clients generiert wird, kann der Server die Seite einfach als statisches Asset senden, sodass die Inhalte direkt vom Browser dargestellt werden können. Während bei dem Serverseitigen Ansatz Verzögerungen durch das Abrufen der Daten zum Zeitpunkt des Requests entstehen, läuft der Prozess bei der Static Site Generation bereits im Voraus ab und bietet daher eine optimale Performance.³⁹
- Die statisch generierten HTML-Seiten sind auf keine weiteren Operationen eines Servers angewiesen, sodass diese auch einfach über ein Content Delivery Network gehostet werden können. Selbst die Nutzung eines eigenen Webserver führt zu keiner großen Belastung von diesem, da keine aufwendigen Prozesse mehr durchgeführt werden müssen und die Seiten einfach als statische Assets ausgeliefert werden.³⁹

³⁹ (Riva, 2022)

⁴⁰ (Moore, 2021)

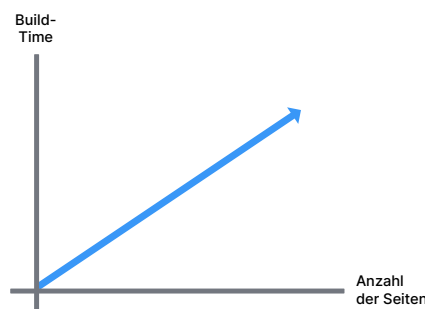
- Da das HTML und der Content bereits vor dem Senden an den Client vorliegen, können die Seiten einfach von den Crawlern der Suchmaschinen erfasst werden, ohne dass diese erst auf das Rendering im Client warten müssen. Diese Methode bietet daher auch Vorteile im Hinblick auf die Suchmaschinenoptimierung.⁴⁰



Nachteile

- Die Static Site Generation ist eine optimale Lösung für performante, skalierbare Frontend-Anwendungen. Das größte Problem dieses Ansatzes ist es jedoch, dass der Content der Seiten bis zum nächsten Deployment gleichbleibt. Somit ist diese Rendering Technik tendenziell eher für Webseiten mit selten wechselnden Inhalten geeignet, deren Daten während des Builds einmal abgerufen und anschließend als statische Assets gehostet werden. Für Anwendungen, deren Inhalte sich regelmäßig ändern und aktualisiert werden, muss ansonsten dieser Prozess jedes Mal neu durchgeführt werden.³⁹

Abbildung 19 ►
Zusammenhang zwischen Dauer des Builds und Anzahl der Seiten



- Der Build-Prozess kann je nach Größe der Anwendung sehr aufwendig sein und vergleichsweise lange dauern, insbesondere wenn dabei viele einzelne statische Seiten generiert werden.⁴⁰

2.4 Inkrementell Statische Regenerierung

Die Nachteile der Static Site Generation zeigen, dass gerade kleinere Content-Änderungen in großen komplexen Applikationen mit vielen verschiedenen Seiten zu Problemen führen können. Wenn beispielsweise in einem Online-Shop mit einigen tausend Produkten, für die jeweils eine statische Seite vorliegt, eine Bezeichnung oder ein Preis von einzelnen Artikeln geändert wird, so müsste gegebenenfalls stundenlang gewartet werden, um diese Änderung im Store sichtbar zu machen, da der komplette Build-Prozess der Anwendung erneut durchlaufen wird.⁴²

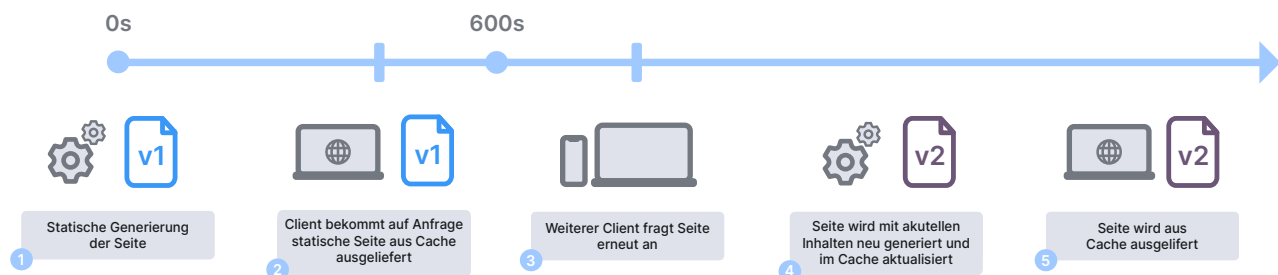
³⁹ (Riva, 2022)

⁴⁰ (Moore, 2021)

⁴² (Robinson, 2021)

Daher sollte im Idealfall die App intelligent genug, sein, um zu erkennen, welche Produkte geändert wurden und dessen Seiten einzeln inkrementell aktualisieren, ohne einen kompletten Rebuild zu durchlaufen. Für solche großen, umfangreichen Websites steht mit der inkrementell statischen Regenerierung eine flexiblere hybride Lösung zur Verfügung.⁴²

Diese Methode erlaubt es, neue statische Seiten zur Laufzeit anzulegen oder zu aktualisieren, ohne dabei den kompletten Build-Prozess zu wiederholen und Vorteile wie die schnelle Auslieferung beizubehalten. Um aktuelle dynamische Inhalte zur Verfügung zu stellen, kann bei der inkrementellen Regenerierung ein bestimmtes Zeitintervall definiert werden. Wird dies beispielsweise auf zehn Minuten gesetzt, so hat jeder Nutzer in den ersten zehn Minuten nach dem Build Zugriff auf die gleiche statische Seite. Wird diese nach der entsprechenden Zeit erneut angefragt, so löst der Request ein erneutes Rendering auf der Serverseite aus, bei dem die aktuellen Informationen aus einer Datenbank oder CMS abgefragt werden. Anschließend wird die Seite als statisches Asset gespeichert und die vorherige Version überschrieben. Beachtet werden muss dabei allerdings, dass dieser Prozess nicht von selbst aktiv wird und nach Ablauf des Intervalls erst bei einer erneuten Anfrage greift.³⁹



▲ **Abbildung 20** Ablauf der Inkrementell Statischen Regenerierung

+ Vorteile

- Diese Methode bietet schnellere und optimierte Build-Zeiten, da im Vergleich zur Static Site Generation nur bestimmte Seiten, wie beispielsweise die am häufigsten angeforderten, zum Build-Prozess statisch generiert werden können und sich sehr große Anwendungen besser skalieren lassen. Zudem können aktuelle Inhalte bereitgestellt oder ausgetauscht werden, ohne die komplette Applikation neu erstellen und hosten zu müssen.⁴²

³⁹ (Riva, 2022)

⁴² (Robinson, 2021)

- Im Gegensatz zu clientseitig gerenderten Apps sind Anwendungen, die die inkrementelle Regeneration nutzen, besser für die Crawler der Suchmaschinen optimiert. Zudem werden sie auch in Browsern mit deaktiviertem JavaScript dargestellt und Nutzer können ohne Verzögerung die Inhalte direkt sehen.⁴²



Nachteile

- Die Inkrementelle Statische Regeneration eignet sich nicht für alle Arten von Websites. Gerade bei kleineren Anwendungen mit relativ wenig verschiedenen Seiten und wenn das Revalidierungsintervall größer als die Zeit eines Rebuilds ist, macht es mehr Sinn, auf den Ansatz der Static Site Generation zu setzen.⁴²
- Auch für hochdynamische Anwendungen mit aktuellen und sich ständig veränderndem Content wie beispielsweise ein News Feed ist diese Methode keine Ideallösung, da der erste Nutzer nach Ablauf des Zeitintervalls zunächst die vorherige Version als Fallback zu sehen bekommt und erst die Neuerstellung für die nächsten User auslöst. In solchen Anwendungsfällen wäre somit das Server Side Rendering besser geeignet, um keine veralteten Inhalte anzuzeigen.⁴³
- Ein weiterer Nachteil dieses hybriden Ansatzes ist es, dass die Anfrage für Seiten, die noch nicht zum Build generiert wurden, für den ersten Nutzer etwas länger dauern, da hier die Daten erst abgerufen und das statische HTML-Asset generiert werden muss.⁴³

2.5 Vergleichender Überblick

Nachdem auf den vorherigen Seiten die einzelnen Ansätze mit ihren Vor- und Nachteilen vorgestellt und deren Ablauf vom Build-Prozess bis zur letztendlichen Darstellung im Browser erläutert wurden, soll dieses Kapitel abschließend einen anschaulichen Überblick über die verschiedenen Rendering-Methoden geben und diese hinsichtlich der Aspekte Datenaktualität, Suchmaschinenoptimierung, Performance und Build-Time vergleichen.

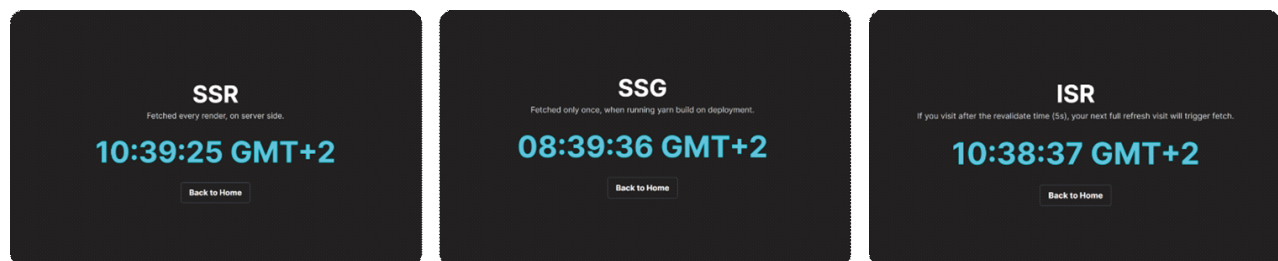
⁴² (Robinson 2021)

⁴³ (Frana, 2020)

Zur Unterstützung wird dabei eine Webanwendung des Entwicklers Theodoruns Clarence (<https://next-render.theodorus-clarence.com/render/>) hinzugezogen. Diese setzt sich aus einzelnen, Unterseiten, jeweils auf eine andere Art und Weise gerendert, zusammen und gibt über eine Zeit-API die Uhrzeit aus.

2.5.1 Datenintegrität

Eine hohe Aktualität bedeutet, dass bei jedem Rendering die neusten und zuletzt geänderten Daten, beispielsweise aus einer Datenbank oder einem Content Management System, abgerufen werden. Dies ist insbesondere für sehr interaktive Anwendungen mit zeitkritischen Parametern entscheidend, deren Informationen nicht veraltet sein dürfen.⁴⁴

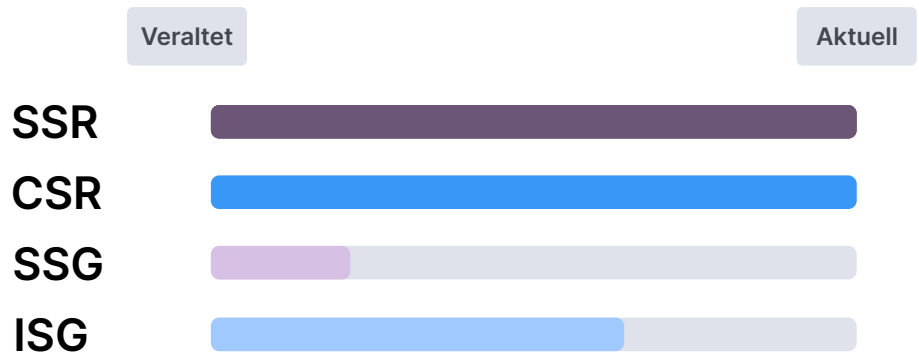


▲ **Abbildung 21** Datenaktualität bei SSR,SSG und ISR im Vergleich

Da bei clientseitigem und serverseitigem Rendering die Daten jeweils erst auf eine eintreffende Anfrage des Clients angefordert werden, bieten diese Ansätze die höchste Integrität. So zeigt die untersuchte Anwendung auf der serverseitig und clientseitig generierten Seite die aktuellste Angabe im Vergleich zur tatsächlichen Uhrzeit, während bei der statisch erzeugten Seite die Uhrzeit zum Zeitpunkt des Builds ausgegeben wird und diese sich auch nach erneutem Laden nie verändert. Bei der inkrementell regenerierten Ansicht, die alle fünf Sekunden validiert wird, liegt die Anzeige in Nähe der Echtzeit, ist jedoch nicht so aktuell wie die der server- und clientseitig erzeugten Seite. Hier löst ein Request erst den Rebuild, bei dem die neue Uhrzeit angefordert wird, aus und es kann zunächst nur die vorherige Version für den Client bereitgestellt werden.

⁴⁴ (Clarence, 2021)

Abbildung 22 ►
Datenaktualität verschie-
dener Renderingansätze

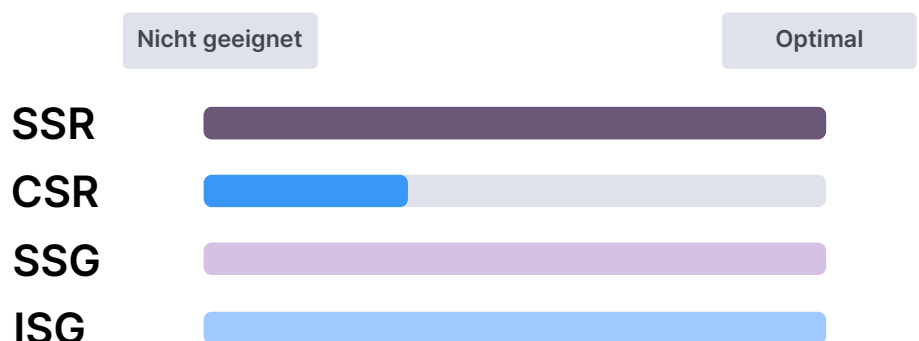


2.5.2 Suchmaschinenoptimierung

Suchmaschinenoptimierung spielt vor allem dann eine wichtige Rolle, wenn ein Ziel der Seite die Auffindbarkeit des Contents mittels Suchmaschinen wie Google und damit der Gewinn von vielen Besuchern ist.

Dabei sind die Methoden des Server Side Rendering und der Static Site Generation bzw. der statischen Regenerierung deutlich besser geeignet als der clientseitige Ansatz. Grund dafür ist, dass bei diesen Verfahren das HTML bereits vor der Antwort an den Client erzeugt wird. Indexiert nun ein Bot einer Suchmaschine die Seite, so sind alle Inhalte direkt im angeforderten HTML-Quellcode enthalten und können von diesem erfasst werden. Im Gegensatz dazu erhält der Crawler bei clientseitigem Rendering, wie in *Kapitel 2.1* beschrieben auf Anfrage in der Regel zu Beginn nur ein leeres <div>-Element, in das die eigentlichen Inhalte nach Herunterladen und Ausführen des JavaScripts erst eingefügt werden. Zwar sind die Bots mittlerweile in der Lage, auch diese Art von Seiten zu erfassen, jedoch bei weitem nicht so gut wie bei den zuerst genannten Methoden.⁴⁴

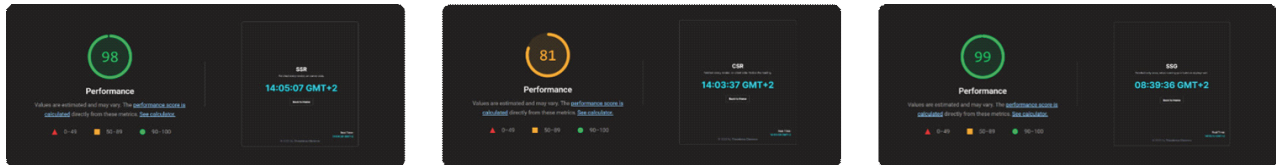
Abbildung 23 ►
SEO-Freundlichkeit von
verschiedenen Render-
ingansätzen



44 (Clarence, 2021)

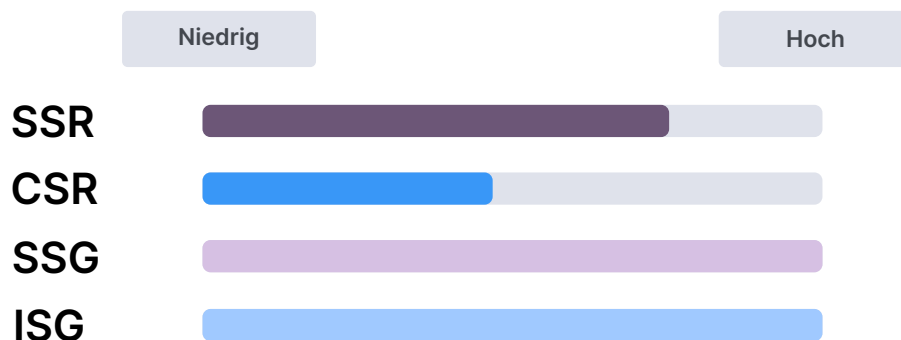
2.5.3 Performance

Ein Faktor, der mitunter Einfluss auf die User Experience einer Anwendung hat, ist die Ladegeschwindigkeit und Wartezeit, bis nach dem Aufruf der Seite erste Inhalte für den Nutzer zu sehen sind. Ein grundlegender Unterschied bei der Betrachtung der jeweiligen Renderingansätze ist, ob die Seite dabei erst auf den Request eines Clients erzeugt werden muss oder das HTML bereits im Voraus generiert wurde.⁴⁴



▲ **Abbildung 24** Google Lighthouse Performance Bewertungen bei SSR, CSR und SSG im Vergleich

Wird die Qualität der einzelnen Seiten der beispielhaften Anwendung in Google Lighthouse gemessen und das Performance-Rating betrachtet, so fällt auf, dass gerade die clientseitig gerenderte Ansicht im Vergleich etwas abfällt und nur einen durchschnittlichen Gesamtwert von 81 erreicht, während die statisch und serverseitig erstellten Seiten deutlich besser abschneiden. Da hier das HTML bereits vor der Antwort an den Browser erstellt wurde, wird die Wartezeit für den User verkürzt, bis dieser erste Inhalte sieht. Zudem kann er schneller mit der Seite interagieren, was sich positiv auf Googles Core Web Vitals (siehe 1.5) auswirkt. Da es sich in diesem Beispiel nur um eine recht simple Anwendung handelt, bei der auf Nutzeranfrage im Backend lediglich die Uhrzeit abgerufen werden muss, ist zwischen der serverseitig und statisch erzeugten Seite nur ein marginaler Unterschied erkennbar. Gerade bei komplexen Anwendungen bietet der statische Ansatz hier Vorteile, da die Seite ohne Verzögerung als fertiges Asset zurückgegeben wird.



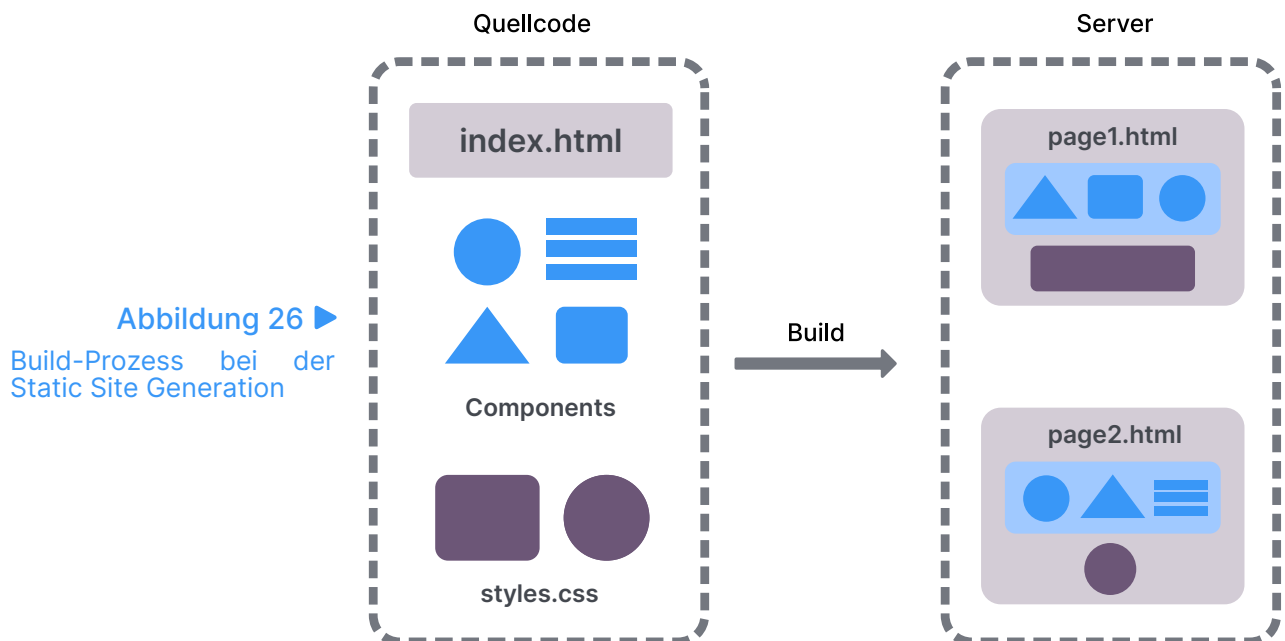
◀ **Abbildung 25**

Performance verschiedener Renderingansätze

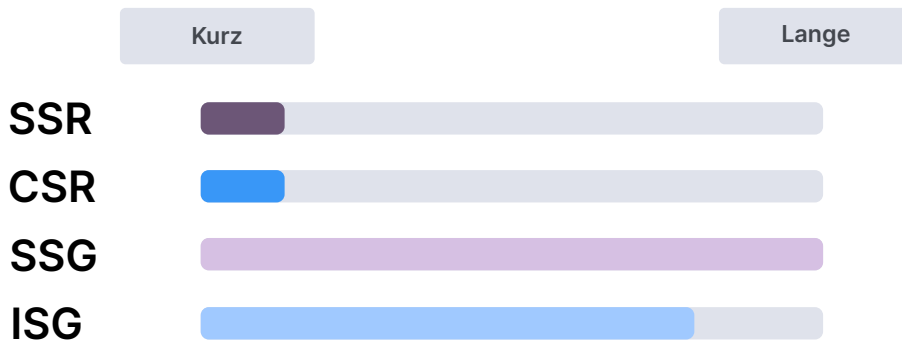
⁴⁴ (Clarence, 2021)

2.5.4 Build-Time

Die Build-Time beschreibt die Zeit, die benötigt wird, um die gesamte Anwendung produktionsfertig zusammenzusetzen um auf einen Server oder Content Delivery Network zu deployen. Dieser Parameter spielt typischerweise erst eine Rolle, wenn sich eine Anwendung aus mehreren verschiedenen Seiten zusammensetzt und beispielsweise dynamisch generierte Routen wie z.B. in einem Online-Shop für jedes Produkt nutzt (*www.shop.com/produkte/[name]*).



Da, wie bereits erläutert, bei dem server- und clientseitigen Ansatz die Daten erst nach dem Request eines Nutzers für eine Seite z.B. *www.shop.com/produkte/smartphone*, abgerufen werden und anschließend die Seite zum Ausliefern gerendert wird, ist die Zeitdauer des Buildprozesses relativ kurz. Für kürzere Antwortzeiten ohne Rendering-Prozess muss hingegen bei der Static Site Generation eine längere Build-Time in Kauf genommen werden. Hier werden bereits zum Deployment die erforderlichen Daten abgerufen und alle Seiten direkt zu diesem Zeitpunkt erstellt. Beinhaltet der beispielhafte Online-Shop 1000 Produkte, müssen in diesem Fall so viele Seiten generiert werden, was im Vergleich deutlich länger dauert. Bei der inkrementellen Regeneration kann diese Zeit etwas verkürzt werden, indem nur ausgewählte Seiten, wie beispielsweise die beliebtesten Produkte, sofort und alle weiteren HTML-Assets erst zur Laufzeit erzeugt werden.



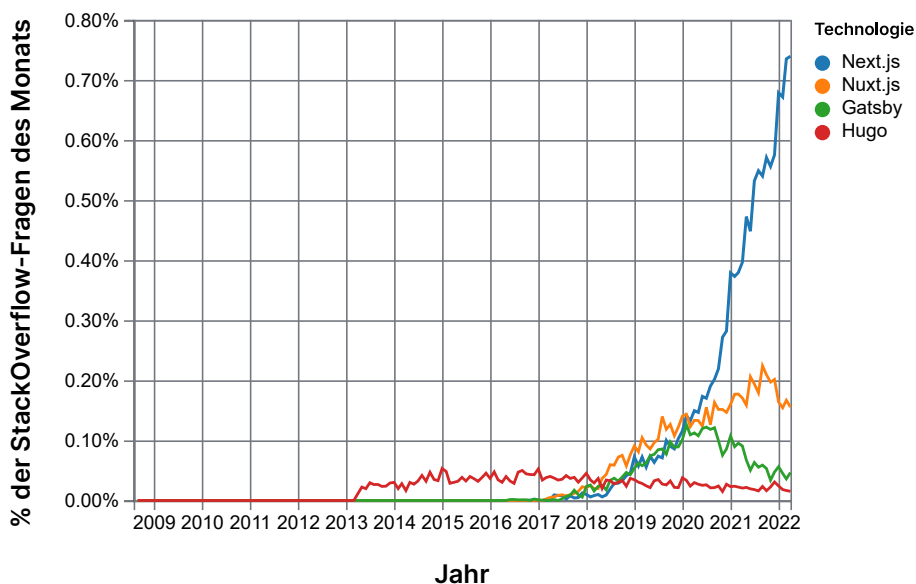
◀ **Abbildung 27**

Build-Time verschiedener Renderingansätze

2.6 Technologieansätze

Nachdem in den vorherigen Kapiteln verschiedene Renderingmethoden ausführlicher beleuchtet wurden, soll der folgende Abschnitt nun die zugehörigen Technologieansätze und hilfreiche Tools vorstellen.

Während die in *Kapitel 1.2.3* erläuterten Frontend-Frameworks React, Vue und Angular typischerweise ein clienseitiges Renderingkonzept verfolgen, werden für serverseitiges Rendering darauf basierende Erweiterungen benötigt und für den statischen Ansatz sogenannte Static Site Generatoren genutzt.



◀ **Abbildung 28**

Vergleich verschiedener Static Site Generator und SSR-Frameworks in den Stack Overflow Trends

2.6.1 Next.js

Das Framework Next.js wurde im Jahr 2016 in seiner Version 1.0 released und sollte dabei die Vorteile von Facebooks beliebter Frontend-Library React mit entsprechenden Lösungsansätzen für deren Probleme und Nachteile kombinieren. Mittlerweile ist das Open-Source-Projekt Next.js, unterstützt von dem Cloudhosting-Anbieter Vercel, sehr weit verbreitet und wird von vielen Entwicklern aufgrund seiner vielseitigen Features in ihrem Technologiestack eingesetzt.⁴⁵

Um die in *Kapitel 2.1* erläuterten Nachteile des clientseitigen Renderings von React-Applikationen zu lösen, werden die mit Next.js erstellten Seiten nicht mehr leer, sondern bereits vorgerendert mit allen HTML-Elementen an den Browser ausgeliefert. Diese werden anschließend über das Virtual DOM mit React-Code verknüpft und sind nun auch weiterhin auf der Clientseite interaktiv. Dieser Prozess wird als Hydrierung bezeichnet. Das Framework bietet dabei zum einen die Möglichkeit, einzelne Seiten statisch zu generieren als auch serverseitig zu rendern. Vorteil ist, dass Entwickler für jede einzelne Seite entscheiden können, welcher Ansatz geeigneter ist und somit ermöglicht wird, flexible und hybride Webanwendungen zu erstellen.⁴⁵

Standardmäßig werden alle Next.js-Seiten statisch erzeugt. Um diese dynamisch zu erweitern und mit nutzerspezifischen persönlichen Informationen anzureichern, können Daten von externen Ressourcen wie beispielsweise Datenbanken, Content Management Systemen oder externen REST-APIs abgerufen werden. Next stellt hierfür mit *getServerSideProps* und *getStaticProps* zwei vorgefertigte Funktionen zur Verfügung. Die Erste kann genutzt werden um API-Aufrufe bei jeder Client-Anfrage auszuführen, während die Zweite zum Build-Prozess aktiv wird. Somit können Seiten je nach Bedarf nach dem Ansatz des Server Side Renderings bzw. der Static Site Generation zu implementiert werden.³⁹

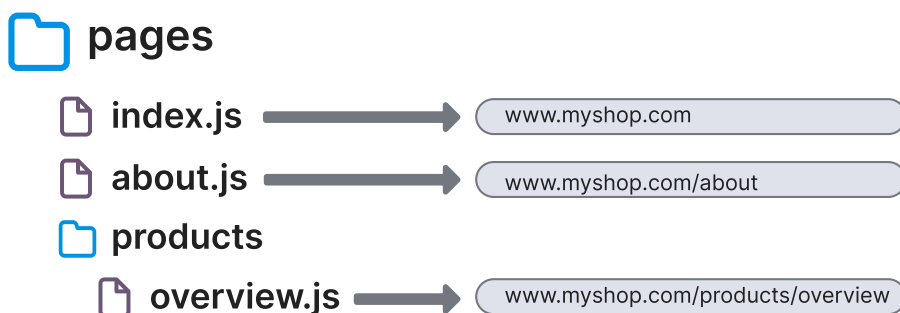
Während React typischerweise für die Entwicklung von Single Page Apps genutzt wird, bietet Next.js von Haus aus ein Routing-System, das es Webdevelopern vereinfacht, mehrere Seiten unter separaten URLs zu erstellen und so Nutzer zwischen diesen navigieren können.³⁸

38 (Thakkar, 2020)

39 (Riva, 2022)

45 (Stoll, 2021)

Einzelne Unterseiten lassen sich hierfür als React-Komponenten in einem bereits definierten Pages-Ordner anlegen und einfach über die URL, die dem Dateinamen entspricht, aufrufen. Eine weitere Besonderheit ist dabei die Link-Komponente, mit deren Hilfe der Wechsel zwischen den Seiten clientseitig über JavaScript durchgeführt wird und nicht wie üblich erneut ein Request des Browsers an den Server gesendet wird.³⁸



◀ **Abbildung 29**

Integriertes Routing in Next.js

Zudem unterstützt Next.js eine ganze Reihe an weiteren Funktionen, die den Entwicklern die Arbeit mit dem Framework erleichtern. Dazu zählen beispielsweise Code-Splitting, automatischer SCSS-Support und sogenanntes *Hot Reloading*, bei dem Änderungen in den Stylesheets oder JavaScript-Dateien direkt im Browser sichtbar werden, ohne einen manuellen Refresh durchführen zu müssen.⁴⁵

2.6.2 Gatsby

Eine beliebte Alternative zu Next.js ist der Static Site Generator Gatsby, dessen Grundgerüst ebenfalls auf React basiert, im Gegensatz zu Next jedoch nur die statische Generierung von Webanwendungen unterstützt. So können die einzelnen Seiten zwar zur Build-Time erzeugt werden und schließlich durch das Ausliefern als statische Assets über ein Content Delivery System nach dem Ansatz des JAMStacks (*siehe 1.4.3*) eine bestmögliche Performance und Ladezeiten ermöglichen. Auf serverseitiges Rendern für hochdynamische Apps mit Echtzeit-Informationen muss jedoch verzichtet werden.³⁹

Ähnlich wie bei Next.js wird Gatsby nicht unbedingt für die Entwicklung von Single Page Applikationen, sondern von umfangreicheren Anwendungen mit verschiedenen Seiten eingesetzt.⁴⁶

³⁸ (Thakkar, 2020)

³⁹ (Riva, 2022)

