FOM Hochschule für Oekonomie & Management

university location Bonn

Term paper

in the study course Business Informatics

as part of the course
IT-Infrastructure

on the subject

Performance comparison between WASM and JavaScript based on code implementing computationally intensive operations on the clientside

by

Mads Fuchs, Nils Rüber, Janis Wiesen

Advisor: Christian Frank
Matriculation Numbers: 675314, 674514, 670300
Submission: 29.02.2024

Table of Contents

List of Figures

List of Tables

1 Grundlagen

1.1 Anforderungen an moderne Webanwendungen

Die Entwicklung moderner Webanwendungen stellt Entwickler vor verschiedene Herausforderungen. Dabei geht es nicht nur um die reine Funktionalität, wie zum Beispiel das Umsetzen eines funktionierenden Webshops, der Webseite an sich. Bei der Entwicklung einer Webseite geht es darüber hinaus auch um die Performance, Benutzerfreundlichkeit, Flexibilität und Skalierbarkeit. Dabei können die unterschiedlichen Aspekte, je nach Anforderung, unterschiedlich stark gewichtet werden, wobei die reine Performance und die Benutzerfreundlichkeit der Webseite für den Nutzer meistens von hoher Bedeutung sind. Die Performance spielt eine große Rolle, da sie den Erfolg der Webseite beeinflussen kann. Nutzer erwarten kurze Ladezeiten und einen einwandfreien Ablauf der Funktionalität[1]. Die Benutzerfreundlichkeit einer Webseite ist deshalb so wichtig, da sie überhaupt erst die Akzeptanz zur Nutzung einer Webseite beeinflusst. Eine Webseite muss grundsätzlich gut lesbar, navigierbar und anpassbar für viele Verschiedene Geräte und Browser sein[2]. Auch die Flexibilität einer Webseite spielt eine übergeordnete Rolle. Sowohl ständig wechselnde Geschäftsanforderungen als auch fortschreitende Technologien setzen voraus, dass Webseiten in der Lage sein müssen auf diese Veränderungen beispielsweise mithilfe von modularen Technologien wie React reagieren zu können[3]. Zuletzt ist die Skalierbarkeit einer Webseite insofern wichtig, dass durch unterschiedliche Nutzerzahlen, der Ressourcenbedarf einer Webseite stark schwanken und dynamisch anpassbar sein sollte[4]. Um die genannten Anforderungen möglichst gut umzusetzen, ist es essenziell wichtig, Frontend und Backend grundsätzlich voneinander zu trennen. Die modulare Vorgehensweise wird hierdurch noch einmal bestätigt und ermöglicht eine flexible Vorgehensweise in der Entwicklung. Die Trennung von Frontend und Backend ist außerdem hilfreich, damit die Entwicklung der beiden Komponenten parallel und unabhängig voneinander erfolgen kann[5]. Dadurch kann bei der Entwicklung Zeit gespart werden und es können schneller Funktionalitäten der Webseite fertiggestellt und getestet oder veröffentlicht werden.

[1] *Deinhard*, 2024.
[2] *Autor, K.*, 2024j.
[3] *Autor, K.*, 2024o.
[4] *anna.cherniavska*, 2023.
[5] *Autor, K.*, 2024n.

