FH University of Applied Sciences
TECHNIKUM WIEN

BACHELORARBEIT

zur Erlangung des akademischen Grades
„Bachelor of Science in Engineering“
im Studiengang Informatik/Computer Science

Technologische Effizienz im Web: Einsparung von Ressourcen und Reduzierung der CO_2-Emissionen durch Optimierung

Ausgeführt von: Artjom Moisejev

Personenkennzeichen: 2110257055

BegutachterIn: Markus Holzer, MSc

Wien, den 17. Mai 2024

Eidesstattliche Erklärung

„Ich, als Autor / als Autorin und Urheber / Urheberin der vorliegenden Arbeit, bestätige mit meiner Unterschrift die Kenntnisnahme der einschlägigen urheber- und hochschulrechtlichen Bestimmungen (vgl. Urheberrechtsgesetz idgF sowie Satzungsteil Studienrechtliche Bestimmungen / Prüfungsordnung der FH Technikum Wien idgF).

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig angefertigt und Gedankengut jeglicher Art aus fremden sowie selbst verfassten Quellen zur Gänze zitiert habe. Ich bin mir bei Nachweis fehlender Eigen- und Selbstständigkeit sowie dem Nachweis eines Vorsatzes zur Erschleichung einer positiven Beurteilung dieser Arbeit der Konsequenzen bewusst, die von der Studiengangsleitung ausgesprochen werden können (vgl. Satzungsteil Studienrechtliche Bestimmungen / Prüfungsordnung der FH Technikum Wien idgF).

Weiters bestätige ich, dass ich die vorliegende Arbeit bis dato nicht veröffentlicht und weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt habe. Ich versichere, dass die abgegebene Version jener im Uploadtool entspricht.“

Wien, 17. Mai 2024

Unterschrift

Kurzfassung

Die vorliegende Bachelorarbeit untersucht den Einfluss der Webtechnologien, die auf österreichischen Webseiten eingesetzt werden, auf CO_2-Emissionen und das Potenzial zur Energieeinsparung. Durch den Einsatz eines, speziell für diese Forschung entwickelten, Webcrawlers wurden Performance-Daten gesammelt, in Google Sheets verwertet und anschließend in MATLAB analysiert. Die Analyse umfasst 3246 Webseiten und bestätigt das Bild des „verschlafenen Österreichs“. Es zeigt ein immenses Potential für Verbesserungen, sowohl in der Aktualität der verwendeten Software, als auch in Bereichen der Performance und Energieeffizienz. Die größte Schwankungsbreite lag bei WordPress-Webseiten, was das Potenzial für Extremfälle bei No-Code-Plattformen unterstreicht und die Betreiber*innen in die Verantwortung zieht, die eigene Webpräsenz zu optimieren. Aufgrund dieser Erkentnisse empfiehlt die vorliegende Arbeit eine bewusste Auswahl und Implementierung von Web-Technologien, kombiniert mit effektiven Optimierungsstrategien, um die Umweltauswirkungen im digitalen Raum zu senken. Dabei liegt eine besondere Empfehlung für die Benutzer*innen von WordPress und anderen No-Code Plattformen, zumindest auf die Mediengrößen und die Anzahl der Drittanbieter-Software zu achten, da in diesen Bereichen die größten Potentiale zur Einsparung von Energie und CO_2-Emissionen liegen.

Schlagworte: Web-Technologien, CO_2-Emissionen, Energieeffizienz, Webcrawler, Leistungsoptimierung, ökologischer Fußabdruck, nachhaltige IT, Datenanalyse, Optimierung von Webseiten

Abstract

This bachelor thesis examines the influence of web technologies used on Austrian websites on CO_2 emissions and the potential for energy savings. Using a web crawler specially developed for this research, performance data was collected, used in Google Sheets and then analyzed in MATLAB. The analysis covers 3246 websites and confirms the image of „sleepy Austria“. It shows immense potential for improvement, both in the up-to-dateness of the software used, as well as in performance and energy efficiency. WordPress websites showed the greatest variation, which underlines the potential for extreme cases with no-code platforms and makes operators responsible for optimizing their own web presence. Based on these findings, this paper recommends a conscious selection and implementation of web technologies, combined with effective optimization strategies to reduce the environmental impact in the digital space. A particular recommendation for users of WordPress and other no-code platforms is to at least pay attention to the media sizes and the number of third-party software, as these are the areas with the greatest potential for saving energy and CO_2 emissions.

Keywords: Web technologies, CO_2 emissions, energy efficiency, web crawler, performance optimization, ecological footprint, sustainable IT, data analysis, website optimization

Danksagung

Die erste und auch die größte Danksagung spreche ich an meine heiß- und viel-geliebte Mercedes-C. aus. Sie war nicht nur die größte positive Wendung meines Lebens, sondern hat mir diesen Traum vom IT-Studium überhaupt erst ermöglicht. Ihre Liebe und Fürsorge, sowie ihr Glaube an mich, mit einhergehender dauerhaften Unterstützung, sei es seelisch, emotional, physisch oder finanziell, haben mir das Durchhaltevermögen geschenkt, das Studium und diese Bachelorarbeit nicht nur durchzustehen, sondern auf höchstem Niveau auskosten und für persönlichen Wachstum zu nutzen.

Gefolgt, nach meiner besseren Hälfte, findet sich immer mein bester Freund Maciej, der oft meine einzige Auszeit und Ausgleich darstellte, zum Studium selbst, zum Schreiben dieser Arbeit und zur beruflichen Tätigkeit. Seine Loyalität und Gastfreundschaft gaben mir den stabilen Halt in dieser stürmischen Zeit und spielten somit eine grundlegende Rolle auf meinem Weg. An dieser Stelle auch ein Dank an seine Mutter Krystyna, die mir immer ein Gefühl des Willkommen-seins vermittelte und stets ein paar nette Worte für mich fand. Und natürlich auch an ihren Kater Jakus.

Ein weiterer wichtiger Grundpfeiler meines Werdegangs ist Dominik, mein Studienkollege und Gründer-Partner des IT-Unternehmens 27vier Development. Mit ihm an meiner Seite war ich stets beflügelt, das Beste aus dem Studium und dieser Bachelorarbeit zu machen, um das Wissen wieder in das gemeinsame Unternehmen zu investieren. Mit diesem Antrieb gelangen mir die Extrameilen für die hohe Qualität und den breiten Umfang dieser Arbeit.

Weiters bedanke ich mich bei allen anderen Menschen, die mich auf dem Weg unterstützt haben: Meinen Betreuer Markus Holzer, MSc, für die Begutachtung und Betreuung dieser Arbeit; Shane, für seinen Glauben an mich; meine beiden Brüder Patrick und Mark Moisejev, sowie meine Schwester Diana Moisejev, für die kleinen Auszeiten an Feiertagen und Ferien.

Den letzten besonderen Dank möchte ich an ChatGPT aussprechen. Es begleitete mich mit umfassender Beratung durch diese Arbeit und lehrte mir unter anderem LaTeX und MATLAB, die ich vor dieser Arbeit nicht kannte. Es half immens bei der Identifizierung von Mustern zur Kategorisierung der gesammelten Daten und beim Lösen komplexer Herausforderungen.

Abschließend möchte ich ein Wort der Trauer aussprechen für all die anderen interessanten Bachelorthemen †, die ich gerne bearbeitet hätte, aber letztendlich den Webcrawler gewählt habe. RIP

- Plan A: CO_2 Einsparung durch Performance Optimierung von Webseiten (Erledigt)
- Plan A_1: Betriebssysteme analysieren: staatliche Ämter, ÖBB, etc. (Linux > Windows)
- Plan B: B-Side (altes Projekt für Musiker-Plattform)
- Plan C: ChatGPT (Auswirkungen auf Lernfähigkeit - Mails schreiben, Coden verlernen)
- Plan G: Grafische Turing Maschine/Automat mit Pixel, statt Band (Ähnlich wie Game of Life von John Conway †)
- Plan Q: Quadtrees untersuchen (Travelling Salesman Problem?)
- Plan O: OpenGL - irgendwas mit Shadern machen
- Plan W: Internetwerbung (Wieso das Konzept zum Scheitern verurteilt ist)

Inhaltsverzeichnis

1 Einleitung

1.1 Hintergrund des Problems

Das World-Wide-Web und die damit einhergehende digitale Revolution haben die Art und Weise wie wir kommunizieren, arbeiten und leben, grundlegend verändert. Diese Technologien sind jedoch nicht ohne ökologische Kosten. Es wird geschätzt [1], dass die Informations- und Kommunikations-Technologie (IKT) ca. 1,8 % - 2,8 % der weltweiten Treibhausgasemissionen im Jahr 2020 ausmachten, mit Schätzungen der Emissionen der IKT, die zwischen 0,8 und 2,3 $GtCO_2$ variierten. Die höchsten Schätzungen setzen den Anteil der IKT an den weltweiten Emissionen auf etwa 6,3 % fest, wobei aktuellere Bewertungen ein Best Case Szenario von etwa 1,5 % als realistisch betrachten. Vor dem Hintergrund des Klimawandels wird die Notwendigkeit, den ökologischen Fußabdruck digitaler Technologien zu reduzieren, immer dringlicher. An anderer Stelle [2] zeigen Prognosen für 2030, dass der Trend des steigenden Energiebedarfs sich nicht nur fortsetzen wird, mit einem Wachstum von 2,8 % - 3,4 %, sondern auch nur zu 29 % im Best Case und 16 % im Worst Case, von erneurbaren Energien betrieben sein wird.

1.2 Problemstellung

Die voranschreitende Digitalisierung öffnet für User*innen und Unternehmen neue Möglichkeiten und macht es für alle einfacher, eine Webpräsenz aufzubauen. Ein besonderes Augenmerk liegt auf dem stetig wachsenden Marktanteil von WordPress, der mittlerweile auf 43 % angestiegen ist [3], aber auch auf anderen No-Code-Lösungen (wie Wix, SquareSpace, Webflow, etc.), die aufgrund ihrer Einfachheit und Zugänglichkeit immer attraktiver werden. Diese Entwicklungsplattformen ermöglichen es Nutzer*innen ohne tiefe Fachkenntnisse, Webseiten zu erstellen und zu verwalten. Jedoch kann das Fehlen von technischem Wissen in vielen Fällen zu suboptimalen Lösungen führen, die nicht nur ineffizient und energieintensiv sein können, sondern auch den Gebrauch von proprietärer Black-Box-Software machen und somit den Webseitenbesitzer*innen nur wenige Möglichkeiten bieten, die Effizienz zu steigern. Insbesondere die übliche Praxis von Unternehmen, gewinnorientierte Entscheidungen zu treffen, wie aggressive Sparmaßnahmen, sich für die günstigsten Anbieter zu entscheiden oder ein Übermaß an Diensten von Drittanbietern zu nutzen, birgt ein erhebliches Potential für Energieverschwendung bzw. Einsparungspotential [4]. Dass es einen signifikanten Unterschied in den CO_2-Emissionen zwischen verschiedenen Plattformen gibt, zeigte die Studie [5], bei der moderne Plattformen, wie YouTube, Facebook, Netflix und TikTok, hinsichtlich ihrer CO_2-Emissionen

untersucht wurden. Die Tatsache, dass moderne Hardware leistungsfähiger und verzeihender gegenüber ressourcenintensiven Prozessen ist [6], verstärkt dieses Problem weiter und trägt dazu bei, dass Webseiten oft viel mehr Ressourcen verbrauchen als notwendig, was sowohl die direkten als auch die indirekten Umweltauswirkungen erhöht. Selbst unter professionellen Entwickler*innen herrscht Unwissen bezüglich Energy Patterns [7], welches auf Energiesparpotenziale in der zugrunde liegenden Software der Plattformen selbst hindeuten kann.

1.3 Ziel der Arbeit

Angesichts der dringenden Notwendigkeit, die Auswirkungen des Klimawandels zu bekämpfen, ist es entscheidend, die Problemstellung rund um den Energieverbrauch und die damit verbundenen CO_2-Emissionen von Webseiten und Web-Apps zu analysieren. Die Herausforderung besteht darin, effektive Lösungsansätze zu identifizieren, die nicht nur die ökologische Nachhaltigkeit fördern, sondern auch die breite Palette von User*innen einschließen – von Einzelpersonen ohne technischen Hintergrund bis hin zu großen Unternehmen. Durch die Entwicklung und Anwendung von nachhaltigeren Web-Technologien und Optimierungsstrategien [8] kann ein bedeutender Beitrag zur Reduzierung des digitalen ökologischen Fußabdrucks geleistet werden. Die vorliegende Arbeit zielt darauf ab, den Einfluss von Web-Technologien und Optimierungsstrategien auf den Energieverbrauch und die CO_2-Emissionen von Webseiten zu untersuchen, um Einsparungspotenziale aufzuzeigen. Durch die Analyse verschiedener Webseiten soll herausgefunden werden, welche Technologien und Methoden sich positiv auf die Umweltbilanz auswirken und somit den digitalen Raum umweltfreundlicher gestalten und welche nicht. Dieses Wissen soll als Grundlage dienen, um Richtlinien und Best Practices für die Entwicklung nachhaltiger Web-Anwendungen zu formulieren, oder vorhandene Forschung zu stützen und somit einen Beitrag zur Bewältigung der ökologischen Herausforderungen unserer Zeit zu leisten.