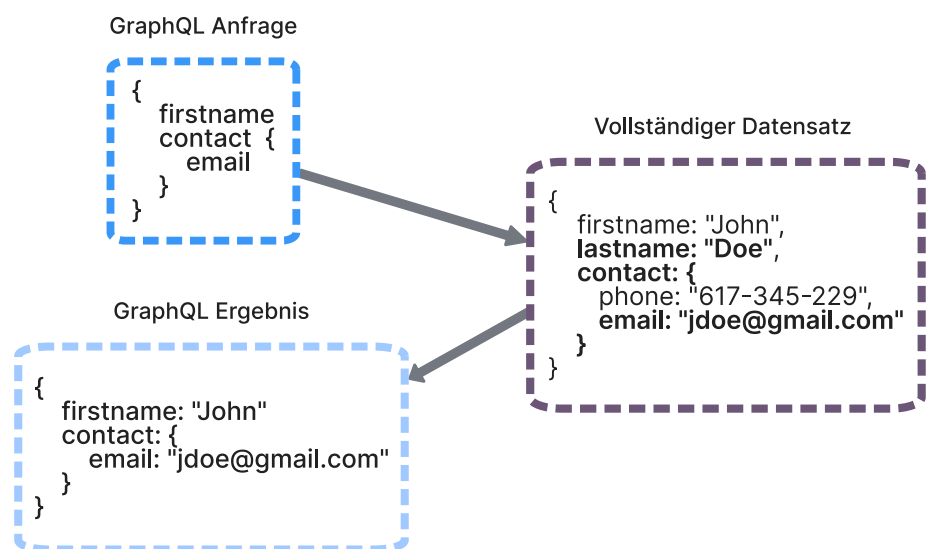
⁴⁵ (Stoll, 2021)

⁴⁶ (Attardi, 2020)

Um eine neue Seite anzulegen, kann hierbei einfach eine neue JavaScript-Datei mit einer React-Komponente im pages-Verzeichnis angelegt werden, die dann über ihren Datei-Namen als URL aufgerufen werden kann.⁴⁶

Eine wichtige Rolle bei Gatsby spielt die Datenabfragesprache GraphQL, um Daten verschiedener Schnittstellen flexibel in die HTML-Templates einbinden zu können. Während bei herkömmlichen Aufrufen einer REST-API immer der komplette Datensatz abgeholt wird, ermöglichen die Befehle in GraphQL, nur eine Teilmenge der Daten zu erhalten, die auch wirklich für die Seite benötigt wird. Die Datenstruktur wird dabei zum Build-Prozess erstellt und kann beispielsweise Markdown-Dateien, Meta-Daten oder Bilder enthalten. Um die Inhalte zu generieren, können die React-Komponenten auf den einzelnen Seiten schließlich die Daten über GraphQL-Befehle abfragen. Für Entwickler, die noch nicht mit der Sprache vertraut sind, erfordert dies zu Beginn etwas Eingewöhnung, kann aber gerade bei komplexen, datenintensiven Anwendungen die Performance verbessern und die Abfrage und Einbindung vereinfachen.⁴⁶

Abbildung 30 ►
GraphQL-Abfrage eines
Datenbestandes



Während es auch möglich ist, Gatsby-Anwendungen *from scratch* aufzusetzen, gibt es hier hilfreiche Boilerplates für viele verschiedene Anwendungszwecke wie beispielsweise Blogs, Portfolio-Webseiten oder E-Commerce-Apps, die bereits benötigte Module installiert, Content Management Systeme verknüpft oder vorgefertigte Layouts definiert haben und Entwicklern dadurch einiges an Erstaufwand abnehmen können. Eine weitere Stärke des Static Site Generators liegt im umfangreichen Ökosystem, das mit seinen vielen verschiedenen Plugins und Themes an Wordpress erinnert.⁴⁶

⁴⁶ (Attardi, 2020)

2.6.3 Nuxt.js

Für Entwickler, die bereits mit dem Frontend-Framework Vue vertraut sind, kann die Open-Source Bibliothek Nuxt.js ein optimaler Ersatz zum React-basierten Next.js sein, von dem sich das Vue-Entwicklerteam sowohl bei der Namensgebung als auch den grundlegenden Features hat inspirieren lassen. So bietet Nuxt ebenso Unterstützung für serverseitiges oder statisches Rendering, ohne dabei hinsichtlich Performance, Suchmaschinenoptimierung oder Workflow in der Implementierung große Abstriche machen zu müssen. Der signifikanteste Unterschied ist dabei die zugrunde liegende Library.³⁹

Weitere nützliche Funktionen des Frameworks sind ein automatisches Routing, Code Splitting, bei dem nur diejenigen Komponenten geladen werden, die gerade benötigt werden, die Möglichkeit zur SEO-Optimierung der Seiten und einige hilfreiche Plugins zur Erweiterung um beispielsweise PWA-Funktionalitäten oder eine Nutzerauthentifizierung.⁴⁷

2.6.4 Hugo

Anders als die bereits vorgestellten Frameworks baut Hugo, wie sich aus dem Namen ableiten lässt, nicht auf JavaScript, sondern auf der von Google entwickelten Programmiersprache Go auf. Ähnlich wie bei Gatsby handelt es sich um einen reinen Static Site Generator, das Markdown-Dateien durch vordefinierte Templates laufen lässt und dem Entwickler die fertigen HTML-Seiten bereit zum Hosten ausgibt.⁴⁸

Eine Besonderheit des Open-Source-Projekts, das sich selbst als „*the worlds fastest framework for building websites*“ bewirbt, ist tatsächlich dessen Performance bei der Erstellung der statischen Seiten. So benötigt Hugo nur ungefähr 1ms für die Generierung einer HTML-Page und kann somit beispielsweise eine 10.000-seitige Webanwendung in nur zehn Sekunden rendern, was deutliche Geschwindigkeitsvorteile gegenüber oben beschriebenen Technologien bietet.⁴⁹

Hugo ist jedoch nicht nur performant, was den Build-Prozess der Webseiten angeht, sondern lässt sich auch schnell einrichten. Da der Static Site Generator als eigenständige, ausführbare Datei ausgeliefert wird, kann dieser ohne Probleme auf jedem Betriebssystem installiert werden.⁴⁸

39 (Riva, 2022)

47 (Tappert, 2020)

48 (Li, 2021)

49 (Hugo, kein Datum)

Entwickler müssen sich dabei nicht um weitere Pakete oder Abhängigkeiten kümmern, und es wird nicht wie bei Nuxt oder Next.js eine JavaScript-Laufzeitumgebung wie Node benötigt.⁴⁸

Standardmäßig bietet Hugo eine übersichtliche Ordnerstruktur, in der die Inhalte, typischerweise als Markdown-Datei, Daten verschiedener Formate wie JSON, statische Medien, vorgefertigte Themes, HTML-Layouts oder Stylesheets angelegt und verwaltet werden.⁴⁸

Zur Erstellung dynamischer HTML-Seiten werden hier keine besonderen Programmierkenntnisse oder Vorwissen in JavaScript-Frameworks benötigt. Um das HTML später zur Generierung der Seiten mit Inhalten anzureichern, werden Variablen genutzt und diese als Platzhalter in geschweiften Klammern in den Elementen platziert. Wie das Beispiel in *Abbildung 31* zeigt, wird dann zur Erstellungszeit automatisch der Titel der Inhaltsvariable erfasst und als Überschrift zwischen den beiden `<h1>`-Tags eingesetzt.⁴⁹

Abbildung 31 ►
Dynamische HTML-Templates in Hugo

Content	Template	Seite
<pre>{ title: "Mein Blog", ... }</pre>	<pre><h1>{{.Title}}</h1></pre>	<pre><h1>Mein Blog</h1></pre>

⁴⁸ (Li, 2021)

⁴⁹ (Hugo, kein Datum)

Content Management Systeme

Um mit seiner Website möglichst viele Besucher zu erreichen, werden ansprechende Inhalte benötigt. So bieten Bilder, Videos und Grafiken einen Mehrwert für die Leser und User. Zudem werden diese, wie bereits im ersten Kapitel beschrieben, positiv von Suchmaschinen registriert. Doch diese Inhalte müssen erst einmal erstellt, verwaltet, publiziert sowie gegebenenfalls in regelmäßigen Abständen aktualisiert und ausgetauscht werden. Für erfahrene Webentwickler ist es keine Herausforderung, mit einfachem HTML und CSS verschiedenste Inhalte im Web bereitzustellen und zu veröffentlichen.⁵⁰ Gerade für Personen mit eingeschränkten Programmierkenntnissen und kleinere Unternehmen ohne die Ressourcen für das Engagement eines Webdevelopers oder einer externen Agentur stellen *Content Management Systeme*, kurz CMS, eine gute Lösung dar. Hierbei handelt es sich um eine Software, mit der die Anwender Multimedia-Inhalte oder Texte gemeinschaftlich über eine Benutzeroberfläche einpflegen können, ohne dass sich diese direkt mit dem Quellcode auseinandersetzen müssen.⁴⁶

In diesem Kapitel soll daher zunächst der traditionelle Ansatz mit herkömmlichen monolithischen Content Management Systemen vorgestellt werden, der die letzten Jahre oder sogar Jahrzehnte des Webs maßgeblich geprägt hat. Demgegenüber steht das noch recht junge Konzept der Headless CMS, auf dessen Grundidee sowie dessen Vor- und Nachteile im Vergleich zum klassischen CMS genauer eingegangen wird und verschiedene Vertreter der Gattungen in diesem Abschnitt vorgestellt werden.

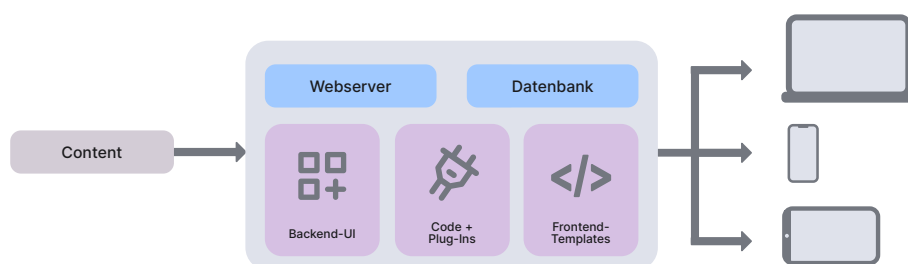
⁵⁰ (IONOS, 2021)

⁴⁶ (Attardi, 2020)

3.1 Traditionelle CMS

Die ersten Content Management Systeme kamen Anfang der 2000er Jahre auf dem Markt, mit Hilfe derer Nutzer im Web 2.0 nicht nur statische HTML-Seiten konsumierten, sondern auch immer mehr selbst zu Content-Produzenten werden konnten. Mittlerweile sind diese CMS kaum mehr aus der Welt des Webs wegzudenken. So basieren laut einer Erhebung von W3Tech mittlerweile über 65 Prozent der im Internet öffentlich zugänglichen Webseiten auf einem Content Management System, wobei der wohl bekannteste Vertreter Wordpress mit knapp 43 Prozent den mit Abstand größten Anteil ausmacht.⁵¹

Grundlegend handelt es sich bei einem traditionellen CMS um eine große zusammengesetzte Webanwendung, die sowohl für die Verwaltung und Speicherung der Medien als auch die Darstellung dieser Inhalte für den Nutzer zuständig ist.⁴⁶ Diese Art von Content Management Systemen basiert in der Regel auf einer monolithischen Architektur, bei der das Backend und Frontend eng miteinander verknüpft und genau aufeinander abgestimmt sind. Ist das CMS einmal eingerichtet und Design der Seiten festgelegt, können recht einfach Inhalte, Beiträge und verschiedene Assets im Backend eingefügt, aktualisiert sowie ganze Seiten erstellt werden, ohne dass dafür besondere Fähigkeiten in HTML, CSS, JavaScript oder einer weiteren Programmiersprache vorhanden sein müssen. Die klassischen Ansätze basieren oftmals auf dem LAMP-Stack (*siehe 1.4.1*), bei denen der Content in einer verknüpften MySQL-Datenbank gespeichert wird, bei einer Seitenanfrage abgerufen und anschließend dynamisch über PHP, zur Darstellung für die Besucher im Frontend, in vorgefertigte Templates eingefügt werden.⁵¹



◀ Abbildung 32

Aufbau und Funktionsweise eines traditionellen CMS

46 (Attardi, 2020)

51 (IONOS, 2022)

Mittlerweile gibt es eine riesige Auswahl von weit über 200 verschiedenen CMS auf dem Markt, die teils unterschiedliche Zielgruppen ansprechen und verschiedene Lösungsansätze für diverse Anwendungsfelder bereitstellen. Das Angebot reicht dabei von simplen Blog-Anwendungen bis hin zu komplexen Enterprise-Systemen.⁵¹ Für einfache Webanwendungen oder Unternehmensseiten erfreuen sich sogenannte Homepage-Baukastensysteme, zu deren bekanntesten Vertretern Square-space oder Jimdo zählen, weiterhin hoher Beliebtheit. Mit Funktionen wie einfachen Content-Editoren, vorgefertigten Themes und Styles oder diversen Plugins zur Erweiterung lassen sich so ohne großen Programmieraufwand anschauliche und moderne Webauftritte oder Blogs realisieren, auch wenn diese von Entwicklern häufig für den unsauberen, automatisch generierten Code und die umständliche Implementierung von zusätzlichen Features kritisiert werden. Quelloffene Webshop-Lösungen wie Shopify bieten als CMS E-Commerce-Lösungen, über die sich mit kurzer Einrichtungsdauer einfache Onlineshops realisieren lassen.⁵²

Viele Anbieter stellen ihre Programme dabei als Open-Source-Software frei zugänglich zur Verfügung. Es gibt jedoch auch proprietäre Ansätze, bei denen das dahinterstehende Unternehmen die Lizenzen zur Nutzung als kommerzielles Produkt vertreibt.⁵¹



Vorteile

- Traditionelle Content Management Systeme ermöglichen es, auch ohne Programmierkenntnisse Webseiten und Anwendungen zu erstellen und zu verwalten. Der Aufwand bei der Installation und Einrichtung ist dabei relativ gering.⁵¹
- Über vorgefertigte Themes oder WYSIWYG-Editoren (*What you see is what you get*) lassen sich die Webseiten, die den Usern im Frontend ausgeliefert werden, einfach über das Backend designen, wobei der CSS-Code automatisch generiert wird.⁵³
- Über große Bibliotheken mit verschiedenen Plug-Ins können die Content Management Systeme um gewünschte Funktionalitäten wie beispielsweise Bildergalerien, Social-Media-Verknüpfungen oder SEO-Tools erweitert werden.⁵³

⁵¹ (IONOS, 2022)

⁵² (Wyllie 2019)

⁵³ (T3N, 2021)

- Heutzutage gibt es eine Vielzahl an verschiedenen Angeboten für unterschiedliche Anwendungsfelder, die häufig als Open-Source-Software ohne Lizenzgebühren eingesetzt werden können und somit für die Unternehmen Kosten eingespart werden.⁵¹



Nachteile

- Aufgrund ihrer monolithischen Architektur sind die Inhalte im Backend in der Regel an ein einziges Frontend gebunden. Andere, neuere Endgeräte wie beispielsweise Smartwatches, Internet-of-Things-Anwendungen oder zugehörige Apps müssen separat bedient werden.⁵¹
- Die abgeschlossene Struktur resultiert in einer geringen Flexibilität und limitiert die Auswahl an einsetzbaren Technologien für Entwickler. Das wirkt sich negativ auf die Skalierbarkeit aus, was zu eingeschränkter Stabilität und Performanceverlusten bei hohem Traffic führen kann.⁵¹
- Werden keine regelmäßigen Updates des CMS oder der genutzten Erweiterungen und Plug-Ins durchgeführt, offenbaren die Anwendungen oftmals Sicherheitslücken und sind ein beliebtes Ziel für Angreifer.⁵⁰
- Trotz der inzwischen weitreichenden Möglichkeiten sind traditionelle CMS in immer noch in ihren Funktionen limitiert. Gerade bei komplexen Anwendungen sind spezielle Anforderungen und gewünschte Features nur schwer umsetzbar.⁵¹

3.2 Headless CMS

Im Gegensatz zu den klassischen Content Management Systemen dienen Headless CMS nur zur Speicherung und Verwaltung der Daten und Inhalte, es ist jedoch keine Präsentationsschicht mit inbegriffen. Das System ist, wie die Bezeichnung schon erkennen lässt, kopflos und umfasst nur das Backend, nicht jedoch wie in der herkömmlichen Variante ein fest gekoppeltes Frontend, bei dem die Inhalte dem Nutzer automatisch aufbereitet und als Webseite zur Verfügung gestellt werden.⁵¹

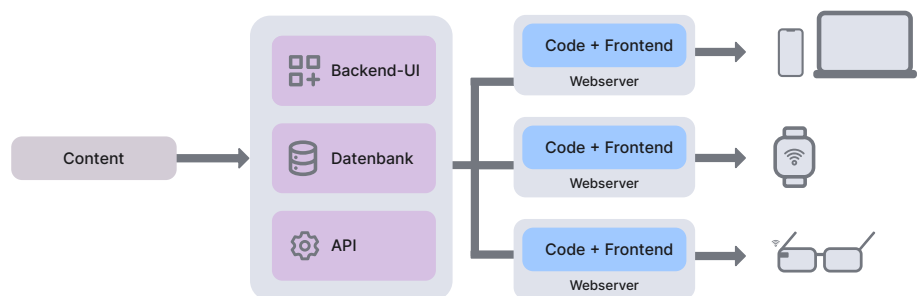
⁵⁰ (IONOS, 2021)

⁵¹ (IONOS, 2022)

Genau wie die traditionellen Varianten bieten Headless CMS den Administratoren ein grafisches User Interface, über das der Content editiert und in einer verknüpften Datenbank gesichert werden kann.⁵¹

Die gespeicherten Inhalte werden bei diesem Ansatz jedoch nicht nur einer einzigen, verknüpften Webseite zur Verfügung gestellt, viel mehr dient das Backend hierbei als universelle Schnittstelle. Über eine vom CMS bereitgestellte API kann der Content von verschiedenen digitalen Plattformen konsumiert und somit nicht nur von Webseiten für Desktop- oder Mobilgeräte, sondern beispielsweise auch von nativen Apps, Smartwatches oder Internet-of-Things-Anwendungen abgerufen werden. Aus technischer Sicht werden hier Schnittstellenaufrufe genutzt, um die Inhalte für die User zu rendern, anstatt direkte Beziehungen zwischen Content und Code herzustellen. Dadurch wird auch die Wiederverwendung der zentral verwalteten Inhalte für verschiedene Kanäle im Vergleich zu den herkömmlichen Systemen erleichtert.⁵³

Abbildung 33 ►
Aufbau und Funktionsweise eines Headless CMS



Die Entkopplung von Backend und Frontend, man spricht auch von einem *Decoupled CMS*, bietet Entwicklern mehr Flexibilität bei der Wahl der eingesetzten Technologie zur Erstellung der teils verschiedenen Frontends, sodass oftmals modernere Frameworks genutzt sowie bessere Ladezeiten und eine optimierte User Experience erreicht werden können. So eignen sich die Headless CMS auch sehr gut für eine Integration in den Ansatz des JAMStacks (siehe 1.4.3), bei dem externe Inhalte über standardisierte REST-APIs abgerufen und zu statisch generierten Seiten verarbeitet werden.⁵¹

51 (IONOS, 2022)

53 (T3N, 2021)

Je nach Art des Hostings lassen sich die verschiedenen Systeme in zwei Kategorien unterteilen. Bei sogenannten *On-Premise-CMS* wird die Software lokal auf der unternehmenseigenen Hardware installiert, gehostet und ausgeführt. Mit diesem System-Typ fällt die Verantwortung auf die Anwender selbst, sich um die Updates, Infrastruktur und hauseigene oder externe Servermöglichkeiten zu kümmern. Zugleich bietet diese Möglichkeit jedoch die volle Kontrolle über die Datenbestände, mehr Freiheit im Management oder die Erweiterung der Backend-API und der Administrationsoberfläche. Auf der anderen Seite können Headless CMS als *Software-as-a-Service*-Angebote konzipiert sein, wobei das System als Service in einer Cloud-Umgebung bereitgestellt wird und so der Verwaltungsaufwand für die Unternehmen auf die Seite des Anbieters verlagert wird und dieser für Infrastruktur, Hosting und Wartung oder Updates zuständig ist.⁵⁴

In den letzten Jahren hat sich die Anzahl der verschiedenen Optionen deutlich erhöht, sodass mittlerweile eine ganze Reihe an unterschiedlichen Headless CMS, teils als Open-Source-Software, den Anwendern zur Verfügung stehen. Neben den bereits beschriebenen API-gestützten Systemen verfolgen andere Vertreter der Headless CMS einen Git-basierten Ansatz, bei dem die Inhalte und Assets nicht in einer Datenbank gespeichert und über eine API abgerufen, sondern in einem Repository verwaltet und aktualisiert werden.⁴⁶



Vorteile

- Während bei traditionellen CMS das Backend fest mit einem einzigen Frontend gekoppelt ist, können bei Headless CMS über eine einzige zentrale Schnittstelle die Inhalte auf mehreren Plattformen, von klassischen Webseiten über iOS- und Android-Apps und Smart-Tvs bis hin zu Virtual Reality Devices, ausgespielt werden.⁵⁵
- Webentwickler sind bei der Nutzung eines kopflosen CMS im Backend deutlich flexibler in der Wahl der Technologie des entsprechenden Frontends, sodass diese je nach Projekt und Anforderungen passende Frameworks und Tools frei wählen können, während dies bei der monolithischen Variante durch die feste Verknüpfung des Frontends nur schwer möglich ist.⁵⁶

46 (Attardi, 2020)

54 (Upsilon IT, 2021)

55 (Nagels, 2019)

56 (Luksha, kein Datum)

- Die Trennung von Frontend und Backend sorgt für eine bessere Skalierbarkeit der Anwendungen. Auch bei Wartungen oder Ausfall eines Frontend ist in der Regel nicht das ganze System betroffen.⁵⁶
- Im Gegensatz zur monolithischen Architektur bieten Anwendungen, die auf einem Headless CMS aufbauen, seltener Sicherheitslücken. Da eine einzelne API deutlich weniger Angriffspunkte als eine komplette und gekoppelte Webseite bietet, wird das Risiko von Angriffen deutlich verringert.⁵⁵



Nachteile

- Im Gegensatz zu traditionellen CMS, die eine Vielzahl an verschiedenen Templates und vorgefertigten Designs zur Verfügung stellen, muss bei einem Headless CMS das entsprechende Frontend selber aufgesetzt und gestaltet werden. Dies ist in der Regel mit einem erheblichen Mehraufwand verbunden. Zudem werden Programmierkenntnisse zur Entwicklung benötigt.⁵⁶
- Während sich die Einrichtung eines herkömmlichen CMS recht schnell und kostengünstig lösen lässt, werden hier durch zusätzliche Infrastruktur wie einen separaten Webserver für das Frontend oder ein eigenes Entwicklerteam mehr Ressourcen und Zeit veranschlagt.⁵⁶
- Bei einem Headless CMS erhalten die Content-Creator keine direkte Live-Vorschau der neuen und aktualisierten Inhalte, da hier das CMS lediglich zur Verwaltung und nicht zur Darstellung für die User dient.⁴⁶

46 (Attardi, 2020)

55 (Nagels, 2019)

56 (Luksza, kein Datum)

3.3 Vergleichender Überblick

Anhand der in den vorherigen Kapiteln vorgestellten Vor- und Nachteile der verschiedenen Arten von Content Management Systemen wird deutlich, dass durchaus beide Varianten ihre Daseinsberechtigung haben. So besteht kein Zweifel daran, dass sowohl klassische als auch Headless CMS zumindest in näherer Zukunft weiterhin eine wichtige Rolle im Bereich der Webentwicklung einnehmen werden. Die Entscheidung, welcher Lösungsansatz der richtige ist, hängt dabei auch immer stark vom Einsatzgebiet und der Art des Projekts ab.⁵⁷

Headless CMS bieten dabei die fortschrittlichere und modernere Lösung, insbesondere dann, wenn Unternehmen planen, mehr als nur eine einfache Webseite mit Inhalten zu versorgen und auf Multichannel-Angebote für ihre Kunden und Nutzer setzen wollen. Die kopflosen Systeme stehen für mehr Flexibilität, eine bessere Skalierbarkeit und eine einfache Integration von verschiedenen digitalen Plattformen. Gleichzeitig erfordern sie mehr Ressourcen, eine aufwendigere Integration und Entwickler, die sich mit modernen JavaScript-Frameworks auskennen.⁵⁷

Traditionelle CMS sind einfacher einzurichten und zu verwalten, da sie Frontend und Backend in einer einzigen Plattform zusammenfassen. Gerade als unkomplizierte Lösung, um beispielsweise einen einfachen Blog oder eine kleine Homepage einzurichten, ist die herkömmliche Variante weiterhin eine gute Option. Durch die inzwischen zahlreichen vorgefertigten Themes und Plug-ins, kann auch ohne Programmierkenntnisse in kurzer Zeit eine funktionsfähige und zeitgemäße Webseite aufgebaut werden.⁵¹

	Traditionelles CMS	Headless CMS
Architektur	Frontend und Backend	Backend und API
Features	Contentspeicherung, Verwaltung, Editieren, Veröffentlichen + Design	Contentspeicherung, Verwaltung, Editieren + Bereitstellung
Ressourcen	Weniger Aufwand bei Einrichtung, keine zusätzlichen Entwickler benötigt, geringe Hostingkosten	Aufwendige Implementierung des Frontends durch Webentwickler, zusätzliche Hostingkosten
Skalierbarkeit	Schwer zu skalieren	Gute Skalierbarkeit, kann mit mehr Workload umgehen
Performance	In der Regel langsamer und häufiger mit Sicherheitslücken	Hohe Performance und weniger anfällig für Angriffe
Flexibilität	Schwer durch eigene Erweiterungen anzupassen, mangelnde Flexibilität	Durch entkoppelte Struktur kann Technologie für Frontend frei gewählt werden

◀ Abbildung 34

Vergleich zwischen einem Headless CMS und einem traditionellen CMS

⁵¹ (IONOS, 2022)

⁵⁷ (Boyarko, 2021)

Ein herkömmliches monolithisches CMS sollte dabei am besten eingesetzt werden, wenn... ⁵⁷

- es sich um eine simple Webseite handelt, bei der vorgefertigte Templates genügen.
- das Budget des Unternehmens limitiert ist.
- der Content nur auf einer einzigen Webseite ausgespielt werden muss.
- mit einer einzigen Plattform die volle Kontrolle über den Inhalt, von Verwaltung bis zur Darstellung, benötigt wird.

Der Einsatz von einem Headless CMS macht vor allem dann Sinn, wenn... ⁵⁷

- moderne JavaScript Frameworks oder Static Site Generators für die Entwicklung der Webseite oder Anwendung eingesetzt werden.
- der Content vom Technologie-Stack des Frontends getrennt werden soll, um die Performance zu verbessern.
- die Nutzer neben einer Webseite über weitere digitale Kanäle mit Inhalten bedient werden sollen.
- spezielle Features oder komplexe Anwendungen durch ein vorhandenes Entwicklerteam umgesetzt werden können.

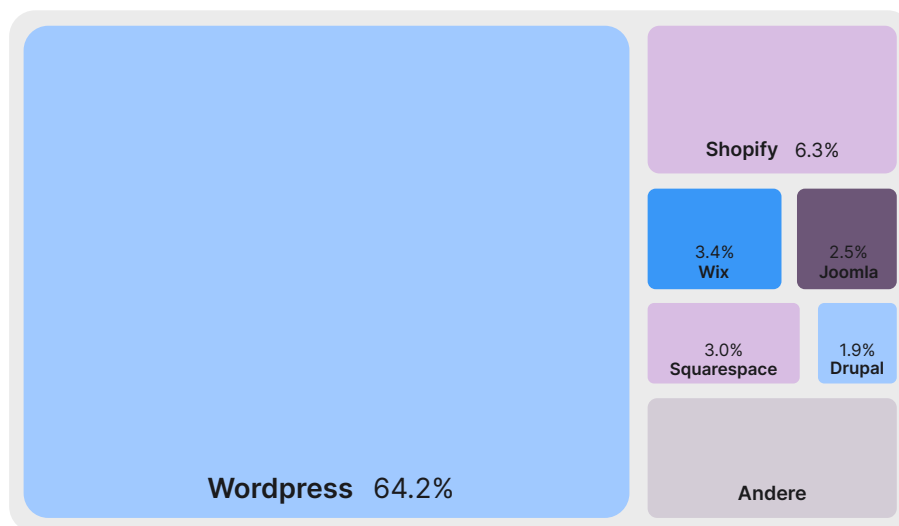
3.4 Technologieansätze

Im Folgenden sollen verschiedene Vertreter von zunächst den traditionellen und anschließend entkoppelten Content Management Systemen vorgestellt werden, die jeweils verschiedene Lösungsansätze verfolgen und sich für unterschiedliche Anwendungszwecke und Aufgabengebiete eignen.