1.2 Überblick der eingesetzten Technologien

1.2.1 React

React ist das von uns bevorzugte Frontend. Im Folgenden werden wir erklären, was React ist und wie es grundsätzlich aufgebaut ist. React wurde von Facebook entwickelt und ist eine leistungsstarke Open-Source-Bibliothek, basierend auf JavaScript. Dabei ist React auf die Erstellung von Benutzeroberflächen, also auf das sogenannte Frontend, spezialisiert[6]. Darüber hinaus ist React eine komponentenbasierte Bibliothek, welche einen effizienten und strukturierten Ansatz zur Gestaltung von interaktiven Benutzeroberflächen bietet[7]. Aus diesen Gründen eignet sich die Bibliothek hervorragend für komplexe Benutzeroberflächen, welche eben aus diesen kleinen und modularen Komponenten bestehen. Für die Benutzer bedeutet der Einsatz von React anwenderfreundliche Oberflächen und für die Entwickler eine verbesserte Code-Organisation, was unter anderem zu einer erleichterten Wartung führt[8]. Zudem verwendet React ein virtuelles Document Object Model (DOM). Das virtuelle DOM ist ähnlich zum tatsächlichen DOM. Es ist allerdings wesentlich leichtgewichtiger und wird als JavaScript-Objekt dargestellt. Auf Grund der Darstellung als JavaScript-Objekt werden lediglich die wirklich notwendigen Änderungen am tatsächlichen DOM durchgeführt, wodurch wiederum die Performance optimiert wird[9]. React Komponenten verwenden grundsätzlich zwei verschiedene Konzepte zur Datenverwaltung. Diese beiden Konzepte nennen sich States und Props und arbeiten im Zusammenspiel, um dynamische und interaktive Benutzeroberflächen zu erstellen[10]. States repräsentieren den internen Zustand einer Komponente und können sich im Laufe der Zeit verändern. Sie werden von der React Komponente selber verwaltet und können sowohl durch Benutzeraktionen als auch durch interne Ereignisse verändert werden[11]. Dies ermöglicht überhaupt erst die interaktive Nutzung der Webseite im laufenden Betrieb. Props dagegen sind als Kurzform für Properties, zu Deutsch Eigenschaften, zu verstehen und sind unveränderlich. Sie dienen dazu, bestimmte Funktionen und Daten von einer sogenannten Elternkomponente an andere Komponenten zu übertragen[12].

[6] *Acharya*, *D. P.*, 2023.
[7] *P*, *V.*, 2024.
[8] *Autor*, *K.*, 2024c.
[9] *Autor*, *K.*, 2023.
[10] *Autor*, *K.*, 2024i.
[11] *Autor*, *K.*, 2024k.
[12] *Autor*, *K.*, 2024l.

1.2.2 Strapi

Als nächstes gehen wir auf unsere verwendete Backendstruktur ein. Die von uns gewählte Backend Technologie, nennt sich Strapi. Strapi ist ein Headless Content Management System (CMS), das Open-Source betrieben wird. Strapi zeichnet sich durch seine Flexibilität und durch seine Entwicklerfreundlichkeit aus. Die Idee hinter Strapi ist die Trennung der Inhaltsverwaltung vom Frontend[13]. Es ermöglicht eine effiziente Erstellung, Verwaltung und Bereitstellung der Inhalte über eine integrierte und benutzerfreundliche Oberfläche. Der Headless-CMS-Ansatz bedeutet, dass alle Inhalte von der Präsentationsschicht getrennt sind. Alle Daten die in Strapi abgelegt und verwaltet werden, werden per API an das Frontend bereitgestellt[14]. Durch die Bereitstellung der Daten per API, ist Strapi in der Wahl des Frontends flexibel, da APIs grundsätzlich einem standardisierten Format folgen und jeweils leicht angepasst werden können. Im Folgenden werde ich etwas konkreter auf die einzelnen Vorteile von Strapi eingehen. Ein wichtiger Aspekt von Strapi ist die vorhandene Flexibilität. Strapi ermöglicht es, Inhalte und Strukturen präzise an die eigenen Projektanforderungen anzupassen. Dabei erlaubt es die No-Code-Konfiguration, das Backend nahezu ohne eigene Programmierung einzurichten. Durch die mitgelieferte, grafische Benutzeroberfläche, können sowohl Datenfelder und Inhaltstypen als auch Relationen zwischen den Daten bequem erstellt werden[15]. In diesem Zusammenhang ist die einfache Integration von Strapi in das Gesamtprojekt zu erwähnen. Es können unterschiedlichste Datenbanksysteme mit Strapi genutzt werden. Darunter SQL-basierte, aber auch NoSQL-basierte. Die Daten wiederum können ganz einfach per API an das unabhängige Frontend übermittelt werden[16]. Ein weiterer äußerst wichtiger Aspekt, der von Strapi abgedeckt wird, ist die IT-Sicherheit. Strapi verfügt bereits von Haus aus, über grundlegende Sicherheitsfunktionen wie Authentifizierung und Authorisierung. Zudem liefert Strapi die Möglichkeit zur Konfiguration von Rollen- und Berechtigungskonzepten direkt mit[17].