1.4 Forschungsfrage und Hypothesen

Die zentrale Forschungsfrage dieser Arbeit lautet: „Inwiefern korrelieren bestimmte Web-Technologien und Optimierungen mit dem Energieverbrauch und den CO_2-Emissionen von Webseiten?“ Diese Frage zielt darauf ab, die Beziehung zwischen der technischen Umsetzung von Webseiten und ihren Umweltauswirkungen zu verstehen.

1.5 Hypothesen

- Hypothese 1: Es besteht eine signifikante Korrelation zwischen der Verwendung moderner Web-Technologien (wie Frontend-Frameworks, Bibliotheken, Content Management System (CMS), etc.) und einer Reduzierung der CO_2-Emissionen von Webseiten.
- Hypothese 2: Webseiten, die für Performance und Effizienz optimiert sind (z.B. durch die Minimierung des Datenverkehrs und effiziente Ressourcennutzung), haben im Vergleich zu weniger optimierten Webseiten einen geringeren ökologischen Fußabdruck.

Diese Hypothesen sollen durch den Einsatz eines selbst entwickelten Webcrawlers überprüft werden, der relevante Daten und Metriken von einer Vielzahl von Webseiten sammelt und analysiert. Die Ergebnisse dieser Analyse werden aufzeigen, inwieweit Webentwickler*innen und -designer*innen durch gezielte Technologie- und Optimierungswahl zur Reduzierung des CO_2-Fußabdrucks beitragen können.

2 Methodik

Die Forschungsfrage und die Hypothesen sollen durch den kombinierten Einsatz eines selbst entwickelten Webcrawlers, der die Datensammlung ausführt, Google Sheets als Speicher der Daten und MATLAB für die Analyse der gesammelten Daten, überprüft werden. Der Crawler ermöglicht die automatisierte Sammlung der Daten und die Analyse der Webseiten hinsichtlich ihrer Performance, des Technologieeinsatzes und der Optimierungsstrategien einer breiten Palette von Webseiten. Die Ergebnisse dieser Analyse werden in Google Sheets gesammelt und gesäubert. Anschließend werden die Verarbeitung und Darstellung der Daten in MATLAB durchgeführt, um aufzuzeigen, inwieweit Webentwickler*innen und -designer*innen durch gezielte Technologie- und Optimierungswahl zur Reduzierung des CO_2-Fußabdrucks beitragen können.

2.1 Forschungsdesign

Das Forschungsdesign orientiert sich an einer quantitativen Methodologie, um objektiv messbare Daten über den Zusammenhang zwischen Web-Technologien, Optimierungsstrategien und deren Auswirkungen auf den Energieverbrauch sowie die damit verbundenen CO_2-Emissionen zu erfassen. Die Entwicklung und Nutzung eines Webcrawlers als zentrales Instrument dieser Forschung ermöglicht es, Daten in großem Umfang zu sammeln und damit eine solide Datenbasis für die Analyse zu schaffen. Diese Methodik erlaubt eine objektive, effiziente und durch die Open-Source-Veröffentlichung des Quellcodes, eine reproduzierbare Datenerhebung, wodurch die Validität und Zuverlässigkeit der Forschungsergebnisse erhöht wird.

2.2 Datenerhebung

Der Kern der Datenerhebung liegt in der Programmierung und dem Einsatz des Webcrawlers. Dieser Crawler wird konfiguriert, um spezifische Daten von einer Vielzahl von Webseiten zu extrahieren, darunter:

- Performance-Metriken (z.B. Ladezeiten, Menge heruntergeladener Daten)
- Verwendete Web-Technologien und Frameworks (z.B. React, Angular)
- Implementierte Optimierungsstrategien (z.B. Minifizierung von Code[1], Bildoptimierung)

[1] Verkleinerung des Quellcodes [9]

Bei der Datenerhebung werden die Ergebnisse vollautomatisch in Google Sheets hochgeladen, um daraufhin analysiert und verwertet zu werden. Die Auswahl der zu analysierenden Webseiten wurde so getroffen, dass sie ein breites Spektrum an Anwendungstypen, Branchen und Technologieeinsätzen abdeckt. Dies stellt sicher, dass die Ergebnisse der Studie generalisierbar und auf verschiedene Kontexte anwendbar sind.

2.3 Datenanalyse

Die Datenanalyse zielt darauf ab, Muster und Korrelationen zwischen den eingesetzten Technologien, den Optimierungsstrategien und den Performance-Metriken mithilfe von MATLAB zu identifizieren. Anhand einer eigenen Score-Berechnung werden Kalkulationen gemacht, um für jede untersuchte Webseite Energieverbrauchs- und CO_2-Emissionsschätzungen auszurechnen. Dabei werden verschiedene Diagramme und Tabellen erstellt, um die Verteilungen visuell darzustellen. Um Mittelwerte und Ausreißer aufzuzeigen werden Boxplots verwendet, wobei für Korrelationen wiederum Streudiagramme zum Einsatz kommen. Die Ergebnisse dieser Analysen werden genutzt, um die aufgestellten Hypothesen zu überprüfen und ein tieferes Verständnis für die Effektivität verschiedener Web-Technologien und Optimierungsansätze in Bezug auf die Reduktion des ökologischen Fußabdrucks zu entwickeln. Weiterhin soll die Auswertung der Daten einen Einblick in den aktuellen Digitalisierungsfortschritt der österreichischen Unternehmen, mit Fokus auf Wien, geben.

3 Entwicklung und Untersuchung

3.1 Entwicklung des Webcrawlers

3.1.1 Konzeption und Zielsetzung

Die Entwicklung eines Webcrawlers für die Untersuchung des ökologischen Fußabdrucks von Webseiten und Web-Apps entsprang der Notwendigkeit, eine umfangreiche und objektive Datenbasis zu generieren. Diese Datenbasis sollte Aufschluss darüber geben, inwieweit verschiedene Web-Technologien und Optimierungsstrategien den Energieverbrauch und die CO_2-Emissionen beeinflussen können. Die Konzeption des Crawlers war daher von Anfang an darauf ausgerichtet, eine Datenerhebung zu ermöglichen, die sowohl in der Breite (Anzahl der untersuchten Webseiten) als auch in der Tiefe (Vielfalt der erfassten Daten) aussagekräftige Informationen liefert. Das primäre Ziel dieses Projekts war die Sicherstellung, dass der Webcrawler zuverlässig, effizient und in einem mittel-hohem Maße skalierbar entwickelt wird. Hierbei sollten insbesondere folgende Aspekte berücksichtigt werden:

- Umfassende Datensammlung: Der Crawler sollte in der Lage sein, eine breite Palette von Webseiten zu besuchen und dabei nur relevante Daten zu sammeln, die für die Bewertung ihres Energieverbrauchs und ihrer CO_2-Emissionen erforderlich sind. Dazu gehören technische Spezifikationen, wie der Einsatz von Frameworks und Bibliotheken, sowie Performancedaten.
- Präzise Analysefähigkeit: Neben der Sammlung von Daten war es essentiell, dass der Crawler diese Informationen in einem Format bereitstellt, das eine detaillierte Analyse ermöglicht. Dies beinhaltet die Fähigkeit, die Daten so zu strukturieren und zu speichern, dass sie leicht zugänglich, gut analysierbar und vor allem verlässlich konsistent gespeichert werden.
- Skalierbarkeit und Effizienz: Angesichts der schieren Menge an Webseiten, die für eine umfassende Analyse erforderlich sind, musste der Crawler auch skalierbar und so effizient in seiner Arbeitsweise sein, wie nur möglich. Dies bedeutet, dass er in der Lage sein sollte, eine große Anzahl von Seiten in einem vertretbaren Zeitrahmen zu verarbeiten, ohne dabei die Genauigkeit oder die Qualität der gesammelten Daten zu beeinträchtigen.

Die Konzeption des Webcrawlers wurde mit dem Bewusstsein vorgenommen, dass die Qualität und Relevanz der gesammelten Daten direkt die Glaubwürdigkeit und Aussagekraft der gesamten Forschungsarbeit beeinflussen. Daher wurde besonderer Wert auf die Auswahl der zu

sammelnden Daten, die Architektur des Crawlers und die Strategien zur Datenanalyse gelegt. Durch diesen methodischen Ansatz soll sichergestellt werden, dass die Ergebnisse der Arbeit einen signifikanten Beitrag zum Verständnis und zur Reduzierung des ökologischen Fußabdrucks digitaler Technologien leisten.

3.1.2 Technische Spezifikationen

Die technische Umsetzung des Webcrawlers basiert auf einer Kombination moderner Technologien und Tools, die zusammenarbeiten, um eine effiziente Datensammlung und -analyse von Webseiten zu ermöglichen. Im Folgenden werden die Schlüsseltechnologien und deren Einsatz im Rahmen dieses Projekts erläutert.

Node.js

Node.js [10] bildet die Grundlage des Webcrawlers und wurde aufgrund seiner Leistungsfähigkeit bei der Handhabung asynchroner Operationen und Netzwerkanfragen sowie der Vielfalt an Open-Source-Bibliotheken ausgewählt. Es ermöglicht die schnelle und einfache Entwicklung eines performanten und skalierbaren Crawlers, da es eine umfangreiche Bibliothek von Modulen bietet, die für verschiedene Aspekte des Crawlings selbst, der Datenverarbeitung und der Datensammlung genutzt werden können.

Cheerio für statische Analyse

Cheerio [11] wird eingesetzt, um die heruntergeladenen HTML-Dokumente der Webseiten zu analysieren. Es bietet eine einfache und effiziente Methode, um mit dem DOM der Seite zu interagieren, ähnlich wie jQuery. Durch den Einsatz von Cheerio kann der Crawler spezifische Daten aus dem HTML extrahieren, wie z.B. Meta-Tags, Header, bestimmte Dateinamen und andere Informationen. Diese Fähigkeit ist essenziell, um Daten über den Technologie-Stack, die Implementierung von Optimierungstechniken und die allgemeine Integration und Klassifizierung der verwendeten Web-Technologien zu erfassen.

Puppeteer und Lighthouse für Performance-Tests

Puppeteer [12] dient als Headless Chrome Browser, der es ermöglicht, Webseiten unter realen Bedingungen zu rendern und mit ihnen automatisiert zu interagieren. In Kombination mit Lighthouse [13], einem von Google entwickelten Tool für die Performance-Analyse von Webseiten und Webanwendungen, ermöglicht Puppeteer die Durchführung detaillierter Performance-Tests. Lighthouse bietet umfassende Berichte über verschiedene Aspekte der Webseiten-Performance, einschließlich der Ladezeit, der Performance-Einsparungspotenziale und des überschüssigen Datenverkehrs. Diese Daten sind entscheidend, um zu bewerten, wie verschiedene Technologien und Optimierungen den Energieverbrauch und die Effizienz von Webseiten beeinflussen.

CO2.js

CO2.js [14] ist eine Open-Source-Bibliothek der Green Web Foundation und dient zur Schätzung der CO_2-Emissionen anhand der übertragenen Datenmenge. Es ermöglicht auch den Abgleich mit einer internationalen Datenbank [15], die Hosting-Anbieter*innen auflistet, welche grünen Strom für ihre Dienstleistungen verwenden. Dies hilft bei der Einschätzung der CO_2-Emissionen und gibt einen weiteren Einblick in den aktuellen Stand der Verbreitung von grünem Hosting.

Google Sheets für Datensammlung

Google Sheets dient als zentrale Plattform für die Speicherung und erste Verarbeitung der gesammelten Daten. Die Verwendung von Google Sheets ermöglicht eine flexible und online zugängliche Methode zur Organisation der Daten, die von verschiedenen Webseiten gesammelt wurden. Außerdem bietet es eine einfache Benutzeroberfläche und die wichtigsten Werkzeuge auf einem Fleck, anders als bei der Microsoft-Alternative MS Excel, was den zeitlichen Aufwand der Forschung verzögern würde.

Google API zur Kommunikation mit Google Sheets

Um die Interaktion zwischen dem Webcrawler und Google Sheets zu ermöglichen, wird die Google Sheets API [16] verwendet. Diese API ermöglicht es dem Crawler, automatisch Daten in Google Sheets einzufügen, zu aktualisieren oder zu lesen. Durch die Programmierung spezieller Skripte kann der Crawler effizient mit der Google Sheets-Datenbank kommunizieren, was die manuellen Dateneingaben eliminiert und den gesamten Prozess der Datensammlung und -analyse weiter automatisiert.

MATLAB

MATLAB [17] wird verwendet, um genauere mathematische Analysen und daraus resultierende Ergebnisse/Daten visuell auszuarbeiten. Durch die exakte Genauigkeit und die Möglichkeit, die erhobenen Daten flexibel zu kombinieren sowie darzustellen, erhält MATLAB eine zentrale Rolle in der Auswertung und Analyse der gesammelten Daten. Weiterhin wird durch die Offenlegung des verwendeten MATLAB-Codes die Reproduzierbarkeit dieser Forschung weiter verstärkt.