⁵⁷ (Boyarko, 2021)

3.4.1 Traditionelle CMS

Wie bereits in *Kapitel 3.1* erläutert, ist Wordpress nach einer Erhebung von W3Tech mit weitem Abstand und einem Anteil von nahezu 65 Prozent Marktführer im Bereich der Content Management Systeme, was mit Blick auf das gesamte Web sogar knapp 43 Prozent aller Seiten ausmacht. Dahinter folgt mit einem vergleichsweise geringen Anteil Shopify und weitere Homepage-Baukästen zur Erstellung und Administration von einfachen Webseiten. Neben Wordpress und Shopify soll im Folgenden allerdings zudem auch das Enterprise-CMS TYPO3 detaillierter beleuchtet werden, da dies vor allem im deutschsprachigen Raum bei professionellen Online-Projekten eine beliebte Alternative ist.⁵⁸



◀ **Abbildung 35**

Marktanteile verschiedener Content Management Systemen nach einer Erhebung von W3Tech

Wordpress

Die Open-Source Software Wordpress, ursprünglich als simples Blogsystem konzipiert, ist das mit Abstand am öftesten eingesetzte CMS im Web. Die riesige Community profitiert dabei von einer großen Menge an Plugins und Designvorlagen. Kritische Stimmen bemängeln jedoch, dass Wordpress dadurch etwas schwerfällig und unübersichtlich geworden sei. Zudem ist das Management System aufgrund seiner enormen Verbreitung auch immer wieder Ziel von Hackerangriffen, sodass die Administratoren stets alle Updates durchführen und die Erweiterungen auf dem neusten Stand halten sollten.⁵⁰

⁵⁰ (IONOS, 2021)

⁵⁸ (W3Tech, 2022)

Durch den geringen Einrichtungsaufwand eignet sich das CMS, welches technisch auf einer MySQL-Datenbank und der Programmiersprache PHP aufsetzt, jedoch weiterhin optimal für kleine und wenig umfangreiche Webseiten mit regelmäßig wechselnden Inhalten, bei dem durch die übersichtliche Oberfläche auch ohne technisches Hintergrundwissen modern wirkende responsive Seiten zu erstellen und deren Content zu verwalten.⁵⁰

Mit zunehmender Komplexität der Projekte oder steigendem Besucheraufkommen nimmt jedoch die Usability und die Performance der Software deutlich ab. Theoretisch lassen sich auch umfangreiche Enterprise-Anwendungen mit Wordpress realisieren, zusätzliche und spezielle Funktionen gehen aber mit einem stark zunehmenden Aufwand einher.⁵⁰

TYPO3

Nach Wordpress gehört auch TYPO3, vor allem im deutschsprachigen Raum zu den beliebtesten lizenzfreien Software-Projekten. Während Wordpress jedoch eher für kleinere Unternehmensseiten oder Blogs gedacht ist, versteht sich TYPO3 als Enterprise-Management-System, das vor allem für große Unternehmensportale oder E-Commerce-Anwendungen eine optimale Lösung darstellt.⁵⁰

Technisch basiert das TYPO3, ähnlich wie viele andere klassische CMS, auf einer MySQL-Datenbank, PHP als Middleware sowie einem Apache Webserver und überzeugt ebenso wie Wordpress durch seine große Community, die es gerade Einsteigern erleichtert, sich in der recht komplexen Software zurechtzufinden. Der Einsatz des leistungsstarken CMS erfordert definitiv Fachkenntnisse, einen höheren Administrationsaufwand und eine längere Einarbeitungszeit, bietet dafür aber auch einen umfangreichen Funktionsumfang, um selbst komplexe Business-Anwendungen optimal umzusetzen. Die weite Verbreitung und der gute Ruf von TYPO3 tragen dazu bei, dass dies oftmals auch bei kleineren Online-Projekten und Webseiten zum Einsatz kommt. Gerade im Vergleich zu anderen Vertretern muss hier jedoch ein größerer Mehraufwand für Installation und Wartung berücksichtigt werden.⁵⁰

50 (IONOS, 2021)

Shopify

Bei Shopify handelt es sich, anders als bei den bereits vorgestellten Vertretern, nicht um ein typisches Content Management System, sondern um ein spezielles Shopsystem für den Einsatz bei E-Commerce-Anwendungen, das insbesondere für kleinere bis mittelgroße Unternehmen ausgerichtet ist, die in einem Online-Shop Produkte verkaufen wollen. Shopify ist zudem nicht Open-Source, sondern bietet den Nutzern als proprietäre Software nach einer kostenlosen Testphase verschiedene Angebote, die sich in Funktionsumfang und Preis unterscheiden.⁵⁹

Über Shopify entworfene Shops sind als Software-as-a-Service (SaaS) konzipiert, sodass diese cloudbasiert sind und vollständig über einen Browser verwaltet werden können. Das CMS setzt sich dabei ebenfalls aus einem Backend und einem integrierten Frontend inklusive abgesicherter Checkout- und Bezahlungsfunktion, zusammen, das über verschiedene kostenlose oder kostenpflichtige Themes und Anpassungsmöglichkeiten individuell gestaltet werden kann. Neben den vorgefertigten Design-Templates bietet Shopify in einem eigenen Store zusätzlich verschiedene hilfreiche Apps an, über die Shopbetreiber Sale- und Rabattaktionen automatisieren, ihren Social Media-Auftritt mit dem Shop verknüpfen oder sich bei der Suchmaschinenoptimierung unterstützen lassen können.⁶⁰

Wirft man einen Blick unter die grafische Benutzeroberfläche des Shop-Systems, so erkennt man eine auf der Programmiersprache Ruby basierende Applikation, die auf eine Redis und MySQL-Datenbank zurückgreift und durch die JavaScript-Library React erweitert wird. Zur zentralen Verwaltung werden die Shop-Komponenten über APIs miteinander verknüpft, die aktuell zwar noch die Arbeit mit REST erlauben, für zukünftige Projekte wird jedoch die Verwendung von GraphQL (*siehe Abbildung 30*) empfohlen.⁶⁰

3.4.2 Headless CMS

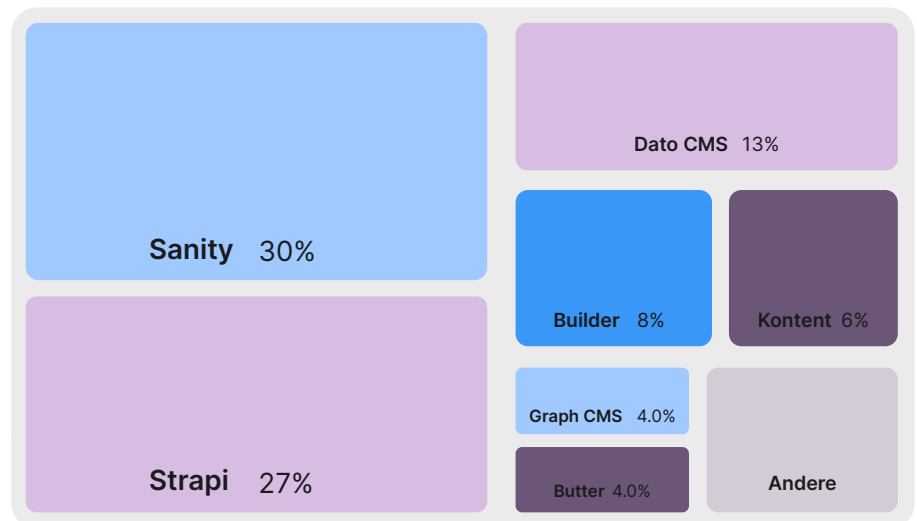
Aufgrund der noch recht jungen Vergangenheit der Headless CMS und dem daher aktuell noch, im Vergleich zu den herkömmlichen Management System, geringen Anteil auf dem Markt gibt es derzeit keine aussagekräftigen Statistiken über die tatsächliche Nutzung der verschiedenen Varianten in der Praxis.

59 (IONOS, 2022)

60 (Zenner, 2020)

Da sich das Angebot in den letzten Jahren jedoch immer mehr erweitert hat, lohnt sich ein Blick auf die Trends-Kategorie der Seite *Built-With.io*, die den Einsatz von Headless CMS bei den 1 Mio. meist besuchten Webseiten im Web analysiert.

Abbildung 36 ►
Einsatz von verschiedenen Content Management Systemen nach einer Erhebung von Built-With



Sanity.io

Bei Sanity.io handelt es sich um eine Softwarelösung, die im Gegensatz zu cloudbasierten SaaS-Programmen lokal installiert und ausgeführt wird. Dabei versteht sich Sanity in erster Linie als Plattform zur Verwaltung von strukturierten Inhalten und nicht als reines CMS. Sanity.io ist zwar nicht vollständig Open-Source, bietet neben verschiedenen Preispaketen je nach Unternehmensgröße ebenfalls eine kostenfreie Version an, die auch erst später bei Bedarf aufgerüstet werden kann.⁶¹

Das Herzstück der Software ist die eigens entwickelte Datenbank *Content Lake*, die zur Speicherung aller Daten dient und diese in Echtzeit bereitstellt. Sie ist speziell auf die Verwaltung von schemalosem Content ausgelegt und soll deutlich flexibler als zweidimensionale starre Tabellen einer herkömmlichen SQL-Datenbank sein, bei der die Struktur bereits im Voraus festgelegt werden muss. In Sanity können Entwickler so beliebige Daten in verschiedenen Formaten in die Datenbank einfügen oder ändern.⁵⁴

⁵⁴ (Upsilon IT, 2021)

⁶¹ (Pecoraro & Gambino, 2021)

Dafür kommt die speziell entworfene und recht kompakte Abfragesprache *Graph Orientated Query Language*, kurz QROQ, zum Einsatz. Des Weiteren dient eine HTTP-API als Endpunkt, um mit dem Datenbestand interagieren zu können.⁵⁴

Das eigentliche Content Management System heißt *Sanity Studio* und ist im Gegensatz zur kompletten Plattform lizenzfrei nutzbar. Die Single-Page-Applikation basiert auf JavaScript und React und bietet den Anwendern ein modernes User Interface, über das die Inhalte intuitiv eingefügt und gepflegt werden können sowie ähnlich zu einem herkömmlichen CMS, eine Live-Vorschau der Inhalte, wie diese späteren Besucher dargestellt werden. Weitere Vorteile sind eine Echtzeit-Kollaboration von Redakteuren im CMS und die Möglichkeit, über die standardisierte HTTP-Schnittstelle die Technologie und das Framework für verschiedene Frontends frei zu wählen.⁶¹

Strapi

Das beliebte Headless CMS Strapi, dessen Name sich aus dem Wort *Bootstrap API* ableitet, soll nach diesem Konzept Entwicklern helfen, schnell und ohne großen Einrichtungsaufwand eine Schnittstelle zu entwerfen. Durch ein benutzerfreundliches, übersichtliches Admin-Panel und einige hilfreiche Kernfunktionen fällt Redakteuren, Frontend- oder Backendentwicklern die Einarbeitung recht einfach.⁶²

Strapi ist ein Open-Source CMS, das technisch auf JavaScript mit Unterstützung der Laufzeitumgebung Node.js (*siehe 1.3.1*) aufbaut. Der Code ist dabei öffentlich auf GitHub zugänglich und somit leicht erweiterbar. Dazu erlaubt das Headless CMS die Arbeit mit verschiedenen SQL oder No-SQL-Datenbanksystemen wie beispielsweise SQLite, PostgreSQL oder MongoDB. Auch sonst ist das System in hohem Maße konfigurierbar, um den spezifischen Anforderungen eines jeden Projekts gerecht zu werden. So können alle Datentypen von der grafischen Nutzeroberfläche erstellt werden. Über das hauseigene Plugin-System lässt sich die Anwendung um Features wie beispielsweise Datenbank-dokumentationen, eine Medienbibliothek oder E-Mail-Konfiguration erweitern.⁶²

54 (Upsilon IT, 2021)

61 (Pecararo & Gambino, 2021)

62 (Elshafie & Haider, 2022)

Für die Anbindung verschiedener Frontends bietet Strapi eine standardmäßig eingerichtet REST-API, die von nahezu jedem Webframework wie React, Vue und Angular, nativen Apps oder selbst Internet-of-Things-Anwendungen konsumiert werden kann und sich durch das Nutzer- und Berechtigungsmodell des CMS detailliert festlegen lässt, welche Aktion bei welchem Endpunkt durchgeführt werden darf. Bei Bedarf lässt sich die API des CMS durch ein weiteres Plugin ebenfalls um eine GraphQL-Anbindung ergänzen, um gerade große Datenbestände effizienter abfragen zu können.⁶²

Contentful

Im Gegensatz zu Strapi oder Sanity ist das Headless CMS Contentful, als *System-as-a-Service* konzipiert, das somit vollständig in einer Cloud-Umgebung läuft. Der Anbieter kümmert sich hier um alle Fragen des Hostings, der Wartung oder um Updates der Anwendung. Contentful nutzt zur Bereitstellung der Inhalte eine RESTful-API, über die Entwickler alle Assets und Medien abrufen können, ohne dabei an ein bestimmtes Framework oder einen Static-Site-Generator gebunden zu sein.⁵⁴

Das CMS lässt dabei die Content-Modelle individuell und ohne Vorgaben definieren oder zu Gruppen anordnen und über verschiedene Kanäle in unterschiedlichen Sprachen ausspielen. Contentful ist nicht Open-Source, lässt sich aber in kleinem Umfang auch ohne zusätzliche Kosten nutzen. Gerade größere Unternehmen mit komplexeren Softwareprojekten können für Preise von 35 Euro bis 800 Euro im Monat auf eine erweiterte Version zurückgreifen.⁵²

Netlify

Netlify CMS ist ein Open-Source System des gleichnamigen Cloud-Hosting Anbieters Netlify. Trotz des Namens ist das CMS nicht zwingend an diese Plattform gebunden, es ist jedoch deutlich einfacher, dies dort zu deployen und zu hosten. Während bei anderen Headless CMS die Inhalte in einer zugehörigen Datenbank gespeichert werden, wählt dieses Content Management System einen anderen Ansatz und sichert den Content als Markdown-Dateien in einem zugehörigen Git-Repository.⁴⁶

46 (Attardi, 2020)

52 (Wyllie, 2019)

54 (Upsilon IT 2021)

62 (Elshafie & Haider, 2022)

Dies macht Netlify zu einer idealen Wahl für JAMStack-Anwendungen und Static Site Generatoren, die darauf ausgelegt sind, mit Markdown-Daten zu arbeiten und diese zu rendern.⁴⁶

Bei der grafischen Oberfläche für Administratoren handelt es sich um eine in React basierte Single-Page-App mit einem integrierten Text-Editor, sodass die Redakteure keinen eigenen Markdown-Code schreiben müssen, sondern dieser automatisch generiert wird. Auch wenn das User Interface selbst auf Facebooks React basiert, so sind die zugehörigen Frontends und Webseiten in ihrer Technologie nicht eingeschränkt und es lassen sich alle gängigen Static Site Generators oder Frameworks wie Gatsby, Next, Nuxt, Hugo oder Jekyll einsetzen. Neben verschiedenen Frontends unterstützt Netlify CMS zudem unterschiedliche Möglichkeiten wie GitHub, GitLab oder Bitbucket, um die Daten im Backend zu verwalten. Das zugehörige GitGateway erlaubt außerdem den Zugang für das CMS für weitere Editoren, ohne dass diese direkt auf das Git-Repository zugreifen können oder einen jeweiligen Account benötigen.⁴⁶

Nebenbei überzeugt Netlify auch durch eine einfache Installation flexible und uneingeschränkte Content-Modelle. Über benutzerdefinierte React-Komponenten lässt sich auch eine spätere Live-Vorschau der Webseite oder des Blog-Eintrags erhalten.⁴⁶

DatoCMS

Ähnlich wie Contentful wird das DatoCMS als proprietäres SaaS-Modell angeboten und beinhaltet dabei neben dem Hosting auch verschiedene Pakete, die sich je nach Unternehmens- oder Projektgröße preislich unterscheiden. Ein Hauptmerkmal dieses Content Management Systems ist die sehr performante und durchdachte GraphQL-Schnittstelle. Eine Besonderheit dabei ist unter anderem die Einbindung von Bildbearbeitungsparametern in der API, über die in beliebig verschachtelter Tiefe passend transformierte oder responsive Bilder oder auch Videos in responsivem Format mit automatisch erzeugten Thumbnails generiert werden können.⁶³

46 (Attardi, 2020)

63 (Bluepick, 2021)

Abbildung 37 ►
Vergleich verschiedener
Headless CMS

	Lizenz	Technologie	Datenmodell
Sanity	Teilweise O. S.	On-premise	API-Basiert
Strapi	Open Source	On-premise	API-Basiert
Contentful	Kostenpflichtig	SaaS	API-Basiert
Netlify	Open Source	On-premise	Git-Basiert
GraphCMS	Kostenpflichtig	SaaS	API-Basiert
Tina	Open Source	On-premise	Git-Basiert
DatoCMS	Kostenpflichtig	SaaS	API-Basiert
Butter	Kostenpflichtig	SaaS	API-Basiert

3.4.3 Hybride CMS

Klassische CMS wie beispielsweise Wordpress, die über eine REST-API verfügen, lassen über diese Schnittstelle Posts, Medien und weitere Inhalte abfragen und das Content Management System auf diese Weise auch als Headless CMS umfunktionieren. Somit wird dies nur noch als reines Redaktionssystem im Hintergrund genutzt, während das Frontend separat entwickelt wird und sich die Inhalte sogar auf mehreren digitalen Kanälen ausspielen lassen.⁶⁴

Bei einem sogenannten hybriden CMS wird der traditionelle und der entkoppelte Ansatz miteinander vereint, bei dem Entwickler auf der einen Seite die Inhalte über eine Schnittstelle abrufen und in beispielsweise mobilen Apps, Wearables oder Internet-of-Things-Anwendungen einbinden. Auf der anderen Seite kann eine fest gekoppelte Webseite mit vorgefertigten Designs und Templates veröffentlicht werden.⁶⁵

⁶⁴ (Höhn, 2021)

⁶⁵ (Schürmanns, 2022)

Progressive Web Apps

Der Begriff der *Progressive Web App*, oftmals auch nur als PWA abgekürzt, ist ein Technologieansatz, der trotz seiner anfänglich eher schleppenden Verbreitung mittlerweile jedem Webentwickler zumindest einmal zu Ohren gekommen ist. Der Ansatz wird dabei vor allem von Google, mit dem mobilen OS Android und dem Webbrowser Chrome zwei große und weitverbreitete Plattformen im Portfolio, als Anwendungsmodell der Zukunft angepriesen. So sind es die PWAs, die mit den Features, die zuvor nur herkömmlichen Apps vorbehalten waren auf der einen Seite, sowie der plattformübergreifenden Ausführung von Webanwendungen auf der anderen, die Vorteile aus genau diesen beiden Welten vereinen sollen.⁶⁶

In diesem Kapitel soll zunächst ein kurzer Überblick über die nativen Apps gegeben werden, damit die Progressive Web Apps später gegenüber diesen abgegrenzt werden können sowie deren grundlegende Funktionen und technologischer Hintergrund erläutert werden, bevor abschließend der Status-Quo des PWA-Konzepts und einige Beispiele aus der Praxis betrachtet werden.

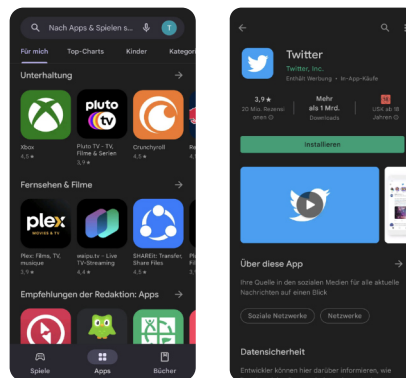
66 (Lieber, 2018)

4.1 Abgrenzung zu nativen Apps

Als native Apps werden Anwendungen bezeichnet, die speziell für die Nutzung auf einem einzigen mobilen oder auch stationären Betriebssystem ausgelegt sind und für die Nutzung auf den jeweiligen Endgeräten separat konzipiert und entwickelt werden. Diese werden in der Regel in den ebenfalls an das jeweilige Betriebssystem gekoppelten App Stores kostenpflichtig oder kostenfrei vertrieben und können hier von den Usern installiert werden.⁶⁷

So werden beispielsweise native Android Apps in Googles *PlayStore* angeboten und können nur auf einem entsprechenden Gerät mit Android-Betriebssystem genutzt werden, während Applikationen für Apples iOS nur im hauseigenen *App Store* bereitgestellt werden und nicht auf einem Android-Smartphone ausgeführt werden können.

Das User Interface der Apps entspricht dabei meist der Design-Sprache des zugrundeliegenden Systems und bietet so den Nutzern eine optimale User Experience und geringe Eingewöhnungszeit.⁶⁷



◀ Abbildung 38

Googles Play Store bietet die Möglichkeit native Apps zu installieren

Durch die native Entwicklung einer Anwendung ist die App zudem auch in technischer Hinsicht optimal auf das Betriebssystem angepasst und diese zeichnen sich neben einer sehr guten Performance auch, aufgrund der Möglichkeit auf die Hardware und Software des OS zuzugreifen, durch umfangreiche Features aus. Da native Apps allerdings speziell für bestimmte Betriebssysteme zugeschnitten sind, müssen diese auch jeweils in unterschiedlichen Programmiersprachen implementiert werden. Dafür wird ein sogenanntes SDK oder auch *System Development Kit* als integrierte Entwicklungsumgebung genutzt. Während für Android-Apps *Android Studio* als SDK zum Einsatz kommt und die Anwendung selbst in den Programmiersprachen Java oder Kotlin geschrieben wird, benötigt man für die Entwicklung von iOS-Apps das Development Kit *XCode*, das wie Android Studio ebenfalls kostenlos zur Verfügung steht. Zur Umsetzung selbst wird schließlich hier Objective-C oder Apple eigens entworfene Sprache Swift genutzt.⁶⁸

⁶⁷ (Bright Solutions, kein Datum)

⁶⁸ (Hahn, kein Datum)

Hier lässt sich bereits erkennen, dass aufgrund der unterschiedlichen Sprachen der Quellcode einer App nicht für beide Plattformen verwendet werden kann und ein deutlicher Mehraufwand in der Entwicklung anfällt, sofern die App sowohl für iOS- als auch Android-Geräte oder zusätzliche Desktop-Systeme bereitgestellt werden soll.⁶⁸



Vorteile

- Eine native App kann auf sämtliche Hard- und Software-Funktionen des zugrunde liegenden Endgeräts zugreifen und so beispielsweise die Kamera, Push-Benachrichtigungen oder GPS nutzen.⁶⁷
- Native Anwendungen zeichnen sich durch eine sehr gute Performance sowie eine gute User Experience aus, sodass die App durch die vorgegebenen Design-Richtlinien des Systems für die Nutzer wie ein fester Bestandteil von diesem erscheint.⁶⁷
- Klassische Apps sind in der Regel recht sicher, da durch Zugriff auf Hard- oder Softwarefunktionen des Gerätes komplexere Sicherheitsalgorithmen implementiert werden können. Zudem werden die Anwendungen vor der Veröffentlichung in den App Stores geprüft.⁶⁹
- Native Apps können problemlos auch bei fehlender oder schlechter Internetverbindung geöffnet und genutzt werden, während dies bei Webanwendungen trotz Browser-Caching nur beschränkt möglich ist. Daten können hier auf dem Endgerät gespeichert werden, das in der Regel mehr Kapazität als der Cache des Webbrowsers bieten.⁶⁸



Nachteile

- Soll die Anwendung für nahezu alle Nutzer als native App verfügbar sein, so muss diese für Apples iOS und Googles Android entwickelt werden, wodurch die größten Nachteile die längere und aufwendigere Entwicklungszeit und die daraus resultierenden höheren Entwicklungskosten und Wartungskosten sind.⁶⁷

67 (Bright Solutions, kein Datum)

68 (Hahn, kein Datum)

69 (Konoplenko, 2021)

- Auch die Fixkosten für native Apps, die für die Verbreitung in den App Stores anfallen, sind in der Regel höher. Zudem müssen bei kostenpflichtigen Anwendungen bis zu 30 Prozent der Einnahmen an Google bzw. Apple als Gebühr abgegeben werden.⁶⁷
- Native Apps werden nicht von Suchmaschinen indiziert und in den organischen Suchergebnissen angezeigt. Neben den verschiedenen Schritten, die zur Veröffentlichung in den Stores durchlaufen werden müssen, fällt so zusätzlicher Aufwand und weitere Kosten für entsprechende Werbedienste oder Kampagnen zur Verbreitung an.⁶⁸

4.2 Entstehung

Im Zeitalter der Smartphones spielt vor allem der Browser als Zugang zum World Wide Web auf den Endgeräten eine zentrale und wichtige Rolle. So kommen Nutzer mit vielen Diensten, die eine zusätzliche native App anbieten, zunächst über eine Webseite in Kontakt, die über Banner oder Hinweismeldungen den User dann zur Installation in den entsprechenden App Store weiterleiten.⁶⁶

Zwar boten sich in dieser Zeit auch für Webanwendungen immer mehr Möglichkeiten und durch die Weiterentwicklung der Browser und dem Aufkommen vieler neuer Webschnittstellen ließen sich Funktionen umsetzen, die zuvor nur nativen Apps vorenthalten waren. Dazu zählen beispielsweise Features wie das Erfassen von Aufnahmen über das Mikrofon oder die Webcam, der Zugriff auf den Standort oder die Nutzung von APIs zur Echtzeitkommunikation. Und doch fehlten Webapps weiterhin einige zentrale Eigenschaften wie Offlinefähigkeit, Pushbenachrichtigungen und Hintergrundaktivität oder die separate Ausführung außerhalb des Browserfensters, um wirklich wie native Apps auszusehen und sich so zu verhalten.⁶⁶

Im Jahre 2015 ist es schließlich Googles Produktmanager Alex Russel, der in einem Blogartikel die Vision des neuen Anwendungsmodells der sogenannten Progressive Web App vorstellt, bei dem das Leben der Applikation zunächst im Browser beginnt und der User bei Interesse eine Verknüpfung der Anwendung auf den Homebildschirm hinzufügen kann.⁶⁶

⁶⁶ (Liebel, 2018)

⁶⁷ (Bright Solutions, kein Datum)

⁶⁸ (Hahn, kein Datum)

Von dort aus aufgerufen, werden PWAs jedoch nicht mehr im Browser geöffnet, sondern wie die nativen Apps in einem eigenen Anwendungsfenster ohne Status oder Menüzeilen und werden auch wie die klassischen Anwendungen im App Switcher oder auf dem Desktop in der Taskleiste angezeigt. Im Hintergrund des nativen Fensters läuft jedoch weiterhin die Browser-Engine.⁶⁶

4.3 Eigenschaften

Wie bereits in den vorherigen Kapiteln angedeutet, bietet das Konzept der Progressive Web Apps zusätzliche Features, die deutlich über die einer herkömmlichen Webseite oder Anwendung hinausgehen. Grundlegend können PWAs dabei als eine Art funktionsreicher Webanwendungen gesehen werden, die, falls es der Browser zulässt, im Look and Feel von klassischen, nativen Apps daherkommen. Dabei lassen sich PWAs nicht nur als reinen Technologieansatz, sondern mehr als übergeordnetes Entwurfsmuster sehen, das bestimmte Eigenschaften von einer Webanwendung fordert, welche im Folgenden genauer erläutert werden.

Responsiveness

Responsives Design beschreibt einen Ansatz für Layouts von Webseiten, bei dem sich die einzelnen Elemente und Inhalte flexibel an die jeweilige Bildschirm- oder Fenstergröße unterschiedlicher Endgeräte anpassen. Da mittlerweile über die Hälfte des weltweiten Internet-Traffics auf mobile Geräte zurückzuführen ist, reicht es nicht mehr aus, eine Webanwendung nur noch für einige große Desktop-Geräte zu gestalten, da ansonsten die User Experience, gerade auf Smartphones, deutlich eingeschränkt ist und sich eine nicht-responsive Seite zudem negativ auf die Suchmaschinenoptimierung auswirkt.⁷⁰

Abbildung 39 ►
Responsives Layout einer
Webseite



66 (Lieber, 2018)

70 (Duò, 2022)

Grundlage für die Responsiveness einer Webseite ist die Kombination aus den Sprachen HTML und CSS (*siehe 1.2*), die Inhalt und Layout in jedem Webbrowser beeinflussen. Eine besondere Rolle spielen die sogenannten CSS-Media Queries, über die sich bestimmte Triggerpunkte (*Breakpoints*) definieren lassen, an denen das Layout umbricht und sich beispielsweise an die Größe eines Smartphone-Displays oder eines Tablets anpasst.⁷⁰

```
1 p{
2   font-size: 12px;
3   margin: 0;
4   @media screen and (min-width: 650px) {
5     font-size: 15px;
6     margin-bottom: 3%;
7   }
8 }
9
```

◀ Codesnippet 5

Responsives Layout mit
CSS Media-Queries

Auffindbarkeit

Da Progressive Web Apps in ihren Kerntechnologien nur herkömmliche Webseiten sind, soll die Eigenschaft *Discoverable* dafür sorgen, dass sich PWAs von diesen unterscheiden lassen. Dazu wurde das Web App Manifest eingeführt, das als JSON-Datei (*siehe 1.1.3*) alle Informationen und Metadaten wie beispielsweise der Titel, Anzeigart, beschreibende Texte und zugehörige Icons zentral zusammenfasst, sodass Suchmaschinen und Browser diese Angaben nutzen und verarbeiten können.⁶⁶

Installierbarkeit

Die Eigenschaft *Installable* beschreibt, dass eine PWA auf dem Gerät installiert werden kann und fortan als Icon auf dem Homebildschirm oder in der Programmliste des Betriebssystems zu finden ist. Damit eine Webanwendung diese Möglichkeit bietet, wird ebenfalls das zuvor beschriebene Web App Manifest benötigt, um eine Verknüpfung auf dem Homescreen anzulegen und die Progressive Web App ähnlich wie native Anwendungen als eigenes Fenster mit voller Fläche und ohne Menüleisten oder anderen Tabs des Browsers laufen.⁶⁶

⁶⁶ (Lieber, 2018)

⁷⁰ (Duò, 2022)

App-ähnlich

Neben der Installierbarkeit soll auch die Anforderung *App-like* dazu dienen, dass PWAs tatsächlich mit nativen Apps konkurrieren können. So können sich zum einen die bereits erwähnten mächtigen neuen Web-APIs zu Nutze gemacht werden, um so unter anderem auf die Kamera und das Mikrofon sowie das Dateisystem des Geräts zuzugreifen, den lokalen Standort abzurufen oder auch hardwarebeschleunigte 2D- und 3D-Inhalte darzustellen. Zudem sollen sich PWAs auch auf dem User Interface ihren nativen Vorbildern annähern und Steuerelemente, Navigationsstrukturen und Layouts einsetzen, die den Usern von klassischen Apps vertraut sind.⁶⁶

Verlinkbarkeit

Eine weitere Eigenschaft von Progressive Web Apps ist, dass diese *Linkable*, d.h. verlinkbar sein müssen und daher wie klassische Webanwendungen über eine URL im Browser aufgerufen werden können. Diese lassen sich im Gegensatz zu nativen Apps auch ohne Installation und Setup ausführen und einfach über einen Link teilen.⁶⁶

Sicherheit

Die nächste Eigenschaft *Safe*, besagt, dass PWAs die Sicherheit von mobilen oder stationären Endgeräten nicht gefährden dürfen. Da Progressive Web Apps in ihrem Kern nur eine Webseite mit bestimmten Zusatzfunktionen ist, bestehen hier die gleichen Gefahren und Risiken und sie unterliegt den gleichen Sicherheitsbestimmungen wie jede andere Internetseite auch.⁶⁶

Für eine risikofreie Auslieferung der Quelldatei sowie eine sichere Übertragung von Anwenderdaten wird das Protokoll HTTPS (*Hypertext Transfer Protocol Secure*) eingesetzt, sodass sich beispielsweise der Server mit einem Zertifikat für den Domainnamen authentifizieren muss und sämtliche Daten, sowohl Quellcode als auch Nutzerinformationen, verschlüsselt übertragen werden.⁶⁶

⁶⁶ (Liebel, 2018)

Verbindungsunabhängigkeit

Durch das Feature *Connectivity Independent* soll gewährleistet werden, dass der User auch bei schwacher oder fehlender Internetverbindung die Anwendung öffnen kann und bei der Durchführung in seiner Kernaufgabe nicht beeinträchtigt werden soll. Das Prinzip der Verbindungsunabhängigkeit wird in der Regel durch einen sogenannten Service Worker (*siehe 4.4.1*) umgesetzt, der außerhalb und parallel zur Clientanwendung ebenfalls noch innerhalb des Webbrowsers läuft und bei Bedarf entscheiden kann, ob bei einer Anfrage der Nutzer über das Netz oder offline aus dem Cache bedient werden soll.⁶⁶

Nutzerbindung

Anwender von Progressive Web Apps sollen durch die Eigenschaft *Re-engageable* zur Wiederverwendung der Applikation bewogen werden. Dafür sollen ähnlich wie bei nativen Apps Push-Benachrichtigungen eingesetzt werden, die den User über extern eingetretene Ereignisse wie beispielsweise neue Inhalte, Neuigkeiten und Aktualisierungen oder abgeschlossene Aktivitäten informieren. Dabei erlaubt eine Notifications-API Webanwendungen auf das native Benachrichtigungszentrum des Betriebssystems zuzugreifen.⁶⁶

4.4 Technologischer Hintergrund

Wie bereits zuvor angedeutet, basieren Progressive Web Apps auf den selben Technologien wie herkömmliche Webanwendungen, setzen in ihrem Kern auf HTML, CSS und JavaScript oder zusätzlich verschiedene Frameworks und Bibliotheken (*siehe Kapitel 1*) und können so von einer Browser-Engine ausgeführt werden. Eine zentrale Komponente zur Umsetzung von den Funktionen wie Offline-Fähigkeit oder Push-Benachrichtigungen, die User zuvor nur aus nativen Apps kannten, ist der *Service-Worker*, der zusammen mit dem *Web App Manifest* die Webanwendungen erst zur PWA werden lässt.