1.2.3 JavaScript, HTML und CSS

Neben dem eigentlichen Front- und Backend, gibt es weitere Programmiersprachen, die bei der Entwicklung einer Webseite eine Rolle spielen. In diesem Kapitel gehen wir genauer auf diese Sprachen und deren Funktionen ein.

---

[13] *Autor*, *K.*, 2024a.
[14] *Autor*, *K.*, 2024e.
[15] Ebd.
[16] *Autor*, *K.*, 2024m.
[17] *Autor*, *K.*, 2024b.

JavaScript ist eine diversifizierte und leistungsfähige Programmiersprache, die in vielen Webentwicklungen eine Rolle spielt. JavaScript wird dabei clientseitig, also lokal auf dem jeweiligen Endgerät des Nutzers, ausgeführt. Dies ermöglicht unter Anderem den Einsatz von clientseitiger Validierung, wodurch Daten bereits lokal überprüft und somit die Serverlast reduziert werden kann[18]. Der Einsatz von JavaScript ermöglicht außerdem die Implementierung von Logik und Interaktivität, wodurch die dynamischen Benutzeroberflächen im Frontend erst geschaffen und funktional gestaltet werden können. Konkret wird mit JavaScript die Manipulation des DOM ermöglicht[19]. Darüber hinaus können mithilfe von JavaScript Benutzerinteraktionen wie Klicks, Tastatureingaben oder Mausbewegungen verarbeitet werden. Durch den gleichzeitigen Einsatz von AJAX-Technologien können Daten asynchron mit dem Server ausgetauscht werden, wodurch die nahtlose Aktualisierung von Webseiteninhalten und eine echte und dynamische Interaktivität sichergestellt wird[20]. Die grundsätzliche Konzeption von JavaScript sieht vor, dass wiederverwendbare Funktionen als Kernbaustein genutzt und bei Bedarf aufgerufen werden. JavaScript unterstützt verschiedene, gängige Datentypen wie Zahlen, Zeichenketten und Objekte. Dadurch ist es auch möglich, einen objektorientierten Ansatz zu verfolgen.

Auch HTML spielt bei der Entwicklung von modernen Webseiten eine Rolle. Über HTML wird die grundlegende Strukturierung der Benutzeroberfläche vorgenommen. Anders als bei klassischen Webseiten, wird diese Strukturierung allerdings in JavaScript XML (JSX), eine Syntaxerweiterung für JavaScript, vorgenommen. JSX ermöglicht es Entwicklern, HTML-artige Elemente direkt in JavaScript umzusetzen, ohne explizite HTML-Syntax zu nutzen[21].

Bei Cascading Style Sheets (CSS) handelt es sich um eine deklarative Sprache, die es ermöglicht Webseiten und deren HTML-Elemente visuell zu gestalten. Dabei gibt es verschiedene Ansätze und Möglichkeiten, wie CSS implementiert und verwaltet wird. Zum einen gibt es die klassische CSS-Variante, bei der eine separate .css-Datei erstellt wird. In dieser Datei werden alle Styles für die Webseite definiert, was bei größeren Projekten allerdings zu Problemen bei der Wartbarkeit führen kann[22]. Eine weitere Möglichkeit bieten CSS Module, die beliebig oft wiederverwendet werden können. Die dritte Möglichkeit bieten CSS Frameworks. In diesem Fall handelt es sich um vorgefertigte Komponenten und Stile, die ebenfalls wiederverwendet werden können. Auch hier gibt es wieder verschiedene Frameworks wie zum Beispiel Bootstrap, Tailwind CSS oder Foundation.

