DHBW
Duale Hochschule
Baden-Württemberg

CO2-Runter-App: Eine Studie zur Anpassung und Erweiterung basierend auf Nutzerbedürfnissen

Studienarbeit

des Studiengangs Informatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Julian Stadler, Kevin Beier

3. Mai 2024

Bearbeitungszeitraum	24 Wochen
Matrikelnummer, Kurs	2757628 TINF21B4, 2876490 TINF21B5
Gutachter	Dr. Oliver Rettig

Erklärung zur Eigenleistung

Wir versichern hiermit, dass wir unsere Studienarbeit mit dem Thema: *CO2-Runter-App: Eine Studie zur Anpassung und Erweiterung basierend auf Nutzerbedürfnissen* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, 3. Mai 2024

Julian Stadler, Kevin Beier

Abstract

Im Zuge der Energiewende und kontinuierlicher Maßnahmen gegen den Klimawandel ist es von großer Bedeutung, das Bewusstsein der Menschen für ihren individuellen CO2-Verbrauch zu schärfen und Wege zur Reduzierung ihres CO2-Fußabdrucks aufzuzeigen. Die CO2-Runter-App, entwickelt von der Stadt Karlsruhe im Rahmen des Projekts "CO2-Runter-App", ermöglicht es den NutzerInnen, ihren CO2-Fußabdruck zu berechnen und mit anderen zu vergleichen. Diese Studienarbeit widmet sich der Analyse der CO2-Runter-App und der Entwicklung einer verbesserten Version, die auf die Bedürfnisse der NutzerInnen zugeschnitten ist. Um dieses Ziel zu erreichen, wurden Umfragen durchgeführt, um die Anforderungen der NutzerInnen zu ermitteln und die App entsprechend der Ergebnisse benutzerfreundlicher zu gestalten, um die aktive Teilnahme der NutzerInnen zu fördern. Die Ergebnisse der Umfrage werden in dieser Arbeit präsentiert, und die daraus resultierenden Änderungen an der App werden ausführlich erläutert.

Abstract

In the context of the energy transition and ongoing measures against climate change, it is of great importance to raise awareness among people about their individual CO2 consumption and to demonstrate ways to reduce their carbon footprint. The CO2-Reduction App, developed by the city of Karlsruhe as part of the "CO2-Reduction App"project, allows users to calculate their CO2 footprint and compare it with others. This research project is dedicated to the analysis of the CO2-Reduction App and the development of an enhanced version tailored to user needs. To achieve this goal, surveys were conducted to identify user requirements and make the app more user-friendly, thus encouraging active user participation. The survey results are presented in this paper, and the resulting changes to the app are elaborated upon.

Inhaltsverzeichnis

Abkürzungsverzeichnis

SDLC	Software Development Lifecycle
HCD	Human Centered Design
UCD	User Centered Design
API	Application Programming Interface
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
UI	User Interface
JS	JavaScript
TS	TypeScript
JSX	JavaScript XML
XML	Extensible Markup Language
SPA	Single Page Application
NPM	Node Package Manager
MUI	Material UI
NGINX	engine x
PWA	Progressive Web App
JSON	JavaScript Object Notation
SPA	Single Page Application

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

1.1 Einführung in das Thema

Der Klimawandel wird in vielen Teilen unserer Welt in den letzten Jahren deutlich sichtbar. Die Klimapolitik nimmt einen höheren Stellenwert in unserer Gesellschaft an, Länder und Nationen setzen sich ambitionierte Klimaziele zum Schutz der Erde, es bilden sich vermehrt Gruppierungen, die gegen den Klimawandel ankämpfen. Aber vor allem ist der Begriff Klimakrise heutzutage jedem bekannt. Viele Städte planen deshalb Maßnahmen, um ihre BürgerInnen über ihr Konsumverhalten aufmerksam zu machen und so zum Beispiel die Emissionsproduktion einzuschränken. Die Stadt Karlsruhe hat deshalb in Kooperation mit dem OK Lab Karlsruhe und der Dualen Hochschule Baden-Württemberg eine CO2-Runter-App veröffentlicht. Die Webseite befasst sich mit dem ökologischen Fußabdruck der BürgerInnen Karlsruhes und besteht im Grunde aus zwei Hauptseiten: dem CO2-Rechner und dem Dashboard. Da das Projekt bereits vor einigen Jahren gestartet wurde und zwischenzeitlich Verbesserungsvorschläge und verschiedenste Bugs und Fehler aufgetreten sind, soll die Arbeit an dem Projekt erneut aufgenommen werden.

1.2 Ziel und Fragestellung der Arbeit

Das Ziel dieser Studienarbeit liegt darin, die CO2-Runter-App der Stadt Karlsruhe weiterzuentwickeln. Die Weiterentwicklung der Applikation lässt sich hierbei in zwei große Themenblöcke aufteilen. Zum einen sollen vorhandene Bugs innerhalb der Applikation identifiziert und behoben werden. Da nicht alle Funktionen der Webseite einwandfrei funktionieren, ist eine umfassende Analyse der Applikation notwendig, um alle gefundenen Probleme zu lösen. Auf der anderen Seite soll bei der Weiterentwicklung der CO2-Runter-App die NutzerInnen mehr in den Vordergrund rücken. Mithilfe einer Umfrage sollen potenzielle NutzerInnen der Webseite zu verschiedenen Kategorien befragt werden. Unter Verwendung der erhobenen Daten soll die CO2-Runter-App so weiterentwickelt werden, dass möglichst viele Personen die Webseite nutzen und dazu angeregt werden, ihren CO2-Fußabdruck zu berechnen. Aus diesem Grund werden in der Studienarbeit und in der Umfrage folgende Fragestellungen im Vordergrund stehen:

- Wie können Anreize geschaffen werden, um Personen zur Reduktion ihres CO2-Fußabdrucks zu motivieren?
- Welche Strategien können eingesetzt werden, um Menschen zu einem umweltfreundlichen Denken hinsichtlich des Klimawandels zu bewegen?
- Welches Webseitendesign ist am effektivsten in der Steigerung des Interesses an klimafreundlichem Verhalten?

2 Aktueller Stand und Analyse der CO2-Runter-App

Im Rahmen dieser Studienarbeit steht die Erfassung des aktuellen Zustands und die Analyse der CO2-Runter-App im Fokus. Dies ist insbesondere wichtig, um vor Beginn unserer eigentlichen Arbeit eine umfassende Bestandsaufnahme vorzunehmen und die bestehenden Probleme zu identifizieren. Zunächst vorweg sollte geklärt werden, dass häufiger der Begriff *App* fallen wird. Dies soll nicht bedeuten, dass es sich hierbei um eine mobile Applikation (App) handelt. Eigentlich ist es nämlich eine Webseite und im Verlauf dieser Studienarbeit wird App und Webseite synonym verwendet. Die Startseite der CO2-Runter-App dient als Einstiegspunkt für alle NutzerInnen. Die Startseite spielt eine entscheidende Rolle, da sie den ersten Eindruck vermittelt und die NutzerInnen dazu anregen sollte, sich weiter mit der App zu beschäftigen. Da das Hauptziel der App darin besteht, die NutzerInnen dazu zu bewegen, ihren persönlichen CO2-Fußabdruck zu ermitteln, ist es von besonderer Bedeutung, dass die Webseite die NutzerInnen aktiv dazu motiviert, diese Aufgabe anzugehen. Die Abbildung 2.1 zeigt die Startseite der Webseite[1]. Bei einem ersten Blick auf die Webseite fallen mehrere Unstimmigkeiten auf. Die Dropdown-Felder wirken unverhältnismäßig lang und stören die Proportionen des Layouts. Die Webseite selbst erscheint unstrukturiert und wenig organisiert, was zu einer geringen Benutzerfreundlichkeit führt. Besonders auffällig ist der Mangel an klaren Handlungsanweisungen, die den NutzerInnenn zur Interaktion und aktiven Nutzung der App ermutigen sollen.