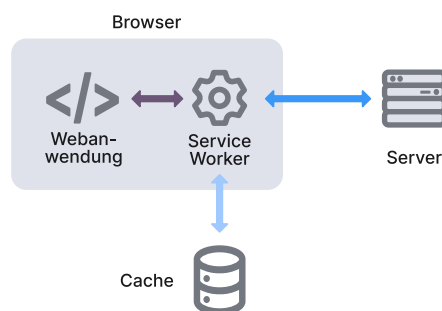
⁶⁶ (Liebel, 2018)

4.4.1 Service Worker

Grundlegend handelt es sich bei einem Service-Worker um JavaScript-Code, der parallel zum eigentlichen Thread der Webanwendung ausgeführt wird. So können diese weder direkt auf die Webanwendung noch auf die Elemente des DOM zugreifen. Stattdessen arbeiten die Service Worker mehr als eine Art Proxy zwischen der eigentlichen Anwendung und dem Server und können HTTP-Requests der Web-App abfangen, gesondert behandeln und gegebenenfalls manipulieren. Die Response des Servers wird dann, möglicherweise abgeändert und wieder zurück an die Webanwendung gesendet. Dabei hat der Service Worker Zugriff auf einen speziellen Cache-Speicher, in dem dieser die Daten wie beispielweise Request-Response-Paare und ganze Seiten und Medienassets ablegen kann.⁷¹

Auf diese Weise ist es möglich, die Offlinefähigkeit der PWA zu realisieren, indem der Service Worker bei einer Anfrage entscheidet, ob er diese an das Netzwerk weiterleitet oder bei fehlender Verbindung den Request mit Daten aus dem Cache beantwortet. Sobald der Service Worker registriert wurde, arbeitet dieser unabhängig von Tab oder Browserfenster als Hintergrunddienst, der auch dann läuft, wenn die Webanwendung gar nicht geöffnet ist. So können mit Hilfe der Notification-API Push-Benachrichtigungen gesendet werden können, auch wenn die PWA weder im Browser noch als eigenständiges Fenster geöffnet ist.⁷²

Abbildung 40 ►
Funktionsweise des Service Workers einer PWA



4.4.2 Web App Manifest

Das Web App Manifest im JSON-basierten Dateiformat erfüllt für eine PWA gleich mehrere wichtige Aufgaben und Teile der in *Kapitel 4.3* beschriebenen Eigenschaften. Dabei dient es zum einen zur Unterscheidung zwischen einer normalen Webseite und einer PWA, es erlaubt auf Wunsch des Users die Installation der App auf dem Homebildschirm und definiert letztendlich die Darstellung der Applikation auf dem Zielgerät.⁶⁶

⁶⁶ (Liebel, 2018)

⁷¹ (Siegrist, 2021)

⁷² (Liebel, 2017)

```

1 {
2   "short_name": "Weather",
3   "name": "Weather: Do I need an umbrella?",
4   "icons": [
5     {
6       "src": "/images/icons-vector.svg",
7       "type": "image/svg+xml",
8       "sizes": "512x512"
9     },
10    {
11      "src": "/images/icons-192.png",
12      "type": "image/png",
13      "sizes": "192x192"
14    },
15    {
16      "src": "/images/icons-512.png",
17      "type": "image/png",
18      "sizes": "512x512"
19    }
20  ],
21  "start_url": "?source=pwa",
22  "background_color": "#3367D6",
23  "display": "standalone",
24  "theme_color": "#3367D6",
25  "description": "Weather forecast information"
26 }
27

```

◀ Codesnippet 6

Beispielhaftes Web App Manifest

So wird in der hier dargestellten Datei mit der Eigenschaft *name* definiert, wie die Anwendung in der Programmliste, dem App-Switcher oder auf dem Homebildschirm heißen soll. Zusätzlich dazu nimmt Eigenschaft *icons* einen Array entgegen, das auf verschiedene Bildressourcen in unterschiedlichen Abmessungen oder Formaten je nach Betriebssystem verweist und das zum Namen passende Icon dafür bereitstellt. Das Manifest erlaubt außerdem die Definition zweier Farbangaben, die passend zum Farbschema der Webanwendung gewählt werden können. Die erste Eigenschaft *theme_color* kann von Browsern genutzt werden, um Teile des User Interfaces entsprechend darzustellen, während die zweite Angabe *background_color* die erwartete Hintergrundfarbe entgegennimmt und beispielsweise das sichtbare Fenster beim Ladevorgang der PWA einfärbt. Zusätzlich kann über die Eigenschaft *display* mit dem zugehörigen Wert *standalone* definiert werden, dass sich die PWA als eigenständiges Fenster außerhalb des Browsers öffnet.⁶⁶

Grundsätzlich wird das Web App Manifest von allen aktuellen Webbrowsern wie Microsoft Edge, Chrome oder Firefox auf Windows und Android sowie Apples Safari unterstützt, auch wenn bei letzterem Vertreter noch nicht alle Angaben ausgewertet werden.⁶⁶

⁶⁶ (Liebel, 2018)

4.5 Status Quo und Zukunft von PWAs

Vor allem die Fortschritte in der Browser-Technologie in den letzten zehn Jahren waren ausschlaggebend für das Aufkommen einer neuen Generation von Webanwendungen, die mit dem Ansatz der Progressive Web Apps neue Möglichkeiten boten, funktionsreiche und eindrucksvolle Applikationen umzusetzen und damit eine ernstzunehmende Alternative zu den nativen Apps darzustellen, die lange Zeit den Markt dominierten.

4.5.1 Praxisbeispiele

Aktuell sind PWAs noch nicht in der Lage, die klassischen, nativen Apps in allen Anwendungsbereichen vollständig zu ersetzen, da diese immer noch bessere Möglichkeiten haben, auf die Endgeräte und deren Systemressourcen zuzugreifen. Da viele Applikationen keinen derart tiefen Zugriff auf Hard- oder Software benötigen, eignet sich der plattformübergreifende Ansatz der PWAs gerade bei formular- und datengetriebene Businessanwendungen, To-Do- oder Notizapps, Nachrichten- und Wetterdiensten oder sozialen Netzwerken sowie Messengern hervorragend und wird, wie die folgenden Beispiele zeigen, inzwischen auch von größeren Unternehmen erfolgreich eingesetzt.⁶⁶

Twitter

Aufgrund der Tatsache, dass über 80 Prozent der User des Microblogging-Netzwerks Twitter den Dienst von einem mobilen Endgerät aus nutzen, führte das Unternehmen im Jahre 2017 mit Twitter Lite eine Progressive Web App ein, die mittlerweile zu den bekanntesten und am weit verbreitetsten PWAs auf dem Markt zählt.⁷³

Neben einem deutlich geringeren Datenverbrauch und schnellen Ladezeiten konnte die Lite-App nun auch auf dem Homescreen installiert werden, was zu 65 Prozent mehr Seitenaufrufen, 75 Prozent mehr gesendeten Tweets und einer um 20 Prozent verringerten Absprungrate bei den Nutzern der mobilen Seite führte. Aufgrund dieses Erfolgs führte Twitter 2019 auch eine Desktop-Version der beliebten Lite-Version als Progressive Web App ein, die in Microsofts App Store heruntergeladen werden kann.⁷³

66 (Lieber, 2018)

73 (Hodakovskis 2019)

Telegram

Der schnell wachsende cloudbasierte Instant-Messenger stellt einen weiteren, recht bekannten Anwendungsfall für den Einsatz der PWA-Technologie dar. Den mittlerweile über 100 Millionen registrierten Nutzern bietet Telegrams Progressive Web App eine optimale User Experience und steht funktionell den nativen Android- und iOS-Apps in nichts nach.⁷³

Financial Times

Auch die Financial Times hat mittlerweile für die mobilen Nutzer eine Progressive Web App implementiert, die sich der parallel dazu verfügbaren nativen Version in ihren Funktionen gleicht und so beispielsweise Nachrichten, einen Ticker oder eine Suchfunktion bietet und sich Artikel zum späteren Lesen markieren lassen. Außerdem wird auch die aktuelle Ausgabe im Cache zwischengespeichert und ist so mit der Unterstützung des Service Workers offline lesbar. Die Applikation aktualisiert sich dabei im Hintergrund selbst, sodass die User im Vergleich zu den nativen Vorbildern Updates nicht eigenständig in den App Stores durchführen müssen.⁶⁶

Pinterest

Wie auch Twitter war Pinterest einer der ersten Global Player, der sich mit dem Technologieansatz der PWAs auseinandersetzte. Durch die neu entworfene Webanwendung konnten nahezu alle wichtigen Nutzerzahlen deutlich verbessert werden, sodass beispielsweise die auf der Seite verbrachte Zeit um 40 Prozent anstieg, nutzergenerierte Werbeeinnahmen um 44 Prozent gesteigert wurden und sich sämtliche Klickraten und Kerninteraktionen verdoppelten.⁷³

4.5.2 Chancen

Der Kerngedanke der PWAs, der besagt, dass diese *progressiv*, sprich plattformübergreifend ausführbar sind, ist zugleich auch der größte Vorteil. So sind diese Art von Webanwendungen überall lauffähig, wo auch ein Browser verfügbar ist und können dadurch mit einer einzigen Codebasis auf Mobilgeräten, Smart TVs und Uhren, Desktoprechnern, Spielekonsolen oder Entertainmentssystemen in modernen Autos eingesetzt werden.⁶⁶

⁶⁶ (Liebel, 2018)

⁷³ (Hodakovskis, 2019)

Während ansonsten mehrere Entwicklerteams mit Fachwissen in den Programmiersprachen für die jeweilige Plattform beauftragt werden müssen, genügt bei diesem Ansatz ein einziges Team, das eine Webapplikation entwirft und implementiert, was gerade für Start-Ups und kleinere Unternehmen mit limitierten finanziellen oder personellen Möglichkeiten einige Chancen ermöglicht.⁶⁶

Mittlerweile wird der technologische Ansatz von allen großen Webbrowsern wie Google Chrome, Edge, Safari und Firefox unterstützt. Insbesondere Google treibt die Entwicklung und Verbreitung der Progressive Web Apps weiter voran und hat dabei vor allem mit Blick auf das eigene browserbasierte Betriebssystem *Chrome OS*, erweiterte Möglichkeiten und neue Schnittstellen für den Desktop in Planung, die PWAs noch mächtiger machen sollen. Auch bei Microsoft spielen Progressive Web Apps spätestens seit *Windows 10* eine wichtige Rolle, sodass das plattformübergreifende Anwendungsmodell mittlerweile fester Bestandteil des hauseigenen Stores ist.⁶⁶

4.5.3 Einschränkungen

Das Applikationskonzept der PWAs hat neben seinem enormen Potential aber auch Grenzen. Eine der größten Einschränkungen ist dabei der fehlende Zugriff auf alle beliebigen Schnittstellen des Betriebssystems oder die Hardware der Endgeräte. Da die PWAs auf die Ausführung über einen Browser angewiesen sind, stehen lediglich die Webschnittstellen zur Verfügung, die zwar in den letzten Jahren immer weiterentwickelt wurden, jedoch noch nicht ganz den Umfang der nativen APIs bieten.⁶⁶

Eine zusätzliche Limitation ist außerdem der Hard- und Softwareriese Apple, der grundsätzlich die Technologie der Progressive Web Apps mit seinem Browser Safari unterstützt, im Vergleich zu Google und Microsoft jedoch deutlich weniger an einer weitreichenden Verbreitung dieses Ansatzes interessiert scheint. So stellt sich Apple dem Gedanken der PWAs nicht entgegen, lässt sich aber gerade beim Implementieren von benötigten Schnittstellen deutlich mehr Zeit als die Konkurrenz, sodass hier nach wie vor keine Unterstützung der Notification-API für Push-Benachrichtigungen vorhanden ist. Grund dafür ist, dass Apple weiterhin seinen sehr profitablen App Store mit nativen Applikationen erhalten will und diese nicht zwingend durch die browserbasierten PWAs ersetzen möchte.⁶⁶

66 (Liebel, 2018)

Konzeption der proto- typischen Anwendung

Neben der Analyse und Betrachtung der modernen Technologieansätze des hybriden Renderings, Headless CMS und Progressive Web Apps in den vorangegangenen Kapiteln soll in dieser Thesis mit den gewonnenen Erkenntnissen und erarbeiteten theoretischen Grundlagen parallel schließlich eine Webapplikation konzipiert und letztendlich prototypisch implementiert werden, die diese drei technologischen Möglichkeiten miteinander kombiniert.

So soll in diesem Kapitel zunächst die grundlegende Idee der Applikation beschrieben, deren Anforderungen als User Stories definiert und die eingesetzten Technologien, Systeme und Frameworks für die spätere Implementierung vorgestellt werden. Zudem soll bereits die Designsprache des User Interfaces inklusive Logo sowie ein ausführliches Datenmodell und Sequenzdiagramme entworfen werden, um die spätere Entwicklung vorzubereiten und zu vereinfachen.

5.1 Motivation und Idee

Die Motivation hinter der Konzeption und der anschließenden Implementierung der Applikation ist es, die in den letzten Jahren aufgekommenen Trends im Bereich des Webdevelopments nicht nur theoretisch zu erarbeiten und zu verstehen, sondern auch zu lernen, diese mit bestimmten Technologien praktisch umzusetzen und in einem realitätsnahen Projekt zu vereinen. Auf diese Weise lassen sich zudem neue Fähigkeiten hinsichtlich des Workflows und der Programmierung im weiten Feld der Webentwicklung aneignen.

Da im Vordergrund die drei Technologien des hybriden Renderings von Webanwendungen, Headless CMS sowie das Konzept der PWAs stehen, musste eine Idee für eine Anwendung gefunden werden, die es zum einen möglich macht, diese drei Ansätze sinnvoll miteinander zu vereinen, sodass alle Aspekte abgedeckt werden können und die Vorteile der jeweiligen Technologien zur Geltung kommen. Auf der anderen Seite soll diese App jedoch nicht nur als eine reine Präsentation der vorgestellten Ansätze dienen, sondern auch für mögliche spätere User einen tatsächlichen Mehrwert bieten.

Bei der Webapp *MovieRec* handelt es sich grundlegend um ein Filmportal, das Nutzern die Möglichkeit bietet, Informationen und Daten zu Filmen zu finden und abzurufen. Neben einem rein informativen und statischen Anteil soll die Applikation darüber hinaus auch dynamische Features bieten, sodass persönliche Listen erstellt, über ein eigenes Nutzerkonto gespeichert und verwaltet und im Sinne eines sozialen Netzwerks Filmempfehlungen, Bewertungen oder Kommentare mit anderen Nutzern geteilt werden können. Je nach Art der Unterseite soll clientseitiges, serverseitiges oder statisches Rendern sinnvoll eingesetzt und miteinander kombiniert werden.

5.2 Anforderungsanalyse

In dieser Anforderungsanalyse werden zunächst die bekanntesten Anwendungen mit einer ähnlichen Idee betrachtet und anschließend die Features der App als User Stories festgehalten.

5.2.1 Recherche

Bevor die endgültigen Anforderungen an die Anwendung erstellt sowie genauere Funktionen festgelegt und definiert werden, soll zunächst ein Überblick über aktuelle Webseiten und Apps in diesem Bereich geschaffen und dabei deren Kernfunktionen, Stärken und Schwächen analysiert werden.

IMDb

Die Internet Movie Database, kurz *IMDb*, zählt zu den bekanntesten und ältesten Webseiten rund um das Thema Filme und Serien. Das Portal wurde bereits im Jahre 1990 vom Briten Col Needham gegründet und seither kontinuierlich erweitert, sodass sich mittlerweile Millionen an Einträgen zu Filmproduktionen und zugehörigen Schauspielern, Regisseuren und Studios sowie News und Trailer in der mittlerweile von Amazon verwalteten Datenbank befinden.⁷⁴

Informationen lassen sich direkt von der Seite abrufen, jedoch wird für die Erstellung einer Watchlist, das Bewerten oder Kommentieren der Filme oder Serien ein Benutzeraccount benötigt. Als Bewertungssystem nutzt IMDb eine Skala von einem bis zehn Sternen. Es gibt zudem eine Rangliste, in der die einzelnen Filme der Bewertung nach sortiert sind. Zudem lassen sich zu jedem Eintrag weitere Verlinkungen zu Schauspielern und Regisseur sowie einige Vorschläge von anderen ähnlichen Filmen finden. Die Seite ist vollständig responsiv und bietet eine mobile Ansicht für Smartphones. Außerdem lässt sich eine zugehörige native App in Apples App Store oder dem Play Store von Google installieren.

Rotten Tomatos

Eine weitere recht beliebte Webseite ist Rotten Tomatos, die 1998 wie zuvor auch IMDb als Hobbyprojekt ins Leben gerufen und mittlerweile von Warner Bros verwaltet und betrieben wird. Wie auch bei IMDb können sämtliche Informationen und Bewertungen zu Filmen ohne Registrierung angesehen werden.. In das bekannte *Tomatometer*, das als Skala immer der Prozentzahl der positiven Kritiken für einen Film entspricht, sind jedoch nur ausgewählte Kritiker zugelassen, die von der Seite selbst bestimmt werden.⁷⁵

74 (Lewis, kein Datum)

75 (Krinner, 2018)

Auf der Startseite werden neben demnächst erscheinenden Filmen und Neuigkeiten auch aktuelle Trendlisten, basierend auf dem Bewertungssystem, präsentiert. Dies ist zwar durch eine responsive Ansicht auf mobilen Endgeräten möglich, jedoch wirkt das User Interface der Seite etwas unübersichtlich, mittlerweile etwas veraltet und es steht keine native App zur Verfügung.

Letterboxd

Letterboxd ist ein Netzwerkdienst für Filmbegeisterte, der 2011 veröffentlicht wurde und von Gründer Matthew Buchanan als „*a social site for sharing one's taste in film*“ zusammengefasst wird. Nutzer können dabei eine Art Tagebuch führen, welche Filme sie gesehen haben, ähnlich wie bei Facebook, diese mit einem Like markieren oder von einem halben bis fünf Sterne bewerten. Die Aktivitäten anderer User können auf deren Profil eingesehen werden.⁷⁶

Eine Mitgliedschaft auf dem Portal ist kostenlos. Es wird mittlerweile jedoch eine kostenpflichtige Premium-Membership angeboten, die zusätzliche Features wie personalisierte Statistiken über seine gesehenen Filme und keine Werbeanzeigen auf der Seite bietet. Letterboxd ist als responsive Webanwendung aufgebaut, die im Vergleich zu den beiden oben genannten Portalen die modernste UI aufweist. Es wird zudem auch eine native App für Apple und Android angeboten, auf die bereits auf der Landingpage der Webseite hingewiesen wird.

5.2.2 User Stories

Als User möchte ich alle Filme in der Datenbank angezeigt bekommen, um mir einen Überblick schaffen zu können.

Als Nutzer von MovieRec möchte ich in der Filmübersicht nach Filmen und Regisseuren suchen sowie nach bestimmten Kriterien wie Genre, Erscheinungsdatum oder Bewertung sortieren können, um so schneller einen entsprechenden Film zu finden.

Als User möchte ich zu jedem Film genauere Informationen zu Handlung, Regisseur und Erscheinungsdatum auf einer separaten Seite erhalten.

⁷⁶ (Gettell, 2012)

Als Nutzer von MovieRec erwarte ich, dass meine Anmeldedaten nach dem Login gespeichert werden, damit ich mich nicht bei jedem neuen Aufruf erneut anmelden muss.

Als Verwalter der Anwendung möchte ich, dass die Formular-eingaben validiert werden, um keine ungültigen Nutzerdaten zu speichern.

Als Nutzer der App erwarte ich, dass meine persönlichen Daten bei Registrierung oder Login über sicher über HTTPS übertragen werden, damit diese vor Angriffen geschützt sind.

Als User möchte ich gesehen Filme bewerten und Kommentare zu diesen abgeben, um meine Meinung zu diesen mit anderen teilen zu können.

Als User erwarte ich, dass aus allen Bewertungen der Nutzer zu einem Film der Durchschnitt errechnet wird, um so ein repräsentatives Rating sehen zu können.

Als Nutzer würde ich gerne die Aktivitäten und Reviews anderer Nutzer in einem Feed angezeigt bekommen, um zu sehen, welche Filme diese empfehlen oder gesehen haben.

Als Nutzer erwarte ich ein responsives Design der Web App, damit ich diese auch auf verschiedenen Endgeräten problemlos nutzen kann.

Als User möchte ich einen Dark- und Lightmode, um das User Interface der App je nach Tageszeit oder persönlicher Präferenz anpassen zu können.

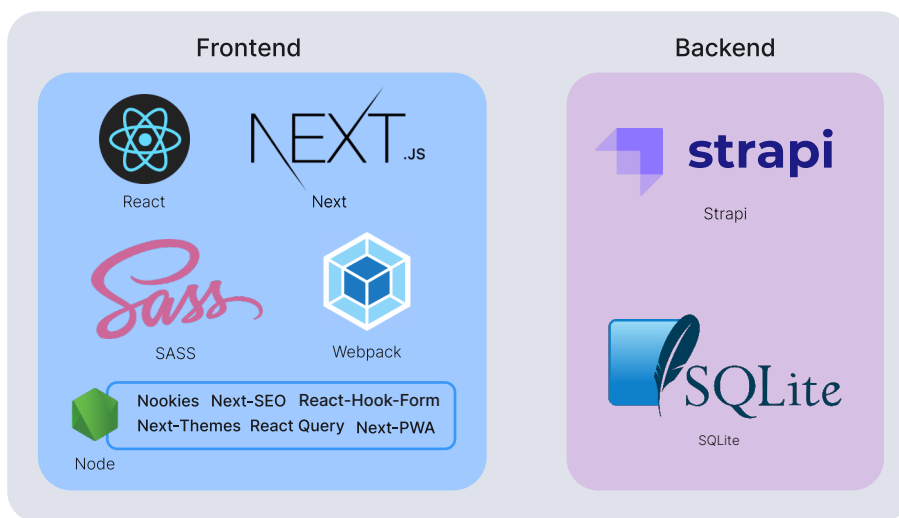
Als Inhaber der Seite möchte ich, dass sie für Suchmaschinen optimiert ist, sodass diese von den Crawlern gut indiziert werden kann und möglichst weit vorne in den organischen Suchergebnissen angezeigt wird.

Als Entwickler der Anwendung möchte ich, dass diese durch den Einsatz von serverseitigem Rendering und statischen Seiten optimale Ladezeiten der einzelnen Seiten bietet, um die User Experience für die Nutzer zu verbessern.

Als Seiteninhaber erwarte ich, dass die User die Anwendung als PWA installieren können, um so die Nutzungshäufigkeit zu erhöhen und mehr Nutzer gewinnen zu können.

5.3 Technologiestack

Um die definierten Anforderungen der Webanwendung umzusetzen sowie die Konzepte des Headless CMS, des hybriden Renderings und den Progressive Web Apps miteinander zu verbinden, wird ein bestimmter Stack an verschiedenen Technologien und Erweiterungen für das Frontend und Backend genutzt, deren Funktion und Zweck für die spätere Implementierung im Folgenden kurz erläutert werden.



◀ Abbildung 41

Technologiestack der Anwendung „MovieRec“

5.3.1 Frontend

Im Frontend der Anwendung wird zum einen ein Framework benötigt, das die Daten aus dem Backend abrufen und verarbeiten kann und für das Rendering des HTML und der UI-Elemente zuständig ist sowie zudem ein Ansatz um das User Interface ansprechend und modern gestalten zu können.

React und Next.js

Als Haupttechnologie des Frontends wurde die Library React (siehe 1.2.4) gewählt. Die leichtgewichtige Bibliothek zeichnet sich vor allem durch den Einsatz von wiederverwendbaren Komponenten aus, was gerade in einem umfangreicheren Projekt von Vorteil ist. So kann Aufwand eingespart werden, wenn viele Elemente der Benutzeroberfläche nur einmal implementiert und gestaltet werden müssen und anschließend beliebig oft auf verschiedenen Unterseiten eingesetzt werden können.

Da React typischerweise für die Entwicklung von Single Page Apps genutzt wird, die clientseitig gerendert werden, wird die Bibliothek auf der Seite des Frontends durch das auf React basierende Framework Next.js (*siehe 2.6.1*) erweitert. Dadurch ist es möglich, aus einer Single-Page- eine Multipage-Applikation zu erstellen, die eine einfache Navigation zwischen den Unterseiten bietet und neben der statischen Generierung zur Build-Time und serverseitigem Rendering auch Inkrementell Statische Regeneration unterstützt.

SASS

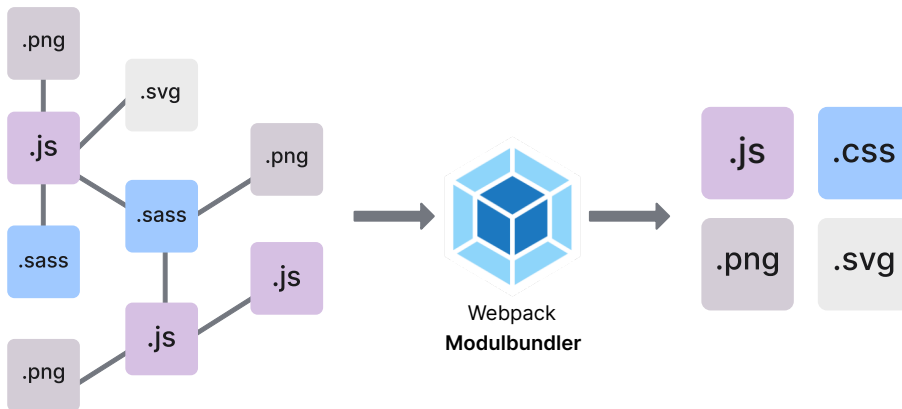
Zur Gestaltung der einzelnen User Interface-Elemente, der Wahl der Farben und der Schrift sowie der Konzeption des responsiven Layouts wird anstelle von herkömmlichen CSS SASS, kurz für *Syntatically Awesome Stylesheets*, genutzt. Wie bereits in *Kapitel 1.2.2* angedeutet, handelt es sich um einen sogenannten Präprozessor, mit Hilfe dessen sich die CSS-Syntax vereinfachen lässt, die Struktur und Organisation der Files in einem solch großen Projekt besser gelingt und hilft Code-Wiederholungen über Variablen zu vermeiden. Da ein herkömmlicher Browser jedoch nur CSS-Dateien verarbeiten und interpretieren kann, muss die SASS-Syntax vor der Veröffentlichung und dem Hosting in herkömmliches CSS kompiliert werden, was hier automatisch vom Framework Next.js übernommen wird.⁷⁷

Webpack

Für viele Funktionen und Features von React und Next.js ist im Hintergrund der Modulbundler Webpack verantwortlich, dessen Hauptaufgabe darin liegt, für die Ausführung der Anwendung die vielen einzelnen, voneinander abhängigen JavaScript-Modulen und Stylesheets des Next.js-Projekts zusammenzuführen und aus diesen eine einzige, vollständige JavaScript-Datei, sozusagen ein Bundle, erzeugt.⁷⁸

⁷⁷ (Hellwig, 2021)

⁷⁸ (Hartmann, 2015)



◀ Abbildung 42

Bündeln der einzelnen Module und erfassen der Abhängigkeiten

Für die Entwicklungszeit stellt Webpack außerdem noch einen einfachen, lokalen Webserver bereit, auf dem die Anwendung zur Entwicklung und zum Testen ausgeführt werden kann. Ein Vorteil bei diesem ist es, dass durch einen Hot-Mode des Servers Änderungen in den HTML- CSS- oder JavaScript-Dateien des Quellcodes automatisch erkannt und diese automatisch transformiert werden, wodurch die Seite immer in Echtzeit aktualisiert wird und ein manuelles Neuladen im Browser überflüssig wird.⁷⁸

Node Package Manager

Ein Bestandteil der bereits in *Kapitel 1.3.1* vorgestellten JavaScript-Laufzeitumgebung Node ist der *Node Package Manager*, meist mit NPM abgekürzt. In dem hilfreichen und sehr beliebten Paketmanager stehen mittlerweile über 350.000 Open-Source-Pakete zur freien Nutzung bereit, die neben kompletten Anwendungsframeworks auch unzählige Hilfsbibliotheken und Erweiterungen umfassen, welche lokal einfach über einen „install“-Befehl in der Kommandozeile zusammen mit dem Namen des entsprechenden Moduls zu dem Projekt hinzugefügt und ab dann in diesem eingesetzt werden können.⁷⁹

Nookies

Eine dieser vielfältigen Hilfsbibliotheken im Repository des NPM ist Nookies. Diese Erweiterung für Next.js stellt einige nützliche Funktionen für den Umgang und Verwaltung von Cookies bereit, sodass sich beispielsweise mit wenigen Zeilen Code nutzerspezifische Cookies innerhalb der Anwendung setzen lassen oder diese bei Bedarf aus dem Cache gelesen werden können.

⁷⁸ (Hartmann, 2015)

⁷⁹ (Roden, 2012)

React Query

Bei React Query handelt es sich um eine mächtige Library für React um externe Schnittstellenaufrufe durchzuführen, die erhaltenen Daten zu verarbeiten und automatisch, temporär speichern zu können. Die Bibliothek lässt sich für viele Anwendungszwecke nutzen, in MovieRec dient diese jedoch primär zum Filtern nach verschiedenen Parametern und Suchen innerhalb der Filmliste, was durch die Nutzung von React Query deutlich performanter und intuitiver wird.

Next-Themes

Next-Themes ist ein kleines, leichtgewichtiges Tool für das Framework Next.js, mit Hilfe dessen globale, seitenübergreifende Themes verwaltet und einfach angepasst werden können, wodurch mit wenigen Zeilen Code ein Dark- und Lightmode implementiert werden kann, welcher sogar in der Lage ist, die festgelegte Einstellung des Endgerätes abzurufen, sodass die Applikation je nach Präferenz des Nutzer in einer dunklen oder hellen Oberfläche erscheint.

React-Hook-Form

Um die Validierung der Formulare nicht komplett eigenständig umzusetzen, bietet die React-Erweiterung React Hook Form, einige vorgefertigte Funktionen, die sich den Formularfeldern hinzufügen lassen und die vor dem Absenden einfach auf definierte Kriterien zu überprüfen.

Next-SEO

Um beispielsweise Titel oder Seitenbeschreibungen in den Head der einzelnen Seiten hinzuzufügen und so die Suchmaschinenoptimierung zu verbessern wird die Bibliothek Next-SEO installiert, die diesen Prozess für Next-Applikationen vereinfacht und vielfältige Anpassungsoptionen bietet.

Next-PWA

Durch das Paket Next-PWA ist es möglich die Next.js-Anwendung in eine Progressive Web App umzuwandeln, ohne dabei den Service Worker eigenständig implementieren und registrieren zu müssen.

5.3.2 Backend

Im Backend soll mit Strapi (*siehe 3.4.2*) ein Headless CMS eingesetzt werden, in dem die Inhalte wie beispielsweise Informationen zu Filmen oder den persönlichen Listen registrierter Nutzer verwaltet und gespeichert werden können. Ganz im Sinne dieser Art des Content Management Systems besteht hier nicht die Möglichkeit die Daten auf einer direkt gekoppelten Webseite darzustellen, sondern diese werden über eine Schnittstelle für das externe Frontend zur Verfügung gestellt.

Strapi und SQLite

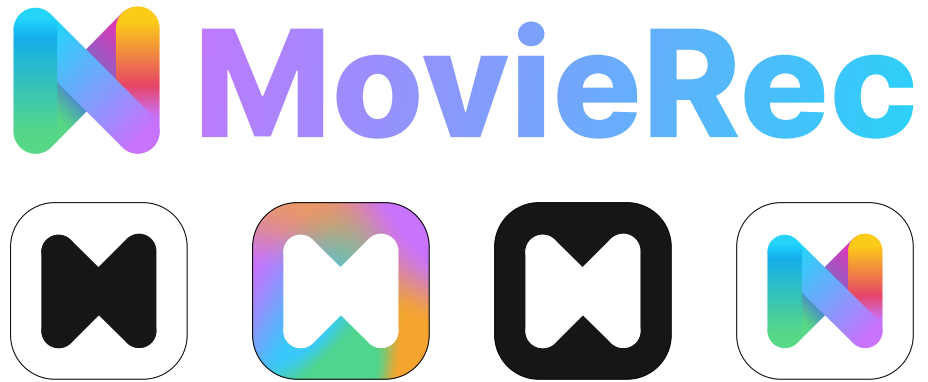
Als Technologie für die verbundene Datenbank fällt die Entscheidung auf den relationale Vertreter SQLite, die als Standardoption bei der Installation von Strapi direkt miteingerichtet wird und mit dem CMS verknüpft ist und somit nicht eigenhändig implementiert und aufgesetzt werden muss.