[18] *Autor*, *K.*, 2024g.
[19] *Autor*, *K.*, 2024d.
[20] *Autor*, *K.*, 2024f.
[21] *Autor*, *K.*, 2024h.
[22] *Autor*, *K.*, 2019.

1.3 Zusammenspiel der Technologien

Für eine funktionsfähige und moderne Webseite ist das Zusammenspiel der zuvor genannten Technologien unerlässlich. Für unser Projekt bilden die klare Trennung von Backend und Frontend sowie die Integration von HTML, CSS und JavaScript, das grundlegende Fundament.

Auf Grund der genannten klaren Trennung von Frontend und Backend, können die Arbeiten an den beiden Teilprojekten unabhängig voneinander und parallel stattfinden. Außerdem bewirkt die Trennung eine gewisse Flexibilität, da Änderungen in einem der beiden Bereiche, nur geringe Auswirkungen auf den anderen Bereich haben. Darüber hinaus können beide Bereiche ebenfalls unabhängig voneinander erweitert oder skaliert werden. Die Kommunikation zwischen den beiden Bereichen erfolgt über RESTFUL APIs. Diese APIs werden von Strapi im Backend automatisch generiert und von React im Frontend genutzt. Durch diese Vorgehensweise wird eine effiziente Datenverwaltung ermöglicht. Strapi verwaltet die Inhalte und React präsentiert sie. Die in Strapi gespeicherten Inhalte können dabei mehrfach von React im Frontend wiederverwendet werden. HTML per JSX, CSS und JavaScript müssen im Gesamtkontext eingegliedert werden. HTML wird in React per JSX umgesetzt, um die Arbeit für Entwickler zu vereinfachen und um die grundlegende Webseitenstruktur zu bauen. CSS setzt an dem Punkt an, an dem das Design angepasst und verfeinert wird, um eine anschauliche Benutzeroberfläche zu bauen. JavaScript ist das Bindeglied zwischen den verschiedenen Technologien. Insbesondere in Form von React, wird per JavaScript der Datenfluss und die dynamische Aktualisierung der Benutzeroberfläche logisch umgesetzt und verwaltet.

1.4 Vorteile der gewählten Architektur

Im Folgenden werde ich nochmal kurz und knapp die Vorteile unserer gewählten Struktur und Technologien hervorheben.

Ein Vorteil von der von uns gewählten Struktur, ist die Modularität und Wiederverwendbarkeit. Diese wird vor allem durch die Trennung von Front- und Backend erreicht. Hinzu kommt die Verwendung einzelner Komponenten innerhalb von React, wodurch Wartung und Erweiterung der Webseite verbessert werden[23].

Darüber hinaus ist, ebenfalls durch die Trennung von Front- und Backend, aber auch durch die Nutzung von Strapi als headless CMS, eine hohe Flexibilität bei der Entwicklung gege-

[23] *Acharya, D. P.*, 2023.

ben. Dadurch kann jederzeit auf äußere Einflüsse wie neue Technologien oder Kundenanforderungen eingegangen werden[24].

Die Verwendung von APIs für den Datenfluss zwischen Front- und Backend, bietet den Vorteil, dass dieser effizient und standardisiert ablaufen kann. Strapi stellt die standardisierten APIs bereit und React greift darüber auf die benötigten und wiederverwendbaren Daten zu[25].

Strapi bietet zudem einen großen Vorteil in Sachen Sicherheit. Die integrierten Sicherheitsfunktionen in Strapi schützen das Backend und im Frontend können unabhängig davon weitere Sicherheitsmaßnahmen gezielt getroffen werden.

[24] *Autor*, *K.*, 2024n.
[25] *Autor*, *K.*, 2024e.

2 Hauptteil

Einleitende Worte zum Hauptteil.