3.1.3 Konfiguration und Funktionsweise des Webcrawlers

Ein wesentlicher Aspekt bei der Entwicklung des Webcrawlers ist seine Flexibilität in Bezug auf die Datenspeicherung. Standardmäßig ist der Crawler so konfiguriert, dass er die gesammelten Daten in Google Sheets einfügt, was eine einfache und effiziente Möglichkeit bietet, die Ergebnisse in Echtzeit zu überwachen und zu analysieren. Diese Konfiguration eignet sich besonders für Teams, die an der Datensammlung beteiligt sind, da der Zugriff und die Kollaboration dadurch erleichtert wird. Außerdem ermöglicht die zentrale Online-Speicherung, den Betrieb von mehreren Crawlern gleichzeitig, von verschiedenen Geräten aus. Dies war jedoch bei der Datensammlung für diese Arbeit nicht der Fall. Für Anwender*innen, die eine lokale Lösung bevorzugen oder eine Offline-Analyse der Daten durchführen möchten, bietet der Crawler eine alternative Konfiguration. Durch Setzen der Flag `--local`, bei der Ausführung des Crawlers, können die Daten stattdessen in eine lokale CSV-Datei geschrieben werden. Diese Option erhöht die Flexibilität des Crawlers erheblich, da User*innen ohne Zugang zu Google Sheets oder diejenigen, die eine strengere Kontrolle über ihre Daten wünschen, diese auch lokal sammeln und speichern können. Die lokale Datenspeicherung ist besonders nützlich für detaillierte Offline-Analysen und kann leicht in andere Datenanalysetools importiert werden.

3.1.4 User Stories

User*in

- Einfache Bedienung und Konfiguration: Als User*in möchte ich eine einfache Möglichkeit haben, den Crawler zu starten, zu stoppen und seine Einstellungen zu konfigurieren, damit ich ohne umfangreiches technisches Wissen Webseiten analysieren kann.
- Ergebnisvisualisierung: Als User*in möchte ich eine intuitive Darstellung der Crawling-Ergebnisse, direkt in Google Sheets oder lokal als CSV Datei, um die Analyseergebnisse leichter verstehen und präsentieren zu können.

Crawler

- Node.js-basiert: Als Developer des Crawlers möchte ich Node.js verwenden, um eine effiziente und skalierbare Anwendung zu erstellen, die auf einer Vielzahl von Systemen leicht zu implementieren ist.
- Verwendung von Cheerio für statische Analyse: Als Developer möchte ich Cheerio nutzen, um die HTML-Struktur von Webseiten schnell und effizient zu analysieren, insbesondere um Header-Informationen und tech-stack-typische Muster zu extrahieren.
- Puppeteer und Lighthouse für Performance-Tests: Als Developer möchte ich Puppeteer für die Automatisierung von Browseraktionen und Lighthouse für Performance-

Messungen einsetzen, um eine detaillierte Analyse der Ladegeschwindigkeit und Optimierungsmöglichkeiten von Webseiten zu ermöglichen.

- Integration der Google API zur Ergebnisübertragung: Als Developer möchte ich die Google API nutzen, um die Ergebnisse direkt in Google Sheets hochzuladen, was eine einfache Überprüfung und Weiterverarbeitung durch den*die User*in ermöglicht.
- Statische Analyse für Technologien und Schlagwörter: Als Developer möchte ich spezifische Muster in JavaScript- und CSS-Dateinamen sowie Schlüsselwörter in Webinhalten erkennen, um Rückschlüsse auf den verwendeten Tech-Stack und Inhalte zu ermöglichen.
- Performance-Tests für verschiedene Ladezeiten und Datenmengen: Als Developer möchte ich verschiedene Performance-Metriken wie Largest Contentful Paint (LCP) und Time to Interactive (TTI) messen sowie die Größe unoptimierter Medien und Ressourcen analysieren, um Optimierungspotenziale zu identifizieren.
- Zuverlässigkeit durch Mehrfachversuche: Als Developer möchte ich, dass der Crawler bis zu drei Versuche unternimmt, eine Ziel-Webseite zu erreichen, um die Zuverlässigkeit der Daten und die Vollständigkeit der Analyse zu gewährleisten.

Google Sheets

- Als User*in möchte ich, dass die Ergebnisse automatisch in Google Sheets eingetragen und aktualisiert werden, damit ich immer Zugang zu den neuesten Daten habe, ohne manuelle Uploads durchführen zu müssen.
- Ergebnisfilterung und -sortierung: Als User*in möchte ich in Google Sheets Filter und Sortierfunktionen nutzen können, um die Ergebnisse nach verschiedenen Kriterien wie Performance-Metriken, verwendeten Technologien oder Optimierungspotenzialen zu organisieren und zu analysieren.

3.1.5 Herausforderungen und Lösungsansätze

Ursprünglich war geplant, dass der Webcrawler auch die Unterseiten einer Webseite untersucht, um den Einfluss von purer Frontpage-Optimierungen zu vermeiden. Jedoch würde dies in vielen Fällen gegen die `robots.txt`[1] Vereinbarung verstoßen und den zeitlichen Rahmen dieser Arbeit sprengen, weshalb davon grundsätzlich Abstand genommen wurde. Außerdem verlassen viele Besucher*innen die Seite nach wenigen Sekunden wieder und würden die Unterseiten gar nicht erst aufrufen. Die Webseiten wurden servertechnisch schonend inspiziert, und die Datenerhebung erfolgte spät in der Nacht, um den üblichen Internetverkehr nicht zu stören und um die Performance-Daten sauber zu halten.

[1] Eine Datei auf einer Webseite, die Suchmaschinen und Crawlern anweist, welche Unterseiten oder Bereiche der Webseite nicht besucht oder indiziert werden sollen.

3.2 Datensammlung und -analyse

Die Auswahl der Webseiten für diese Studie folgte einem zweigleisigen Ansatz, um eine umfassende Datenbasis zu schaffen, die sowohl die Vielfalt des österreichischen Webraums abdeckt, als auch spezifische Einblicke in die Webpräsenz von Wiener Unternehmen ermöglicht. Einerseits wurden etwa 120 Webseiten mit hohem Traffic aus Listen der „top österreichischen Webseiten“ ausgewählt. Diese repräsentieren eine breite Palette von Inhalten und Diensten, von Nachrichten und sozialen Medien bis hin zu E-Commerce und Unternehmenswebseiten. Andererseits wurden rund 3500 Webseiten von Wiener Unternehmen quer durch alle Branchen, mit einem besonderen Fokus auf kleine und mittlere Unternehmen (KMU), basierend auf Einträgen von firmenabc.at ausgewählt. Dieser Ansatz zielte darauf ab, die typische Diversität von KMUs zu erfassen und gleichzeitig den Effekt von mangelndem Fachwissen im Website-Management zu untersuchen. Es wurden pro Wiener Gemeindebezirk durchschnittlich 600 Unternehmen nach einer Webseite gescannt, auf fehlerhafte Serverantworten geprüft und anschließend Duplikate beseitigt. Jeder Wiener Gemeindebezirk wurde berücksichtigt, mit der Ausnahme des ersten Gemeindebezirks, da dort, mit 2000 untersuchten Unternehmen, überproportional viele Unternehmen ansäßig sind. Übrig blieben 3246 einzigartige, funktionierende Webseiten, die für die Untersuchung mittels Webcrawler herangezogen wurden.

Um die Forschungsfragen dieser Arbeit zu adressieren, wurde eine umfangreiche Palette von Performance-Metriken und technologischen Merkmalen der ausgewählten Webseiten erfasst:

Lighthouse Performance Score (0-100)
: Eine aggregierte Bewertung [18], die von Googles Lighthouse-Tool berechnet wird, um die Performance einer Webseite in verschiedenen Aspekten, einschließlich Ladezeit, Datengrößen und -verkehr, zu bewerten.

LCP (ms)
: Misst die Zeit, die benötigt wird, um den größten sichtbaren Inhaltsteil im Viewport zu laden und darzustellen. Ein schneller LCP wird mit einem besseren Nutzungserlebnis assoziiert und ist eine bekannte Metrik im Bereich Web-Optimierung [19].

TTI (ms)
: Die Zeit, die vergeht, bis eine Seite vollständig interaktiv wird. Eine kurze TTI bedeutet, dass Nutzer*innen schneller mit der Seite interagieren können. Auch diese Metrik wird oft in Zusammenhang mit positivem Nutzererlebnis gesetzt [20].

Speed Index (ms)
: Zeigt, wie schnell die Inhalte einer Webseite während des Ladeprozesses sichtbar werden. Ein niedrigerer Wert steht für eine bessere Performance.

Main-Thread-Aufwand (ms)

Misst die Auslastung des Hauptthreads durch Skriptausführung, Layoutberechnungen und anderen Rendering-Aufgaben. Eine geringere Auslastung führt zu einer reaktionsschnelleren Seite.

Render-Blocking-Ressourcen (ms)

Ressourcen, die das Laden oder Darstellen der Seite verzögern können, weil sie vom Browser verarbeitet werden müssen, bevor der Rest der Seite geladen wird. Eine Reduzierung dieser Ressourcen verbessert die Ladegeschwindigkeit.

Third-Party-Ressourcen (ms und bytes)

Inhalte oder Skripte, die von Dritten bereitgestellt werden und die Performance beeinflussen können. Die Messung in Millisekunden und Bytes hilft, ihren Einfluss auf die Ladezeit zu verstehen.

Bildcodierung (bytes)

Bezieht sich auf die Art und Weise, wie Bilder digital codiert sind. Effiziente Bildcodierung kann die Dateigröße reduzieren, ohne die Qualität merklich zu beeinträchtigen, was die Ladezeiten verkürzt.

Verwendung von modernen Web-Medienformaten (bytes)

Modernere Bild- und Videoformate wie WebP oder AVIF bieten eine bessere Kompression bei gleichbleibender Qualität, was die benötigte Bandbreite reduziert.

Bildoptimierung (bytes)

Die Praxis, Bilder so zu verkleinern und zu komprimieren, dass ihre Dateigrößen minimiert werden, ohne an visueller Qualität für den Benutzer*innen zu verlieren.

ungenutztes CSS/JavaScript (bytes)

CSS-Regeln und JavaScript-Code, welche nicht verwendet, aber dennoch geladen werdem. Ihre Entfernung[2] kann die Größe der übertragenen Dateien reduzieren und somit die Ladegeschwindigkeit verbessern.

unverkleinertes CSS/JavaScript (bytes)

Dateien, die nicht minifiziert wurden, d.h., überflüssige Leerzeichen, Kommentare und andere nicht notwendige Zeichen enthalten. Minifizierung reduziert die Dateigröße und verbessert die Ladezeiten.

Netzwerk-Payload (bytes)

Dateimenge, die über das Netzwerk übertragen wurde. Die Reduzierung der Menge dieser Dateien kann die Ladezeiten verkürzen und die allgemeine Performance verbessern.

[2] z.B. durch „Tree-Shaking“

Grünes Hosting (0 oder 1)

Gibt an, ob der Hosting Provider in der Liste der zertifzierten „Grünes Hosting Anbieter*innen“ [15] ist.

CO_2 mit grünem Hosting (Gramm)

Geschätze CO_2-Emissionen, mit Einbezug des „grünen Hostings“-Status.

CO_2 ohne grünem Hosting (Gramm)

Geschätze CO_2-Emissionen, ohne Einbezug des „grünen Hostings“-Status.

Neben den Performance-Metriken wurden umfangreiche Analysen zum technologischen Aufbau der Webseiten durchgeführt:

Identifizierung des Servers (z. B. nginx, Apache usw.)

Die Server-Software spielt eine wichtige Rolle bei der Auslieferung von Webinhalten. Bekannte Webserver wie Nginx und Apache bieten unterschiedliche Features und Konfigurationsoptionen, die die Geschwindigkeit, Sicherheit und Effizienz der Inhaltsauslieferung beeinflussen können. Die Wahl des Servers kann basierend auf Leistungsfähigkeit, Flexibilität und Sicherheitsmerkmalen getroffen werden.

Verwendete Frontend-Frameworks (z. B. Angular usw.)

Moderne Frameworks wie Angular oder Vue.js unterstützen Entwickler*innen bei der Erstellung interaktiver und dynamischer Benutzeroberflächen. Sie bieten Struktur und Wiederverwendbarkeit, was die Entwicklung beschleunigt und die Performance durch effizientes Rendering und clientseitige Optimierungen verbessern kann.

Bibliotheken (z. B. jQuery usw.)

JavaScript-Bibliotheken wie jQuery oder D3.js bieten vorgefertigte Funktionen für häufige Aufgaben. Die richtige Auswahl und effiziente Nutzung dieser Bibliotheken können die Codebasis vereinfachen und die Ladezeiten verkürzen.

Externe Services (z. B. Google Analytics usw.)

Dazu zählen APIs, Zahlungsdienste, Analytik-Dienste oder Content Delivery Network (CDN), welche die Funktionalität erweitern, aber auch zusätzliche Latenzzeiten oder Abhängigkeiten einführen können. Eine sorgfältige Auswahl und Integration externer Services sind essenziell für die Performance und Sicherheit.

Caching-Technologien (z. B. x-cache usw.)

Caching reduziert die Notwendigkeit, Daten bei jeder Anfrage neu zu generieren oder vom Ursprungsserver zu laden. Technologien wie Varnish, Redis oder einfache HTTP-Caches können die Ladegeschwindigkeit erheblich verbessern und den Server entlasten.

Sicherheitspraktiken (z. B. strict-transport-security usw.)

Zu den Best Practices gehören die Implementierung von HTTPS, Content Security Policy (CSP), regelmäßige Updates und Patches von Softwarekomponenten, sichere Authentifizierungsmechanismen und der Schutz vor gängigen Angriffen wie Cross-Site Scripting (XSS) und SQL-Injection. Gute Sicherheitspraktiken sind grundlegend, um die Integrität der Webseite und die Datenschutzanforderungen zu gewährleisten.

Performance-Optimierungen (z. B. content-encoding usw.)