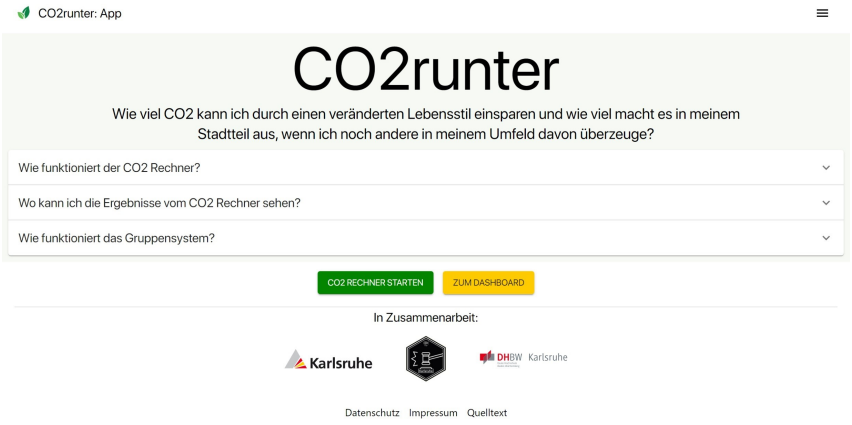

Abbildung 2.1: Startseite der CO2-Runter-App

[1] Die Startseite ist unter folgendem Link erreichbar: https://co2runter.karlsruhe.de/ [20]

Erst nach längerem Verweilen auf der Seite werden die Buttons *CO2 Rechner Starten* und *zum Dashboard* sichtbar. Dennoch bleibt unklar, was sich hinter dem Begriff *Dashboard* verbirgt und warum es für die NutzerInnen von Interesse sein sollte. Es fehlt eine klare Aufforderung, die NutzerInnen zur aktiven Teilnahme zu bewegen und sie dazu zu inspirieren, sich eingehender mit der Webseite zu beschäftigen.

Die ideale Startseite sollte den BesucherInnen eine übersichtliche und strukturierte Präsentation bieten und sie unmittelbar dazu motivieren, ihre persönlichen CO2-Fußabdruck zu ermitteln oder die Daten der Karlsruher Stadtteile zu sehen.

Wenn die NutzerInnen sich dazu entscheiden, den Button *CO2 Rechner Starten* zu betätigen, werden sie auf eine Seite weitergeleitet, die in Abbildung 2.2 dargestellt ist.

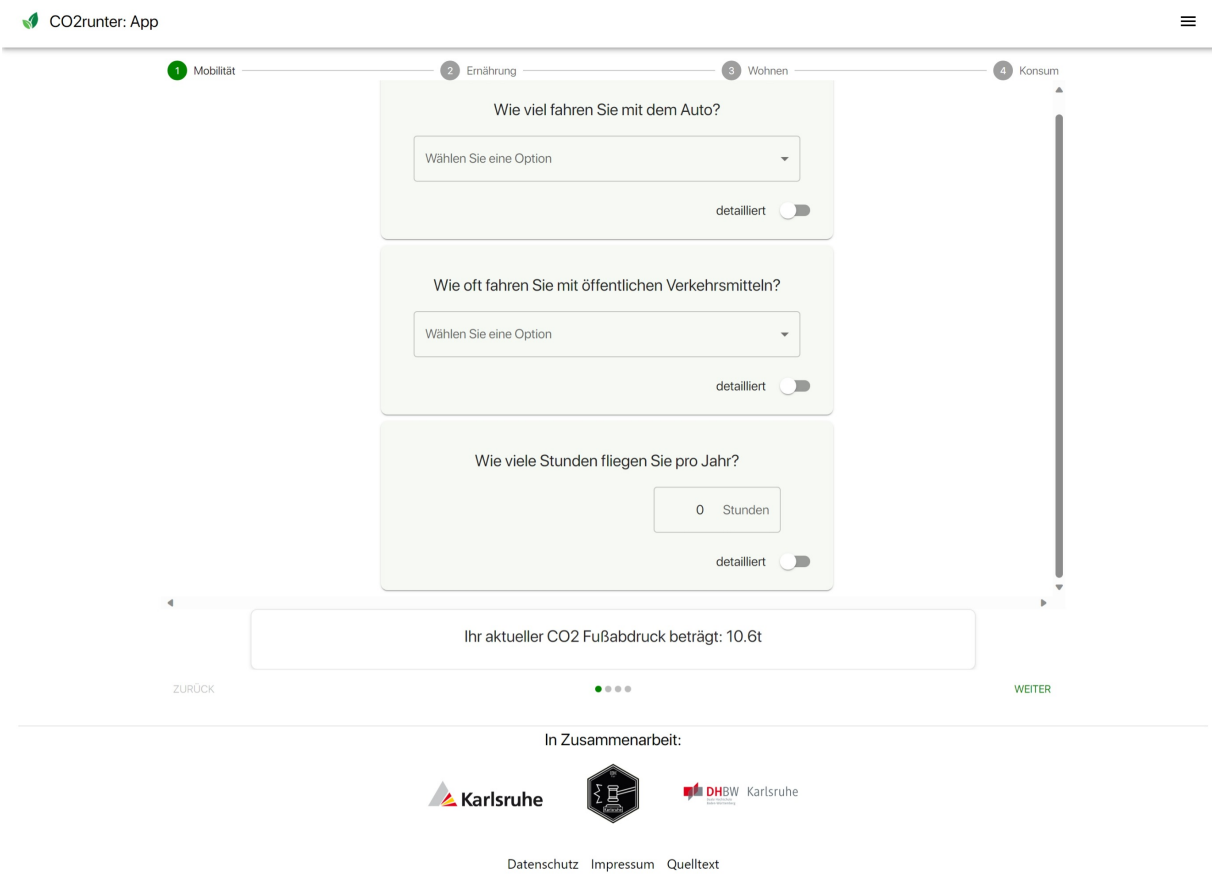

Abbildung 2.2: CO2 Rechner Formular der CO2-Runter-App

Auf dieser Seite wird den NutzerInnenn ein Formular präsentiert, das sie ausfüllen müssen, um ihren endgültigen CO2-Fußabdruck zu bestimmen. Anschließend können sie diese Daten einem Stadtteil von Karlsruhe zuordnen und versenden oder ohne Datenübertragung fortfahren, was sie direkt zum Dashboard führt. Die Möglichkeit, diese Entscheidung zu treffen, wird in Abbildung 2.3 illustriert.