2.1 Architektur

Server Frontend API Backend Shopify

2.2 Strapi

Strapi bietet von Haus aus eine gute Grundstruktur. Alle notwendigen Dateien werden von Strapi grundlegend mitgeliefert und müssen im Anschluss nur noch an die individuellen Anforderungen angepasst werden. In diesem Kapitel werden wir auf die von uns veränderten und auf unsere Bedürfnisse zugeschnittenen Dateien eingehen und deren Funktionalitäten erläutern.

2.2.1 Vite.config.js

Diese Datei ist abgelegt unter: „rfinnotrade/strapi/src/admin/vite.config.js“. Um zu erläutern, was in dieser Datei passiert, muss das grundlegende Prinzip von Cross-Origin Ressource Sharing (CORS) verstanden werden. CORS ist ein Sicherheitsmechanismus, welcher in Webbrowsern verwendet werden kann, um Zugriff auf Ressourcen von einer anderen Domain kontrolliert zulassen zu können. Daher auch der Begriff „Cross-Origin“. Es handelt sich hierbei um eine Erweiterung der standardmäßig verwendeten Same-Origin-Policy (SOP), welche Zugriffe nur von der selben Domain zulässt (Vgl. Cross-Origin Resource Sharing – Wikipedia, o. J.). In unserem Fall erlaubt diese Struktur also den Fremdzugriff auf Ressourcen innerhalb von Strapi. Diese Datei wird ebenfalls grundsätzlich von Strapi mitgeliefert und wurde von uns leicht angepasst. HIER CODE EINFÜGEN In Zeile 13 und 14 erlauben wir jeglichen Domain Zugriff auf Strapi und geben zudem die Info mit, dass alle credentials weitergeleitet werden sollen. Hier könnte man auch weitere Restriktionen vornehmen, was von uns allerdings in der Datei „Middleware.js“ vorgenommen wurde.

2.2.2 Middlewares.js

Diese Datei liegt unter „rfinnotrade/strapi/config/middleware.js“. Der Begriff Middleware in unserem Webprojekt ist als eine Softwareschicht zwischen unserer Webanwendung und dem Server zu verstehen. Die Middleware verarbeitet dabei eingehende Anfragen und ausgehende Antworten, um zum Beispiel Login- und Authentifizierungsfunktionen zu ermöglichen. HIER CODE EINFÜGEN Die hier gezeigte Middleware.js, arbeitet die aufgeführten Middleware-Funktionen nacheinander und in genau dieser Reihenfolge ab. Somit wird sichergestellt, dass aufeinander aufbauende Funktionen auch in der richtigen Reihenfolge ausgeführt werden. Es ist außerdem noch wichtig zu erwähnen, dass dies Strapi Middleware-Funktionen sind, da wir uns nach wie vor innerhalb unserer Strapi Konfiguration befinden. Neben der Anmeldung unter unserer rf-innotrade.de Admin Seite oder für andere API-Calls, welche nur von einem angemeldeten Admin durchgeführt werden dürfen, ist diese Datei für die im vorherigen Kapitel erwähnte CORS-Konfiguration zuständig. HIER CODE EINFÜGEN, ZEILE 6-16 Diese Konfiguration sorgt dafür, dass Authentifizierungsanfragen immer nur über die in Zeile 9 genannten Domains erlaubt sind und die Anfrage nur dann weiterverarbeitet werden darf.

2.2.3 Token.js

Die nächste Datei ist unter dem Pfad „rfinnotrade/strapi/src/middlewares/Token.js“ abgelegt. Diese Datei ist ebenfalls sehr wichtig, da Strapi von Haus aus keine Cookies verarbeiten kann. Die Verarbeitung von Cookies ist in unserem Fall notwendig, damit Informationen wie der aktuelle Login-Status oder welcher User, mit welchen Berechtigungen, startet gerade eine Anfrage an Strapi, abgefragt werden können. HIER CODE EINFÜGEN Realisiert wird diese Funktion im ersten Schritt durch das Auslesen des Cookies aus dem mitgelieferten Gesamtkontext der Webseite in Zeile 2 und 3. Im nächsten Schritt wird in den Zeilen 6 bis 8 der JSON Web Token (JWT) extrahiert. Sollte die Abfrage für ein JWT erfolgreich sein, wird in Zeile 9 und 10 das Token um „Bearer“ erweitert und dem „Authorization-Header“ des Webseitenkontextes wieder hinzugefügt. Durch diese Middleware Funktion versteht Strapi, ob ein User berechtigter Weise eingeloggt ist oder Serveranfragen durchführen darf.