Dazu zählen Techniken wie Lazy Loading, Bildoptimierung, Minifizierung von CSS und JavaScript, Verwendung moderner Code-Splitting-Techniken und die effiziente Nutzung von Browser-APIs. Diese Optimierungen können die Ladezeiten verkürzen und die Nutzererfahrung verbessern.

Backend-Technologien (z. B. WordPress, Joomla, etc.)

Die Auswahl des Backends (z. B. Node.js, Ruby on Rails, Django) hat direkten Einfluss auf die Entwicklungsgeschwindigkeit, Performance und Skalierbarkeit. Moderne Backend-Technologien bieten oft integrierte Lösungen für Datenbankanbindungen, Authentifizierung und RESTful-APIs.

3.2.1 Ablauf der Datensammlung

Der*die User*in startet mittels eines Bash-Scripts, wie im nächsten Code-Abschnitt (Listing 1) dargestellt, eine Instanz des Webcrawlers in einer Node.js-Umgebung und kann dabei wählen, ob die Analyseergebnisse lokal in einer CSV-Datei gespeichert oder mittels Google Sheets API in Google Sheets hochgeladen werden. Wenn der*die User*in sich für die lokale Speicherung entscheidet, muss er auch einen Dateinamen für die lokale CSV-Datei definieren.

```
1  # Startet den Crawler und lädt die Ergebnisse in Google Sheets hoch
2  sh run.sh
3
4  # Startet den Crawler und speichert die Ergebnisse lokal
5  sh run.sh --local example.csv
```

Listing 1: Bash-Befehle zur Steuerung des Webcrawlers

Der Webcrawler verarbeitet eine bestimmte Menge an Webseiten und legt anschließend eine Pause von 30 Sekunden ein, um die CPU abzukühlen. Dies ist notwendig, um die Konsistenz der Analysequalität zu gewährleisten, da ein langanhaltender Betrieb des Webcrawlers die verwendete Hardware stark beansprucht und dadurch die Benchmarking-Ergebnisse verfälschen könnte.

Die Datensammlung besteht aus vier Schritten und Hauptfunktionen (Listing 2):

```
async function runAnalysis(url, isOnline) {
    const detectedTechs = await analyzeTechStack(url);
    await analyzePerformance(url, detectedTechs);
    await runCo2Check(url, detectedTechs);
    await writeCSV(detectedTechs, isOnline);
}
```

Listing 2: Ablauf der Analyse des Technologie-Stacks einer Webseite

1. analyzeTechStack()

Mittels Axios.js wird eine einfache Anfrage für die index.html der zu untersuchenden Webseite gestellt. Anschließend werden nacheinander folgende Aspekte untersucht: Header, Meta-Tags, die Namen importierter Skript-, Stil- und anderer Dateien nach typischen Mustern in der Namensgebung sowie andere DOM-Elemente, wie im Code-Abschnitt (Listing 3) beschrieben. Die gefundenen Technologien werden in einem Sammelobjekt notiert und für die nächsten Schritte weitergereicht.

```
export const detectBackendTechnologies = ($, headers, detectedTechs) => {
    const bodyHTML = $('body').html();

    // patterns for detecting backend technologies
    const patterns = {
        'WordPress': /wp-content|wp-includes/,
        'Drupal': /sites\/default\/files/,
        'Joomla': /\/components\/|\/templates\//,
        'Express': /Express/,
        'Django': /csrfmiddlewaretoken|admin\/login\/django/,
        'Ruby': /<meta name="csrf-param" content="authenticity_token" \/>/
    };

    Object.keys(patterns).forEach(tech => {
        if (bodyHTML.match(patterns[tech]) ||
            (headers['x-powered-by'] &&
                headers['x-powered-by'].match(tech))) {
            detectedTechs.backendTechnology.add(tech);
        }
    });
}
```

Listing 3: **1.) analyzeTechStack** - Beispiel einer Mustererkennung von Backend-Technologien

2. analyzePerformance()

In der Vorbereitungsphase, noch vor der eigentlichen Analyse, wird ein Puppeteer-Headless-Browser gestartet, damit in diesem Schritt ein neues Fenster im Headless-Browser geöffnet und die zu untersuchende Webseite geladen werden kann. Googles Lighthouse führt einen ausgiebigen und rechenintensiven Benchmarking-Test aus, um die Performance und andere Metriken der Webseite zu messen. Aus den Ergebnissen werden nur die Daten extrahiert und gespeichert, die für die Forschung relevant sind, wie im nachfolgenden Code-Abschnitt (Listing 4) dargestellt. Wie nach dem ersten Schritt, werden auch hier die Daten im Sammelobjekt ergänzt und weitergereicht.

```
export const analyzePerformance = async (url, detectedTechs) => {
    ...
    const options = {
        onlyCategories: ['performance'],
        flags: { emulatedFormFactor: 'mobile' }
    };
    const { lhr } = await lighthouse(url, options, undefined, page);

    if (lhr.categories.performance.score === null) {
        throw new Error('Score NULL')
    }
    detectedTechs.score = parseInt((lhr.categories.performance.score) * 100);
    detectedTechs.lcp =
        parseInt(lhr.audits['largest-contentful-paint'].numericValue);
    detectedTechs.tti = parseInt(lhr.audits['interactive'].numericValue);
    ...
};
```

Listing 4: **2.) analyzePerformance** - Performance Messung mittels Google Lighthouse und Extraktion einiger Metriken

3. runCo2Check()

Diese Phase startet die CO2.js-Bibliothek, um aus dem berechneten Netzwerk-Payload der Lighthouse-Analyse einen geschätzten CO_2-Emissionswert zu ermitteln, wie im nächsten Code-Abschnitt (Listing 5) dargestellt. Zunächst wird geprüft, ob der*die Hosting-Anbieter*in ein*eine zertifizierte*r „grünes Hosting-Anbieter*in“ [15] ist. Entsprechend werden die geschätzten CO_2-Emissionen sowohl mit, als auch ohne den Faktor grünes Hosting notiert und weitergereicht. Falls der*die Hosting-Anbieter*in kein grünes Hosting anbietet, unterscheiden sich diese Werte nicht.

```
export const runCo2Check = async (url, detectedTechs) => {
    const isGreen = await hosting.check(extractDomain(url))
    const co2Emissions = new co2()
    const bytesSent = detectedTechs.enormousNetworkPayloads * AMOUNT_OF_VISITS;
    detectedTechs.greenHosting = isGreen ? 1 : 0;
    detectedTechs.co2WithGreenHosting =
        parseInt(co2Emissions.perByte(bytesSent, isGreen));
    detectedTechs.co2WithoutGreenHosting =
        parseInt(co2Emissions.perByte(bytesSent, false));
}
```

Listing 5: **3.) runCo2Check** - Auswertung der CO_2-Emissionen mittels CO2.js

4. writeCSV()

Je nachdem, ob sich der*die User*in für die lokale oder Online-Speicherung der Daten entschieden hat, speichert der Webcrawler die gesammelten Ergebnisse des Sammelobjekts entweder in eine lokal angelegte CSV-Datei oder lädt sie in Google Sheets hoch, wie im nächsten Code-Abschnitt (Listing 6) beschrieben. Um die Google Sheets API zu nutzen, muss vorher ein Service-Agent in der Google Cloud Console erstellt, mit den entsprechenden Rechten ausgestattet und dessen Schlüssel als JSON-Datei heruntergeladen und in das Projektverzeichnis kopiert werden.

```
export const writeToSheet = async (data) => {
    await client.authorize();
    const request = {
        spreadsheetId: spreadsheetId,
        range: range,
        valueInputOption: 'RAW',
        insertDataOption: 'INSERT_ROWS',
        resource: {
            values: [data]
        },
        auth: client,
    };
    ...
    const response = await sheets.spreadsheets.values.append(request);
    ...
}
```

Listing 6: **4.) writeCSV** - Upload der Daten zu Google Sheets

3.2.2 Analyse der Ergebnisse

Um die große Menge an gesammelten Daten zu analysieren und zu verarbeiten, wurde MATLAB verwendet, um deskriptive Grafiken zu erstellen. Dabei wurden Analysen aus verschiedenen Perspektiven durchgeführt, um die Gesamtheit der Daten zu illustrieren. Besonders wurde darauf geachtet, dass nur aussagekräftige Visualisierungen erstellt wurden, was zu weiteren Anpassungen und Kalkulationen führte.

3.3 Ergebnisse

3.3.1 Allgemeine Performance und Tech-Stack Lage in Wien

Name	Einh.	Mittel	Minimum	Maximum
Score	Score	68	8	100
LCP	Sek	8,29	0.6	223,7
TTI	Sek	8,00	0.6	87,0
Speed Index	Sek	5,67	0.6	61,4
Main-Thread-Aufwand*	Sek	1,97	0	71,2
Render-Blocking*	Sek	1,32	0	54,0
Third-Party-Ressourcen*	Sek	0,13	0	12,0
Third-Party-Ressourcen*	Byte	643175	0	41368259
Bildcodierung*	Byte	1449	0	191360
Web-Medienformate*	Byte	4592	0	216905
Bildoptimierung*	Byte	2113	0	205650
ungenutztes CSS*	Byte	621	0	17965
ungenutztes JS*	Byte	1154	0	22980
unverkleinertes CSS*	Byte	35	0	4350
unverkleinertes JS*	Byte	59	0	9355
Gesamt Netzwerk-Payloads	Byte	3047671	139	161622664
Grünes Hosting	%	49,8	-	-
CO_2 mit grünem Hosting	Gramm	1,1	0	39,5
CO_2 ohne grünem Hosting	Gramm	1,2	0	64,1

Tabelle 1: Tabelle der Performance-Ergebnisse von 3246 einzigartigen Webseiten aus ganz Wien. Abgebildet sind Mittel-, Minimum- und Maximumwerte. Mit (*) gekennzeichnete Werte sind als Einsparungs-/Optimierungspotenziale zu sehen und können als vergeudete Ressourcen betrachtet werden.

Eine durchschnittliche Webseite erreicht einen Score von 68/100 und setzt pro Aufruf 1,1 g CO_2 bei einem grün gehosteten Anbieter*in und 1,2 g CO_2 bei einem*einer nicht-grün hostenden Anbieter*in, lediglich durch die Übertragung der Daten, frei. Die durchschnittliche Netzwerkbelastung beträgt 3 MB und erreicht bis zu 161 MB pro Seitenaufruf. Die größte Belastung entsteht dabei durch Drittanbieter-Ressourcen, wie Code-Bibliotheken, Plugins oder Medien, und beläuft sich auf durchschnittlich 643 KB. Ebenso zeigt sich, dass eine der effektivsten Optimierungsmethoden die Verwendung moderner Web-Medienformate, wie WEBP statt PNG

und JPG, ist, was zum zweithöchsten Einsparungspotenzial von durchschnittlich 4 KB bis hin zu 216 KB pro Seitenaufruf führt. Bezüglich der Ladezeiten beträgt der Durchschnitt 8,29 Sekunden für LCP und 8,0 Sekunden für TTI. Weiterhin wurde festgestellt, dass bereits 49,8 % der untersuchten Webseiten einen*eine Hosting-Anbieter*in verwenden, der*die grünen Strom für seine*ihre Server bezieht. Es wurden drei vollständige Durchläufe der 3246 Webseiten gemacht, um starke Abweichungen in der Webseiten-Performance zu minimieren. Dabei wurden je Durchlauf laut CO2.js 3,6 kg CO_2-Emissionen freigesetzt, was insgesamt 10,8 kg CO_2 für die ganze Studie ergibt, nur durch die Übertragung der Daten und ohne den CO_2-Fußabdruck des verwendeten Endgeräts.

3.3.2 Backend Technologien

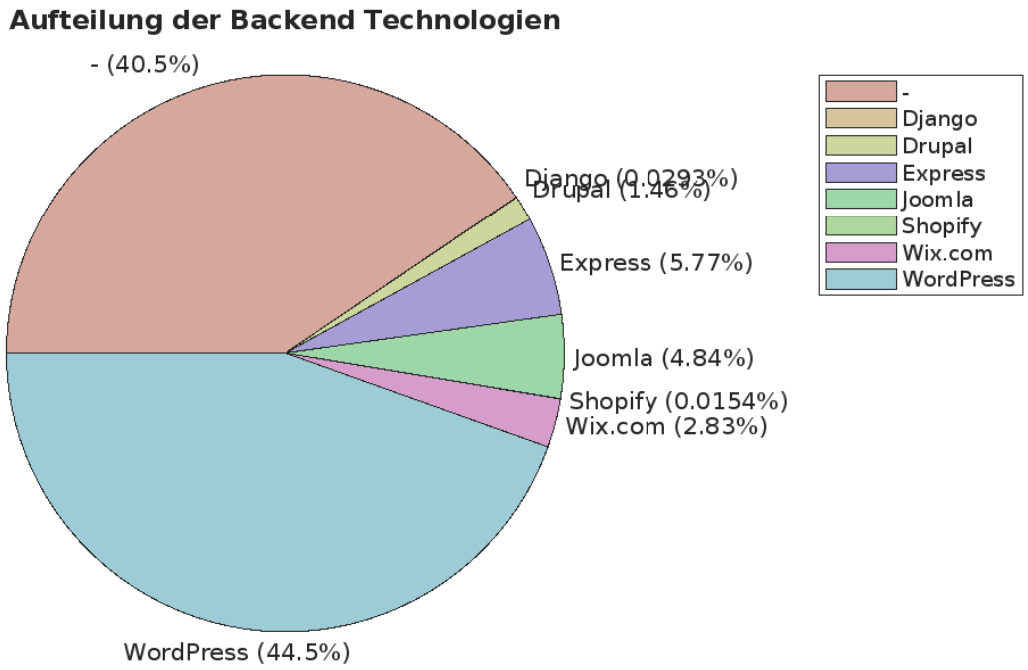

Abbildung 1: Verteilung der erkannten Backend Technologien in einem Tortendiagramm. (Quelle: Eigene Darstellung)

Mit 44,5 % ist WordPress der klare Marktführer. Weitere 40,5 % der Webseiten konnten keiner spezifischen Backend-Technologie zugeordnet werden. Die restlichen 15 % teilen sich andere Anbieter*innen. Django und Shopify wurden jeweils nur bei einer Webseite erkannt.