5.4 User Interface

Die grafische Benutzeroberfläche der Webanwendung soll recht minimalistisch gestaltet werden, sodass diese den User nicht von der Nutzung der eigentlichen Funktionen der App abhält und sich dieser schnell zurechtfindet, die UI zeitgleich trotzdem sehr modern wirkt und sich an derzeitigen Standards und Trends orientiert.

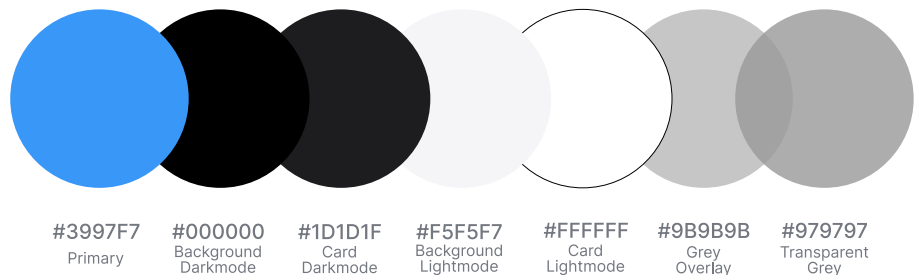
Das Logo von MovieRec setzte sich aus vier an den Enden abgerundeten Rechtecken zusammen und greift den dabei Anfangsbuchstaben *M* des Namens der Applikation auf. Jede einzelne Komponente des Logos ist in einem anderen Farbverlauf gehalten, die an ihren Enden jedoch immer ineinander übergehen und somit nach wie vor harmonisch wirken.

Abbildung 43 ►
Logo und Iconkonzepte

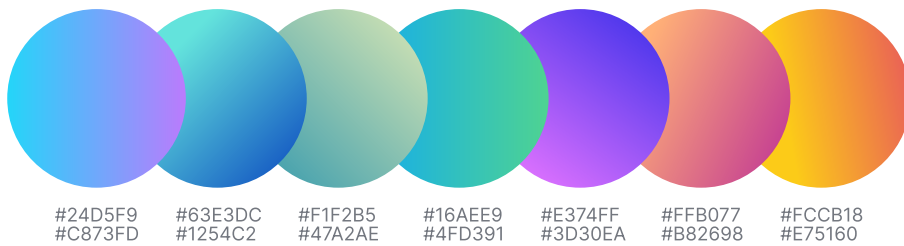


Farblich wird für die Ansicht im Light-Mode auf viele weiße Flächen und ein helles Grau gesetzt, die im Dark-Mode durch Schwarz und ein dunkles Grau ersetzt werden. Als primäre Akzentfarbe kommt ein kräftiges Blau zum Einsatz, das beispielsweise bei Call-to-Action-Buttons direkt den Blick des Users auf sich zieht. Für die Schriftfarbe wird konträr zu den Hintergründen im hellen Design klassischerweise Schwarz genutzt, während dies im dunklen Modus durch Weiß ausgetauscht wird, um einen guten Kontrast und eine optimale Lesbarkeit zu erhalten. Zwischen längeren Fließtexten und Überschriften wird neben der Schriftgröße und dem Schnitt auch durch die Deckkraft unterschieden, die bei längeren oder untergeordneten Textpassagen hinabgesetzt wird.

Abbildung 44 ►
Farbpalette der Webanwendung



Wie auch im Logo spielen auch für die Filmtitel Farbverläufe eine wichtige Rolle. Um das User Interface visuell aufzuwerten sowie die einzelnen Filme nach ihrem Genre zu gliedern, wird für jede Kategorie ein anderer Gradient genutzt, der den Titel des Films einfärbt.



◀ Abbildung 45

Gradients der Filmüberschriften

Als Schriftart wird die Open-Source-Font *Inter* in der gesamten Webanwendung verwendet. Die serifenlose und recht simple, aber auch modern anmutende Schrift führt dazu, dass sich die Typographie perfekt in das Konzept der Benutzeroberfläche einfügt und im Zusammenspiel mit der Farbwahl der Font für eine gute Lesbarkeit auf digitalem Screen sorgt. Da nur eine einzige Schriftart zum Einsatz kommt, sorgen die verschiedenen Schriftschnitte für eine visuelle Hierarchie zwischen den unterschiedlichen Ebenen von Überschriften und Lauftexten.

Lorem Ipsum Dolor Font Weight 800

sit amet, consetetur Font Weight 600

sed diasit consetetur Font Weight 500

onumy eirmod tempor invidunt ut Font Weight 400

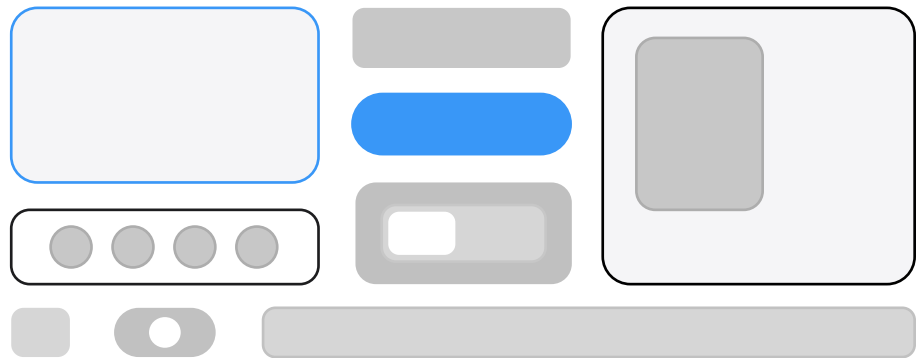
labore et dolore magna aliquyam erat Font Weight 300

◀ Abbildung 46

Typographie der Webanwendung

Die UI-Komponenten wie Karten, Buttons und Formularfelder werden alle mit abgerundeten Ecken gestaltet, sodass die einzelnen Bestandteile organisch und weich wirken, gut miteinander zusammenspielen und die Oberfläche für Nutzer einladend und freundlich erscheint und dieser die App gerne verwenden soll. Überlappende Elemente sind leicht transparent und lassen darunter liegende Teile verschwimmen und sorgen so neben einem ansprechenden Effekt für mehr räumliche Tiefe.

Abbildung 47 ►
Konzepte und Formen der
UI-Komponenten

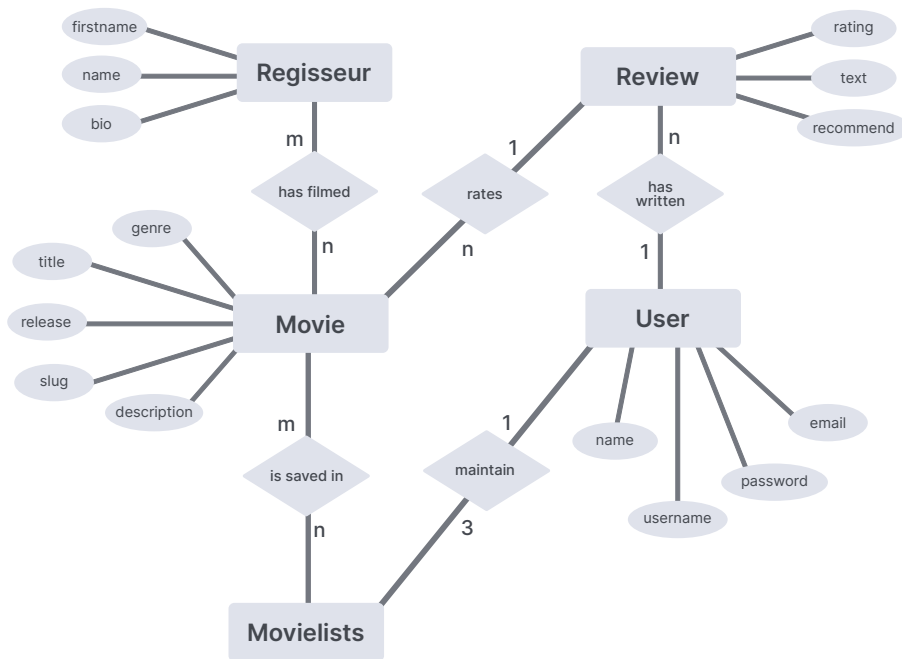


5.5 Datenmodell

Sämtliche Daten und Informationen der Anwendung werden wie bereits bei der Vorstellung des Technologiestacks beschrieben in der mit dem CMS im Backend verknüpften SQLite-Datenbank gespeichert. Dafür müssen zunächst auf der grafischen Benutzeroberfläche des CMS die einzelnen Modelle erstellt werden, die dann anschließend die jeweiligen Datensätze der unterschiedlichen Typen entgegennehmen und in den Tabellen der relationalen Datenbank speichern können.

Um die spätere Implementierung im Backend von Strapi zu vereinfachen, wird bereits in der Konzeption ein konkretes Datenmodell in dem in der Softwareentwicklung gängigen *Entity-Relationship*-Schema angelegt, das die einzelnen Datentypen, deren Attribute sowie die Beziehungen zwischen den Objekten untereinander visuell und übersichtlich darstellt.

Die wichtigste Komponente in der Datenstruktur der Applikation ist die Entität *Movie*. Diese besitzt die zugehörigen Eigenschaften *title* als Namen des Films, ein entsprechendes *genre*, mit *release* das Erscheinungsjahr des Films und *description* nimmt einen kurzen Beschreibungstext der Handlung des jeweiligen Films entgegen. Das Attribut *slug* wird genutzt, um später einen entsprechenden URL-Pfad für die einzelnen Informationsseite des Films zu generieren. Typischerweise hat jeder Datensatz einer Entität einen zugehörigen Primärschlüssel wie beispielsweise eine eindeutige ID. Da dies später von dem Content Management System automatisch für jeden Eintrag generiert wird, kann es als Attribut in diesem ER-Modell vernachlässigt werden.



◀ Abbildung 48

Datenmodell der Anwendung als ER-Schema

Eine weitere Entität ist der *Regisseur*, der neben den Attributen *firstname* und *name* mit *bio* auch noch zusätzlich einen kurzen, zugehörigen Text erhält. Dieses Objekt ist mit dem Datenschema *Movie* verbunden. So besitzt jeder Film einen einzigen, zugehörigen Regisseur, während jeder Regisseur mit mehreren Filmen, die dieser gedreht hat, verknüpft sein kann.

Für den Login-Prozess und das spätere Bewerten und Markieren von Filmen ist ein weiteres zentrales Element die Entität *User*, in der die bei der Registrierung erfassten Daten wie *name*, *username*, *email* und *password* gespeichert werden.

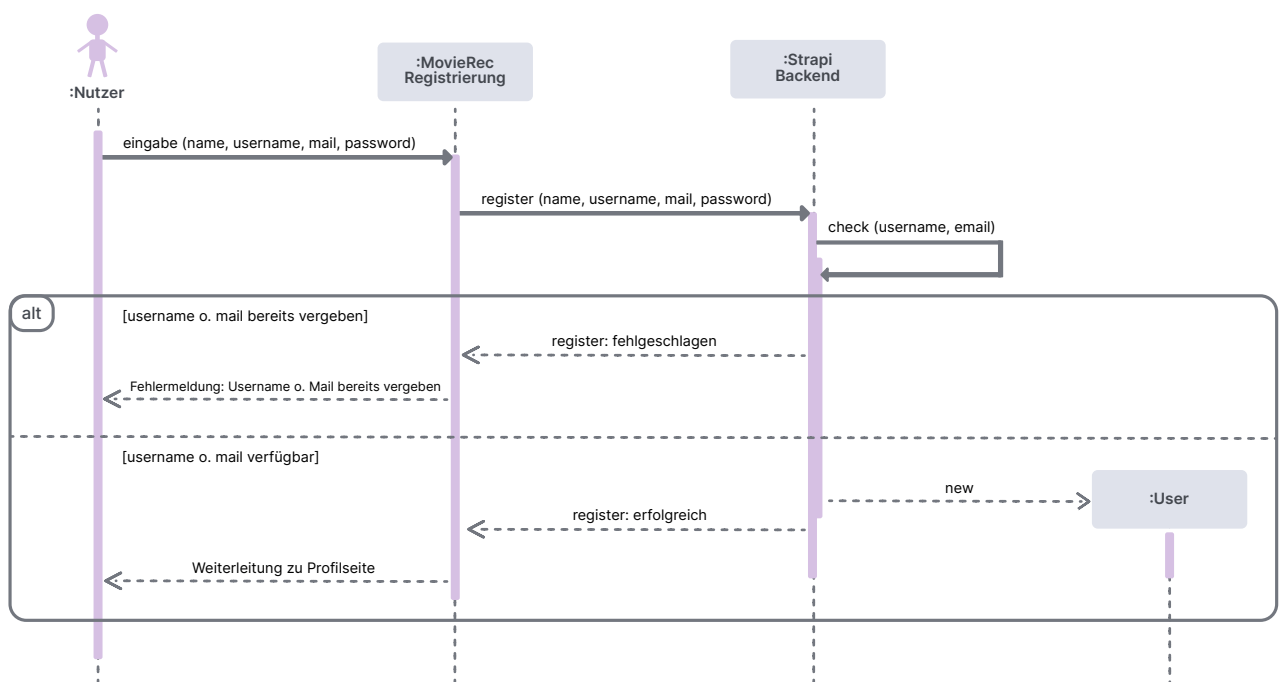
Bewertungen werden als Entität *Review* dargestellt und haben neben einem *rating* von null bis fünf auch mit *text* einen kurzen Kommentar und *recommend* mit einem Boolean-Wert, ob es sich um eine Empfehlung handelt, als Attribute. Jede *Review* bewertet dabei genau einen Film, während zu einem Film mehrere Bewertungen erfasst werden.

Jeder Nutzer kann mit einer Watchlist, einer Empfehlungsliste und einer Liste mit gesehenen Filmen insgesamt drei *Movielists* verwalten. Filme können hier in den Listen von mehreren Usern gespeichert werden.

5.6 Nutzerinteraktionen

Um zeitliche Abläufe der Interaktionen von Nutzern der Anwendung und die Verarbeitung dieser in Front- und Backend zu visualisieren, werden diese in Sequenzdiagrammen dargestellt. Hierfür werden im Folgenden die Registrierung, die Anmeldung von Usern sowie das Hinzufügen eines Films zur Watchlist und das Teilen einer Bewertung durch angemeldete Benutzer betrachtet.

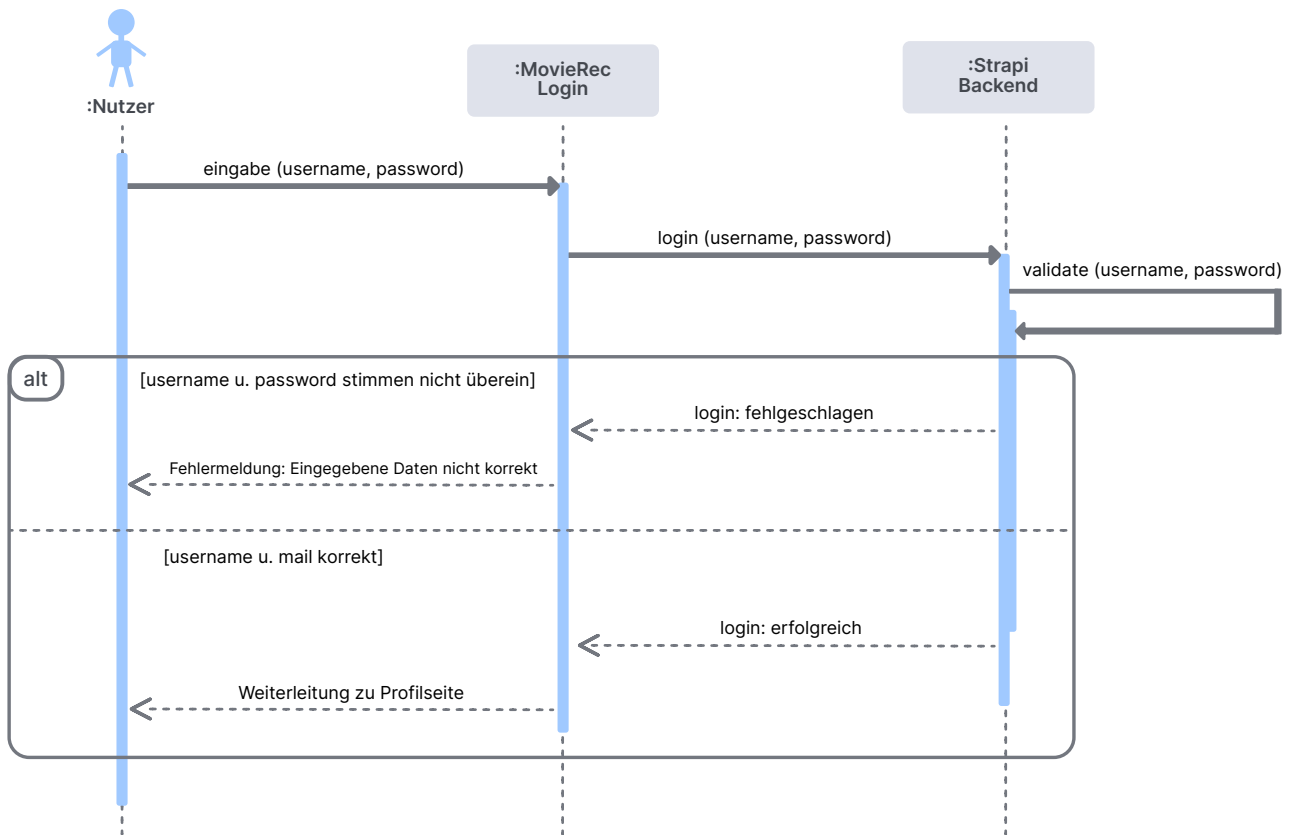
Registrieren



▲ **Abbildung 49** Sequenzdiagramm zur Nutzerinteraktion „Registrierung“

Bei der Registrierung eines neuen Nutzers werden dessen Daten zunächst in einem Formular erfasst und nach dem Absenden an das CMS weitergeleitet. Um Duplikate bei Nutzernamen oder Mailadressen zu vermeiden, werden diese Angaben im Backend mit bisherigen Einträgen überprüft. Falls es keine Übereinstimmung gibt, wird ein neuer Nutzer in der Datenbank angelegt, während andernfalls der User darauf hingewiesen wird, seine Daten zu ändern.

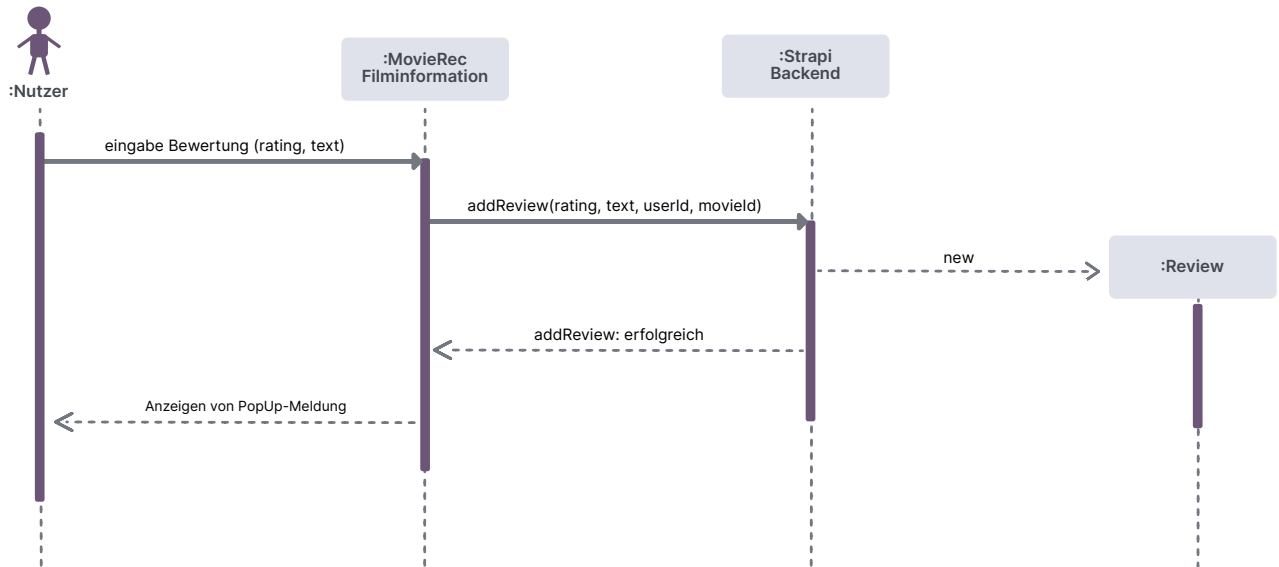
Anmelden



▲ **Abbildung 50** Sequenzdiagramm zur Nutzerinteraktion „Anmelden in der App“

Nahezu ähnlich läuft die Interaktion bei dem Versuch der Anmeldung eines bereits registrierten Nutzers ab. Hier gibt dieser seinen Nutzernamen mit dem entsprechenden, zugehörigen Passwort in die Formularfelder ein. Die Angaben werden im Content Management System mit dem Nutzereinträgen in der Datenbank abgeglichen. Liegt ein Fehler vor, so wird der Nutzer darauf hingewiesen, seine Angaben zu überprüfen. Stimmen Name und Passwort überein, wird er zur Seite seines persönlichen Nutzerprofils weitergeleitet.

Bewertung hinzufügen

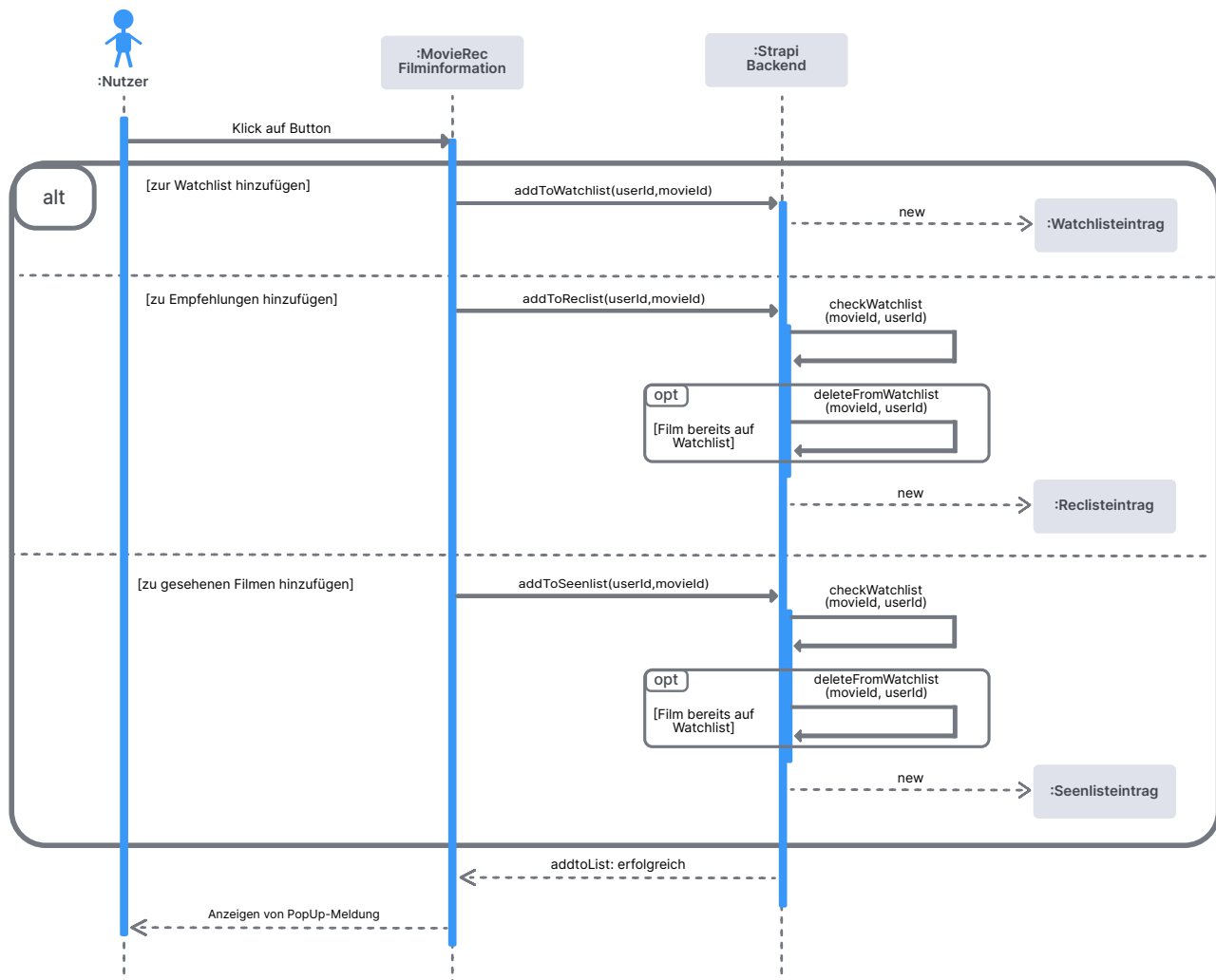


▲ **Abbildung 51** Sequenzdiagramm zur Nutzerinteraktion „Bewertung hinzufügen“

Wird ein Film eines Nutzers bewertet, so wird das Rating und ein optionaler Kommentar an das CMS weitergegeben und dort ein neuer Eintrag für den entsprechenden Film in der Datenbank angelegt. War dieser Vorgang erfolgreich, wird der User über eine kurze Benachrichtigung darüber informiert.

Zu Liste hinzufügen

Auf jeder Filmseite hat der Nutzer die Option, einen Film auf seiner Watchlist zu vermerken, um ihn in Erinnerung zu behalten, diesen als bereits gesehen zu markieren oder zu seinen Empfehlungen hinzuzufügen. Bei letzteren beiden Optionen wird bei der Interaktion im Backend überprüft, ob sich der Film noch auf der Watchlist des Users befindet und davon gelöscht, bevor ein neuer Eintrag auf der Empfehlungs- oder Gesehenenliste erstellt wird.



▲ Abbildung 52 Sequenzdiagramm zur Nutzerinteraktion „Zu Liste hinzufügen“

Implemen- tierung

Während im vorangegangenen Kapitel ausführlich die Konzeption der Webanwendung *MovieRec* beschrieben wurde, soll dieser Abschnitt die anschließende Implementierung des Backends und Frontends sowie die Umsetzung der zuvor definierten Anforderungen behandeln, anhand von Codeausschnitten erklären und dabei besonderes Augenmerk auf den Einsatz des Headless CMS, dem Rendern mit verschiedenen Ansätzen, als auch die Umwandlung zu einer Progressive Web App in einem praktischen Projekt mit dem zuvor erläuterten Technologiestack gelegt werden.

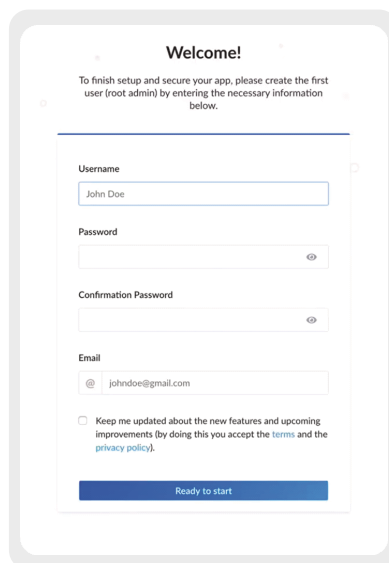
6.1 Backend

Als erster Schritt bei der technologischen Umsetzung von MovieRec wird das entkoppelte Content Management System Strapi eingerichtet und dort die Inhaltstypen nach dem in *Kapitel 5.5* konzipierten Datenmodell erstellt, sodass diese für die spätere Programmierung des Frontends über die Schnittstelle des CMS für den Nutzer bereitgestellt werden können.

6.1.1 Installation und Einrichtung

Da es sich bei Strapi um ein Open-Source-System handelt und als On-Premise-Anwendung zu nutzen ist, kann dies kostenlos eingesetzt und für die Entwicklung lokal installiert und zunächst dort ausgeführt werden. Falls auf dem Computer bereits die Laufzeitumgebung Node.js vorhanden ist, besteht die Möglichkeit, das in JavaScript entwickelte Content-Management-System mit geringem Aufwand einfach über eine Zeile im Command Line Interface der Konsole automatisch einzurichten und zu starten. Hierfür wird der Befehl *npx create strapi-app* genutzt, hinter dem nur noch der entsprechende Projektname hinzuzufügen ist. Mit einer Bestätigung durch Enter wird das komplette Projekt erstellt, entsprechende Pakete installiert, Abhängigkeiten erfasst und ohne zusätzliche Angaben eine MySQL-Datenbank aufgesetzt. Über den weiteren Befehl *npm run develop* wird ein lokaler Entwicklungsserver gestartet und das Content Management System kann unter der entsprechenden Adresse *http://localhost:1337/admin* im Browser aufgerufen werden.

Beim Aufrufen der URL öffnet sich zunächst ein Registrierungs-Fenster, bei dem ein neuer Nutzeraccount angelegt wird, mit dem man sich fortan für den Zugriff als Administrator auf das CMS anmelden kann.

The image shows a web browser window displaying the Strapi admin interface's registration page. The page has a light gray background with a white central form area. At the top, it says 'Welcome!' followed by instructions to create a first user. The form contains fields for 'Username' (with 'John Doe' entered), 'Password', 'Confirmation Password', and 'Email' (with 'johndoe@gmail.com' entered). There is a checkbox for staying updated and a 'Ready to start' button at the bottom.

◀ Abbildung 53

Einrichten eines Administrationsaccounts bei dem CMS Strapi

6.1.2 Erstellen der Content Types

Das CMS Strapi bietet die Möglichkeit, über ein vorinstalliertes Plugin eigene Inhaltstypen zu definieren und anschließend zu jedem Content-Type beliebig viele Einträge innerhalb dieser sogenannten Collection hinzuzufügen. Das Vorgehen orientiert sich dabei an dem Datenmodell, das in der Konzeption als ER-Schema entworfen wurde.

Für den entsprechenden Content-Type, um später die Filme hinzuzufügen und zu verwalten, wird daher ein neuer Collection-Type angelegt und dieser mit dem Namen *Movie* erstellt.

Abbildung 54 ►
Erstellen eines neuen Content-Types in Strapi

CT Create a collection type

Configurations

BASE SETTINGS ADVANCED SETTINGS

Display name UID

Movie movie

The UID is used to generate the API routes and databases tables/collection...

Cancel Continue

Im nächsten Schritt werden die zugehörigen Attribute und Eigenschaften dieses Typs hinzugefügt und deren mögliche Datentypen definiert. Wie sich in *Abbildung 54* erkennen lässt, bietet Strapi hierfür vielfältige Möglichkeiten, sodass sich neben Texten oder Zahlenangaben auch ein Datum und Medieninhalte wie Bilder speichern lassen.