2.2.4 Custom.js

Die nachfolgende Datei liegt unter „rfinnotrade/strapi/src/api/custom/controllers/custom.js“ und beinhaltet verschiedene Funktionen. Im Folgenden werde ich auf die Funktionen „log-

in“, „logout“ und „checkAuthStatus“ eingehen, da diese im Kontext von Strapi und den bisher genannten Funktionen am relevantesten sind.

Als erstes gehen wir auf die Login-Funktion ein. Diese Funktion erledigt die Aufgabe, den eigentlichen Login bei einer Login-Anfrage von der Admin Seite zu verifizieren, Daten aus Strapi abzufragen und darüber hinaus alle notwendigen Daten in den Cookies zu setzen. HIER CODE EINFÜGEN, NUR LOGIN FUNKTION Um diesen Vorgang zu realisieren, wird erneut der gesamte Webseitenkontext in Zeile 33 übergeben und daraus alle „request“-Daten in Zeile 34 geladen. Hier stehen unter anderem die Eingaben des Benutzers drin. Von Zeile 37 bis Zeile 45, wird die Funktion „sanitizeOutput“ deklariert, welche erst später angewendet wird. Ab Zeile 46 wird versucht mithilfe der übergebenen Login-Daten, eine Anfrage an Strapis eigentliche Login API durchzuführen. Ist diese Anfrage erfolgreich, werden die entsprechenden Benutzerdaten aus der Strapi Datenbank geladen, unter anderem die Benutzerrollen abgefragt. Ist alles erledigt, wird eine Cookie mit allen notwendigen Informationen zurückgegeben und in den Webseitenkontext geladen. Somit wurde die Login Abfrage erfolgreich über eine API durchgeführt und alle notwendigen Login Informationen anschließend in den Cookies der Webseite für weiter Abfragen gespeichert. Eine automatische Logout Funktion wurde in Zeile 68 ebenfalls großzügig implementiert. Diese würde spätestens nach 7 Tagen greifen, das Cookie auslaufen und der Logout somit automatisch erfolgen.

Der ebenfalls implementierten Logout-Funktion wird zu Beginn erneut der Webseitenkontext übergeben. HIER CODE EINFÜGEN, NUR LOGOUT FUNKTION In Zeile 208 wird dann zunächst das Cookie aus dem Kontext ausgelesen und und das JWT auf „NULL“ gesetzt, wodurch der Logout theoretisch bereits erfolgt ist. Zusätzlich wird der automatische Logout, welcher in der Login Funktion auf 7 Tage gesetzt wird, auf den Wert 0 gesetzt. Dadurch löst sich das Cookie selber auf und der Logout würde ebenfalls erfolgen. Das Cookie wird am Ende wieder an den Webseitenkontext zurückgegeben.

Die Funktion checkAuthStauts hat die Aufgabe, den aktuellen Login Status bei Aufruf dieser Funktion abzufragen. Diese Funktion ergänzt die Login- und Logout-Funktion also um eine weiter wichtige Funktionalität. Die Funktion ist ähnlich aufgebaut wie Login und Logout. HIER CODE EINFÜGEN, NUR CHECKAUTHSTATUS Sie prüft dabei zunächst ab, ob überhaupt ein JWT vorhanden ist. Sollte das nicht so sein, wird die Anfrage abgelehnt, da kein aktueller Login erfolgt ist. Sollte ein JWT vorhanden sein, wird dieses erneut verifiziert und der Benutzer kann, entsprechend seinen Berechtigungen, Anfragen durchführen.