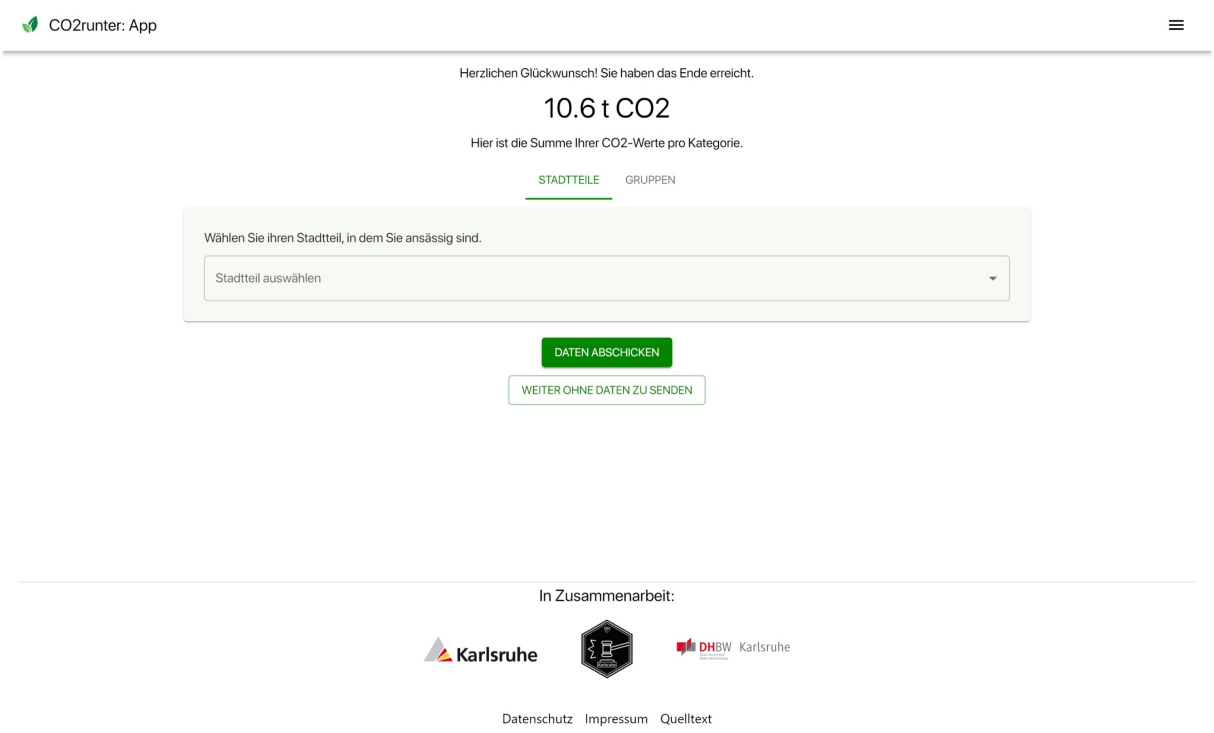

Abbildung 2.3: CO2 Rechner Formular Daten senden der CO2-Runter-App

Die letzte Hauptfunktion ist das Dashboard, das die übermittelten Nutzerdaten in Grafiken anzeigt. In der Kartengrafik werden die CO2-Emissionen der Stadtteile von Karlsruhe dargestellt. Diese kommen zustande, indem NutzerInnen beim Berechnen ihrer CO2-Bilanz angeben, aus welchem Stadtteil sie aus Karlsruhe kommen, wodurch sich diese Daten anpassen können. Neben der Kartengrafik gibt es noch die Charts, in denen der durchschnittliche Betrag folgenden Kategorien zugewiesen werden kann: Mobilität, Wohnen, Konsum, Ernährung und Infrastruktur. Im Gruppen-Tab sieht man wiederum, welche Gruppe wie viel CO2 verbraucht hat, und im letzten Tab, wie der Name schon verrät, wird ein Balkendiagramm angezeigt, welches die Beteiligung der NutzerInnen und in welchem Stadtteil diese wohnen. Die Abbildung 2.4 zeigt die Kartengrafik des Dashboards und die jeweilige Menge an CO2, die den Stadtteilen zugeordnet wurde.

Insgesamt bietet die CO2-Runter-App eine Plattform zur Berechnung und zum Vergleich des CO2-Fußabdrucks, wobei der Fokus auf einer verbesserten Benutzerfreundlichkeit und klaren Handlungsanweisungen liegen sollte, um die aktive Beteiligung der NutzerInnen zu fördern. Die Struktur und Anordnungen der Unterseiten sehen im Vergleich zur Startseite wesentlich strukturierter und organisierter aus, und als NutzerIn weiß man eher, was zu tun ist.

Im nächsten Kapitel werden einige Grundlagen und Grundstrukturen definiert und erklärt, auf die diese Studienarbeit aufgebaut und das Projekt CO2-Runter weiterentwickelt wird. Dabei geht es sowohl um Prozesse, die nichts mit den eigentlichen Programmieraufgaben

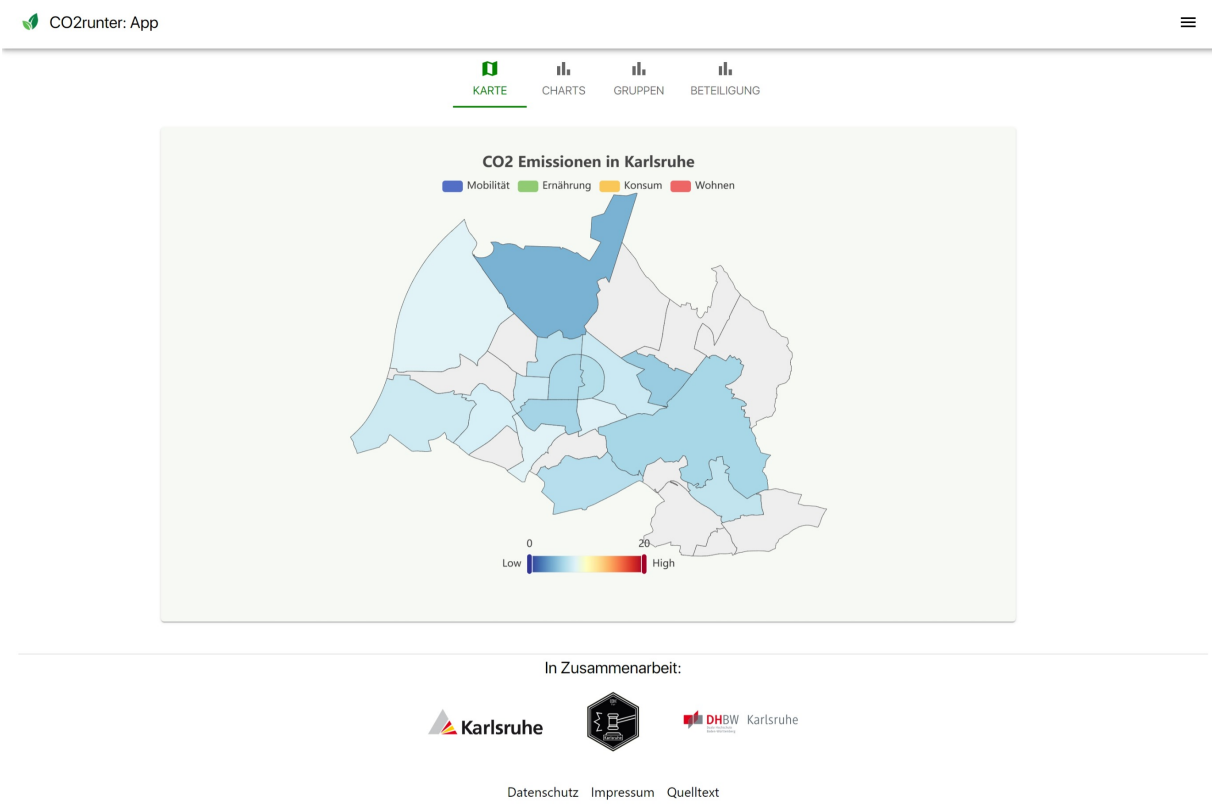

Abbildung 2.4: CO2 Dashboard der CO2-Runter-App

zu tun haben. Auf der anderen Seite werden aber auch wichtige Programmierbegriffe, Frameworks und Werkzeuge erklärt, die im Laufe der Studienarbeit und des Projekts eine Rolle spielen werden.