3.3.3 Frontend Technologien

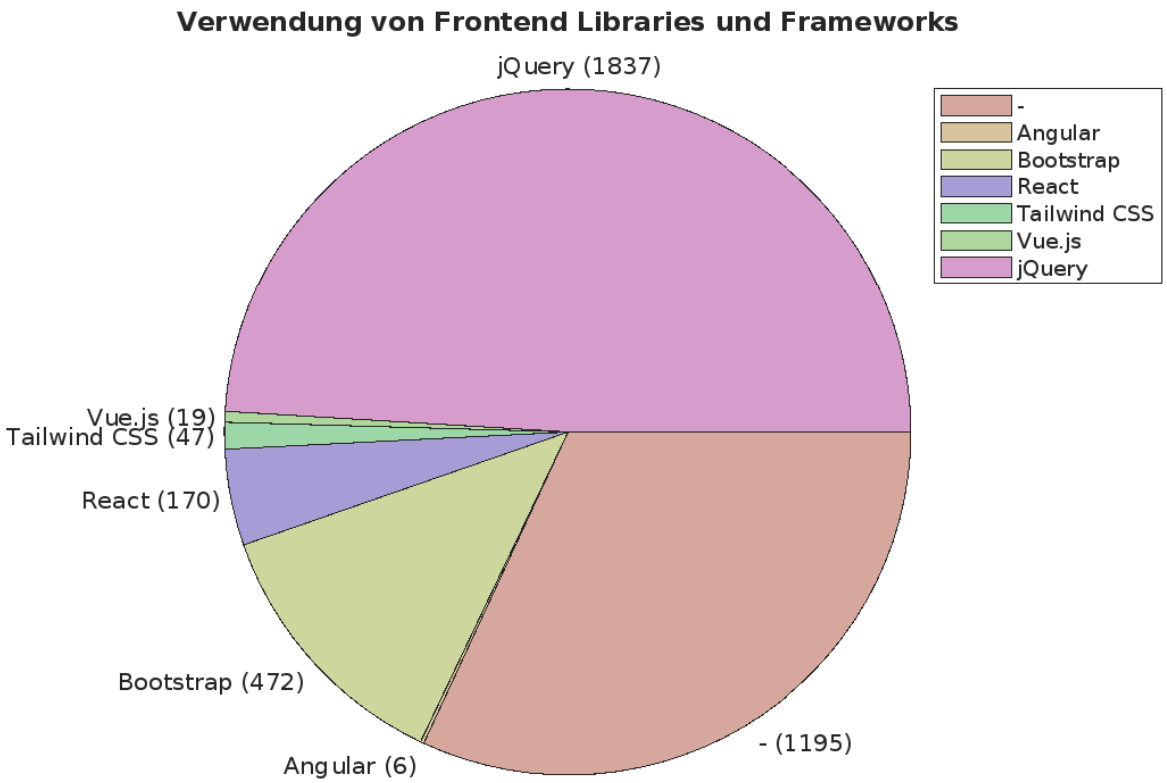

Abbildung 2: Verteilung der erkannten Frontend Libraries und Frameworks. (Quelle: Eigene Darstellung)

Von den untersuchten Webseiten haben 1837 jQuery verwendet, gefolgt von 1195 Webseiten ohne namhafte Frontend-Frameworks oder Bibliotheken. Aufgrund des hohen Interesses an jQuery wurde auch die Version der verwendeten jQuery-Bibliothek untersucht. Ältere Versionen können erhebliche Sicherheitslücken und Performance-Einbußen darstellen.

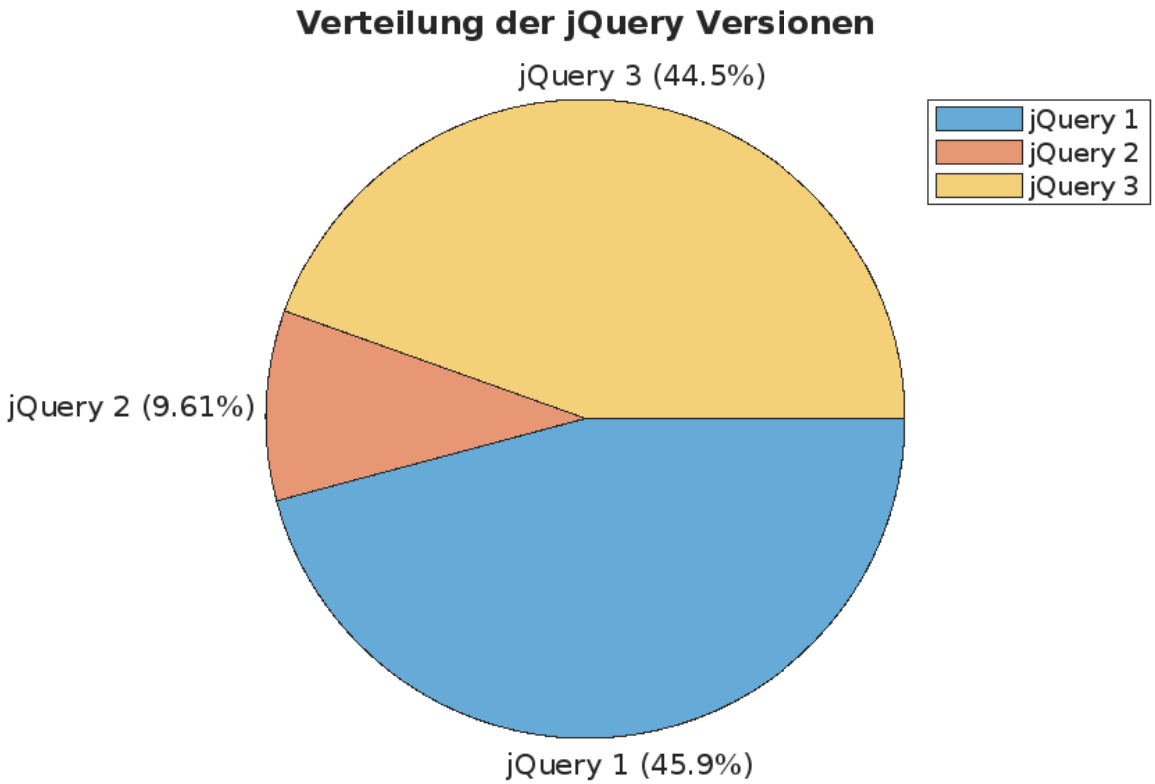

Abbildung 3: Verteilung der erkannten jQuery Versionen. (Quelle: Eigene Darstellung)

3.4 Bewertung der Hypothesen

Der Schwerpunkt dieser Untersuchung lag auf dem Umweltaspekt und der damit verbundenen vergeudeten Energie durch lange Ladezeiten sowie den freigesetzten CO_2-Emissionen bei der Datenübertragung. Um die Ergebnisse der Ladezeiten und übertragenen Datenmengen zu normieren und in Relation zur Gesamtmenge der untersuchten Webseiten zu setzen, wurde folgende Formel entwickelt und angewendet:

T - Ladezeit einer Webseite

$\bar{T}$ - Durchschnittliche Ladezeit aller untersuchten Webseiten

M - Main-Thread-Arbeitszeit einer Webseite

M_f - Faktor der Main-Thread-Arbeitszeit, definiert als $\frac{M}{T}$

E - Energie Faktor

C - CO_2-Emissionen einer Webseite

$\bar{C}$ - Durchschnittliche CO_2-Emissionen aller untersuchten Webseiten

F - CO_2-Faktor

$$Impact = \sqrt{\left(\frac{T}{\bar{T}} \times M_f\right) \times \left(\frac{C}{\bar{C}}\right) \times EF}$$

Die Kalkulation basiert auf der Annahme, dass während des Ladens einer Webseite das Gerät des Endverbrauchers weiterhin Strom verbraucht. Da österreichweit fast 90 % des Website-Traffics von Mobilgeräten stammen [21] und bei diesen oft über 70 % des Energieverbrauchs allein auf den Bildschirm entfallen [22], wird die Ladezeit als Einsparpotenzial für Energie betrachtet. Da eine einzige Webseite potenziell mehrere Millionen Mal von verschiedenen Geräten aufgerufen werden kann, ist das Gesamteinsparpotenzial enorm, aber schwer abzuschätzen, wodurch eine Berücksichtigung der Gesamtanzahl verschiedenster Faktoren den Rahmen dieser Arbeit sprengen würde.

Die normierte Ladezeit L einer Webseite wird berechnet, indem sie durch den Durchschnitt aller Ladezeiten der untersuchten Webseiten $\bar{T}$ geteilt wird. Anschließend wird das Produkt aus Ladezeit und Main-Thread-Arbeitszeit M_f gebildet, um den zusätzlichen Energieaufwand bei langer Main-Thread-Arbeitszeit zu berücksichtigen. Ähnlich werden die von CO2.js geschätzten CO_2-Emissionen C durch den Durchschnitt aller Emissionen $\bar{C}$ geteilt und normiert. Danach werden die beiden normierten Werte – der Energiefaktor E und der CO_2-Faktor F – multipliziert und vom Ergebnis wird die Wurzel gezogen, um Ausreißer zu glätten. Bei der Auswertung für diese Arbeit wurden diese Gewichte EF jedoch nicht berücksichtigt, da zu viele externe Faktoren diese Gewichtung beeinflussen könnten.

Weiter ist es schwer, ein pauschales Verhältnis zwischen der Energieverschwendung von Endgeräten und Datazentren festzulegen. Obwohl global gesehen nur 33 % der CO_2-Emissionen auf Datazentren zurückzuführen sind [23], variiert der Stromverbrauch zwischen den Programmiersprachen erheblich. Beispielsweise kann PHP bis zu 13,8 Mal mehr Strom verbrauchen als JavaScript und 29,3 Mal mehr als C [24] [25]. Auch Faktoren wie Browserwahl [26] und Verbindungsgeschwindigkeit sowie -qualität [27] spielen eine wesentliche Rolle in der Berechnung der tatsächlichen CO_2-Emissionen.

3.4.1 Bewertung der 1. Hypothese

> *Hypothese 1: Es besteht eine signifikante Korrelation zwischen der Verwendung moderner Web-Technologien und einer Reduzierung des Energieverbrauchs sowie der CO_2-Emissionen von Webseiten.*

Um diese Hypothese zu überprüfen, wurde eine Kategorisierung aller untersuchten Webseiten nach ihren verwendeten Frontend- und Backend-Technologien durchgeführt. Zur quantitativen Bewertung dieser Hypothese wurden die CO_2-Emissionen und der zuvor erläuterte $Impact$ für jede Webseite berechnet und verglichen.

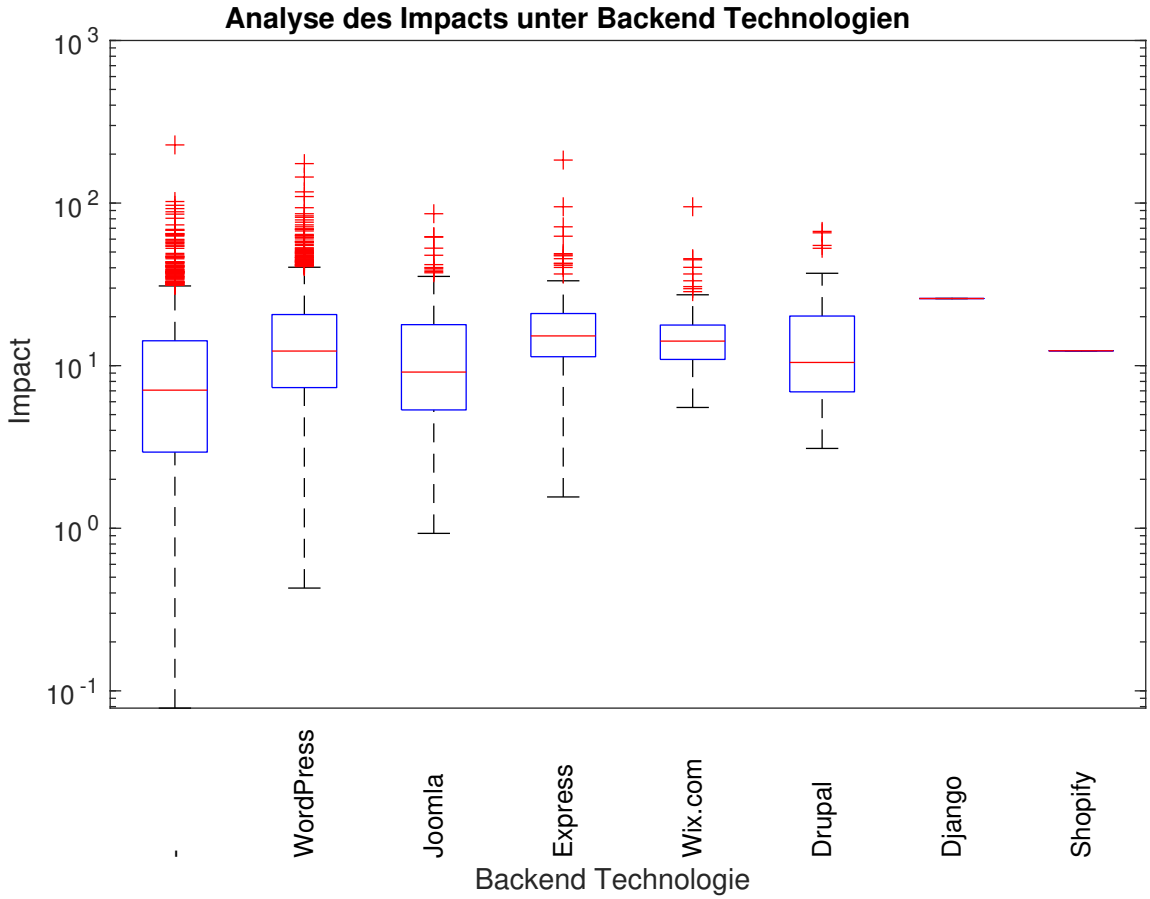

Abbildung 4: Jeder Boxplot repräsentiert die Verteilung der CO_2-Emissionen für die jeweilige Backend Technologie: kein Backend (-), WordPress, Joomla, Express, Wix.com, Drupal, Django und Shopify. (Quelle: Eigene Darstellung)

Die Analyse der Backend-Technologien offenbart bereits signifikante Unterschiede. Insbesondere zeigt WordPress das breiteste Spektrum an CO_2-Emissionen, was die Annahme stützt, dass die Nutzung von No-Code-Plattformen ohne entsprechendem Fachwissen zu erheblichen Performanceeinbußen führen kann.