2.3 React als Frontend

Appendices

Appendix 1: Literature appendices

Together with the paper, the following literature appendices are submitted:

- Sunarto et al. - 2023 - A Systematic Review of WebAssembly VS Javascript P.pdf
- De Macedo et al. - 2022 - WebAssembly versus JavaScript Energy and Runtime.pdf
- javascript-issues-and-solutions.html
- 10-most-common-javascript-mistakes.html
- what-to-understand-callback-and-callback-hell-in-javascript.html
- introduction.html
- what-is-the-dom-explained-in-plain-english.html
- vanilla-javascript-libraries-quest-stateful-dom-rendering.html
- javascript-performance.html
- js_performance.html
- was-ist-ein-assembler-a-756636.html
- using-git-and-github-for-latex-writting.html
- a-git-workflow-for-writing-papers-in-latex-4cfb31be4b06.html
- _Einleitung.html
- ITInfraPaper.html
- WASMvsJS.html
- webassembly-solving-performance-problems.html
- experimental-design.html
- hitlist.html
- experiment.html
- Sieve_of_Eratosthenes.html

- webassembly-mehr-als-nur-ein-web-standard.html
- was-ist-webassembly-a-4243591d831cc5a12d425ede224f5e5b.html
- WebAssembly-Wasm.html
- javascript-vs-webassembly.html
- downsides-of-rust-programming-language.html
- rust-worlds-fastest-growing-programming-language.html
- 2023.html
- was-ist-ein-compiler.html
- was-ist-ein-interpreter.html
- Concepts.html
- FOM-LaTeX-Template.html
- Rust_to_wasm.html

They can also be found in the literature appendices folder of the GitHub repository.

Appendix 2: GitHub Repositories

Here are the links to the GitHub repositories that contain the code and data from the LaTeXProject used in this paper:

- ITInfraPaper
- WASMvsJS

Quellenverzeichnis

Acharya, *Durga Prasad* (2023): 15+ Beste React-Tutorials und Ressourcen für Entwickler, de, o. O., 2023-01, URL: https://kinsta.com/de/blog/besten-react-tutorials/ [Zugriff: 2024-12-26]

anna.cherniavska (2023): Erstellung hochleistungsfähiger Webanwendungen: Die Rolle von .NET in der modernen Webentwicklung, de, o. O., 2023-09, URL: https://chudovo.de/die-rolle-von-net-in-der-modernen-webentwicklung/ [Zugriff: 2024-12-26]

Autor, *Kein* (2019): Welche Rolle spielt CSS bei der Webseiten Gestaltung?, de-DE, o. O., 2019-04, URL: https://www.web4business.de/magazin/technik/welche-rolle-spielt-css-bei-der-webseiten-gestaltung/ [Zugriff: 2024-12-30]

Autor, *Kein* (2023): Mastering React.js Virtual DOM with Practical Examples, en, o. O., 2023-09, URL: https://dev.to/anii1429/mastering-reactjs-virtual-dom-with-practical-examples-4mhf [Zugriff: 2024-12-26]

Autor, *Kein* (2024a): Strapi, de, Page Version ID: 240810628, o. O., 2024-01, URL: https://de.wikipedia.org/w/index.php?title=Strapi&oldid=240810628 [Zugriff: 2024-12-29]

Autor, *Kein* (2024b): Basics Strapi: Plattform für flexible Contentverwaltung, de, o. O., URL: https://friedrich-siever.de/blog/was-ist-strapi/ [Zugriff: 2024-12-30]

Autor, *Kein* (2024c): Die komponentenbasierte Architektur von React: Eine Fallstudie | AppMaster, de, o. O., URL: https://appmaster.io/de/blog/reaktive-komponentenbasierte-architektur [Zugriff: 2024-12-26]

Autor, *Kein* (2024d): Funktionen, o. O., URL: https://lehre.idh.uni-koeln.de/lehrveranstaltungen/wisem19/basissysteme-der-informationsverarbeitung-1-bsi-2/web-technologien/javascript/bsi-javascript-fur-fortgeschrittene/ [Zugriff: 2024-12-30]