3 Grundlagen

Bevor mit der eigentlichen Arbeit der Studienarbeit begonnen wird, sollten wichtige Grundstrukturen geklärt werden. Das folgende Kapitel stellt die wichtigsten Grundstrukturen und Prinzipien vor, die im Laufe dieser Studienarbeit relevant werden. Es werden Konzepte zur Erstellung und Umsetzung der Umfrage sowie zu Programmiertechnologien geboten.

3.1 Humanzentriertes Design

Beim Entwickeln neuer Software setzen erfahrene SoftwareentwicklerInnen häufig auf einen gut durchdachten Plan und Struktur. Dabei wird auf bestimmte Entwicklungsprozesse gesetzt. Einer der bekanntesten Prozesse in der Softwareentwicklung ist hierbei der "Software Development Lifecycle (SDLC)" (kurz: SDLC). Der SDLC wird genutzt, um möglichst effiziente, kostengünstige Software mit hoher Qualität zu designen, entwickeln und zu produzieren.[36] Es gibt eine Menge verschiedene Modelle des SDLC, jedoch haben alle Modelle im Grunde dieselbe Struktur: Planen - Design - Implementieren - Testen. Besonders auf den Aspekt "Design"wird in diesem Kapitel genauer eingegangen.

Eine der weitverbreitetsten Designmodelle ist das "Human Centered Design (HCD)" (kurz: HCD, dt.: menschenzentriertes Design). Dabei handelt es sich um eine Designtechnik, bei der der Mensch im Vordergrund des Entwicklungsprozesses steht. Laut der Webseite der Harvard Business School liegen die Ziele des HCD darin, die Ziele, Wünsche und Vorlieben des Produktnutzers stetig im Auge zu behalten. Dort beschreibt ein Dekan der Harvard Business School vier grundlegende Phasen im HCD. Diese sind laut Dekan Srikant Datar folgende Phasen: Clarify - Ideate - Develop - Implement. Wiederum definiert Sim van der Ryn in seinem Paper Ḧuman Centered Designäus dem Jahr 2013 das Vorgehen des HCD ein wenig anders, jedoch ähnlich. Dieser schreibt, dass man sich zuerst mit den potenziellen NutzerInnen beschäftigen sollte. Anschließend wird das Problem definiert, eine Idee erarbeitet, ein Prototyp erstellt und abschließend getestet.[21, 35]

Zu den Vorteilen des HCD gehört unter anderem eine hohe Nutzerzufriedenheit, da die Meinung der Personen direkten Einfluss auf das Design des Produkts hat und somit sichergestellt wird, dass alle Bedürfnisse und Erwartungen an das Produkt erfüllt werden [38]. Zudem kann dadurch die Wechselwirkung zwischen Menschen und Objekten besser

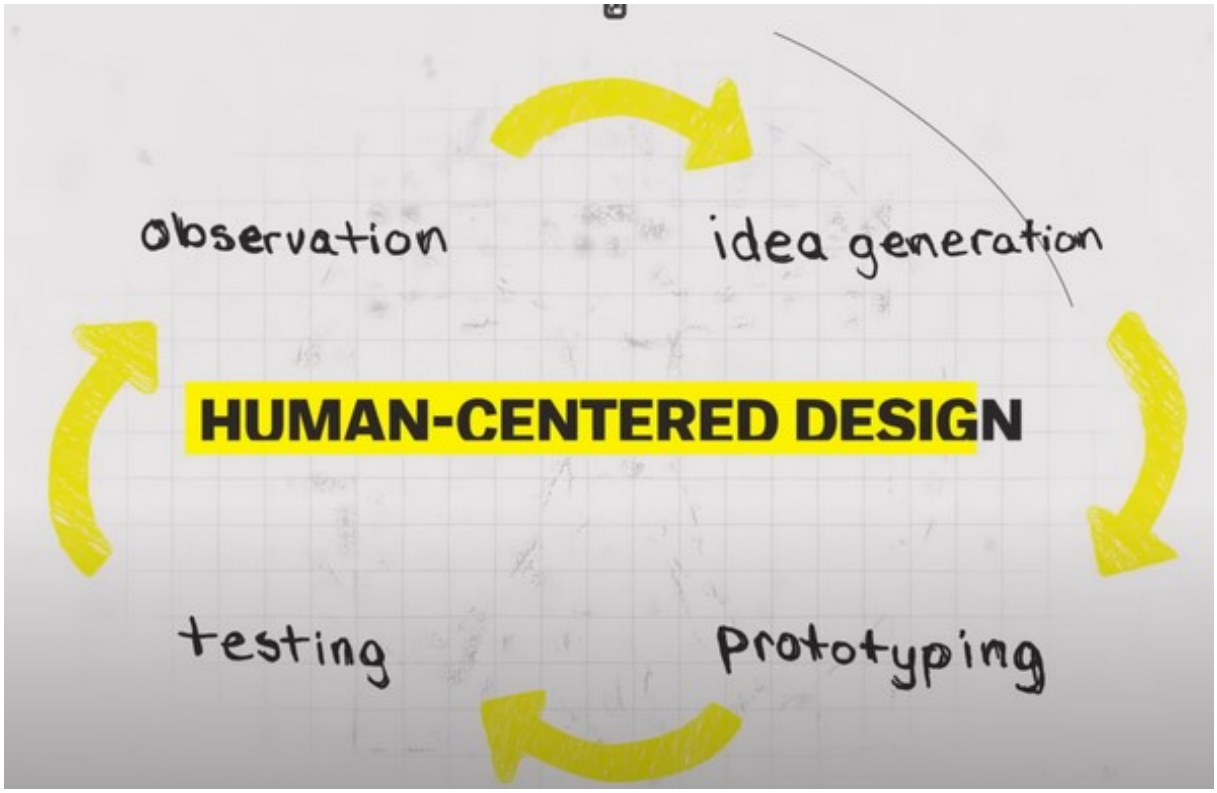

Abbildung 3.1: Humand Centered Design Prozess

verstanden werden, da man durch die Designfrage wichtige Erkenntnisse über das Verhalten bzw. Bedürfnisse der Menschen oder zumindest einer Personengruppe bekommt. [37]

Zu den Vorteilen des HCD gehören unter anderem eine hohe Nutzerzufriedenheit, da die Meinung von Personen unmittelbaren Einfluss auf das Design des Produkts nimmt und so sicherstellt, dass alle Bedürfnisse und Erwartungen an das Produkt garantiert werden.[38] Zudem kann dadurch die Wechselwirkung zwischen Menschen und Objekten besser verstanden werden, da man durch die Designfrage wichtige Erkenntnisse über das Verhalten bzw. die Bedürfnisse der Menschen oder zumindest einer Personengruppe bekommt.[37]

Allerdings gibt es auch einige Nachteile bzw. Probleme des HCD. Ein Aspekt wäre der rapide Produktlebenszyklus. Durch die ständig wechselnden Bedürfnisse und Anforderungen an ein Produkt stehen DesignerInnen und Designteams ständig unter großen Herausforderungen, um die Ansprüche treffen zu können. Häufig könnten aber auch eben diese menschlichen Anforderungen an ein Produkt zum Problem werden, da diese Anforderungen nicht umsetzbar und realisierbar sind.[18]

3.2 Nutzerzentriertes Design

Wenn man sich beim Entwickeln neuer Software besonders auf die NutzerInnen konzentriert und diese nach ihren Wünschen und Anregungen gestaltet, spricht man von Üser-Centered Design(UCD). Ein wesentlicher Unterschied zum Humanzentrierten Design (HCD) besteht darin, dass das Nutzerzentrierte Design soziale und kulturelle Aspekte nicht in das Design einfließen lässt. [16] Diese Aspekte spielen beim Humanzentrierten Design eine wesentliche Rolle. Dennoch wird UCD heutzutage oft mit HCD gleichgesetzt. [25]