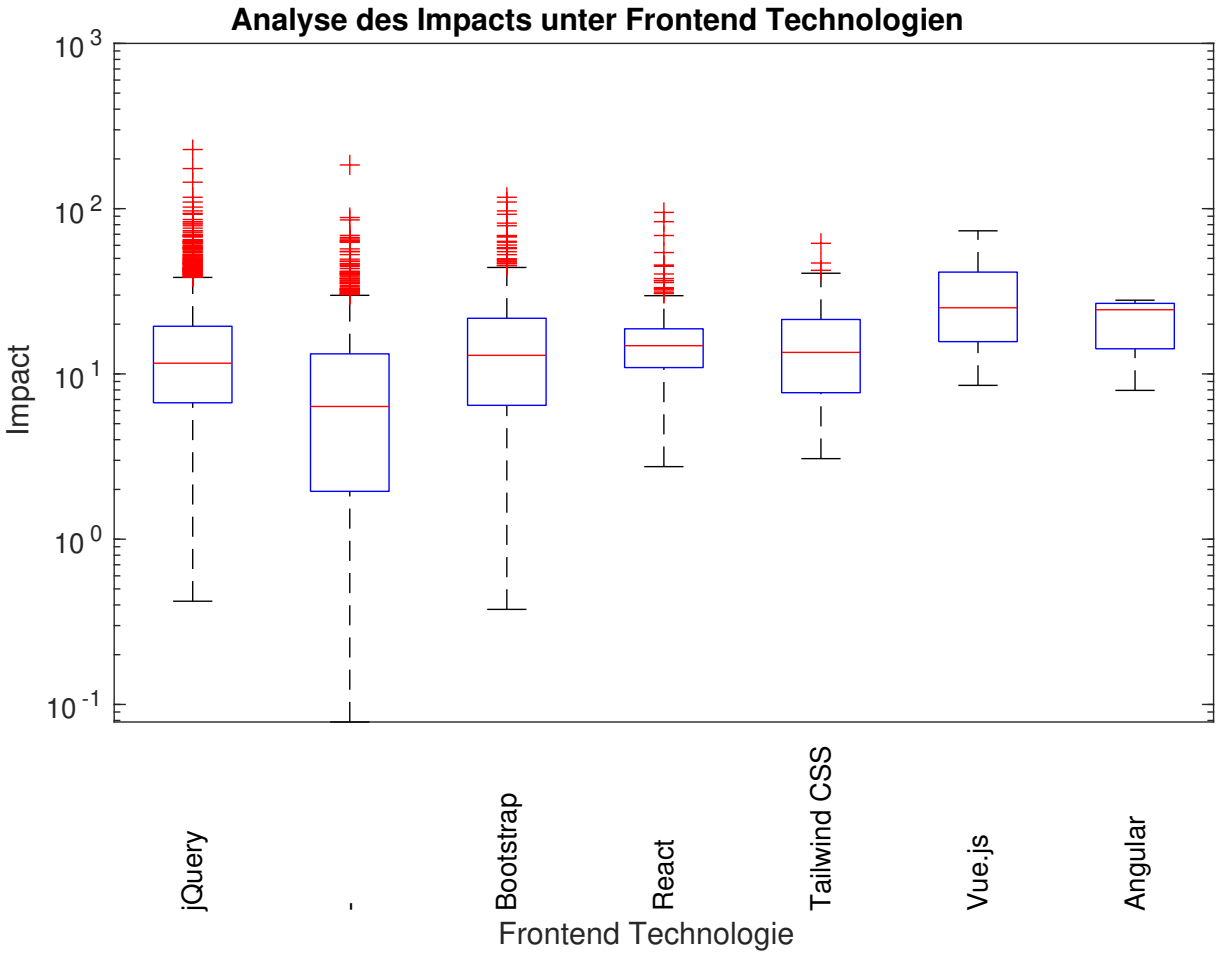

Abbildung 5: Jeder Boxplot repräsentiert die Verteilung der CO_2-Emissionen für die jeweilige Frontend Technologie. (Quelle: Eigene Darstellung)

Bei der Analyse der Frontend-Technologien zeigte sich, dass sowohl die Nutzung als auch der Verzicht auf jQuery oder andere Bibliotheken die endgültige Performance nur geringfügig beeinflusst und zu großen Streuungen in den Daten führt. Vue.js schnitt jedoch am schlechtesten ab, da sowohl sein Median als auch seine Mittelwerte höher ausfielen als bei anderen Technologien. React hingegen zeigte die geringste Streuung der Mittelwerte und scheint eine deterministische Performance zu bieten. Es gab jedoch keine Ausreißer im Bereich niedriger Impact-Werte, während Webseiten mit jQuery, Bootstrap, oder ohne spezielle Frontend-Technologie manchmal bessere Ergebnisse lieferten als jede React-basierte Webseite.

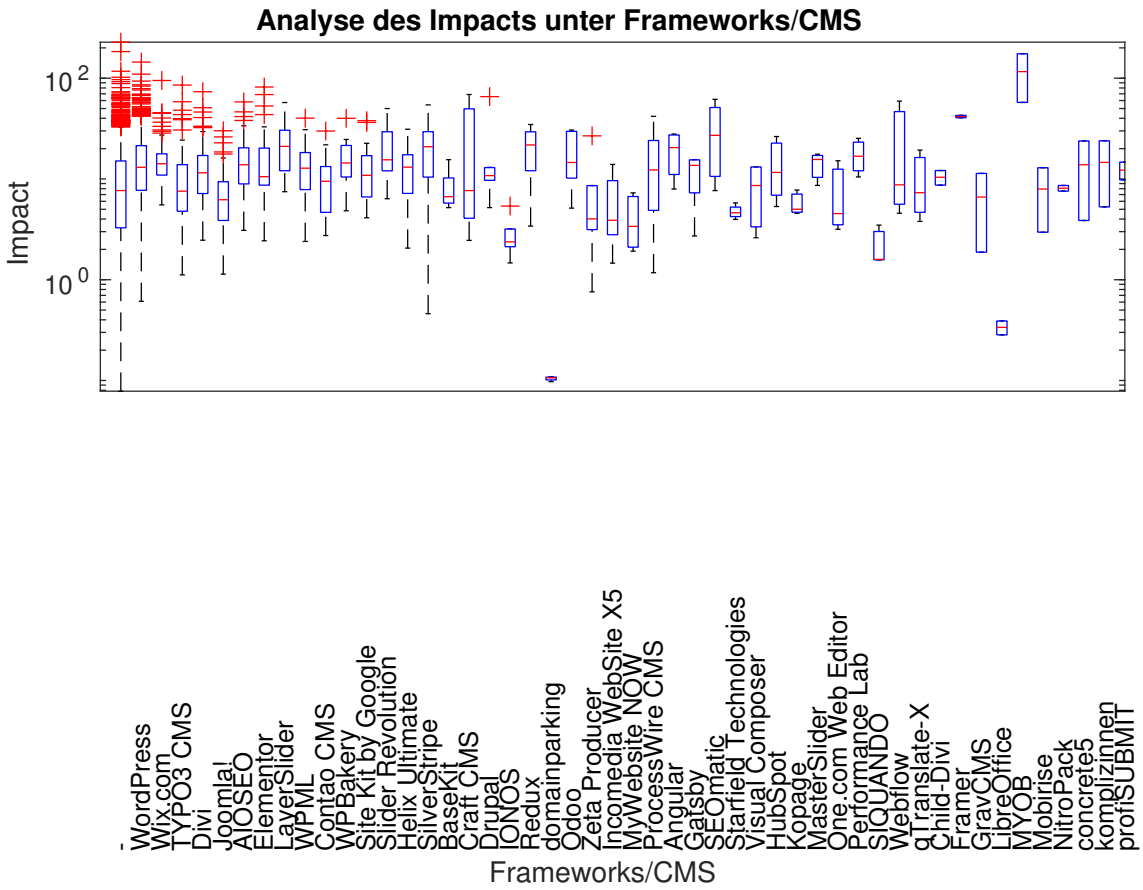

Abbildung 6: Jeder Boxplot repräsentiert die Verteilung der CO_2-Emissionen für die jeweiligen Frameworks/CMS. (Quelle: Eigene Darstellung)

Die Analyse der Frameworks und CMS verdeutlicht deutliche Unterschiede in den erkannten Technologien. Während die größte Streuung bei jenen Webseiten zu beobachten war, bei denen keine Frameworks oder CMS erkannt wurden, zeigten WordPress, Elementor, WPML und andere WordPress-bezogene Technologien eine Platzierung in den oberen Bereichen des Impact-Profils. Wix.com schnitt nur geringfügig besser ab als WPML, platzierte sich jedoch ebenfalls in einem hohen Bereich. Gute Platzierungen erreichen IONOS sowie einige weniger bekannte Technologien wie Zeta Producer, Incomedia Website und My Website NOW.

Zusammenfassend lässt sich zur ersten Hypothese feststellen, dass trotz einer umfassenden Untersuchung, welche die CO_2-Emissionen in Bezug zu Ladezeiten und Main-Thread-Arbeitszeit setzte, keine signifikanten Unterschiede festgestellt werden konnten. Gewisse Unterschiede und Tendenzen in den erhobenen Daten waren erkennbar, jedoch nicht aussagekräftig genug, um als eindeutige Muster identifiziert zu werden. Diese Erkenntnis legt nahe, dass die alleinige Verwendung moderner Web-Technologien möglicherweise nicht ausreicht, um den Energieverbrauch und die CO_2-Emissionen signifikant zu reduzieren.

3.4.2 Überprüfung von Hypothese 2

> *Hypothese 2: Webseiten, die für Performance und Effizienz optimiert sind (z.B. durch die Minimierung des Datenverkehrs und effiziente Ressourcennutzung), haben im Vergleich zu weniger optimierten Webseiten einen geringeren ökologischen Fußabdruck..*

Um diese Hypothese zu überprüfen, wurde der Lighthouse Score als Indikator für eine optimierte Webseite herangezogen und mit den geschätzten CO_2-Emissionen verglichen.