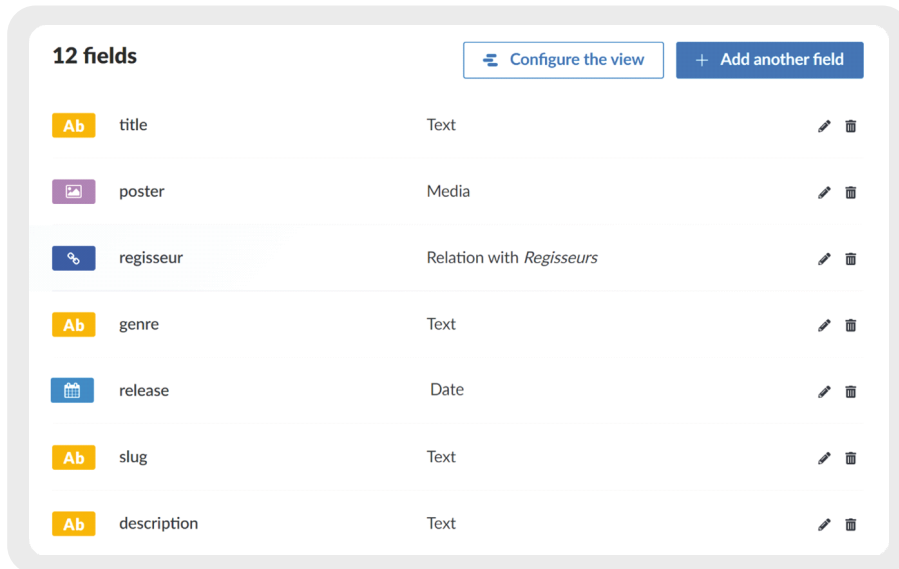
Abbildung 55 ►
Datentypen der Attribute in Strapi

CT Movies

Select a field for your collection type

Ab Text Small or long text like title or description	@ Email Email field with validations format
≡ Rich Text A rich text editor with formatting options	🔒 Password Password field with encryption
123 Number Numbers (integer, float, decimal)	☰ Enumeration List of values, then pick one
📅 Date A date picker with hours, minutes and seconds	🖼️ Media Files like images, videos, etc
🔴 Boolean Yes or no, 1 or 0, true or false	[...] JSON Data in JSON format
🔗 Relation Refers to a Collection Type	🔍 UID Unique identifier

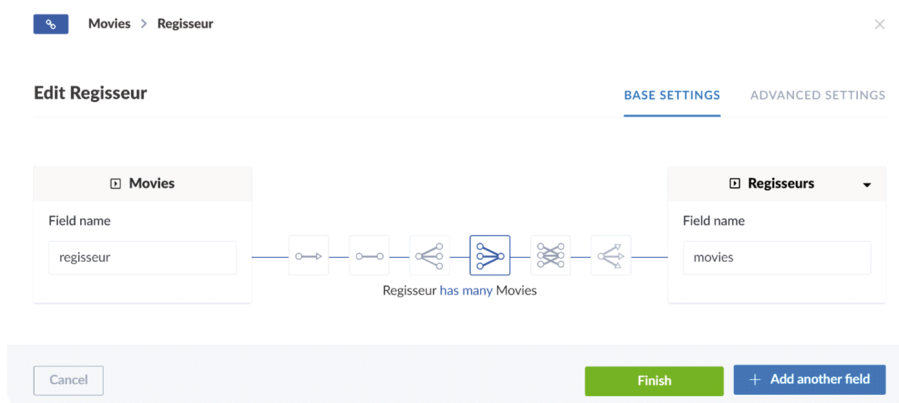
So können nach und nach immer weitere Felder für den Titel, die Beschreibung, das Erscheinungsdatum und ein zugehöriges Poster angelegt werden (siehe *Abbildung 55*). Dieses Vorgehen wird auch für einen neuen Content-Type *Regisseur* wiederholt und dafür Felder des Typs Text für seinen Vor- und Nachnamen sowie seine Biografie erstellt.



◀ **Abbildung 56**

Felder des Collection Types „Movie“

Bei der Betrachtung der *Abbildung* fällt neben den bereits beschriebenen Feldern die Verknüpfung mit dem Content-Type *Regisseur* auf. Eine solche Beziehung zwischen zwei Collections wird realisiert, in dem der Feldtyp *Relation* ausgewählt und anschließend ein anderer, bereits erstellter Collection-Type selektiert werden. Strapi bietet dabei die Möglichkeit, alle Art von Beziehungen zwischen den Komponenten zu implementieren. Da in diesem Fall jedoch ein Film nur von einem einzigen Regisseur gedreht, dieser jedoch mit beliebig vielen Filmen verknüpft werden kann, wird eine Viele-zu-Eins-Beziehung genutzt, wie in der folgenden *Abbildung 56* zu erkennen ist.



◀ **Abbildung 57**

Relation zweier Content-Types „Movie“ und „Regisseur“

Zusätzlich zu den Collections *Movie* und *Regisseur* werden nach dem konzipierten Entity-Relationship-Model die weiteren Content-Types *Review*, *Watchlist*, *Reclist*, und *Seenlist* erstellt, diese entsprechend untereinander verknüpft und die definierten Attribute hinzugefügt. Eine Besonderheit gibt es hier bei dem Modell *User*. Da dies bereits von Strapi automatisch bei der Einrichtung generiert wurde, verfügt dies vorab über bestimmte Felder wie Name, Username, Passwort und E-Mail. Hier müssen nur noch entsprechende Relationen zu den Content-Types der Listen und der Reviews erstellt werden müssen, damit registrierte Nutzer der App später eigene, persönliche Einträge hinzufügen können.

Nach dem Anlegen der einzelnen Content-Types können im nächsten Schritt die Inhalte für die einzelnen Filme und ihre Regisseure hinzugefügt werden. Dazu wird die jeweilige Collection auf dem Dashboard von Strapi ausgewählt und ein neuer Eintrag hinzugefügt. In die verschiedenen, zuvor definierten Felder können nun die benötigten Informationen eingegeben, Medienassets wie das Poster hochgeladen und wie in der Abbildung auf der rechten Seite zu erkennen ist, die Verknüpfung mit bereits erstellten Regisseuren ausgewählt werden.

Abbildung 58 ►
Erstellen eines neuen Eintrags vom Content Type „Movie“

The screenshot shows the 'Create an entry' form for the 'movies' collection in Strapi. The form is divided into several sections:

- Title:** A text input field containing 'Pulp Fiction'.
- Poster:** A large image placeholder with the text 'Click to select an asset or drag & drop a file in this area'.
- Description:** A text area containing a detailed description of the movie 'Pulp Fiction'.
- Genre:** A text input field containing 'Drama'.
- Release:** A text input field containing '1994'.
- Slug:** A text input field containing 'pulp-fiction'.
- Information:** A section with 'LAST UPDATE' and 'BY' fields, both currently empty.
- Regisseur:** A section with a dropdown menu 'Add an item...' and a list of selected directors: 'Quentin Tarantino', 'Christopher Nolan', and 'Martin Scorsese'.
- Reclists:** A section with a dropdown menu 'Add an item...' and a list of selected reclists: '0'.

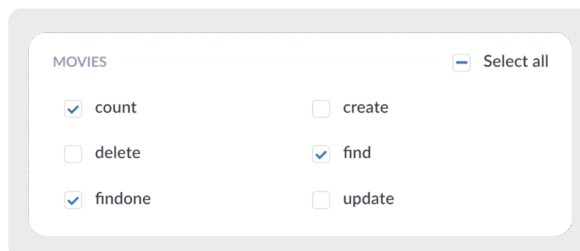
At the top right of the form, there are two buttons: 'Publish' (disabled) and 'Save' (active).

6.1.3 API

Da das Headless CMS Strapi nicht über ein gekoppeltes Frontend zur Darstellung der gespeicherten Inhalte verfügt, stellt es diese über eine Schnittstelle bereit, sodass sie von einem oder mehreren Anwendungen konsumiert und abgerufen werden können. Neben einer GraphQL-Schnittstelle über ein zusätzliches Plugin steht standardmäßig eine RESTful-API (*siehe 1.1.2*) zur Verfügung, die für die Entwicklung der Anwendung genutzt werden soll. Diese erfüllt alle Anforderungen und es muss keine weitere, neue Datenabfragesprache gelernt werden muss.

Um die einzelnen Content-Types zugänglich zu machen, müssen die Genehmigungen für diese erst in den Einstellungen des CMS aktiviert werden. Hierfür wird von Strapi zwischen den zwei vordefinierten Rollen *Public* und *Authenticated* unterschieden. Während unter der ersten Rolle auch unangemeldete Nutzer in der Lage sind, die Inhalte, sobald diese freigegeben wurden, abzurufen, ist dies mit zweiterer Rolle nur authentifizierten, angemeldeten Nutzern möglich.

Da die Informationenseiten über Filme und Regisseure in dieser Anwendung von jeglichen, auch nicht registrierten Nutzern eingesehen werden können, müssen für diesen Content-Type unter der Rolle *Public* die Felder *count*, *find* und *findone* aktiviert werden, sodass die entsprechenden Daten später über das Frontend mittels der HTTP-GET-Methode über die URL-Route mit */movies* für Filme oder */regisseure* für die jeweiligen Filmemacher abgerufen werden.



◀ Abbildung 59

Freigabe der API-Endpunkte für den Content-Type „Movie“

Aufgrund dessen, dass es in der späteren Applikation auch spezifische Nutzerseiten mit deren Empfehlungen und gesehenen Filmen gibt, können auch hier die GET-Routen für unangemeldete Benutzer freigegeben werden. Um Einträge auf den Watchlisten oder neue Reviews und Bewertungen hinzuzufügen oder zu ändern, müssen die HTTP-Methoden *POST* bzw. *PUT* genutzt werden. Da dies nur für Nutzer möglich sein soll, die bereits einen freigegebenen Account angelegt haben, werden die Felder *create* und *update* nur für die Rolle *Authenticated* aktiviert, sodass später nur angemeldete Benutzer Einträge dieses Content-Typs über das Frontend hinzufügen oder manipulieren können.

Sind die zugehörigen Berechtigungen vergeben, ist es möglich, die im CMS verwalteten Informationen über eine URL abzurufen und einzusehen. Die gesicherten Daten werden auf Anfrage dann im JSON-Format (*siehe 1.1.3*) ausgeliefert. Um zu überprüfen, ob die Erlaubnis der entsprechenden Content-Types für öffentliche Nutzer eingerichtet wurde, kann die Adresse unter dem entsprechenden Localhost und dem Namen der Collection als Route im Browser eingegeben werden. In folgendem Beispiel liefert Strapi auf eine GET-Anfrage der Adresse `http://localhost:1337/movies` alle bisher angelegten Einträge der Collection Movie aus.

```
{
  "id": 1,
  "title": "Pulp Fiction",
  "regisseur": {
    "id": 1,
    "name": "Quentin Tarantino",
    "slug": "tarantino",
    "published_at": "2022-05-13T07:39:28.597Z",
    "created_at": "2022-05-13T07:39:20.633Z",
    "updated_at": "2022-07-19T01:05:02.250Z",
    "portrait": { ... } // 17 items
  },
  "genre": "Krimi",
  "release": "1994",
  "slug": "pulp-fiction",
  "description": "Vincent Vega und Jules Winnfield holen für ihren Boss Marsellus Wallace eine schwarze Aktentasche aus einer Wohnung ab. Drei Jungs, die ihnen dabei im Weg stehen, lassen ihr Leben. Die Killer machen sich mit einem vierten Jungen als Geisel auf den Weg ins Hauptquartier. Doch als das Auto über eine Straßenerhöhung schaukelt, erschießt Vincent aus Versehen die Geisel. Um den blutverspritzten Wagen zu reinigen, machen die beiden einen Abstecher zu Jules's Freund Jimmie, wo auf Befehl vom Boss ein Spezialist für schwierige Aufträge zu ihnen stößt: The Wolf...",
  "regisseurShort": "Tarantino",
  "published_at": "2022-05-13T06:36:16.180Z",
  "created_at": "2022-05-13T06:36:12.225Z",
  "updated_at": "2022-07-20T16:56:47.210Z",
  "poster": { ... }, // 17 items
  "watchlists": [ ... ], // 1 item
  "reclists": [ ... ], // 1 item
  "seenlists": [ ... ], // 1 item
  "reviews": [ ... ] // 1 item
},
```

▲ **Abbildung 60** Daten eines Eintrags der „Movie“-Collection im JSON-Format

So lassen sich hier die zuvor angelegten Felder wie *title*, *genre*, *release* oder *slug* und die zugehörigen Einträge erkennen. Verknüpfte Collections wie *Regisseur* oder *Review* und deren Attribute sind hier typischerweise für das JSON-Format als verschachteltes Objekt in jedem einzelnen Eintrag enthalten.

Query Filter

In Strapis REST-API können zusätzliche praktische URL-Parameter genutzt werden, um die Ergebnisse eines GET-Requests an einen API-Endpunkt im Voraus zu filtern, einzuschränken oder zu sortieren.

Die einzelnen Parameter werden dabei, wie in folgenden Beispielen ersichtlich ist, als Query-String nach einem Fragezeichen hinter der eigentlichen URL geschrieben und können so von dem CMS bei einer Anfrage ausgewertet werden. Dabei stehen viele verschiedene Operationen und diverse Filtermöglichkeiten zur Verfügung, die sich beliebig miteinander kombinieren lassen und auch verknüpfte Content-Types erfassen können, sodass sich gezielt nach Einträgen suchen lässt.

Operation	Beispiel	Erklärung
Sortieren	/movies?_sort=release:DESC	Die Filme werden nach ihrem Erscheinungsdatum absteigend sortiert
Filtern	/users?name=John	Alle User deren Name John enthält werden zurückgegeben
Filtern	/reviews?rating_gte=4	Es wird nach allen Reviews, deren Rating 4 oder besser beträgt, gefiltert
Begrenzen	/regisseur?_limit=10	Es werden nur die ersten 10 Einträge des Types Regisseur zurückgegeben
Deepfiltering	/movies?reviews.rating=5	Alle Filme die mit einer Review mit einer 5 Sterne Bewertung verknüpft sind
Kombination	/movies?genre=krimi&release=2010	Alle Filme des Genre Krimi, die im Jahr 2010 erschienen sind

◀ Abbildung 61

Beispiele verschiedener API-Parameter in Strapi

Authentifizierung

Um bestimmte Informationen wie beispielsweise sensible Nutzerdaten vor unbefugten Zugriffen zu schützen, können entsprechende API-Endpunkte wie bereits oben beschrieben statt für die Rolle *Public* nur für *Authenticated* freigegeben werden. Für eine entsprechende Authentifizierung eines Users bietet das CMS Strapi von Haus aus zwei vordefinierte Schnittstellenendpunkte, sodass sich über einen POST-Request mit beigefügten Nutzernamen und Passwort an */auth/local* ein bereits registrierter Benutzer anmelden kann und sich über den Endpunkt */auth/local/register* neue User registrieren können.

Für den Authentifizierungsvorgang setzt das Content Management System *JSON Web Tokens*, kurz JWT, ein. Dabei handelt es sich grundlegend um einen standardisierten Access Token, mit dem sich JSON-Objekte sicher zwischen zwei Kommunikationspartnern, in diesem Fall der Frontend-Applikation und dem CMS, austauschen lassen.⁸⁰

⁸⁰ (Luber, 2022)

Bei einer erfolgreichen Anmeldung erhält der Nutzer einen solchen Token, den er nun beispielsweise bei einer Anfrage mitsenden und sich authentifizieren kann und somit Zugriff auf die bestimmten Ressourcen erhält. Diese Vorgänge sind in der Regel zustandslos, wobei der Server keine Sitzungen oder Zustände speichern muss.

6.2 Frontend

Nachdem das Content Management System Sprapi, das in der Gesamtanwendung als Backend fungiert, installiert, eingerichtet und dort erste Content-Typen erstellt wurden, kann parallel dazu mit der Implementierung des Frontends der Applikation mit den in *Kapitel 5.3* beschriebenen Frameworks und Bibliotheken begonnen werden. Dabei soll im Folgenden die Programmierung der wichtigsten Funktionen für die späteren Nutzer und Erstellung der jeweiligen Seiten und Komponenten nach dem in *Kapitel 5.4* konzipierten User Interface anhand von Ausschnitten des Quellcodes erläutert sowie die grundlegenden Konzepte und Funktionen bei der Entwicklung mit React und Next.js erklärt werden.

6.2.1 React State und Hooks

Bevor mit der eigentlichen Erklärung der Implementierung der Funktionen der Anwendung und dem Aufbau der mittels Next.js serverseitig und statisch gerenderten Seiten begonnen wird, sollen zum allgemeinen Verständnis von bestimmten Codeabschnitten noch zwei wichtige Konzepte der zugrundeliegenden Library React betrachtet werden, die für die Umsetzung eingesetzt wurden.

Wie bereits in *Kapitel 1.2.4* erläutert, ist React für seinen komponentenbasierten Aufbau bekannt, bei dem die UI im *return*-Statement der Funktion steht. Das Aussehen und Verhalten der Elemente lässt sich nun über zwei Wege beeinflussen:

- Die Eigenschaften (*Properties*), werden beim Erzeugen der Komponente von außen optional als Funktionsargument übergeben. Dabei handelt es sich um ein Objekt, das üblicherweise *props* genannt wird und den Namen und Wert des jeweiligen Properties enthält. Properties sind innerhalb der Komponente dann unveränderlich, können allerdings jederzeit extern neu gesetzt werden.⁸¹
- Bei dem sogenannten State handelt es sich um ein Objekt, das nur innerhalb der Komponente sichtbar ist, dafür allerdings im Gegensatz zu den Properties von der Komponente selbst verändert werden kann. Dieser lässt sich am besten als der aktuelle Ist-Zustand der Applikation oder eines einzelnen Elements beschreiben.⁸¹

Die wohl in der Praxis am meisten genutzte Möglichkeit, den State einer Komponente wiederzuverwenden und zu ändern, ist der State Hook. Die von React bereitgestellte Funktion *useState* liefert hierfür einen Array zurück. Während der erste Wert die Repräsentation des aktuellen States darstellt, ist der zweite Wert eine Funktion, über die der State geändert werden kann. Als Argument wird *useState* der initiale Zustand übergeben, der von jeglichem Datentyp sein kann, in diesem Beispiel eine Zahl mit dem Wert 0. In dem Fall wird dann bei einem Button-Klick der State über *setCount* geändert und die Komponente neu gerendert.⁸²



```

1 function Example() {
2   const [count, setCount] = useState(0);
3   return (
4     <div>
5       <p>You clicked {count} times</p>
6       <button onClick={() => setCount(count + 1)}>
7         Click me
8       </button>
9     </div>
10  );
11 }
12

```

◀ Codesnippet 7

Einsatz von *useState*-Hooks in React

6.2.2 Installation und Einrichtung

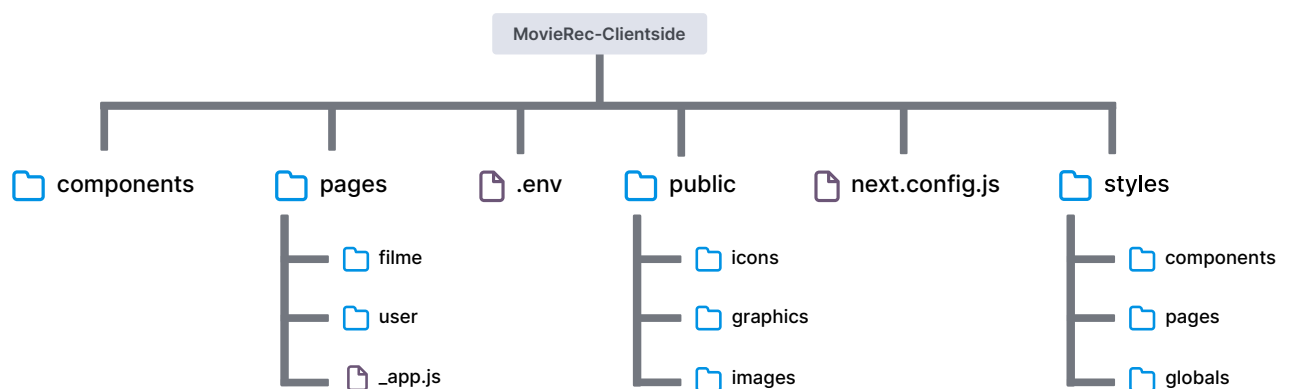
Mit Hilfe der Laufzeitumgebung Node ist es möglich eine Anwendung auf Basis von Next.js mit Hilfe eines einzigen Befehls *npx create next-app* in der Kommandozeile zu installieren. Hierbei müssen weder die Skripte der zugrunde liegenden Bibliothek React als auch das zugehörige Framework Next manuell hinzugefügt noch der Paketbundler Webpack aufgesetzt werden, wodurch Mehraufwand reduziert werden kann.

⁸¹ (Hartmann & Zeigermann, 2019)

⁸² (React.js Org, kein Datum)

Die Basisanwendung bietet zudem eine vorgefertigte Ordnersstruktur, die anschließend den Anforderungen des Projekts entsprechend angepasst werden kann sowie vordefinierte Befehle, mit Hilfe dieser sich ein Entwicklungsserver starten und die Applikation während der Entwicklung im Browser auf einem festgelegten Port 3000 unter der Adresse `http://localhost:3000` testen lässt.

Nachdem als Nächstes die in *Kapitel 5.3.1* vorgestellten und für die spätere Umsetzung einiger Features benötigten Pakete über den Node Package Manager installiert wurden, wird im nächsten Schritt die Verzeichnisstruktur nach der unten zu erkennenden Abbildung erweitert.



▲ **Abbildung 62** Verzeichnisstruktur der Frontend-Applikation

Das wichtigste Verzeichnis auf der obersten Ebene innerhalb des Projekts ist der Ordner *pages*. Dieser wird bereits bei der Installation von Next automatisch erstellt und enthält alle Seiten der Multipage-Applikation als JavaScript-Datei, die sich später unter dem Dateinamen als Ende der URL aufrufen lassen. Wird so beispielsweise ein JavaScript-Modul *login.js* in diesem Ordner angelegt, ist diese als Seite unter */login* verfügbar. Die Seiten sind dabei in ihrem Kern React-Komponenten, die typischerweise ihrem *return*-Statement die auf der Seite zu sehenden Inhalte als JSX enthalten.³⁹

Um später die einzelnen Überblicksseiten für die Filme sowie die Profilseiten der User zu implementieren, werden in *pages* mit *filme* und *user* zwei Unterverzeichnisse erstellt, um eine übersichtlichere Struktur für das Routing zu erhalten.

³⁹ (Riva, 2022)

Mit `_app.js` befindet sich bereits eine bei der Installation der Next-App generierte Datei im `pages`-Ordner. Hierbei handelt es sich jedoch nicht um eine Seite, die sich später über eine URL aufrufen lässt, sondern um eine übergeordnete Komponente, die für alle Seiten gültig ist und bei Initialisierung der Anwendung ausgeführt wird, um so seitenübergreifende Daten wie beispielsweise hier Informationen und die Listen eines gerade angemeldeten Nutzers abzurufen und der kompletten Anwendung zur Verfügung zu stellen. Zudem lassen sich in der `_app.js` auch globale Styles und Komponenten wie das Header-Menü, eine Cookie-Meldung oder das Panel mit den Einstellungen hinzufügen, die nun auf jeder Seite sichtbar sind und nicht mehr einzeln in jeder Datei im `pages`-Ordner ergänzt werden müssen.⁸³

```

1 function MyApp({Component}) {
2   return (
3     <
4       <Header/>
5       <Component {...pageProps} />
6       <HelpPanel/>
7       <MobileNav/>
8     </>
9   )
10 }
11
```

◀ Codesnippet 8

Übergeordnete `_app.js`-Datei mit einem Header, MobileNav und HelpPanel als globale Komponenten

Neben dem `pages`-Ordner werden auch in `components` React-Komponenten als JavaScript-Dateien gespeichert. Da diese jedoch nicht in dem speziell vorgesehenen Verzeichnis für Seiten liegen, handelt es sich hierbei um das Verzeichnis um verschiedene UI-Elemente wie den Header, den Footer, Karten, die Suche und Filter oder Buttons, die sich exportieren und so beliebig oft auf allen Seiten wiederverwenden lassen.

Mit der `.env`-Datei lassen sich Umgebungsvariablen, sprich global auch außerhalb der Anwendung verfügbare Werte, speichern und dient hier lediglich dazu die aktuelle Adresse `http://localhost:1337` unter der Variable `API_URL` zu definieren, sodass während der Entwicklung API-Aufrufe an das CMS durchgeführt werden können. Bei einem späteren Deployment muss nur noch die Variable mit der neuen URL aktualisiert werden und nicht mehr einzeln an jeder Stelle im Code geändert werden muss.

Im Verzeichnis `public`, das ebenfalls direkt bei der Erstellung der Next.js-App erstellt wird, können alle statischen Assets wie Logos oder Bilder und Grafiken für die Landingpage gespeichert werden.³⁹ Um diese übersichtlicher zu verwalten, werden hierzu weitere Unterordner angelegt.

³⁹ (Riva, 2022)

⁸³ (Next.js, kein Datum)

Bei der *next.config.js*-Datei handelt es sich um ein Konfigurationsmodul, das von Next.js während des Build-Prozesses und dem Starten des Servers ausgelesen wird und sich für spezielle Funktionen, wie dem Laden von optimierten Bildern aus einer externen API, anpassen lässt.

Zuletzt wird ein Ordner *styles* angelegt, der spätere alle CSS bzw. SCSS-Dateien enthält. Um auch hier diese übersichtlicher zu verwalten, werden mit *globals*, ein Ordner der Dateien wie ein Reset, Variablen und seitenübergreifende Definitionen enthält, *pages* für die Layouts der einzelnen Seiten und *components* für die Gestaltung der React-Komponenten, drei Unterverzeichnisse generiert.

6.2.3 Serverseitig gerenderte Filmübersicht

Eine zentrale Seite innerhalb der Anwendung ist die Übersicht mit allen Filmen. Dafür wird im Verzeichnis *filme* eine Datei *index.js* angelegt, sodass diese für Benutzer unter der URL */filme* erreichbar ist, da JavaScript-Module mit dem Namen *index* im späteren Routing nicht mit angegeben werden müssen. Die Übersicht soll dabei serverseitig gerendert werden und auf Anfrage eines Users alle gespeicherten Filme aus dem Content Management System abrufen und die Informationen auf der Benutzeroberfläche darstellen.

Next.js stellt mit *getServerSideProps* eine vorgefertigte asynchrone Funktion für genau diesen Zweck zur Verfügung. Bei dieser Methode wird über die JavaScript-Methode *fetch* ein API-Request an die Route */movies* des Strapi-Backends ausgeführt, die wie in *Kapitel 6.1.2* erläutert, die Informationen aller Filme bereitstellt. Hier wird auch noch einmal die Bedeutung der *.env*-Datei deutlich, deren Variable hier verarbeitet wird, da das CMS zunächst noch auf einem lokalen Host ausgeführt wird.

Nachdem die Antwort an das Frontend gesendet wurde, müssen die eigentlichen Daten extrahiert werden und können dann schließlich als *movies* im Property *props* an die Seite zurückgegeben werden.

Codesnippet 9 ►

Serverseitiger Abruf der Daten

```
1 export async function getServerSideProps() {
2
3   const { API_URL } = process.env
4   const res = await fetch(`${API_URL}/movies`)
5   const data = await res.json()
6
7   return {
8     props: {
9       movies: data
10    }
11  }
12 }
```

Das *props*-Objekt wird nun, wie in der unteren Abbildung zu erkennen ist, an die eigentliche Seite, bei der es sich typischerweise für Next im Kern um eine einfache JavaScript-Funktion handelt, weitergegeben. Dessen Daten werden dort allerdings nicht direkt verarbeitet, sondern an die Komponente *MovieList* ausgelagert, die das Objekt entgegennimmt.

Da dieses Objekt aus einem Array mit allen gespeicherten Filmen und deren Informationen besteht, wird dieser mit Hilfe der JavaScript-Funktion *map* Stück für Stück durchlaufen und jeder einzelne Film an eine weitere Komponente *MovieCard* weitergegeben. Anhand dieser Struktur wird ein weiteres Mal das bereits beschriebene komponentenbasierte Konzept von React deutlich, bei dem sich die einzelnen Bestandteile beliebig ineinander schachteln lassen, sodass hier die Komponente *MovieList* mehrere Card-Elemente für jeden einzelnen Film enthält.

Die Komponente *MovieCard* stellt als UI-Element nun bestimmte Informationen zu einem Film kompakt dar. Das Objekt *movie*, das die Komponente übergeben bekommt, enthält hier die im CMS gespeicherten Informationen mit dem Namen der in *Kapitel 6.1.2* angelegten Felder des Content-Types. Über die in der unteren Abbildung zu erkennenden Punktnotation kann dann in JavaScript auf die einzelnen Attribute wie beispielsweise den Filmtitel über *movie.title* zugegriffen, diese so auf der Karte dargestellt und in einer zugehörigen SASS-Datei gestylt werden.

```

1 const Movies = ({movies}) => {
2
3   return (
4     <>
5
6       <div className="container">
7         <div className={styles.movieList}>
8           <h1>Übersicht</h1>
9           <h2>Alle Filme im Blick</h2>
10          <MovieList movie={movie}/>
11        </div>
12      </div>
13      <Footer/>
14    </>
15  );
16 }
17

```

◀ Codesnippet 10

Next.js Seite mit *MovieList*-Komponente, die das Objekt „movie“ übergeben bekommt

```

1 {
2   id: 1,
3   title: "Pulp Fiction",
4   regisseur: {
5     name: "Tarantino"
6   },
7   genre: "Krimi",
8   release: "2009"
9 }
10

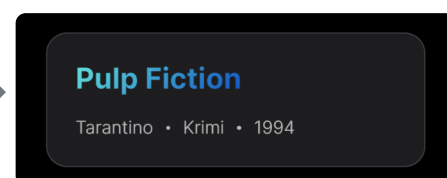
```



```

1 const MovieCard = ({movie}) => {
2
3   return (
4     <div className={styles.movieCard}>
5       <p>{movie.title}</p>
6       <div className={styles.movieInfo}>
7         <span>{movie.regisseur.name}</span>
8         <span>{movie.genre}</span>
9         <span>{movie.release}</span>
10      </div>
11    </div>
12  );
13 }
14 }
15

```



▲ **Codesnippet 11** JSON-Daten werden in der React-Komponente verarbeitet und letztendlich auf dem Screen für den User gerendert

6.2.4 Filtern und Suchen

Für die Filter- und Suchfunktion auf der Filmübersichtsseite wird eine neue React-Komponente *filter.js* angelegt, die sich aus einem Input-Feld für die Suche und zum anderen aus zwei Dropdown-Menüs für das Filtern nach den Genres und dem Sortieren nach Bewertungen oder Erscheinungszeitraum zusammensetzt.

Um den Filterprozess möglichst performant und effizient zu implementieren, wird die Library *React Query* eingesetzt. Diese stellt mit der Hook *useQuery* eine Funktion bereit, die später die gefilterten Daten als Variable *data* zurückgibt, die dann an die *MovieList*-Komponente weitergegeben werden können. *Use-Query* bekommt dabei initial zum einen ein Array von Keys, eine Funktion *getMovies* und die im Property *movies* (siehe *Codesnippet 10*) gespeicherten Filme als initialen Wert übergeben.

Codesnippet 12 ►

UseStates der Input-Felder zum Filtern und Definition von *React-Query*

```

1 const [genre, setGenre]=useState(null)
2 const [searchterm, setSearchterm]=useState(null)
3 const [sortPara, setSortPara]=useState(null)
4
5 const {data, status} =
6 useQuery(['movies', {genre: genre},{sortPara: sortPara}, {searchterm: searchterm}],
7         getMovies, {initialData: movies}
8 )
9

```

In dem Array sind drei darüber definierte States enthalten, sodass immer bei einer Änderung wie einer Eingabe in das Suchfeld oder Auswählen eines Feldes der Filter der entsprechende State geändert wird. Dies resultiert darin, dass *getMovies* aufgerufen und als Argument *filter* die aktuellen States übergeben bekommt.

Innerhalb von *getMovies* werden dann die einzelnen Inhalte aus dem Array extrahiert und jeweils in einer Konstante gespeichert. Diese werden nun im fetch-Request als URL-Query (siehe *Codesnippet 13*) genutzt um jeweils individuelle Anfragen, basierend auf den aktuell ausgewählten Filtern, an Strapi zu senden. In dem Codeausschnitt wird dabei aus übersichtsgründen nur die Anfrage für einen aktiven Genrefilter gezeigt, sodass, je nachdem welches Genre vom User ausgewählt wurde, eine passende Anfrage gesendet wird und in Echtzeit die angezeigten Filme aktualisiert werden, ohne, dass die komplette Seite neugeladen werden muss.

```

1 const getMovies = async(filter) =>{
2
3   const genreFilter=filter.queryKey[1].genre
4
5   if(genreFilter){
6     const res = await fetch(`${API_URL}/movies?genre=${genreFilter}`)
7     return res.json()
8   }
9 }
10

```

◀ Codesnippet 13

Aufruf der Funktion „getMovies“ und beispielhaftes Filtern nach einem Genre

6.2.5 Statisch generierte Einzelseiten

Während die Filmübersichtsseite und später der Feed serverseitig gerendert werden, sollen die jeweiligen undynamischen Einzelseiten aus Performancegründen und einer besseren User Experience für jeden Film nach dem Ansatz der Static Site Generation erzeugt werden. Um in der Applikation im Browser lesbare URLs zu erhalten können in Next.js Routen dynamisch erzeugt werden. Hierfür wird im Ordner *pages* in dem Unterverzeichnis *film* eine neue Seite als JavaScript-Datei angelegt, diese jedoch mit eckigen Klammern als *[slug].js* bezeichnet, in der später bei dem Abruf der Seiten der im Backend ergänzte Parameter *slug* steht.