Ein zentraler Aspekt des UCD sind sogenannte Personas. Laut dem Buch "Persona Design in Participatory Agile Software Development"werden Personas häufig als Methode im UCD genutzt, um fiktionale Charaktere zu erschaffen, die verschiedene NutzerInnengruppen repräsentieren. [33] Eine Persona wird dabei meistens in Form einer Erzählung beschrieben. [25] Das Ziel besteht darin, die Persona wie eine reale Person darzustellen und ihre Bedürfnisse anschaulich innerhalb einer Geschichte zu präsentieren, um sie in die Entwicklung des Produkts einzubeziehen. [25]

Grundsätzlich beginnt eine solche Erzählung mit einer Beschreibung der Person. [25] Während der Beschreibung werden wichtige charakteristische Merkmale der Persona wie Vorlieben, Abneigungen, Beruf usw. hervorgehoben. [25]

3.2.1 Erstellen einer Persona

Die Erstellung einer Persona ist ein wesentlicher Bestandteil des UCD. In einem wissenschaftlichen Paper der Universität Rostock wird der Prozess der Persona-Erstellung beschrieben. Der gesamte Entwicklungsprozess einer Persona ist in der folgenden Abbildung dargestellt:

Zu sehen ist, dass nahezu alle Schritte, welche vor der Erstellung von Personas passieren, mit Daten zutun haben. Besonders das Sammelen und Kategorisieren der Daten wird hier thematisiert.[19] Dazu schreibt der Wissenschaftler Frank Ploß in seiner Diplomarbeit Üsability Engineering in einem Open-Source-Projekt", welches ebenfalls von der Universität Rostock zitiert wird, dass es sich bei den eizenlenen Designstufen um ein iteratives Modell handelt.[34] Das bedeutet im Grunde genommen nur, dass die einzelnen Schritte aufeinandner aufbauend sind. In dem Artikel wird zudem erwähnt, dass bereits im Vorfeld der Personaentwicklung Fragen über die Zielgruppe gestellt werden sollen, um die Erschaffung einer Persona zu vereinfachen.[19] Dieser Schritt wird im Rahmen der Studienarbeit ausführlicher innerhalb einer Umfrage durchgeführt.

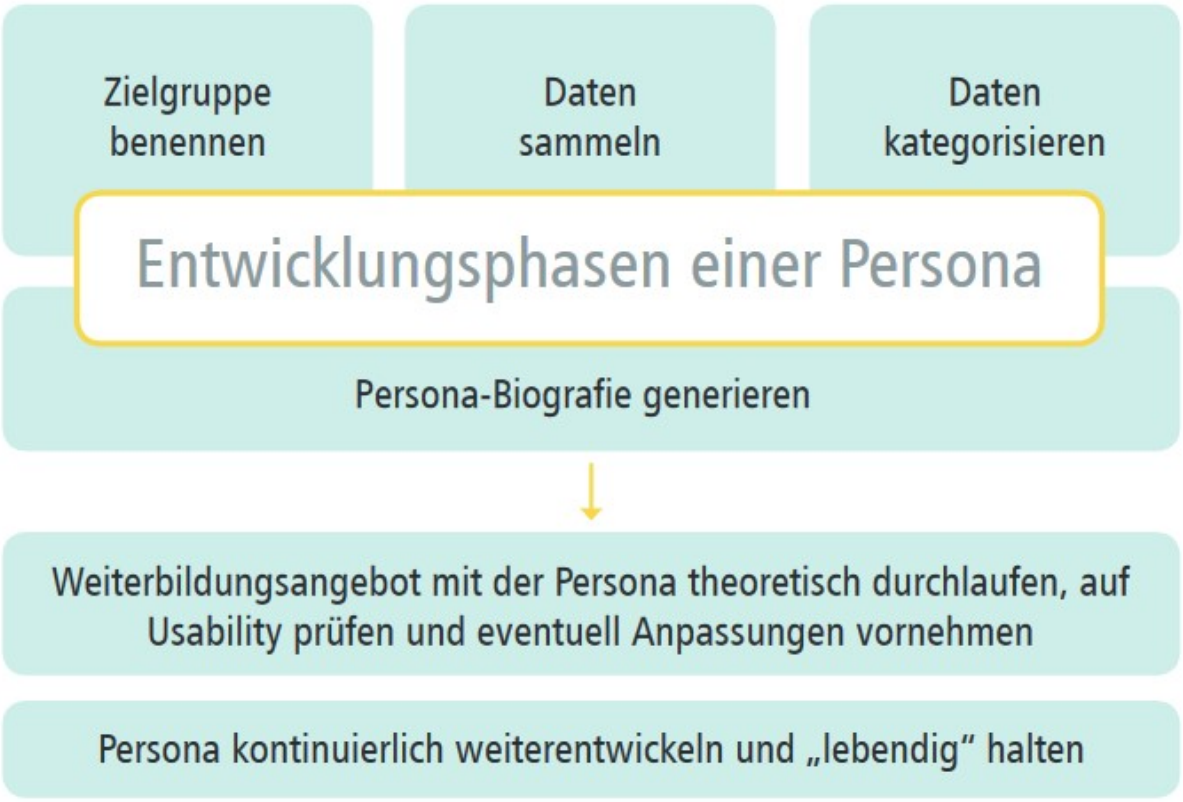

Abbildung 3.2: Entwicklungsprozess der Persona [19]

Nach dem Sammeln und Kategorisieren der Daten wird die Persona definiert.[19] In diesem Schritt wird laut dem wissenschaftlichen Artikel der Universität Rostock eine Art Kurzbiografie für jede Persona erschaffen.[19] Als Beispiel für eine Persona in der Praxis verwendet die Universität Rostock deshalb Steckbriefe, um ihre Personas so kurz wie möglich, mit möglichst vielen Informationen darzustellen.[19]

Zum Abschluss sollten die Personas noch einen so genanntes Weiterbildungsangebot durchlaufen. Das bedeutet, dass man sich beispielsweise in die Sicht eines Kunden versetzt und als die gerade eben erstellte Persona den Einkaufsprozess durchläuft.[19] Dieser Schritt hilft dabei, die Persona noch stärker zu definieren und evtl. Schwächen/Stärken herauszufinden.

3.3 Einführung in die Webentwicklung

Die Webentwicklung ist ein sehr großer Bereich der Softwareentwicklung und umfasst die Entwicklung von Webanwendungen, die auf den aktuellsten Technologien, Frameworks und Tools basieren. Die Schaffung zeitgemäßer Webanwendungen beinhaltet verschiedene Aspekte, von der Frontend-Entwicklung bis zur Backend-Entwicklung und der Nutzung

von Application Programming Interface (API)s und Datenbanken, welche alle miteinander interagieren und kommunizieren müssen.

Der Standard für die Webentwicklung ist die Verwendung von JavaScript (JS), Hyper Text Markup Language (HTML) und Cascading Style Sheets (CSS). Diese Technologien sind die absoluten Grundlagen für die Entwicklung von Webanwendungen, auf denen alles Weitere aufbaut.