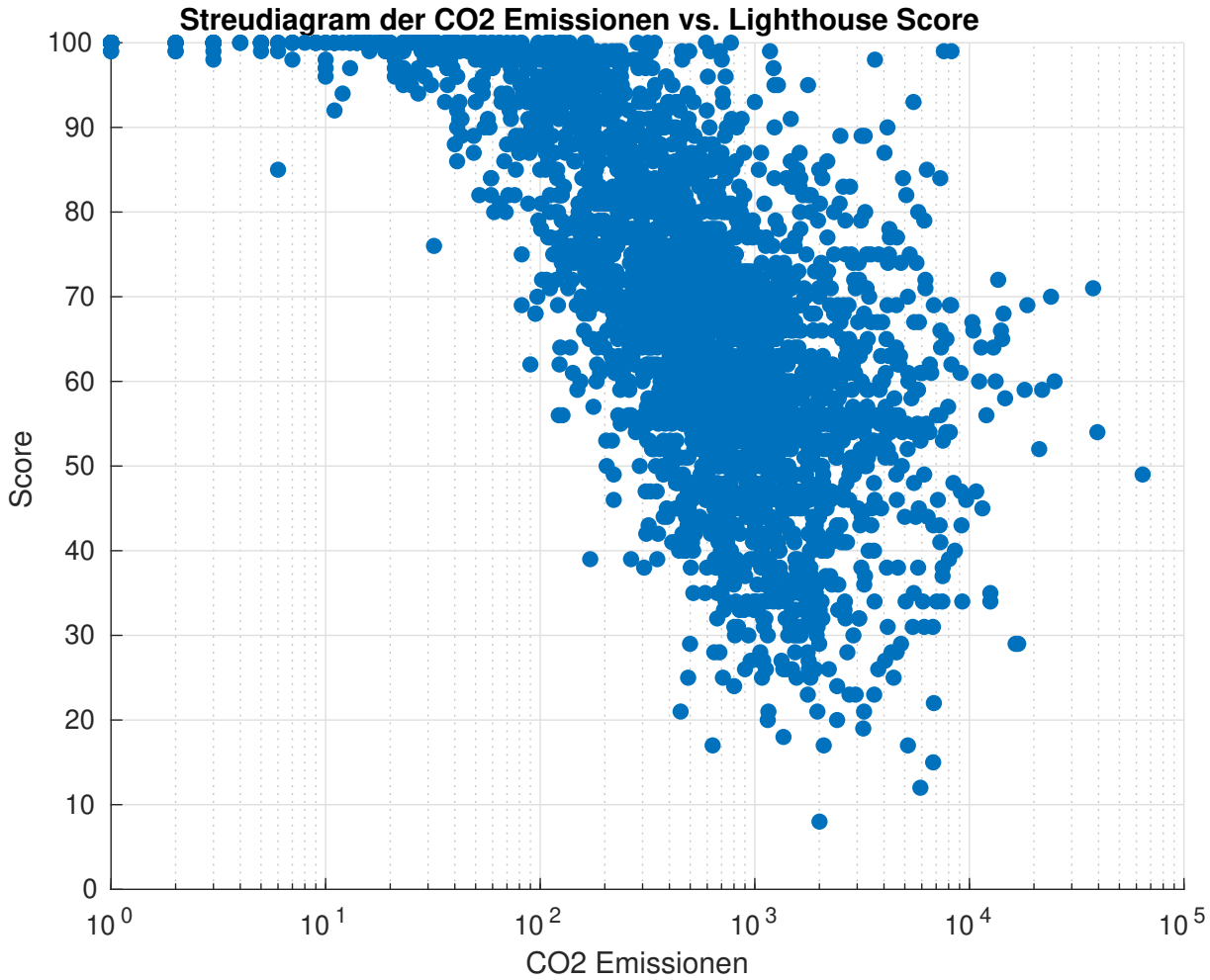

Abbildung 7: Verteilung der CO_2 Emissionen gegenüber dem Lighthouse Score. (Quelle: Eigene Darstellung)

Der eindeutige Abwärtstrend zeigt, dass mit einem sinkenden Lighthouse Score die CO_2-Emissionen ansteigen. Somit wird die zweite Hypothese bestätigt.

4 Diskussion

4.1 Ergebnisinterpretation

Die in der Analyse dargelegten Daten offenbaren die Unterschiede zwischen den verschiedenen Backend- und Frontend-Technologien sowie den verwendeten Frameworks und CMS bezüglich ihres Einflusses auf CO_2-Emissionen und Performance. WordPress, als Beispiel für eine No-Code-Lösung, weist eine breite Spanne an CO_2-Emissionen auf. Dies deutet darauf hin, dass der Einsatz solcher Technologien ohne tiefgreifendes Fachwissen zu Performancenachteilen führen kann, die wiederum erhöhte Umweltauswirkungen nach sich ziehen. Diese Erkenntnis stützt die Vermutung, dass die Wahl der Technologie und das Wissen über ihre effiziente Nutzung entscheidend sind, um umweltfreundlichere Web-Lösungen zu schaffen.

Bei den Frontend-Technologien zeigt sich, dass weder die Nutzung noch der Verzicht auf Bibliotheken wie jQuery die Performance wesentlich beeinflussen, was auf eine hohe Variabilität in der Implementierung hindeutet. Vue.js zeigte im Vergleich zu anderen Technologien schlechtere Leistungswerte, während React konsistentere Mittelwerte aufwies, jedoch ohne signifikante Spitzen in umweltfreundlichen Bereichen. Dies könnte darauf hinweisen, dass trotz der Vorhersehbarkeit in der Performance durch die Nutzung von React, die Optimierungsmöglichkeiten möglicherweise noch nicht vollständig ausgeschöpft werden, oder die Bibliothek ein gewisses Maß an Grund-Leistung benötigt.

Die Analyse der Frameworks und CMS zeigt, dass Technologien, die auf WordPress basieren, tendenziell höhere CO_2-Emissionen verursachen, was deren Platzierung in den oberen Bereichen der Impact-Skala erklärt. Dies unterstreicht die Notwendigkeit für Entwickler*innen, bei der Wahl und Implementierung dieser Technologien sorgfältig zu überlegen, wie sie den Energieverbrauch minimieren können.

4.2 Bedeutung der Ergebnisse

Die Ergebnisse dieser Studie sind besonders relevant für Webentwickler*innen und -designer*innen, sowie Entscheidungsträger*innen in der Technologiebranche. Sie verdeutlichen, dass die bloße Auswahl moderner Web-Technologien nicht ausreichend ist, um den Energieverbrauch und die CO_2-Emissionen signifikant zu reduzieren. Vielmehr ist ein tiefes Verständnis der Funktionsweise und eine optimierte Implementierung dieser Technologien er-

forderlich. Zudem zeigt der klare Zusammenhang zwischen niedrigeren Lighthouse Scores und höheren CO_2-Emissionen, wie wichtig eine umfassende Optimierung für die umweltfreundliche Webentwicklung ist.

4.3 Empfehlungen und Best Practices

Auf Grundlage der Ergebnisse lassen sich mehrere Empfehlungen formulieren:

- Optimierung der Codebasis: Entwickler*innen sollten Techniken wie Lazy Loading, Tree-Shaking und Code Minification anwenden, um den Datenverkehr und die Ladezeiten zu minimieren.
- Bewusste Technologiewahl: Die Auswahl von Technologien sollte nicht nur auf Basis der Funktionalität, sondern auch unter Berücksichtigung ihres Potenzials zur Minimierung des ökologischen Fußabdrucks erfolgen.
- Fortbildung und Bewusstseinsbildung: Es ist essenziell, dass Entwickler*innen und Designer*innen in den Prinzipien der umweltfreundlichen Webentwicklung geschult werden.

4.4 Limitationen der Studie

4.4.1 Technische und methodische Limitationen

Anpassung der Analysemethoden

Im Verlauf der Entwicklung des Webcrawlers und der Analysemethoden wurden mehrere Anpassungen vorgenommen, um die Genauigkeit der Ergebnisse zu maximieren. Trotz dieser Bemühungen stieß man auf Grenzen bei der Erfassung bestimmter Daten, insbesondere bei Webseiten, die ihre verwendeten Technologien effektiv verbergen. Diese Limitation führt zu potenziellen Verzerrungen in den gesammelten Daten, da nicht alle relevanten Informationen erfasst werden konnten.

Hardware-Limitationen und Performance-Ergebnisse

Nach dem ersten Probelauf des Crawlers wurde festgestellt, dass die verwendete Hardware anfällig für Überhitzung war, was die Performance des Crawlers beeinträchtigte. Insbesondere die Ladezeiten der analysierten Webseiten waren teilweise doppelt so hoch im Vergleich zu Prüfdurchläufen, in denen die Hardware abgekühlt war. Als Lösungsansatz wurde der Crawling-Prozess angepasst, indem die Menge der untersuchten Webseiten in 10er Chunks aufgeteilt wurde. Nach jeweils zehn analysierten Webseiten erfolgte eine Wartezeit von 30 Sekunden, um die Hardware abkühlen zu lassen. Diese Maßnahme verbesserte die Zuverlässigkeit der

Performance-Ergebnisse, führte jedoch zu einer Verlängerung der Gesamtdauer des Crawling-Prozesses.

Memory Leaks in der Lighthouse Bibliothek

Eine weitere Herausforderung stellten Memory Leaks in der verwendeten Lighthouse Bibliothek dar. Wenn der Crawler über einen längeren Zeitraum lief, füllte sich der RAM zunehmend, was zu Performance-Einbußen führte. Als Gegenmaßnahme wurde der Crawler in einem neuen Terminalfenster gestartet, um jeweils zehn Webseiten zu testen. Das Bash-Script findet sich im Anhang. Nach Abschluss schloss sich das Fenster automatisch, und nach einer Wartezeit von 30 Sekunden öffnete sich ein neues Fenster für den nächsten Durchlauf. Diese Maßnahme ermöglichte es, den RAM auf Betriebssystemebene zu entleeren und die Leistungsfähigkeit des Crawlers zu erhalten. Allerdings erhöhte diese Lösung die Komplexität des Crawlers.

4.4.2 Herausforderungen bei der Datensammlung

Versteckte Technologien

Wie bereits erwähnt, verwenden manche Webseitenbetreiber*innen Techniken, um ihre eingesetzten Technologien zu verbergen. Die Gründe hierfür können vielfältig sein, von der Absicht, sich vor potenziellen Sicherheitsrisiken zu schützen, bis hin zu dem Wunsch, proprietäre Informationen nicht preiszugeben. Die Identifizierung der verwendeten Technologien war in diesen Fällen oft schwierig, was die Vollständigkeit der gesammelten Daten beeinträchtigte.

Blockierung der Crawler-Software

Eine weitere Herausforderung stellte die Blockierung des automatisierten Zugriffs durch einige Webseiten dar. Um dieses Hindernis zu überwinden, wurde der User-Agent-Header des Crawlers angepasst, um die Erkennung als automatisierte Software zu umgehen und stattdessen einen Menschen vorzutäuschen. Der beschriebene User-Agent-Header findet sich im Anhang. Diese Anpassung war notwendig, um Zugang zu den Webseiten zu erhalten und relevante Daten sammeln zu können.

Variabilität der Server-Performance

Die Performance von Webseiten ist abhängig von externen Faktoren, die außerhalb der Kontrolle des Forschungsprojekts liegen. Beispielsweise kann die Tageszeit einen erheblichen Einfluss auf die Serverauslastung haben, was zu Schwankungen in den Ladezeiten und damit zu Verzerrungen in den Performance-Ergebnissen führen kann. Um diesen Variablen Rechnung zu tragen, wurde beschlossen, die Datensammlung hauptsächlich nachts durchzuführen, wenn die Netzwerkauslastung generell niedriger ist. Zusätzlich wurden die Daten von jeder Webseite insgesamt dreimal gesammelt, um temporäre Schwankungen zu berücksichtigen. Für die

Analyse wurden dann jeweils die besten der drei Ergebnisse verwendet, um die optimale Performance der Server widerzuspiegeln. Diese Strategie sollte dazu beitragen, die Zuverlässigkeit der gesammelten Daten zu erhöhen, führte jedoch zu einem erheblich höheren Zeitaufwand für die Datensammlung.

4.5 Schlussfolgerungen

Die in dieser Arbeit durchgeführte Analyse hat wichtige Erkenntnisse über den Zusammenhang zwischen der Nutzung von Web-Technologien und deren Einfluss auf die CO_2-Emissionen sowie den Energieverbrauch geliefert. Die Untersuchung verschiedener Backend- und Frontend-Technologien, Frameworks und CMS hat gezeigt, dass die Auswahl und Implementierung der Technologien einen deutlichen Einfluss auf die ökologische Nachhaltigkeit von Webseiten haben kann.

Trotz der beobachteten Unterschiede und Tendenzen konnte Hypothese 1, welche eine signifikante Korrelation zwischen der Nutzung moderner Technologien und einer Reduzierung von CO_2-Emissionen sowie Energieverbrauch postulierte, nicht eindeutig bestätigt werden. Dies deutet darauf hin, dass die bloße Verwendung neuerer oder populärer Technologien nicht automatisch zu einer verbesserten ökologischen Effizienz führt. Vielmehr ist es die Art und Weise, wie diese Technologien implementiert und optimiert werden, die entscheidend ist.

Die Bestätigung der zweiten Hypothese, dass Webseiten mit optimierter Performance auch geringere CO_2-Emissionen aufweisen, unterstreicht die Bedeutung einer durchdachten und umfassenden Optimierung. Hierbei spielen sowohl technische Aspekte wie die Minimierung von Datenverkehr und effiziente Ressourcennutzung als auch die Auswahl der richtigen Technologien eine entscheidende Rolle.

Diese Erkenntnisse legen nahe, dass eine umfassende Ausbildung und Sensibilisierung von Entwickler*innen hinsichtlich der ökologischen Auswirkungen ihrer Arbeit von großer Bedeutung sind. Zudem wird deutlich, dass nachhaltige Webentwicklung ein fortlaufender Prozess ist, der sowohl fortgeschrittene technische Fähigkeiten als auch ein tiefes Verständnis für die Wechselwirkungen zwischen Technologie, Performance und Umwelt erfordert.

Abschließend lässt sich festhalten, dass die Entwicklung von umweltfreundlicheren Webanwendungen eine gemeinschaftliche Anstrengung erfordert, die über die Grenzen einzelner Technologien hinausgeht und eine integrierte Betrachtung von Design, Entwicklung und Optimierung erfordert. Die Ergebnisse dieser Arbeit bieten eine wertvolle Grundlage für zukünftige Forschungen in diesem Bereich und für die Praxis der Webentwicklung, die zunehmend auf Nachhaltigkeit ausgerichtet sein muss.

Literatur

[1] C. Freitag, M. Berners-Lee, K. Widdicks, B. Knowles, G. S. Blair und A. Friday, „The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations," *Patterns*, Jg. 2, Nr. 9, S. 100 340, 2021, ISSN: 2666-3899. DOI: https://doi.org/10.1016/j.patter.2021.100340. Adresse: https://www.sciencedirect.com/science/article/pii/S2666389921001884#sec1.1 (besucht am 17. 05. 2024).