Für die statische Generierung wird nun nicht mehr *getServerSideProps*, sondern mit *getStaticProps* eine weitere von Next bereitgestellte Funktion eingesetzt. Der Ablauf hierbei ist ähnlich wie bei dem serverseitigen Abruf der Daten. Nur wird hier der Prozess bereits beim Build ausgelöst und nicht erst auf Anfrage eines Nutzers. Über den URL-Query *?slug* werden nicht die Informationen aller Filme, sondern nur die eines Bestimmten mit dem entsprechenden Parameter angefordert und wieder über das Props-Objekt als *movie* an die Seite weitergegeben.

```

1 export async function getStaticProps({params}){
2
3   const { API_URL } = process.env
4   const slug=params.slug;
5
6   const res = await fetch(`${API_URL}/movies?slug=${slug}`)
7   const data = await res.json()
8
9   return{
10     props: {
11       movie: data[0],
12       params: params
13     }
14   }
15 }
16

```

◀ Codesnippet 14

Abruf der Informationen eines einzelnen Films mit Hilfe des Parameters „slug“

Auffällig ist, dass die `getStaticProps`-Funktion mit `params` einen eigenen Parameter entgegennimmt. Dieser ist Teil des von Next vordefinierten Context-Objekts und ist dafür zuständig, bei dynamischem Routing den über `params.slug` variablen Teil der URL auszulesen.

Da in dieser Anwendung nicht nur eine einzelnen Filmseite, sondern mehrere Seiten zu jedem im Backend gespeicherten Film statisch gerendert werden müssen, kommt unterstützend zu `getStaticProps` zusätzlich die Funktion `getStaticPaths` zum Einsatz, die beim späteren Build-Prozess die komplette Liste an URL-Pfaden erstellt, für die jeweils eine Filmseite gene-riert werden muss.

Codesnippet 15 ►

Erstellen der Liste an Pfaden der jeweiligen Filme für das dynamische Routing

```

1 export async function getStaticPaths() {
2
3   const { API_URL } = process.env
4   const res = await
5     fetch( `${API_URL}/movies` )
6   const data = await res.json()
7
8   const paths = data.map(movie =>{
9     return{
10       params: {
11         slug: movie.slug.toString(),
12       },
13     })
14   return {
15     paths
16   };
17 }
18
```

Über einen `fetch`-Request werden auch hier zunächst wie gewohnt die Liste aller Filme aus dem Backend angefordert, der Array mittels `map` durchlaufen, aber nun als Return-Wert nicht den kompletten Eintrag eines jeden Films zurückgegeben, sondern nur die `slug`, die dann als URL - Parameter für jeden einzelnen Film fungieren kann.

Über diesen Ansatz können mit einem einzigem JavaScript-Template hunderte Seiten zu den einzelnen Filmen im Backend erzeugt werden, die sich nun von den Usern über die im Content-Type in Strapi eingetragene `slug` als URL aufrufen lassen. Ist beispielsweise zu dem Film Pulp Fiction die zugehörige Slug `pulp-fiction` in Strapi gespeichert, lässt sich durch den Einsatz der oben beschriebenen Funktionen die statisch generierte Informationsseite zu diesem Film über die Route `/filme/pulp-fiction` öffnen.

Die Informationen im übergebenen `movie`-Objekt wie den Titel, das Jahr der Veröffentlichung, den Regisseur oder die Beschreibung der Handlung eines einzelnen Films lassen sich, wie bereits in *Codesnippet 11* gezeigt, auch hier über die Punktnotation in das JSX einbinden, stylen und schließlich als Informationsseite (*siehe Abbildung 63*) für den Nutzer darstellen.



◀ Abbildung 63

Screenshot einer statisch generierten Einzelseite, abrufbar über eine dynamische Route

6.2.6 Registrierung und Login

Für das Anlegen von Benutzeraccounts wird eine separate Seite *register.js* zur Registrierung von neuen Usern erstellt. In einem Formular kann dieser nun seine Daten wie einen Benutzernamen, seine E-Mail-Adresse oder ein gewünschtes Passwort eingeben. Eine Besonderheit in React ist es, dass statt herkömmlichen Input-Feldern sogenannte *Controlled Components* eingesetzt werden, um die Benutzereingaben entgegenzunehmen, zu validieren und schließlich an das Backend zu senden, sodass in Strapi ein neuer Eintrag des Content-Types *User* angelegt werden können.

Hierfür werden für die einzelnen Nutzerdaten zunächst State-Hooks (siehe Kapitel 6.2.1) erstellt. So werden Informationen des Namens des Users, wie man der ersten Zeile des Codeausschnitts erkennt, im State *name* gespeichert und können über *setName* geändert werden. Der initiale State wird in der Klammer hinter dem Funktionsaufruf definiert und ist hier in allen Fällen ein leerer String.

```
1 const [name, setName]=useState('')
2 const [username, setUsername] = useState('')
3 const [email, setEmail] = useState('')
4 const [password, setPassword] = useState('')
5
```

◀ Codesnippet 16

UseStates die, die Eingaben der Nutzer entgegennehmen

In den erstellten Input-Feldern wird nun mit der Funktion *onChange* immer bei Änderungen *setName* aufgerufen und der State auf die aktuelle Benutzereingabe gesetzt.

```
1 <input
2   type="text"
3   value={username}
4   onChange={e => setName(e.target.value)}
5 />
6
```

◀ Codesnippet 17

Beispielhaftes Controlled Input in React für die Eingabe des Usernames

Versucht ein User das Formular abzusenden, wird dies zunächst mit Hilfe der Bibliothek *React Hook Forms* (siehe *Kapitel 5.3.1*) validiert und überprüft, ob alle Felder korrekt ausgefüllt wurden. Ist dies nicht der Fall, wird darauf hingewiesen, fehlende Felder zu ergänzen oder zu korrigieren. Anderenfalls werden die erfassten Informationen über die recht umfangreiche Funktion *handleRegister* an Strapi gesendet. Da dabei keine Informationen abgerufen werden, kommt im Gegensatz zum Fetching der Filme in *Kapitel 6.2.4* die HTTP Methode POST anstelle von GET zum Einsatz. So werden die eingegebenen Daten in der Konstante *registerInfo* zusammengefasst und anschließend der POST-Request mit einem definierten Header für das JSON-Format sowie den Nutzerdaten im Body an die von Strapi speziell zur Registrierung neuer Nutzer angelegte Route, gestellt.

Codesnippet 18 ►

Funktion „handleRegister“ sendet Eingaben über einen POST-Request an das Backend und registriert den Nutzer

```

1 async function handleRegister() {
2
3   const registerInfo = {
4     name: name,
5     username: username,
6     email: email,
7     password: password
8   }
9
10  const register = await fetch(`${publicRuntimeConfig.API_URL}/auth/local/register`, {
11    method: "POST",
12    headers: {
13      'Accept': 'application/json',
14      'Content-Type': 'application/json'
15    },
16    body: JSON.stringify(registerInfo)
17  })
18
19  const registerResponse = await register.json();
20  setCookie(null, 'jwt', registerResponse.jwt, {
21    maxAge: 30 * 24 * 60 * 60,
22    path: '/',
23  })
24  Router.push('/dashboard')
25
26 }
```

Die Antwort des Backends wird ebenfalls in einer Konstante gespeichert. Damit ein neu registrierter Nutzer anschließend direkt angemeldet ist und sich fortan bei dem CMS authentifizieren kann, wird ein JWT-Token, der automatisch von Strapi erstellt wird, als Cookie gespeichert. Dazu dient die von der in *Kapitel 5.3.1* vorgestellten Bibliothek *Nookies* bereitgestellte Funktion *setCookies*, die den Token in den Cache der Applikation schreibt. Zuletzt wird der neue Nutzer nach erfolgreicher Registrierung zu seiner persönlichen Dashboard-Seite weitergeleitet.

Ein nahezu identischer Vorgang wird für die Anmeldung neuer Nutzer auf einer neuen Seite *login.js* implementiert. Im Gegensatz zur Registrierung werden jedoch hier nur der Nutzernamen und das Passwort in Controlled-Inputs als State entgegengenommen und wie in *Kapitel 6.1.3* dargestellt an den Endpunkt zur Anmeldung */auth/local* von Strapi gesendet.

6.2.7 Filmlisten und Reviews

Wie bereits in dem in *Kapitel 5.6* beschriebenen Sequenzdiagramm deutlich wird, handelt es sich bei dem Anlegen und Hinzufügen der persönlichen Filmlisten sowie dem Bewerten von Filmen um einen recht komplexen Vorgang. Die Hauptkomponente, die auf jeder Filmseite für den Nutzer sichtbar ist und zugleich die Logik der einzelnen Vorgänge enthält, ist ein geteilter Button, der aus drei verschiedenen Icons zum Hinzufügen zur Watchlist, zu den gesehenen Filmen oder zu seinen persönlichen Empfehlungen aufgebaut ist (siehe *Abbildung 64*).

Die simpelste Interaktion ist das Hinzufügen eines neuen Films zur Watchlist des Nutzers. Dazu wird die ID des Films sowie die ID des gerade angemeldeten Benutzers ähnlich wie bei der Registrierung oder Anmeldung in einer Konstante gespeichert und auch hier über einen Request mit der POST-Methode und den entsprechenden Informationen im Body an den API-Endpunkt */watchlists* im Backend gesendet. Da bei der Einrichtung des Backends festgelegt wurde, dass das Hinzufügen nur registrierten Nutzern gestattet ist, muss im Header des Requests noch der JWT zur Authentifizierung hinzugefügt werden.

```
1 const addToWatchlist = async () =>{
2   const addmovie = {
3     movies:[{id:movie.id}],
4     users:[{id:userId}]
5   }
6   const jwt = parseCookies(ctx).jwt
7
8   const add = await fetch(`${API_URL}/watchlists`, {
9     method: "POST",
10    headers: {
11      Authorization: `Bearer ${jwt}`,
12      'Accept': 'application/json',
13      'Content-Type': 'application/json'
14    },
15    body: JSON.stringify(addmovie)
16  })
17
18  showNotification("zur Watchlist")
19  const addResponse = await add.json()
20 }
21
```

◀ Codesnippet 19

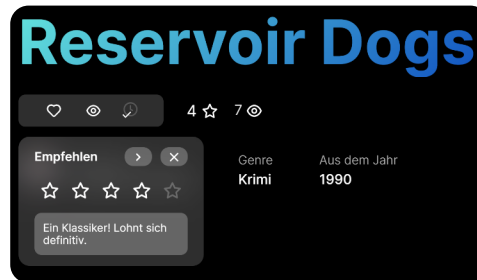
„addToWatchlist“ um Filme im Backend zur eigenen Watchlist hinzuzufügen

Klickt der User hingegen auf eines der beiden anderen Icons, wird der Film nicht direkt zu den gesehenen oder empfohlenen Filmen hinzugefügt, sondern es öffnet sich ein kleines Fenster unterhalb, in das der Nutzer eine Bewertung abgeben und einen kurzen Kommentar zu dem Film schreiben kann. Sendet dieser die Eingaben mit Klick auf den kleinen oberen Button ab, so wird zum einen die ID des Films zusammen mit der ID des Users wie auch schon beim Hinzufügen zur Watchlist mit einem POST-Request an den Endpunkt `/reclists` bzw. `/seenlists` gesendet. Zudem stellt die

Frontend-Applikation einen weiteren POST-Request an die API `/reviews` des CMS, um so einen neuen Eintrag mit der Bewertung und dem Text des Users zu dem jeweiligen Film hinzuzufügen.

Abbildung 64 ►

Ausschnitt des Buttons auf der Filmseite, um den Film zu einer Liste hinzuzufügen und zu bewerten



Filme, die der User auf der persönlichen Watchlist vermerkt hat, sollen außerdem bei einer Empfehlung oder dem Hinzufügen zur Liste mit den gesehenen Filmen automatisch von dessen Watchlist gelöscht werden. Dafür wird eine Funktion `removeFromWatchlist` geschrieben, die ebenfalls bei dem Hinzufügen einer Review ausgeführt wird. Hier wird die HTTP-Methode DELETE genutzt, um bei einer entsprechenden Anfrage an das CMS mit Hilfe von API-Parametern mit der ID des Users und des Films den richtigen Eintrag im Content-Type Watchlist zu finden und so zu löschen.

Codesnippet 20 ►

Löschen eines Films von der Watchlist über die HTTP-Delete-Methode

```

1 const removeFromWatchlist = async () =>{
2
3   const deleted = await fetch( `${API_URL}/watchlists?movie.id=${movie.id}&user.id=${user.id}`, {
4     method: "DELETE"
5   })
6
7   const deleteResponse = await deleted.json()
8 }
9

```

6.2.8 Clientseitig gerenderte Ratinginformationen

Bei der statischen Generierung der jeweiligen Einzelseiten der Filme werden einmal zum Build-Prozess die im Backend verwalteten Informationen erfasst. Da User jedoch zu beliebigen Zeitpunkten bei der Nutzung der App neue Reviews hinzufügen können und sich so das Rating eines Films ständig ändern kann, macht es aufgrund der fehlenden Datenaktualität dieses Ansatzes hier keinen Sinn, die Durchschnittsbewertung ebenfalls schon zur Generierung zu errechnen und im HTML zu rendern.

Stattdessen kann für die Darstellung des Ratings und einer Angabe, von wie vielen Nutzern der Film bereits gesehen wurde, für diese Teilkomponente der Seite ein clientseitiger Renderingansatz verfolgt werden. Bei diesem werden erst bei Aufruf und nach dem Laden der Seite diese Informationen angefordert und somit immer alle bis zu diesem Zeitpunkt abgegebenen Bewertungen berücksichtigt.

React stellt für diesen Zweck mit dem *useEffect*-Hook eine weitere praktische Funktion für das clientseitige Rendern bereit, die jedes Mal aufgerufen wird, wenn sich eine Komponente in ihrem Zustand ändert. Da wie in dem Codeausschnitt zu sehen ist, in den eckigen Klammern hier kein Argument übergeben wird, greift der Funktionsaufruf nicht bei einer Änderung, sondern nach dem Laden der Seite. Dabei wird wie gewohnt über die *fetch*-Methode ein GET-Request an den Endpunkt */reviews* des Backends gesendet. Um hier nur die Reviews zum aufgerufenen Film zu erhalten, kommen erneut die praktischen URL-Queries in Strapi zum Einsatz, die bereits auf der Serverseite angeforderte Daten filtern können und man auf diese Weise nur die Einträge zu einem bestimmten Film erhält.

```
1 useEffect(()=>{
2   fetch(`${API_URL}/reviews?movies.id=${movie.id}`)
3   .then((res) => res.json())
4   .then((data) => {
5     const ratingList = data.map(item => item.rating)
6     var ratingSum = 0;
7     ratingList.forEach(number => ratingSum += number)
8     ratingSum = ratingSum/ratingList.length
9     var ratingSumRounded =Math.round((ratingSum + Number.EPSILON) * 10) / 10;
10    setAverageRating(ratingSumRounded)
11  })
12 },[])
13
```

◀ Codesnippet 21

Clientseitiger Abruf der Ratinginformationen und errechnen eines Durchschnittswertes

Um letztendlich die durchschnittliche Bewertung aller Reviews auszurechnen, sind noch weitere Schritte in dieser Funktion implementiert. So werden zunächst mittels der *map*-Methode aus dem gesamten Array der Reviewobjekte nur die Ratings erfasst und in einen separaten Array geschrieben. Schließlich werden die darin enthaltenen Werte aufaddiert und durch dessen Länge geteilt und zuletzt auf eine Nachkommstelle gerundet.

Die Ratinginformation wird schließlich zusammen mit der Anzahl, wie oft der Film bereits gesehen wurde, auf der Filminformationsseite dargestellt (siehe *Abbildung 63*).

6.2.9 Serverseitig gerenderter Feed

Wie in *Kapitel 2.2* erläutert, bietet der Ansatz des Server Side Renderings eine hohe Datenintegrität, was insbesondere bei dem Nutzerfeed der Anwendung von Bedeutung ist, da hier immer auf Anfrage alle neuen geteilten Empfehlungen und Reviews aus dem CMS erfasst werden sollen.

Die Datei *feed.js* ist daher ähnlich aufgebaut, wie schon die Seite zur Filmübersicht. So wird auch hier die Funktion *getServerSideProps* genutzt, um bei einem Request mittels der *fetch*-Methode auf der Seite des Servers die Daten vom Endpunkt */reviews* abzurufen. Da hier für den User immer die neusten geteilten Beiträge zuerst angezeigt werden sollen, diese in Strapi jedoch immer mit der nächsthöheren ID zu unterst eingefügt werden, können diese über den URL-Query *?sort=id:DESC* in ihrer Reihenfolge umgekehrt werden.

Codesnippet 22 ►

Serverseitiger Abruf der Reviews zur Bereitstellung auf der Seite „Feed“

```

1 export async function getServerSideProps() {
2
3   const { API_URL } = process.env
4   const res = await fetch(`${API_URL}/reviews?_sort=id:DESC`)
5   const data = await res.json()
6
7   return {
8     props: {
9       reviews: data
10    }
11  }
12 }
13
```

In der JavaScript-File werden die Daten nun wieder zuerst an die Seite und dort als Property der Komponente *ReviewList* übergeben. Auch hier lassen sich Parallelen zur *MovieList* auf der Filmübersichtseite erkennen.

Nachdem sich über *map* der empfangene Array wieder Eintrag für Eintrag erfassen lässt, wird hier nun jeder Eintrag nicht der Komponente *MovieCard*, sondern *ReviewCard* übergeben, die dann die Informationen des Review-Objekts zusammen mit der zugehörigen SCSS-Datei für den User als gestaltetes Element auf der Seite rendert.

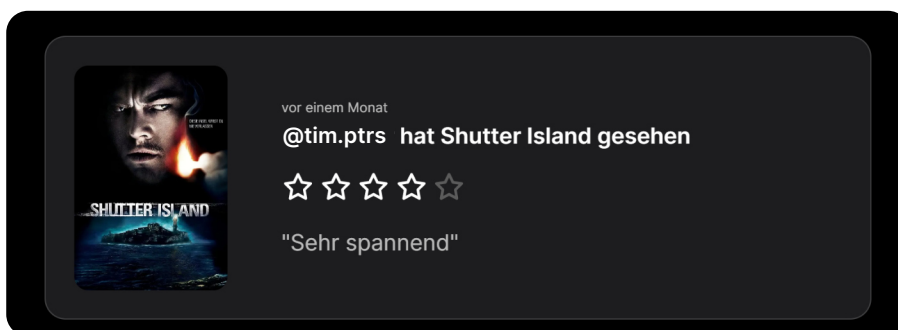
```

1 const ReviewList = ({reviews}) => {
2   return (
3     <div className={styles.reviewbox}>
4       {reviews.map(review => (
5         <ReviewCard review={review}/>
6       ))}
7     </div>
8   );
9 }
10

```

◀ Codesnippet 23

Durchlaufen aller Reviews und übergeben eines einzelnen Objekts an die Komponente der „Review Card“



◀ Abbildung 65

Ausschnitt einer Review-Card mit den entsprechenden Informationen aus dem Backend

6.2.10 Suchmaschinenoptimierung

Da die Anwendung aus serverseitig gerenderten und statisch generierten Seiten aufgebaut ist, die sich optimal zum Indizieren durch die Crawler der Suchmaschinen eignen, sollen nun auch für eine verbesserte Suchmaschinenoptimierung zusätzlich Titel und Meta-Descriptions in den Head der einzelnen Seiten geschrieben werden.

Um dies zu vereinfachen, kommt die bereits in *Kapitel 5.3.1* vorgestellte Bibliothek Next-SEO zum Einsatz. Diese bietet die Möglichkeit, alle erforderlichen Meta-Informationen zusammenzufassen. Dafür wird in der entsprechenden JavaScript-Datei der Seite im *pages*-Ordner eine neue Konstante mit dem Namen *SEO* definiert und der jeweilige Titel und die Beschreibung der Seite hinzugefügt. Diese Konstante wird dann an eine spezielle von der Bibliothek bereitgestellte React-Komponente *<NextSeo/>* übergeben und auf diese Weise bei dem Rendern der Seite automatisch im HTML-Head übernommen.

Codesnippet 24 ►

Hinzufügen von Metadaten zu einer Seite über „Next-SEO“

```

1 const SEO = {
2   title: 'Übersicht - MovieRec',
3   description: 'Erhalte mit MovieRec alle Informationen zu deinen Lieblingsfilmen auf einen Blick'
4 }
5
6 return (
7   <>
8     <NextSeo {...SEO}/>
9     <div>
10      /... Content
11    </div>
12    <Footer/>
13  </>
14 );
15

```

6.2.11 Upgrade zur PWA

Um die Webanwendung mit den typischen Features einer PWA zu erweitern, wird das Paket Next-PWA installiert und anschließend die *next.config.js*-Datei ergänzt, sodass später bei dem Build-Prozess die Funktionen von Next-PWA dafür sorgen, dass automatisch ein Service-Worker implementiert und registriert wird. Standardmäßig bietet dieser einen Offline-Support, in dem der User im Falle einer fehlenden Verbindung zu einer bestimmten, ausgewählten Seite weitergeleitet wird. Es lassen sich jedoch auch verschiedenen andere Seiten für diesen Fall speichern, die dann für einen Offline-Support aus dem Cache abgerufen werden.

Codesnippet 25 ►

Erweitern der *next.config.js*-Datei für PWA-Funktionen

```

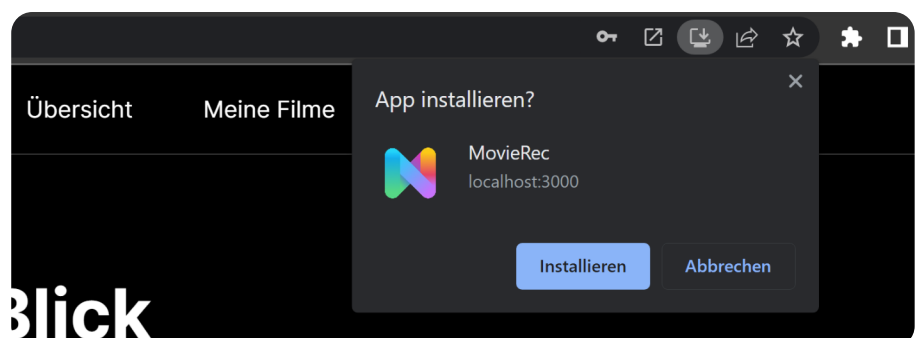
1 const withPWA = require('next-pwa')
2
3 module.exports = withPWA({
4
5   pwa: {
6     dest: 'public',
7     register: true,
8     fallback: 'offline.js'
9   }
10 //Rest der Next.config-Datei
11 })
12

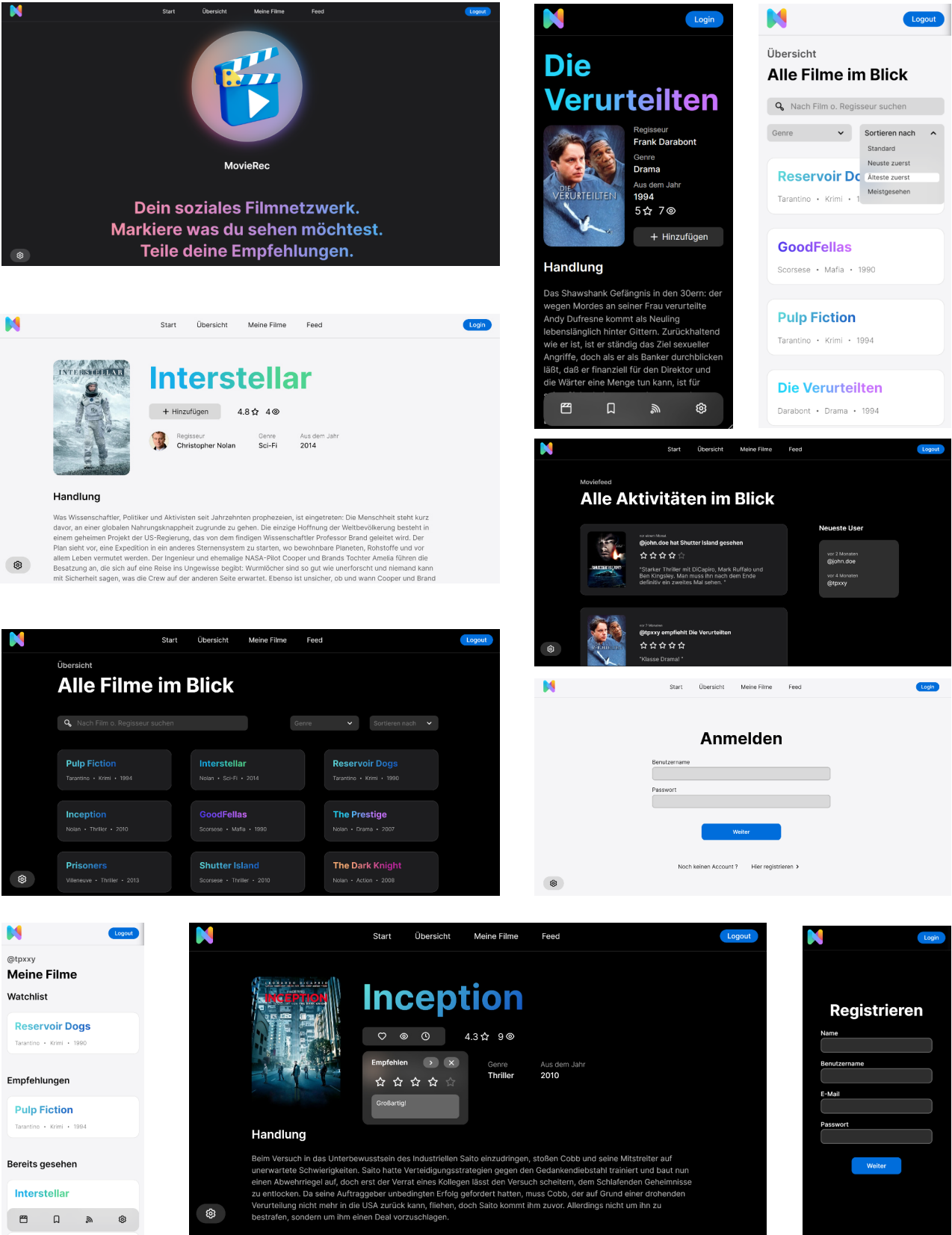
```

Zudem wird eine *manifest.json*-Datei im *public*-Ordner hinzugefügt, die sich an dem Aufbau aus *Codesnippet 6* orientiert und alle erforderlichen Metainformationen und Pfade zu den Icons enthält, sodass MovieRec später installiert und dem Homescreen hinzugefügt werden kann.

Abbildung 66 ►

Möglichkeit MovieRec aus dem Chrome Browser als PWA zu installieren





▲ Abbildung 67 Screenshots der fertigen Anwendung

Fazit und Ausblick

7.1 Fazit

Das übergreifende Ziel der Thesis war zum einen, mit SSR bzw. SSG, Headless CMS und PWAs drei der wichtigsten technologischen Trends aus der Welt der heutigen Webentwicklung zu betrachten, deren Hintergründe genauer zu erläutern sowie deren Stärken, Schwächen und Einsatzmöglichkeiten herauszustellen. Zudem sollte überprüft werden, ob sich diese Technologien miteinander kombinieren lassen, wodurch zu diesem Zweck zunächst eine prototypische Webapplikation konzipiert und anschließend mit Blick auf die zuvor untersuchten Ansätze umgesetzt wurde.

Bei dem Rendering von Webseiten oder Anwendungen gibt es inzwischen vielfältige Möglichkeiten, die jeweils verschiedene Vor- und Nachteile bieten. Während Single-Page-Applikationen typischerweise clientseitig im Browser gerendert werden, wird dieser Vorgang bei dem Server-Side-Rendering wieder auf die Gegenseite verlagert. Die statische Generierung basiert darauf, bereits im Voraus erzeugte HTML-Seiten zu hosten. Durch die inzwischen enorme Anzahl an modernen Librarys oder Static Site Generators stehen Entwicklern mittlerweile vielfältige technologische Optionen zur Verfügung, den für die jeweilige Anwendung passenden Ansatz zu implementieren. Frameworks wie Next.js oder Nuxt bieten dabei sogar die Möglichkeit, die einzelnen Methoden innerhalb einer Applikation durch hybrides Rendering zu vereinen und deren Stärken zu nutzen.

Während der Markt der Content Management Systeme lange Zeit von Wordpress oder typischen Homepage-Baukästen dominiert war, erfreuten sich in den letzten Jahren sogenannte Headless CMS immer mehr Beliebtheit. Wie sich aus deren Bezeichnung ableiten lässt, dienen diese lediglich zur Verwaltung und Speicherung der Inhalte, nicht mehr jedoch zur Darstellung in einem fest gekoppelten Frontend. Dies muss dafür separat implementiert werden und die Inhalte über eine Schnittstelle abrufen, was jedoch so mehr Flexibilität und Freiheit in der Wahl der Technologie oder eines passenden Frameworks bietet und die Möglichkeit, auch weitere digitale Kanäle wie eine Smartwatch oder IoT-Anwendungen zu bedienen.

Progressive Web Apps sollen herkömmliche Webanwendungen erweitern und eine plattformübergreifende Alternative für klassische native Apps darstellen, die nur noch auf einer einzigen Code-Basis entwickelt werden müssen und auf jedem Betriebssystem ausführbar sind.

So bieten PWAs durch den Einsatz von JavaScript-Service Workern und einem Web App Manifest Features wie eine Installationsmöglichkeit, Push-Benachrichtigungen oder Offline-Support.

Mit dem Entwurf und der Implementierung der Anwendung *MovieRec* wurden schließlich all diese Technologien zusammengeführt und versucht, auf diese Weise eine moderne performante Webapplikation umzusetzen, die möglichst alle im Voraus definierten Anforderungen erfüllt und für den Nutzer sinnvolle Funktionen und eine durchdachte UI bietet.

Die Kombination der drei Ansätze verlief dabei insbesondere durch Wahl des Technologiestacks mit dem Headless CMS Strapi und dem Frontend-Framework Next.js nahezu problemlos. Vor allem das Zusammenspiel aus einem entkoppelten CMS im Backend und der funktionsreichen JavaScript Bibliothek Next auf der Clientseite bietet gerade für interaktive und dynamische Webanwendungen mit erhöhtem Datenaustausch enormes Potential. Über die mächtigen Funktionen *getServerSideProps* und *getStaticProps* ist es simpel, die benötigten Daten zur Darstellung im Frontend über die REST-API von Strapi im JSON-Format abzurufen und so direkt zu verarbeiten und gleichzeitig dabei zu entscheiden, ob dieser Vorgang bereits zum Build oder erst auf Anfrage gestartet werden soll. Allein für eine noch effizientere Abfrage, was aber nur bei einer Vielzahl an Usern oder einer enormen Menge an gespeicherten Filmen von Bedeutung ist, wäre es sinnvoll gewesen, GraphQL zu nutzen, allerdings hätte die Sprache zusätzlich noch erlernt werden müssen. Durch das entkoppelte Frontend konnte außerdem nicht nur das Framework frei gewählt werden, sondern auch einige spezielle Features wie einen Darkmode, dynamisches Routing mit statischen Seiten oder eine komplexe Rating- und Reviewfunktion elegant umgesetzt werden, was beispielsweise in einem monolithischen System wie Wordpress nur schwer oder extrem umständlich möglich gewesen wäre.