Die Frontend-Entwicklung von Webanwendungen umfasst hierbei die Verwendung von Frameworks wie **React.js**, **Angular** und **Vue**. Diese wiederum basieren auf der Programmiersprache JavaScript und ermöglichen die Erstellung von wiederverwendbaren Komponenten, auch bekannt als Komponentenarchitektur. Die Verwendung von Frameworks ermöglicht darüber hinaus auch die Anwendung moderner Entwicklungspraktiken wie Komponententests, Test-Driven-Development und Continuous Integration. Außerdem ermöglichen Frameworks den EntwicklerInnenn, die Entwicklungseffizienz zu verbessern und die Wartbarkeit der Anwendung zu erhöhen. Die zuvor angesprochenen Frameworks sind nur einige der vielen JavaScript Frameworks, die existieren. Es gibt eine Vielzahl von Frameworks, die alle ihre eigenen Stärken und Schwächen haben. Die Wahl des richtigen Frameworks hängt von den Anforderungen des Projekts ab und sollte sorgfältig abgewogen werden, um die beste Benutzererfahrung zu gewährleisten.[17, 44]

3.3.1 Komponentenarchitektur

Wie zuvor angesprochen ist die Komponentenarchitektur ein grundlegendes Konzept in der Webentwicklung, das darauf abzielt, die Benutzeroberfläche in wiederverwendbare und modulare Komponenten zu strukturieren. Diese Komponenten können eigenständige Teile der Benutzeroberfläche sein, die bestimmte Funktionen oder Darstellungen bereitstellen. Durch die Verwendung einer Komponentenarchitektur können EntwicklerInnen die Komplexität von Webanwendungen reduzieren, den Entwicklungsprozess beschleunigen und die Wartbarkeit verbessern.

Die Komponentenarchitektur zielt darauf ab, diverse Prinzipien zu erfüllen. In der folgenden Auflistung sind einige der wichtigsten Prinzipien aufgeführt:

- **Wiederverwendbarkeit:** Komponenten können in verschiedenen Teilen einer Anwendung wiederverwendet werden, was die Entwicklung beschleunigt und den Code reduziert.
- **Modularität:** Die Benutzeroberfläche wird in unabhängige Komponenten aufgeteilt, die jeweils eine spezifische Aufgabe erfüllen. Diese Komponenten können dann miteinander kombiniert werden, um komplexe Benutzeroberflächen zu erstellen.

- **Klare Trennung von Anliegen:** Jede Komponente kann ihre eigene interne Logik und ihr eigenes Styling haben, was eine klare Trennung von Anliegen ermöglicht.
- **Wartbarkeit:** Die Wiederverwendbarkeit und Modularität der Komponenten erleichtern die Wartung und Aktualisierung von Anwendungen, da Änderungen an einer Komponente automatisch in allen Instanzen dieser Komponente wirksam werden.
- **Komposition:** Durch die Kombination mehrerer Komponenten können komplexe Benutzeroberflächen erstellt werden, die aus mehreren modularen Teilen bestehen. Diese Komposition ermöglicht es EntwicklerInnenn, die Benutzeroberfläche schrittweise aufzubauen und sie bei Bedarf zu erweitern oder anzupassen.

Diese Architektur ermöglicht es EntwicklerInnenn, die Benutzeroberfläche in kleinere, wiederverwendbare Teile zu zerlegen, die unabhängig voneinander entwickelt, getestet und gewartet werden können. Dies führt zu einer verbesserten Entwicklungseffizienz und einer höheren Codequalität, da Entwickler sich auf die Entwicklung und Wartung einzelner Komponenten konzentrieren können, anstatt sich mit der gesamten Benutzeroberfläche auseinandersetzen zu müssen.[22, 32]

3.3.2 Programmiersprache JavaScript

JavaScript ist eine interpretierte Programmiersprache, welche hauptsächlich Anwendung bei Webanwendungen findet. Sie wird dafür genutzt, um Interaktionen als auch die dynamische Veränderung auf der Webseite zu ermöglichen. Dadurch kommt JavaScript größtenteils im Browser vor, also auf der Seite des Clients. JavaScript kann neben dem sehr verbreiteten objektorientierten Programmierstil auch als prozedurale Programmiersprache eingesetzt werden, dadurch ist JavaScript sehr flexibel, was einerseits vorteilhaft ist, aber auch zum Nachteil führt, da es keine konkreten Richtlinien gibt, woran man sich halten sollte. Aber JavaScript ist nicht nur im Programmierstil sehr flexibel, denn dadurch, dass es nur eine interpretierte Programmiersprache ist, besitzt es weitere Nachteile gegenüber von kompilierten Sprachen.

Einer der größten Nachteile hierbei ist, dass es bei JavaScript keine direkte Typisierung gibt, die im Vorhinein im Quellcode festgelegt wird, wie bei typisierten Programmiersprachen wie z. B. Java. Dadurch können theoretisch jegliche Art von Daten in einer Variable gespeichert werden. Das führt dazu, dass man während des Entwicklungsprozesses einfache Fehler nicht feststellen kann, sondern erst beim Ausführen der Software. Bei Programmiersprachen, welche typisiert sind, kann dieses Problem schon im Vorhinein festgestellt werden. Dadurch können einfache Fehler wie z. B. das Datentypen nicht miteinander übereinstimmen, ob auf diese Variable in diesem Kontext zugegriffen werden kann oder dass eine Variable

nicht existiert, bereits im Vorhinein festgestellt werden. Dazu kommt, dass neben den nicht typisierbaren Variablen in JavaScript die Eingabe- und Ausgabeparameter einer Funktion nicht typisieren kann. Diese Tatsache führt häufig zu Fehlern, welche erst im laufenden Betrieb der Software festgestellt werden können. Dadurch wird JavaScript-Code sehr schwer wartbar, da EntwicklerInnen beim Lesen des Quellcodes nicht genau wissen können, was in einer Variable gespeichert ist und was die Funktionen genau machen. Um den Quellcode letztlich korrekt zu verstehen, ist es nötig, sich intensiv einzuarbeiten, was sowohl viel Zeit, als auch Geld kostet.

Im Folgenden ist ein kleines Beispiel, wie sich JavaScript von Java unterscheidet hinsichtlich der Typisierung von der Deklaration von Variablen und Funktionen.

```
1    // Deklaration von Typisierten Variablen
2    int name = 1;
3    char name = "c";
4    String name = "John";
5
6    // Deklaration von Typisierten Funktionen
7    private int add(int a, int b) {
8        return a + b;
9    }
```

Listing 3.1: Java Variablen und Funktion deklarations Beispiel

```
1    // Deklaration von Variablen
2    let a = 1;
3    let b = "c";
4    let c = "John";
5
6    // Deklaration von Funktionen
7    function add(a, b) {
8        return a + b;
9    }
```

Listing 3.2: JavaScript Variablen und Funktion deklarations Beispiel

Grundsätzlich ist es in JavaScript nicht möglich direkt zu wissen, was in einer Variable sein kann. Schon bei einfachen Funktionen wie einer Additionsfunktion wird in der JavaScript-Variante nicht direkt deutlich, dass eine Zahl zurückkommen soll. Denn es wäre auch möglich einen zusammengefügten String zurückzugeben. Bei Java hingegen wird schnell deutlich, was genau Funktionen machen sollen. Dennoch hat JavaScript seine Vorteile, welcher hauptsächlich in der Erlernbarkeit der Sprache liegt. Ein einfaches „Hallo Welt“ Beispiel ist zügig geschrieben. Wenn man das z. B. mit Java vergleicht, gibt es wesentlich mehr zu verstehen, als auch zu schreiben.

```java
public class HelloWorld {
  public static void main (String[] args) {
    System.out.println("Hello World!");
  }
}
```

Listing 3.3: „Hallo Welt“ Beispiel in Java

```javascript
    console.log("Hallo Welt");
```

Listing 3.4: „Hallo Welt“ Beispiel in JavaScript

Wie schon zuvor erwähnt, ist die objektorientierte Programmierung sehr verbreitet und dadurch auch ein Standard. JavaScript hingegen verwendet eine sehr abstrakte Implementation von Objekten, was das objektorientierte Programmieren erschwert. Um jedoch in JavaScript objektorientiert zu programmieren, wird die Flexibilität der Sprache genutzt, um sich eine eigene Struktur zu bauen. Durch diese abstrakte Implementation von Objekten, ist jede Variable, egal ob String, Zahl oder Funktion ein Objekt. Jedoch existieren Konzepte wie Klassen, Vererbung und Zugriffsbeschränkung nicht direkt. Über Umwege können Klassen und Vererbungen mittlerweile realisiert werden.