Autor, *Kein* (2024e): Headless-CMS Strapi: höchste Flexibilität und Benutzbarkeit, de, o. O., URL: https://aw-studio.de/blog/hochste-flexibilitat-und-benutzbarkeit-mit-dem-headless-cms-strapi [Zugriff: 2024-12-29]

Autor, *Kein* (2024f): JavaScript und interaktive Webinhalte – html-einfach.de, de, o. O., URL: https://html-einfach.de/javascript-und-interaktive-webinhalte/ [Zugriff: 2024-12-30]

Autor, *Kein* (2024g): Professionelles JavaScript für höchste Ansprüche | Slenderiser, o. O., URL: https://www.slenderiser.de/inspiration/wiki/frontend-entwicklung/javascript/ [Zugriff: 2024-12-30]

Autor, *Kein* (2024h): React JSX, en-US, o. O., URL: https://www.w3schools.com/react/react_jsx.asp [Zugriff: 2024-12-30]

Autor, *Kein* (2024i): React State and Props: The Two Pillars of React, en-us, o. O., URL: https://www.dhiwise.com/post/state-and-props-the-pillars-of-react-explained [Zugriff: 2024-12-26]

Autor, *Kein* (2024j): Responsive Design: Der ultimative Leitfaden zur Erstellung einer flexiblen Website, de, o. O., URL: https://www.exovia.de/wissen/website-erstellen/responsive-design/ [Zugriff: 2024-12-26]

Autor, *Kein* (2024k): State Management in React, o. O., URL: https://www.codecentric.de/wissens-hub/blog/state-management-in-react [Zugriff: 2024-12-26]

Autor, *Kein* (2024l): State und Lifecycle – React, en, o. O., URL: https://de.legacy.reactjs.org/docs/state-and-lifecycle.html [Zugriff: 2024-12-26]

Autor, *Kein* (2024m): Strapi – Portallösungen mit Headless CMS | AM GmbH, de, o. O., URL: https://am-gmbh.de/strapi-headless-cms [Zugriff: 2024-12-29]

Autor, *Kein* (2024n): Warum wir bei der Web-Entwicklung eine Trennung zwischen Frontend und Backend favorisieren? – assonos Blog, o. O., URL: https://www.assono.de/blog/warum-wir-bei-der-web-entwicklung-eine-trennung-zwischen-frontend-und-backend-favorisieren [Zugriff: 2024-12-26]

Autor, *Kein* (2024o): Webentwicklung: Die besten Technologien für 2024, de-DE, o. O., URL: https://wf-creative.com/blog/webentwicklung-die-besten-technologien-fuer-2024/ [Zugriff: 2024-12-26]

Deinhard (2024): Überblick über moderne Entwicklungs-Prinzipien und Lösungen für Webanwendungen, de-de, o. O., URL: https://www.it-schulungen.com/wir-ueber-uns/wissensblog/ueberblick-ueber-moderne-entwicklungs-prinzipien-und-loesungen-fuer-webanwendungen.html [Zugriff: 2024-12-26]

P, *Vera* (2024): Was ist React: Funktionen verstehen und wie man es für die moderne Webentwicklung einsetzt, en-US, o. O., 2024-03, URL: https://www.hostinger.de/tutorials/was-ist-react [Zugriff: 2024-12-26]

Declaration of Authenticity

We hereby declare that we produced the submitted paper with no assistance from any other party and without the use of any unauthorized aids and, in particular, that we have marked as quotations all passages which are reproduced verbatim or near-verbatim from publications. Further, we declare that this thesis has never been submitted before to any examination board in either its present form or in any other similar version. we herewith disagree that this thesis may be published. We herewith consent that this thesis may be uploaded to the server of external contractors for the purpose of submitting it to the contractors' plagiarism detection systems. Uploading this thesis for the purpose of submitting it to plagiarism detection systems is not a form of publication.