[2] A. S. G. Andrae und T. Edler, „On Global Electricity Usage of Communication Technology: Trends to 2030," *Challenges*, Jg. 6, Nr. 1, S. 117–157, 2015, ISSN: 2078-1547. DOI: 10.3390/challe6010117. Adresse: https://www.mdpi.com/2078-1547/6/1/117#8RenewableElectricity (besucht am 17. 05. 2024).

[3] Kinsta, „WordPress Market Share Statistics (2011-2024)," Adresse: https://kinsta.com/wordpress-market-share/ (besucht am 17. 05. 2024).

[4] G. Becker, L. Bennici, A. Bhargava, A. D. Miglio, J. Lewis und P. Sachdeva, „The green IT revolution: A blueprint for CIOs to combat climate change," *mckinsey*, Adresse: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-green-it-revolution-a-blueprint-for-cios-to-combat-climate-change (besucht am 17. 05. 2024).

[5] A. Batmunkh, „Carbon Footprint of The Most Popular Social Media Platforms," *Sustainability*, Jg. 14, Nr. 4, 2022, ISSN: 2071-1050. DOI: 10.3390/su14042195. Adresse: https://www.mdpi.com/2071-1050/14/4/2195 (besucht am 17. 05. 2024).

[6] J. Giceva, „Data Processing on Modern Hardware," Adresse: https://db.in.tum.de/teaching/ss21/dataprocessingonmodernhardware/MH_1.pdf?lang=en (besucht am 17. 05. 2024).

[7] P. Rani, J. Zellweger, V. Kousadianos, L. Cruz, T. Kehrer und A. Bacchelli, „Energy Patterns for Web: An Exploratory Study," Jan. 2024. Adresse: https://arxiv.org/html/2401.06482v1 (besucht am 17. 05. 2024).

[8] G. S. Foundation, „Green software," Adresse: https://learn.greensoftware.foundation/ (besucht am 17. 05. 2024).

[9] wiki.selfhtml.org, *Minifizierung*, 2024. Adresse: https://wiki.selfhtml.org/wiki/Minify (besucht am 17. 05. 2024).

[10] Node.js, *Node.js*, 2024. Adresse: https://nodejs.org/en (besucht am 17. 05. 2024).

[11] Cheerio.js, *Cheerio.js*, 2024. Adresse: https://cheerio.js.org/ (besucht am 17. 05. 2024).

[12] Puppeteer, *Puppeteer*, 2024. Adresse: https://pptr.dev/ (besucht am 17. 05. 2024).

[13] Google, *Lighthouse*, 2024. Adresse: https://github.com/GoogleChrome/lighthouse/blob/main/docs/readme.md (besucht am 17. 05. 2024).

[14] Green Web Foundation, *CO2.js*, 2024. Adresse: https://www.thegreenwebfoundation.org/co2-js/ (besucht am 17. 05. 2024).

[15] Green Web Foundation, *Green Web Dataset*, 2024. Adresse: https://www.thegreenwebfoundation.org/tools/green-web-dataset/ (besucht am 17. 05. 2024).

[16] Google, *Google Sheets API*, 2024. Adresse: https://developers.google.com/sheets/api/reference/rest (besucht am 17. 05. 2024).

[17] MATLAB, *MATLAB*, 2024. Adresse: https://de.mathworks.com/products/matlab.html (besucht am 17. 05. 2024).

[18] Google, *Lighthouse Performance Scoring*, 2024. Adresse: https://developer.chrome.com/docs/lighthouse/performance/performance-scoring (besucht am 17. 05. 2024).

[19] B. P. Philip Walton, *Largest Contentful Paint (LCP)*, 2024. Adresse: https://web.dev/articles/lcp (besucht am 17. 05. 2024).

[20] P. Walton, *Time to Interactive (TTI)*, 2024. Adresse: https://web.dev/articles/tti (besucht am 17. 05. 2024).

[21] Statista, „Mobile internet usage penetration in Austria from 2020 to 2029," 2024. Adresse: https://www.statista.com/statistics/567690/predicted-mobile-internet-user-penetration-rate-in-austria/ (besucht am 17. 05. 2024).

[22] P.-E. Moreau, „Argos: Comparing the Energy Consumption of Two Web Stacks," März 2021. Adresse: https://marmelab.com/blog/2021/03/04/argos-comparing-the-energy-consumption-of-two-web-stacks.html (besucht am 17. 05. 2024).

[23] A. Andrae, „Total Consumer Power Consumption Forecast," Okt. 2017. Adresse: https://www.researchgate.net/publication/320225452_Total_Consumer_Power_Consumption_Forecast (besucht am 17. 05. 2024).

[24] R. Pereira, M. Couto, F. Ribeiro u. a., „Energy efficiency across programming languages: how do energy, time, and memory relate?," S. 256–267, Okt. 2017. DOI: 10.1145/3136014.3136031. Adresse: https://marmelab.com/blog/2021/03/04/argos-comparing-the-energy-consumption-of-two-web-stacks.html (besucht am 17. 05. 2024).

[25] R. Pereiraa, M. Coutoa, F. Ribeiroa u. a., „Ranking Programming Languages by Energy Efficiency," Dez. 2020. Adresse: https://repositorium.uminho.pt/bitstream/1822/69044/1/paper.pdf (besucht am 17. 05. 2024).

[26] J. de Macedo, J. Aloísio, N. Gonçalves, R. Pereira und J. Saraiva, „Energy Wars - Chrome vs. Firefox," 2020. Adresse: https://dl.acm.org/doi/pdf/10.1145/3417113.3423000 (besucht am 17. 05. 2024).

[27] H. Ferreboeuf, M. Efoui-Hess und X. Verne, „Environmental impacts of digital technology: 5-year trends and 5G governance,“ 2021. Adresse: https://theshiftproject.org/wp-content/uploads/2023/04/Environmental-impacts-of-digital-technology-5-year-trends-and-5G-governance_March2021.pdf (besucht am 17. 05. 2024).

[28] A. M. (Domain314), *Bachelorarbeit Quellcode*, 2024. Adresse: https://github.com/Domain314/6_Bachelor-Abgabe (besucht am 17. 05. 2024).

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

IKT Informations- und Kommunikations-Technologie

LCP Largest Contentful Paint

TTI Time to Interactive

CMS Content Management System

KMU kleine und mittlere Unternehmen

CDN Content Delivery Network

A QR Code zum GitHub-Repository

Abbildung 8: QR-Code zum GitHub-Repository dieser Arbeit [28]. Enthalten sind der Quellcode für den Crawler, der Quellcode dieser Arbeit in LaTeX, der Quellcode der MATLAB-Grafiken und die Ergebnisse der Analyse als CSV-Datei. (Quelle: Eigene Darstellung durch qr-code-generator.com)

B Webcrawler und MATLAB Quellcode

```
#!/bin/bash

CHUNK_SIZE=10
WAIT_DURATION=10
PAUSE_DURATION=30

TOTAL_URLS=3246
let TOTAL_CHUNKS=($TOTAL_URLS+$CHUNK_SIZE-1)/$CHUNK_SIZE

local_arg=""

for ((i=1; i<=$#; i++)); do
    if [ "${!i}" == "--local" ]; then
        echo "Running in local mode"
        let next_index=i+1
        local_arg="${!next_index}"
        break
    fi
done

for (( i=0; i<$TOTAL_CHUNKS; i++ ))
do
    let START=$i*$CHUNK_SIZE
    let END=($i+1)*$CHUNK_SIZE-1

    konsole --new-tab -e "bash -c 'node index.js $START $END $local_arg'" &
    while [ ! -f "complete.txt" ]; do sleep $WAIT_DURATION; done

    echo "done found"

    rm complete.txt

    sleep $PAUSE_DURATION
done
```

Listing 7: **run.sh** - Startet die Datensammlung. Arbeitet sich in 10er Chunks durch und wartet 30 Sekunden zwischen den Durchläufen.

```
const response = await axios.get(websiteUrl, {
    headers: {
        'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:123.0)
            Gecko/20100101 Firefox/123.0',
        'Accept-Language': 'en-US,en;q=0.9',
        'Accept': 'text/html,application/xhtml+xml,
            application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8',
        'Accept-Encoding': 'gzip, deflate, br',
        'Referer': 'https://www.google.com/',
        'Cache-Control': 'no-cache',
    }
});
```

Listing 8: **User-Agent-Header** zur Umgehung der Crawler-Blockierung

```
data =  readtable('./data/lighthouse_data4_full.csv');

% Initialize a map to hold the count for each major version of jQuery
jqueryVersionCount = containers.Map('KeyType', 'char', 'ValueType', 'double');

% Iterate through each row in the data
for i = 1:height(data)
    % Split the technologies by comma for the current row and trim spaces
    techs = strtrim(strsplit(data.libraries{i}, ','));
    % For each technology in the current row
    for j = 1:length(techs)
        tech = techs{j};

        % Check if the technology starts with 'jQuery'
        if startsWith(tech, 'jQuery')
            % Regular expression to extract version numbers
            versionPattern = '^jQuery\s+(\d+)';
            tokens = regexp(tech, versionPattern, 'tokens');
            if ~isempty(tokens)
                % Build the grouped version name (e.g., 'jQuery 3')
                majorVersion = tokens{1}{1};
                groupedVersionName = ['jQuery ', majorVersion];

                % Count the occurrences
                if isKey(jqueryVersionCount, groupedVersionName)
                    jqueryVersionCount(groupedVersionName) =
                        jqueryVersionCount(groupedVersionName) + 1;
                else
                    jqueryVersionCount(groupedVersionName) = 1;
                end
            end
        end
    end
end

% Extract the technology names and their respective counts
techNames = keys(jqueryVersionCount);
techCounts = zeros(length(techNames), 1);

% Accumulate the counts for each technology
for i = 1:length(techNames)
    techCounts(i) = jqueryVersionCount(techNames{i});
end

% Plotting the pie chart
piechart(techCounts,techNames,LabelStyle="namepercent",
    LegendVisible="on", StartAngle="90");
title('Verteilung der jQuery versionen');

print -depsc pieJQuery.eps
```

Listing 9: **pieJQuery.m** - Beispiel der Erstellung eines Tortendiagramms

```
% Load the CSV file
data =  readtable('./data/lighthouse_data4_impact.csv');

% Initialize arrays to hold CO2 values and their corresponding technologies
allCO2Values = []; allTechs = {};

% Iterate through each row in the data
for i = 1:height(data)
    % Split the technologies by comma for the current row and trim spaces
    techs = strtrim(strsplit(data.frameworks{i}, ','));
    % For each technology in the current row
    for j = 1:length(techs)
        tech = techs{j};
        allCO2Values = [allCO2Values; data.impact(i)];
        allTechs = [allTechs; {tech}];
    end
end

% Calculate the frequency of each technology
[uniqueTechs, ~, idx] = unique(allTechs);
techCounts = accumarray(idx, 1);

% Sort the technologies by frequency in descending order
[~, sortIdx] = sort(techCounts, 'descend');
sortedTechs = uniqueTechs(sortIdx);
sortedCO2Values = []; sortedAllTechs = {};

% Populate the new arrays according to the sorted order of technologies
for i = 1:length(sortedTechs)
    tech = sortedTechs{i};
    techIndices = strcmp(allTechs, tech);
    sortedCO2Values = [sortedCO2Values; allCO2Values(techIndices)];
    sortedAllTechs = [sortedAllTechs; allTechs(techIndices)];
end

boxplot(sortedCO2Values, sortedAllTechs, 'LabelOrientation', 'inline',
    'Whisker', 1.5);
ylabel('Impact');
xlabel('Frameworks/CMS');
title('Analyse des Impacts unter Frameworks/CMS');
xtickangle(45);
set(gca, 'YScale', 'log');
xlim([0, 50]);

print -depsc distImpactbyFrameworksCMS.eps
```

Listing 10: **distImpactByFrameworksCMS.m** - Beispiel einer Funktion zur Erstellung der Boxplot Grafiken

```
% Load the CSV file
data =  readtable('./data/lighthouse_data4_impact.csv');

% Initialize arrays to hold CO2 values and their corresponding scores
xValues = [];
yValues = [];
allTechs = {};

% Iterate through each row in the data
for i = 1:height(data)
    % Split the technologies by comma for the current row and trim spaces
    techs = strtrim(strsplit(data.frameworks{i}, ','));
    % For each technology in the current row
    for j = 1:length(techs)
        tech = techs{j};
        % Append the CO2 and score values to the xValues and yValues arrays
        xValues = [xValues; data.CO2WithGreenHosting(i)];
        yValues = [yValues; data.score(i)];
        allTechs = [allTechs; {tech}];
    end
end

% Plot the scatter diagram
scatter(xValues, yValues, 'filled');
xlabel('CO2 Emissionen');
ylabel('Lighthouse Score');
title('Streudiagram der CO2 Emissionen vs. Lighthouse Score');
grid on;  % Turn on the grid
ax = gca; % Get current axes
ax.XAxis.Exponent = 0; % Prevent scientific notation on x-axis
ax.YAxis.Exponent = 0; % Prevent scientific notation on y-axis

set(gca, 'XScale', 'log');

print -depsc scatterCO2vsScore.eps
```

Listing 11: **scatterCO2vsScore.m** - Erstellung des Streudiagrams für CO_2-Emissionen und Lighthouse Score