JavaScript ist in dieser Arbeit von Relevanz, da jedes Frontend Framework welche in späteren Abschnitten angesprochen wird darauf basiert und diverse Konzepte von JavaScript verwendet. Zudem ist JavaScript die einzige Programmiersprache, welche im Browser ausgeführt werden kann. Das bedeutet, dass JavaScript die einzige Programmiersprache ist, welche direkt mit den NutzerInnen interagieren kann. Das ist ein großer Vorteil, da dadurch die Interaktion mit den NutzerInnen direkt im Browser stattfinden kann, ohne dass der Browser die Webseite neu laden muss. Dadurch können Webanwendungen wie die CO2-Runter-App eine bessere Nutzererfahrung bieten, da die Interaktion mit den NutzerInnen schneller und flüssiger ist. Eine detailliertere Beschreibung von JavaScript findet man auf der Mozilla Developer Network JavaScript Webseite.[29]

3.3.3 JavaScript XML

JavaScript XML, auch bekannt als JSX, erweitert die Syntax um Extensible Markup Language (XML). Dies ermöglicht das Schreiben und Verwenden von Code, der HTML ähnelt, innerhalb einer JavaScript-Datei.

Im vorherigen Abschnitt wurde auf die Programmiersprache JavaScript und deren wichtige Verwendung im Webumfeld eingegangen. Früher war es typisch, den Logikteil vom

Darstellungsteil getrennt zu programmieren und zu speichern, wie in der folgenden Grafik dargestellt:

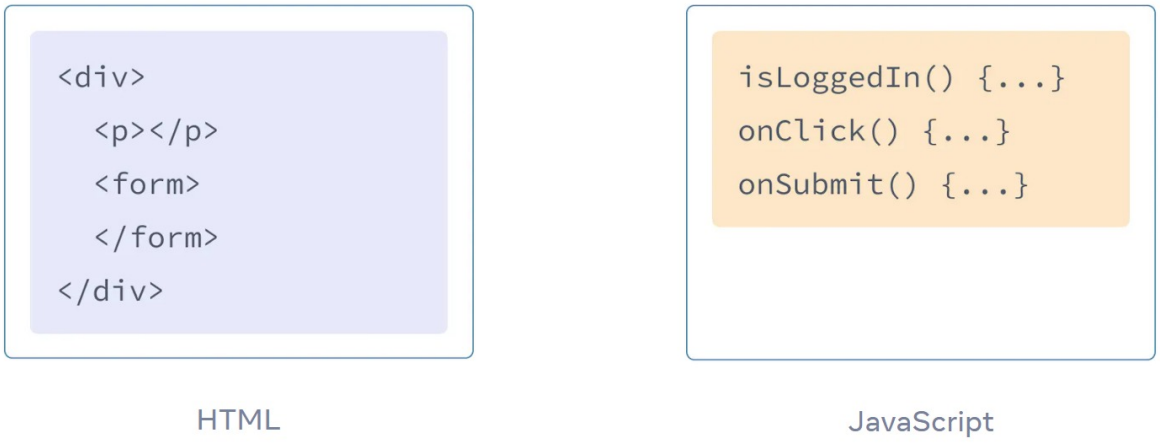

Abbildung 3.3: Einfache Grafik von Darstellung und Logik bei Webseiten[11, 12]

Doch aufgrund der zunehmenden Interaktivität und der wachsenden Größe von Webseiten wird heutzutage vermehrt JavaScript und weniger HTML und CSS verwendet. Dies hat dazu geführt, dass die Grenzen zwischen Logik und Darstellung verschwimmen. Dadurch ist die Trennung zwischen Logik und Darstellung in JSX nicht mehr so klar auseinanderzusetzen. Wie in der folgenden Grafik vereinfacht dargestellt:

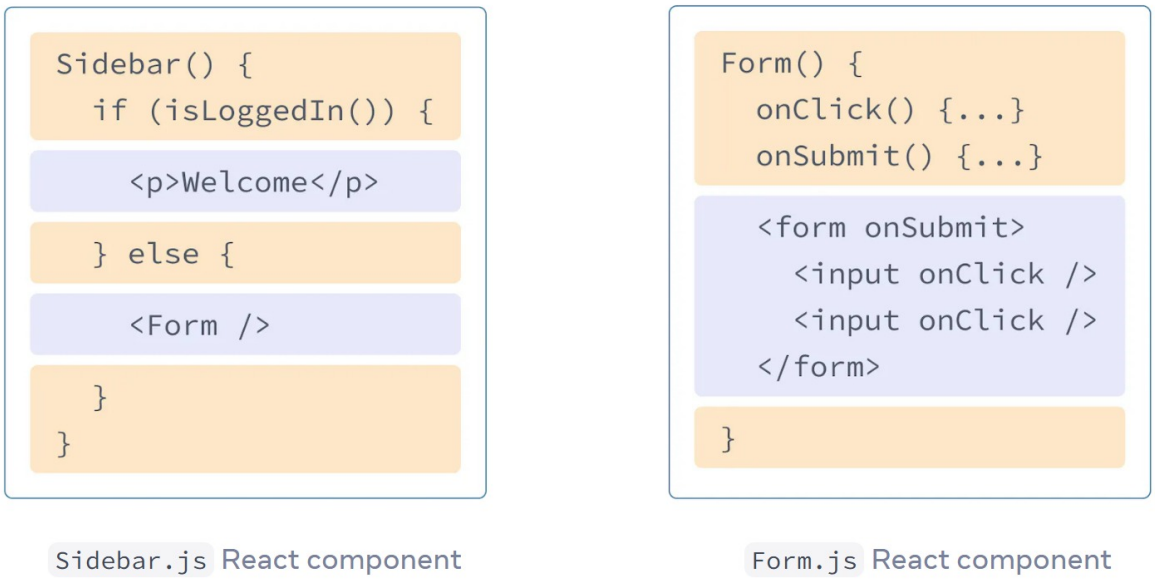

Abbildung 3.4: JSX Grafik von Darstellung und Logikverschmelzung[10, 13]

Dadurch kann die Darstellung und Logik von zusammenhängenden Komponenten beieinander sein, erhöht das die Möglichkeit, dass der Code aktuell bleibt, und Code oder Komponenten, die wiederum nichts miteinander zu tun haben, sind separiert durch verschiedene JavaScript-Dateien. Somit können EntwicklerInnen besser strukturierten Code

schreiben, der sowohl die Logik als auch die Darstellung umfasst. Dieser Ansatz bietet eine nahtlose Integration von JavaScript-Funktionalitäten in die Gestaltung von Benutzeroberflächen. Anstatt verschiedene Dateien für HTML, CSS und JavaScript zu haben, ermöglicht JSX das Verfassen von Komponenten, die sowohl Struktur als auch Verhalten vereinen. Dadurch entsteht eine vereinfachte, effizientere Entwicklung von Webanwendungen.