Zudem verfolgen Progressive Web Apps und entkoppelte CMS in ihren Grundzügen mit einer geräte- und betriebssystem-übergreifenden Idee teils ähnliche Ansätze. Die Funktionen der PWA müssen dabei mittlerweile nicht mehr alle selbst implementiert werden, sondern können inzwischen über vorgefertigte Module einfach der Anwendung hinzugefügt und bei Bedarf angepasst werden, sodass es recht einfach ist, eine herkömmliche Webapplikation wie *MovieRec* mit Funktionen einer nativen App aufzuwerten..

Dies macht bei klassischen statischen Informations- oder Unternehmensseiten und Portfolios meist weniger Sinn, kann jedoch gerade bei einer interaktiven und personalisierbaren Anwendung wie hier vor allem auf Mobilgeräten zu einer optimierten User Experience und besserer Nutzerbindung führen.

Die in MovieRec eingesetzten Technologien im Front- und Backend verfügen außerdem alle über ausführliche Dokumentationen, eine Vielzahl an Tutorials, Lernvideos oder Beispielen und eine große Community, sodass die Einarbeitung recht leicht viel und mit diesem Technologiestack in der vorgegebenen Zeit bereits als Einzelperson eine umfangreiche Webanwendung erstellt werden konnte, was die drei Ansätze auch in Kombination, vor allem für kleinere Start Ups mit begrenzten Personalmöglichkeiten und Ressourcen interessant macht, die gerade mit Headless CMS und modernen JavaScript-Frameworks sehr gut auch plattformübergreifende Projekte realisieren können.

7.2 Ausblick

In kaum einem anderen Bereich in den letzten Jahren sind so viele Fortschritte und neue Möglichkeiten zu beobachten, wie in dem der Webentwicklung. Dabei kommen und gehen neue Trends nicht nur mit Blick auf das Design der UI, sondern auch auf der technologischen Ebene. So hätte es vor einigen Jahren kaum einer für möglich gehalten, dass die damals enorm beliebten Single-Page-Applications durch statisch generierte HTML-Assets abgelöst werden, die die Grundlage des JAMStacks bilden und deren Idee ursprünglich aus den Anfängen des Webs stammt.

Daher ist es schwer, eine sichere Prognose abzugeben, wie das Feld des Web Developments in den nächsten fünf oder zehn Jahren aussehen wird. Mit Sicherheit werden weitere neue Frameworks oder Bibliotheken veröffentlicht und die bereits verfügbaren mit neuen Versionen und Features erweitert, sodass gerade das Rendering der HTML-Seiten noch flexibler wird und Entwickler in Zukunft noch häufiger mit hybriden Ansätzen verschiedene Arten innerhalb einer Anwendung vereinen könnten. Insbesondere der JAMStack mit seiner serverlosen Architektur, bei dem die Seiten auf verteilten, cloudbasierten Content Delivery Networks gehostet werden, wird immer mehr zu einer Alternative zu klassischen Technologiestacks und sogenannte API-First Entwicklungsansätze rücken weiter in den Vordergrund.

Ein solches Konzept geht Hand in Hand mit den inzwischen immer beliebteren schnittstellenbasierten Headless CMS. So dürfte es spannend werden, zu beobachten, wie sich der Markt der Content Management Systeme in der Zukunft entwickelt. Mit Sicherheit haben klassische CMS weiterhin ihre Daseinsberechtigung und Wordpress wird nicht von heute auf morgen seine Monopolstellung verlieren. Die flexibleren und entkoppelten Alternativen dürften im Zeitalter von smarten, internetfähigen Geräten und dem Ausspielen von Inhalten auf eine Vielzahl von digitalen Kanälen immer wichtiger werden und leichtgewichtige Low-Code Builder wie Webflow weitere Kunden abgreifen.

Eine ähnliche Vorhersage könnte auch über Progressive Web Apps getroffen werden, die sich seit ihrer Entstehung 2015 durchaus etabliert, aber noch nicht vollends den Durchbruch geschafft haben, um native Apps langfristig zu ersetzen. Viel wird hier auch von der Einstellung Apples gegenüber dieser Technologie abhängen, die diese im Gegensatz zu Google bis heute zwar nicht ablehnen, jedoch mit Blick auf den enorm profitablen App Store auch nicht fördern wollen.

Anhang und Verzeichnisse

Literaturverzeichnis

- ¹ Ackermann, P. (2021). *Webentwicklung - Das Handbuch für Fullstack-Entwickler*. Bonn: Rheinwerk.
- ² Luber, S., & Donner, A. (19. Juni 2020). *IP-Insider: Was ist das Client Server Modell?* Von <https://www.ip-insider.de/was-ist-das-client-server-modell-a-940627/> abgerufen
- ³ RedHat. (8. Mai 2020). *RedHat: Was ist eine REST-API und was ist REST (Representational State Transfer)?* Von <https://www.redhat.com/de/topics/api/what-is-a-rest-api> abgerufen
- ⁴ Talend. (kein Datum). *REST-API: Definition, Funktionen und Bedingungen*. Von <https://www.talend.com/de/resources/was-ist-rest-api/> abgerufen
- ⁵ Uptrends. (kein Datum). *Uptrends: Was ist eine REST API?* Von <https://www.uptrends.de/was-ist/rest-api> abgerufen
- ⁶ Augusten, S. (20. April 2018). *IP Insider: Was ist JSON?* Von <https://www.dev-insider.de/was-ist-json-a-702243/> abgerufen
- ⁷ Svekis, L. L., Percival, R., & van Putten, M. (2021). *JavaScript from Beginner to Professional: Learn JavaScript quickly by building fun, interactive, and dynamic web apps, games, and pages*. Birmingham: Packt Publishing.
- ⁸ Frontend GmbH. (kein Datum). *Frontend GmbH: Was ist Frontend? Was ist Backend?* Von <https://www.frontend-gmbh.de/blog/frontend-und-backend/> abgerufen
- ⁹ Geißler, O., & Ostler, U. (5. Mai 2018). *Die IT-Schichtentheorie: Was ist ein Frontend und ein Backend?* Von <https://www.datacenter-insider.de/was-ist-ein-frontend-und-ein-backend-a-714429/> abgerufen
- ¹⁰ Mardan, A. (2018). *Full Stack JavaScript: Learn Backbone.js, Node.js and MongoDB*. San Francisco: Apress.
- ¹¹ Northwood, C. (2018). *The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer*. Manchester: Apress.

-
- ¹² Dowden, M., & Dowden, M. (2020). *Architecting CSS: The Programmer's Guide to Effective Style Sheets*. Brownsberg: Apress.
- ¹³ Monus, A. (2. Mai 2018). *Raygun: Popular CSS Preprocessors With Examples: Sass, Less & Stylus*. Von <https://raygun.com/blog/css-preprocessors-examples/> abgerufen
- ¹⁴ IONOS. (20. März 2020). *IONOS Digital Guide: Document Object Model (DOM): Definition, Aufbau und Beispiel*. Von <https://www.ionos.de/digitalguide/websites/web-entwicklung/was-ist-das-document-object-model-dom/> abgerufen
- ¹⁵ Mediaevents. (August 2021). *Mediaevents: Javascript Events*. Von <https://www.mediaevent.de/javascript/events.html> abgerufen
- ¹⁶ MDN Web Docs. (2. Mai 2022). *MDN Web Docs: Introduction to client-side frameworks*. Von https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Clientside_JavaScript_frameworks/Introduction abgerufen
- ¹⁷ Tillmann, J. (12. August 2017). *T3N: React vs. Angular – wann ist was besser?* Von <https://t3n.de/news/react-vs-angular-besser-845241/> abgerufen
- ¹⁸ Sturm, O. (12. März 2019). *Entwickler.de Blog: React – Mal was anderes als Angular*. Von <https://entwickler.de/javascript/react-mal-was-anderes-als-angular/> abgerufen
- ¹⁹ Tappert, I. (1. Januar 2020). *Tappert IT: Angular vs. React vs. Vue.js – JavaScript Frontend Frameworks im Vergleich*. Von <https://tappert-it.de/2020/01/01/angular-react-vue-javascript-frontend-frameworks-vergleich/> abgerufen
- ²⁰ Böhm, R. (19. Oktober 2020). *Angular-Tutorial für Einsteiger*. Von <https://angular.de/artikel/angular-tutorial-deutsch/> abgerufen
- ²¹ Peterke, L. (2019). *Vue.js kurz & gut*. Heidelberg: Dpunkt Verlag.
- ²² Tillmann, J. (1. August 2018). *T3N: Was ist eigentlich Vue.js?* Von <https://t3n.de/news/vuejs-1097879/> abgerufen
- ²³ Lottana, N. (3. Dezember 2020). *LogRocket: How the virtual DOM works in Vue.js*. Von <https://blog.logrocket.com/how-the-virtual-dom-works-in-vue-js/> abgerufen

- ²⁴ Development Support. (kein Datum). *Development Support: Frontend vs. Backend in der Web- und Softwareentwicklung*. Von <https://dev-supp.de/programmierung/frontend-vs-backend> abgerufen
- ²⁵ Thattil, S. (2. Mai 2019). *Yuhiro Blog: PHP vs Java: die ultimative Gegenüberstellung*. Von <https://www.yuhiro.de/php-vs-java/> abgerufen
- ²⁶ Augusten, S. (13. November 2020). *Dev Insider: Definition „Java Virtual Machine“ - Was ist eine JVM?* Von <https://www.dev-insider.de/was-ist-eine-jvm-a-972537/> abgerufen
- ²⁷ IONOS. (25. Januar 2022). *IONOS Digital Guide: Internet-Programmiersprachen: Die besten Sprachen fürs Web*. Von <https://www.ionos.de/digitalguide/websites/web-entwicklung/internet-programmiersprachen/> abgerufen
- ²⁸ Springer, S. (2016). *Node.js: Das umfassende Handbuch. Serverseitige Webapplikationen mit JavaScript entwickeln*. Bonn: Rheinwerk.
- ²⁹ Matzer, M., & Litzl, N. (15. Januar 2018). *BigData Insider: NoSQL-Datenbanken im Vergleich*. Von <https://www.bigdata-insider.de/nosql-datenbanken-im-vergleich-a-666922/> abgerufen
- ³⁰ Subramanian, V. (2019). *Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node*. Bangalore: Apress.
- ³¹ Rogojan, B. (21. Oktober 2019). *ButterCMS: JAMStack vs MEAN vs LAMP: Your Guide to Picking One*. Von <https://buttercms.com/blog/jamstack-vs-mean-vs-lamp-your-guide-to-picking-one/> abgerufen
- ³² Holmes, S., & Harber, C. (2019). *Getting MEAN with Mongo, Express, Angular, and Node*. New York: Manning Publications.
- ³³ Zammetti, F. (2020). *Practical JAMstack: Blazing Fast, Simple, and Secure Web Development, the Modern Way*. Pottstown: Apress.
- ³⁴ Biilmann, M., & Hawksworth, P. (2019). *Modern Web Development on the JAMstack*. Sebastapol: O'Reilly.

- ³⁵ Prerender.io. (3. Februar 2022). *Prerender.io: Der unverzichtbare Leitfaden für SEO für Webentwickler*. Von <https://prerender.io/de/seo-for-web-developers/> abgerufen
- ³⁶ Lammenett, E. (2021). *Praxiswissen Online Marketing: Affiliate-, Influencer-, Content-, SocialMedia-, Amazon-, Voice-, B2B-, Sprachassistenten- und E-Mail-Marketing*, Google. Wiesbaden: Springer.
- ³⁷ Kessler, J. (1. Juni 2021). *SEONative Blog: Core Web Vitals von Google: Das gibt es ab Juni 2021 (nicht nur für SEO) zu beachten*. Von <https://www.seonative.de/core-web-vitals/> abgerufen
- ³⁸ Thakkar, M. (2020). *Building React Apps with Server-Side Rendering: Use React, Redux, and Next to Build Full Server-Side Rendering Applications*. Vadodara: Apress.
- ³⁹ Riva, M. (2022). *Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js, the React framework for production*. Birmingham: Packt Publishing.
- ⁴⁰ Moore, R. (24. Juli 2021). *Nerd for Tech: Compare and Contrast: CSR, SSR, and SSG in NextJS*. Von <https://medium.com/nerd-for-tech/compare-and-contrast-csr-ssr-and-ssg-in-nextjs-58e3caf2e15e> abgerufen
- ⁴¹ Farmer, K. (15. April 2021). *Nerd for Tech: Server-Side Rendering vs Client-Side Rendering*. Von <https://medium.com/nerd-for-tech/server-side-rendering-vs-client-side-rendering-919165ba4c4b> abgerufen
- ⁴² Robinson, L. (21. April 2021). *Smashing Magazine: A Complete Guide To Incremental Static Regeneration (ISR) With Next.js*. Von <https://www.smashingmagazine.com/2021/04/incremental-static-regeneration-nextjs/> abgerufen
- ⁴³ Frana, M. (19. Februar 2020). *Dev.to: Server Side Rendering vs Static Site Generation vs Incremental Static Regeneration*. Von <https://dev.to/matfrana/server-side-rendering-vs-static-site-generation-17nf> abgerufen
- ⁴⁴ Clarence, T. (12. November 2021). *How to choose between Next.js CSR, SSR, SSG, and ISR*. Von <https://theodorusclarence.com/blog/nextjs-fetch-method> abgerufen

- ⁴⁵ Stoll, K. (18. September 2021). *T3N: DX, SEO, E-Commerce – darum ist Next.js so beliebt*. Von <https://t3n.de/news/dx-seo-e-commerce-nextjs-1406707/> abgerufen
- ⁴⁶ Attardi, J. (2020). *Using Gatsby and Netlify CMS: Build Blazing Fast JAMstack Apps Using Gatsby and Netlify CMS*. Billerica: Apress.
- ⁴⁷ Tappert, I. (August 2020). *Nuxt.js: Vue.js mit Extras*. Entwickler Magazin.
- ⁴⁸ Li, B. (29. Oktober 2021). *Kinsta: Wie man mit Hugo eine rasend schnelle statische Seite erstellt*. Von <https://kinsta.com/de/blog/hugo-statische-seite/> abgerufen
- ⁴⁹ Hugo. (kein Datum). *Hugo*. Von <https://gohugo.io/> abgerufen
- ⁵⁰ IONOS . (14. Dezember 2021). *IONOS Digital Guide: CMS-Vergleich 2022: Die beliebtesten Content Management Systeme*. Von <https://www.ionos.de/digitalguide/hosting/cms/cms-vergleich/> abgerufen
- ⁵¹ IONOS. (3. Februar 2022). *IONOS Digital Guide: Headless CMS vs. Traditional CMS*. Von <https://www.ionos.de/digitalguide/hosting/cms/headless-cms-vs-traditionelles-cms/> abgerufen
- ⁵² Wyllie, D. (23. Februar 2019). *T3N: Von klassisch bis headless: Content-Management-Systeme im Überblick*. Von <https://t3n.de/magazin/klassisch-headless-content-management-systeme-ueberblick-247074/> abgerufen
- ⁵³ T3N. (23. September 2021). *T3N: Wieso die Zukunft des Content-Management headless ist*. Von <https://t3n.de/news/zukunft-content-management-1404817/> abgerufen
- ⁵⁴ Upsilon IT. (15. Juli 2021). *Upsilon IT: Top 10 Headless CMS. Market-leading Headless CMS Comparison*. Von <https://upsilon-it.medium.com/top-10-headless-cms-market-leading-headless-cms-comparison-37230a9e86bd> abgerufen
- ⁵⁵ Nagels, P. (18. Juli 2019). *Headless CMS vs. Traditionelles CMS: Was passt besser zu meinem Projekt?* Von <https://codecontrol.io/de/blog/headless-cms-vs-traditionelles-cms-was-passt-besser-zu-meinem-projekt> abgerufen

- ⁵⁶ Luksza, R. (kein Datum). *Soft Kraft: Should I use Headless CMS? Use Cases, Pros and Cons*. Von <https://www.softkraft.co/headless-cms-pros-cons> abgerufen
- ⁵⁷ Boyarko, E. (2021). *Upsilon IT: Headless or Traditional CMS? Schemes, Use Cases, Pros and Cons*. Von <https://www.upsilonit.com/blog/headless-or-traditional-cms-schemes-use-cases-pros-and-cons> abgerufen
- ⁵⁸ W3Tech. (3. Juni 2022). *Usage statistics of content management systems*. Von https://w3techs.com/technologies/overview/content_management abgerufen
- ⁵⁹ IONOS. (12. April 2022). *IONOS Digitalguide: Shopify vs. WordPress: Die Onlineshop-Lösungen im Vergleich*. Von <https://www.ionos.de/digitalguide/hosting/cms/shopify-vs-wordpress/> abgerufen
- ⁶⁰ Zenner, R. (30. November 2020). *Shopify Partners: Ein Blick hinter die Kulissen: So funktioniert die Technik von Shopify*. Von <https://www.shopify.de/partners/blog/die-technik-von-shopify> abgerufen
- ⁶¹ Pecoraro, C., & Gambino, V. (2021). *Jumpstart Jamstack Development: Build and deploy modern websites and web apps using Gatsby, Netlify, and Sanity*. Birmingham: Packt Publishing.
- ⁶² Elshafie, K., & Haider, M. (2022). *Designing Web APIs with Strapi*. Birmingham: Packt Publishing.
- ⁶³ Bluepick. (25. März 2021). *Bluepick: DatoCMS und die 3 besten Alternativen*. Von <https://www.bluepick.de/cms/datocms> abgerufen
- ⁶⁴ Höhn, T. (28. August 2021). *T3N: Headless CMS – warum wir WordPress in Zukunft ganz anders nutzen werden*. Von <https://t3n.de/news/headless-cms-wordpress-zukunft-1397884/> abgerufen
- ⁶⁵ Schürmanns, S. (16. Juni 2022). *CMSStash: Marktübersicht Hybrid CMS*. Von <https://cmsstash.de/empfehlungen/hybrid-cms> abgerufen
- ⁶⁶ Liebel, C. (2018). *Progressive Web Apps - Das Praxisbuch*. Bonn: Rheinwerk.

- ⁶⁷ Bright Solutions. (kein Datum). App entwickeln: *Native App vs Hybride App, Web App & PWA*. Von <https://www.brightsolutions.de/blog/native-vs-hybride-vs-web-app> abgerufen
- ⁶⁸ Hahn, L. (kein Datum). It-Portal24: Native App entwickeln – Überblick für Einsteiger. Von <https://www.itportal24.de/ratgeber/native-app-entwickeln> abgerufen
- ⁶⁹ Konoplenko, A. (15. Januar 2021). Progressive Web Apps vs Native Apps: *Which One Is the Best for Your Business*. Von <https://brocoders.com/blog/progressive-web-apps-vs-react-native-apps-which-one-is-the-best-for-your-business/> abgerufen
- ⁷⁰ Duò, M. (16. Mai 2022). *Kinsta Blog: Der Leitfaden für Anfänger zum responsiven Webdesign (Code-Beispiele & Layout-Beispiele)*. Von <https://kinsta.com/de/blog/responsiven-webdesign/> abgerufen
- ⁷¹ Siegrist, K. (29. Juni 2021). *Informatik Aktuell: Service Worker in Web-Anwendungen*. Von <https://www.informatik-aktuell.de/entwicklung/methoden/service-worker-in-web-anwendungen.html> abgerufen
- ⁷² Liebel, C. (16. Juni 2017). *Heise Blog: Progressive Web Apps, Teil 2: Die Macht des Service Worker*. Von <https://www.heise.de/blog/Progressive-Web-Apps-Teil-2-Die-Macht-des-Service-Workers-3740464.html> abgerufen
- ⁷³ Hodakovskis, A. (1. November 2019). *Progressive Web Apps you use every day*. Von <https://medium.com/progressive-webapps/progressive-web-apps-you-may-use-every-day-164bfa92c498> abgerufen
- ⁷⁴ Lewis, R. (kein Datum). *Britannica: IMDb*. Von <https://www.britannica.com/topic/IMDb> abgerufen
- ⁷⁵ Krinner, J. (30. August 2018). *Chip: Rotten Tomatoes: Neues Regelwerk soll vielfältigere Kritiken fördern*. Von https://www.chip.de/news/Rotten-Tomatoes-Neues-Regelwerk-soll-vielfaeltigere-Kritiken-foerdern_147600124.html abgerufen
- ⁷⁶ Gettell, O. (24. April 2012). *Letterboxd: Social website for film fans launches*. Von <https://www.latimes.com/archives/blogs/24-frames/story/2012-04-24/letterboxd-social-website-for-film-fans-launches> abgerufen

-
- ⁷⁷ Hellwig, J. (18. Mai 2021). *Kulturbanause: Einstieg in Sass – Funktionsweise und Überblick*. Von <https://kulturbanause.de/blog/einstieg-in-sass-funktionsweise-und-ueberblick> abgerufen
- ⁷⁸ Hartmann, N. (5. Juli 2015). *React Anwendungen mit Webpack entwickeln*. Von <https://reactjs.de/artikel/react-webpack> abgerufen
- ⁷⁹ Roden, G. (2012). *Node.js & Co: Skalierbare, hochperformante und echtzeitfähige Webanwendungen professionell in JavaScript entwickeln*. Heidelberg: Dpunkt Verlag.
- ⁸⁰ Luber, S. (31. Januar 2022). *Security Insider: Was ist JSON Web Token (JWT)?* Von <https://www.security-insider.de/was-ist-json-web-token-jwt-a-1094265/> abgerufen
- ⁸¹ Hartmann, N., & Zeigermann, O. (2019). *React, 2nd Edition: Grundlagen, fortgeschrittene Techniken und Praxistipps - mit Typescript und Redux*. Heidelberg: dpunkt Verlag.
- ⁸² React.js Org. (kein Datum). *React Documentation: Einführung in Hooks*. Von <https://de.reactjs.org/docs/hooks-intro.html> abgerufen
- ⁸³ Next.js. (kein Datum). *Next.js Documentation: Custom 'App'*. Von <https://nextjs.org/docs/advanced-features/custom-app> abgerufen

Abbildungsverzeichnis

Abb. 1 - Kommunikation zwischen Client und Server	5
Abb. 2 - Austausch von Daten mittels REST	7
Abb. 3 - Grundlegende Bestandteile und Technologien des Frontends	9
Abb. 4 - Aufbau eines HTML-Elements.....	10
Abb. 5 - Definition von CSS Regeln	11
Abb. 6 - Beispielhafte Darstellung der Knoten des Document Object Models.....	12
Abb. 7 - Vergleich verschiedener Frontend-Frameworks in den Stack Overflow Trends	14
Abb. 8 - Komponentenbasierter Aufbau einer React-App	16
Abb. 9 - Vergleich Framework und Library	16
Abb. 10 - Rendern der UI-Elemente mit einem Virtual DOM.....	18
Abb. 11 - Vergleich verschiedener Programmiersprachen in den Stack Overflow Trends	20
Abb. 12 - Tabellen einer relationalen Datenbank	23
Abb. 13 - Auslieferung einer Webseite im herkömmlichen Ansatz im Vergleich zum JAMStack	27
Abb. 14 - Einflussfaktoren auf die Page Experience.....	29
Abb. 15 - Google Lighthouse Report einer Webseite	30
Abb. 16 - Vorgang des Client Side Renderings vom Seitenaufruf bis zur Darstellung im Browser	33
Abb. 17 - Vorgang des Server Side Renderings vom Seitenaufruf bis zur Darstellung im Browser	35
Abb. 18 - Vorgang der Static Site Generation vom Seitenaufruf bis zur Darstellung im Browser	37

Abb. 19 - Zusammenhang zwischen Dauer des Builds und Anzahl der Seiten	38
Abb. 20 - Ablauf der Inkrementell Statischen Regenerierung	39
Abb. 21 - Datenaktualität bei SSR, SSG und ISR im Vergleich ...	41
Abb. 22 - Datenaktualität verschiedener Renderingansätze	42
Abb. 23 - SEO-Freundlichkeit von verschiedenen Renderingansätzen	42
Abb. 24 - Google Lighthouse Performance Bewertungen bei SSR, CSR und SSG im Vergleich	43
Abb. 25 - Performance verschiedener Renderingansätze	43
Abb. 26 - Build-Prozess bei der Static Site Generation	44
Abb. 27- Build-Time verschiedener Renderingansätze	45
Abb. 28 - Vergleich verschiedener Static Site Generator und SSR-Frameworks in den Stack Overflow Trends	45
Abb. 29 - Integriertes Routing in Next.js	47
Abb. 30 - Graph QL Abfrage eines Datenbestandes	48
Abb. 31 - Dynamische HTML-Templates in Hugo	50
Abb. 32- Aufbau und Funktionsweise eines traditionellen Content Management Systemes	53
Abb. 33 - Aufbau und Funktionsweise eines Headless CMS	56
Abb. 34 - Vergleich zwischen einem Headless CMS und einem traditionellen CMS	59
Abb. 35 - Marktanteile verschiedener Content Management Systeme nach einer Erhebung von W3Tech	61
Abb. 36 - Einsatz von verschiedenen Content Management Systemen nach einer Erhebung von BuiltWith	64
Abb. 37 - Vergleich verschiedener Headless CMS	68

Abb. 38 - Googles Play Store bietet die Möglichkeit native Apps zu installieren	71
Abb. 39 - Responsives Layout einer Webseite.....	74
Abb. 40 - Funktionsweise des Service Workers einer PWA.....	78
Abb. 41 - Technologiestack der Anwendung "MovieRec"	89
Abb. 42 - Bündeln der einzelnen Module und erfassen der Abhängigkeiten.....	91
Abb. 43 - Logo und Icon-Konzepte	94
Abb. 44 - Farbpalette der Webanwendung	94
Abb. 45 - Gradients der Filmüberschriften.....	95
Abb. 46 - Typographie der Webanwendung	95
Abb. 47 - Konzepte und Formen der UI-Komponenten.....	96
Abb. 48 - Datenmodell der Anwendung als ER-Schema.....	97
Abb. 49 - Sequenzdiagramm zur Nutzerinteraktion "Registrierung"	98
Abb. 50 - Sequenzdiagramm zur Nutzerinteraktion "Anmelden in der App"	99
Abb. 51 - Sequenzdiagramm zur Nutzerinteraktion "Bewertung hinzufügen"	100
Abb. 52 - Sequenzdiagramm zur Nutzerinteraktion "Zu Liste hinzufügen"	101
Abb. 53 - Einrichten eines Administrationsaccounts bei dem CMS Strapi	103
Abb. 54 - Erstellen eines neuen Content-Types in Strapi	104
Abb. 55 - Datentypen der Attribute in Strapi	104
Abb. 56 - Felder des Collection-Types "Movie"	105

Abb. 57 - Relation zweier Content-Types "Movie" und "Regisseur"	105
Abb. 58 - Erstellen eines neuen Eintrags vom Content-Type "Movie"	106
Abb. 59- Freigabe der API-Endpunkte für den Content-Type "Movie"	107
Abb. 60 - Daten eines Eintrags der "Movie"-Collection im JSON-Format	108
Abb. 61 - Beispiele verschiedener API-Parameter in Strapi.....	109
Abb. 62 - Verzeichnisstruktur der Frontend-Applikation	112
Abb. 63 - Screenshot einer statisch generierten Einzelseite, abrufbar über eine dynamische Route	119
Abb. 64 - Ausschnitt des Buttons auf der Filmseite, um den Film zu einer Liste hinzuzufügen und zu bewerten	122
Abb. 65- Ausschnitt einer Review Card mit den entsprechenden Informationen aus dem Backend	125
Abb. 66 - Möglichkeit "MovieRec" aus dem Chrome Browser als PWA zu installieren	126
Abb. 67 - Screenshots der fertigen Anwendung	127

Codesnippets

Snippet 1 - Darstellung von Daten im JSON-Format	8
Snippet 2 - Syntax des CSS-Präprozessors SASS mit einer Variable.....	11
Snippet 3 - Hinzufügen eines Event Listeners mit einer Funktion in JavaScript	13
Snippet 4 - React-Komponente als JavaScript-Funktion.....	15
Snippet 5 - Responsives Layout mit CSS Media-Queries	75
Snippet 6 - Beispielhaftes Web App Manifest.....	79
Snippet 7 - Einsatz von useState-Hooks in React	111
Snippet 8 - Übergeordnete _app.js-Datei mit einem Header, MobileNav und HelpPanel als globale Komponenten	113
Snippet 9 - Serverseitiger Abruf der Daten.....	114
Snippet 10 - Next.js-Seite mit MovieList-Komponente, die das Objekt „movie“ über geben bekommt.....	115
Snippet 11 - JSON-Daten werden in der React-Komponente verarbeitet und letztendlich auf dem Screen für den User gerendert	115
Snippet 12 - UseStates der Input Felder zum Filtern und Definition von React Query.....	116
Snippet 13 - Aufruf der Funktion “getMovies” und beispielhaftes Filtern nach einem Genre.....	117
Snippet 14 - Abruf der Informationen eines einzelnen Films mit Hilfe des Parameters “slug”	117
Snippet 15 - Erstellen der Liste an Pfaden der jeweiligen Filme für das dynamische Routing.....	118
Snippet 16 - UseStates, die Eingaben der Nutzer entgegennehmen.....	119

Snippet 17 - Beispielhaftes Controlled Input in React für die Eingabe eines Usernames	119
Snippet 18 - Funktion "handleRegister" sendet Eingaben über einen POST-Request an das Backend und registriert den Nutzer durch einen Cookie	120
Snippet 19 - Funktion "addToWatchlist" um Filme im Backend zur eigenen Watchlist hinzuzufügen	121
Snippet 20 - Löschen eines Films von der Watchlist über die HTTP-Delete-Methode	122
Snippet 21 - Clientseitiger Abruf der Ratinginformationen und errechnen eines Durchschnittwertes	123
Snippet 22 - Serverseitiger Abruf der Reviews zur Bereitstellung auf der Seite "Feed"	124
Snippet 23 - Durchlaufen aller Reviews und übergeben eines einzelnen Objekts an die Komponente der "ReviewCard"	125
Snippet 24 - Hinzufügen von Meta-Daten zu einer Seite über "Next-SEO"	126
Snippet 25 - Erweitern der next.config.js-Datei für PWA-Funktionen.....	126

Abkürzungsverzeichnis

API	Application Programming Interface
CMS	Content Management System
CSR	Client Side Rendering
CSS	Cascading Style Sheets
DB	Datenbank
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IoT	Internet of Things
ISR	Inkrementell Statische Regenerierung
JSON	JavaScript Object Notation
JSX	JavaScript XML
NPM	Node Package Manager
PWA	Progressive Web App
REST	Representational State Transfer
SASS	Syntactically Awesome Style Sheets
SEO	Search Engine Optimization
SQL	Structured Query Language
SSR	Static Side Generation
SSR	Server Side Rendering
UI	User Interface

Eidesstattliche Erklärung

Hiermit versichere ich eidesstattlich, dass die vorliegende Bachelor-Thesis von mir selbstständig und ohne unerlaubte fremde Hilfe angefertigt worden ist, insbesondere, dass ich alle Stellen, die wörtlich oder annähernd wörtlich oder dem Gedanken nach aus Veröffentlichungen, unveröffentlichten Unterlagen und Gesprächen entnommen worden sind, als solche an den entsprechenden Stellen innerhalb der Arbeit durch Zitate kenntlich gemacht habe, wobei in den Zitaten jeweils der Umfang der entnommenen Originalzitate kenntlich gemacht wurde. Ich bin mir bewusst, dass eine falsche Versicherung rechtliche Folgen haben wird.

Ohlsbach, 27.08.22

Ort, Datum

Tim Peters

Name, Unterschrift