Die Einführung von JSX hat die Art und Weise, wie Webentwicklung betrieben wird, maßgeblich verändert, und sie spielt eine bedeutende Rolle in Frameworks wie **React.js**, **Vue** und anderen modernen Bibliotheken zur Entwicklung von Benutzeroberflächen. Diese Entwicklung hat dazu geführt, dass EntwicklerInnen leistungsstarke, interaktive Webanwendungen schneller und effektiver erstellen können.[9]

3.3.4 TypeScript superset JavaScript

TypeScript (TS) ist eine von Microsoft entwickelte Programmiersprache, die eine typisierte Erweiterung von JavaScript darstellt. Es fügt statische Typen hinzu, was die Code-Qualität verbessert und EntwicklerInnenn ermöglicht, robustere und wartbarere Software zu erstellen. [26]

Schlüsselmerkmale von TypeScript

1. **Statische Typisierung:** TypeScript fügt statische Typisierung hinzu, die es EntwicklerInnenn ermöglicht, Typen für Variablen, Parameter und Rückgabetypen von Funktionen festzulegen. Dies hilft, Fehler frühzeitig im Entwicklungsprozess zu erkennen.

2. **Verbesserte Code-Intelligenz:** Durch die Verwendung von TypeScript profitiert man von besserer Code-Intelligenz in Entwicklungsumgebungen, was die Produktivität steigert. Intelligente Autovervollständigung und detaillierte Fehlermeldungen sind einige der Vorteile.

3. **Interfaces und Typen:** TypeScript ermöglicht die Definition von benutzerdefinierten Typen und Schnittstellen, was zu einer klaren und strukturierten Codebasis führt. Dies erleichtert die Zusammenarbeit im Team und das Verständnis des Codes.

4. **ES6-Unterstützung:** TypeScript unterstützt ECMAScript 6 und darüber hinaus, was bedeutet, dass EntwicklerInnen die neuesten Funktionen von JavaScript nutzen können.

5. **Bessere Wartbarkeit und Skalierbarkeit:** Die Verwendung von Typen trägt dazu bei, Bugs zu reduzieren und die Wartbarkeit von Codebasen zu verbessern. Dies ist besonders wichtig für größere Projekte.

3.3.5 Moderne Frontend-Frameworks

Die Entwicklung von Webanwendungen hat sich in den letzten Jahren stark weiterentwickelt, und moderne Frontend-Frameworks spielen dabei eine entscheidende Rolle. Diese Frameworks erleichtern nicht nur die Erstellung von ansprechenden und interaktiven Benutzeroberflächen, sondern bieten auch viele Funktionen und Werkzeuge, die die Entwicklung effizienter und produktiver machen.

- **Angular:** Angular ist ein von Google entwickeltes TypeScript-basierendes Framework. Es bietet eine umfangreiche Sammlung von Werkzeugen und Bibliotheken, die EntwicklerInnenn helfen, dynamische Single Page Application (SPA) zu erstellen. Die Struktur von Angular erleichtert die Organisation des Codes und bietet eine klare Trennung von Logik und Darstellung.[14]
- **Vue:** Vue.js ist ein progressives JavaScript-Framework, das sich durch seine leichte Integration, Flexibilität und eine sanfte Lernkurve auszeichnet. Es ermöglicht die Erstellung interaktiver Benutzeroberflächen und wird von einer aktiven Community unterstützt, die ständig zur Weiterentwicklung und Verbesserung beiträgt.[7]
- **React.js:** React.js, entwickelt von Facebook, ist eine beliebte JavaScript-Bibliothek, die sich auf die Erstellung wiederverwendbarer User Interface (UI)-Komponenten konzentriert. Durch die Verwendung von JavaScript XML (JSX) können EntwicklerInnen eine hierarchische Struktur aufbauen, die zur Erstellung dynamischer Benutzeroberflächen beiträgt.[8]

Parameter	**Angular**	**React.js**	**Vue**
Unterstützung	Google	Facebook	Community
Typ	Framework	Bibliothek	Framework
Größe	Mittel	Klein	Sehr klein
Sprache	TypeScript	JavaScript	JavaScript
Leistung	Gut	Gut	Gut

Die Fortsetzung erfolgt auf der nachfolgenden Seite

Parameter	**Angular**	**React.js**	**Vue**
Datenbindung	Beide	Einfach	Einfach
Lernkurve	Steil	Einfach	Einfach
Nutzung	2.	1.	3.
Entwicklungsgeschwindigkeit	Mittel	Relativ kurz	Schnell
Am besten geeignet für	PWA, SPA	E-Commerce	Schnelle Entwicklung

Tabelle 3.1: Vergleich von Angular, Vue und React.js [4, 28]

Diese Frameworks haben jeweils ihre eigenen Stärken und eignen sich für unterschiedliche Projekte je nach Anforderungen, Teampräferenzen und Zielen.[15, 30] Bei allen Frameworks steht eine Frontendtechnologie besonders im Vordergrund: Single-Page-Applications (SPA). Eine SPA hat den Hintergedanken, dass die gesamte Anwendung auf nur einer Webseite dargstellt wird und läuft. [6] Bei einer SPA wird die gesamte Präsentationsschicht vom Browser durchgeführt. Dabei übernimmt der Client bzw. der Browser die Aufgabe, HTML-Code und Daten zu vereinen. [6] Man kann sich das ganze in folgendem Beispiel vorstellen: Eine NutzerIn hat eine Webseite vor Augen. Auf der Webseite ist ein Button, der beim Drücken Daten anfordert und diese auf einer Webseite anzeigen soll. Anstatt die Daten von einem Server zu holen, auf einer Seite einzubauen und die gesamte Webseite neu zu laden, nutzen SPA's dynamisches Rendering. Es wird also vereinfacht gesagt, platz auf der bereits vorhandenen Webseite gemacht, um den neuen Inhalt (die Daten) dynamisch ohne alles neu zu laden, anzuzeigen. Die NutzerIn merkt aufgrund des dynamischen Einfügens häufig gar nicht, dass im Hintergrund etwas passiert. Dieses Vorgehen nennt man in der Fachsprache auch **clientseitiges Rendering**.

Die CO2-Runter-App wurde ursprünglich mit **React.js** entwickelt. Allerdings fiel die Entscheidung, das Framework zu wechseln, da die EntwicklerInnen mit **React.js** nicht zufrieden waren und auf diverse Probleme stießen. Als Alternative wurde **Vue** ausgewählt, da es eine leicht verständliche Lernkurve aufweist und die EntwicklerInnen bereits Erfahrung damit hatten. Zudem zeichnet sich **Vue** durch hohe Flexibilität aus und lässt sich gut mit anderen Bibliotheken und Frameworks kombinieren. Die Entscheidung für **Vue** wurde auch durch seine Performance und umfassende Dokumentation beeinflusst, was die Entwicklung erleichtert.

3.4 Werkzeuge

Bei der Weiterentwicklung der CO2-Runter-App kommen zahlreiche Tools zum Einsatz, um die Entwicklungsumgebung zu verbessern und die Entwicklung zu vereinfachen. Da jedoch einige Tools auf das Backend und die Datenbank bezogen sind, werden diese hier außen vor gelassen. Es wird ausschließlich auf diejenigen eingegangen, die für die Frontend-Entwicklung relevant sind. Diese Tools sind wichtig für die Weiterentwicklung der CO2-Runter-App und spielen auch für das nutzerzentrierte Design eine Rolle.

3.4.1 NPM - Node Package Manager

NPM ist die Abkürzung für Node Package Manager (NPM). Wie der Name schon sagt, ist NPM ein Paketmanager, der bei der Installation von Node.js mitgeliefert wird. Er ermöglicht somit die Installation und Verwaltung von JavaScript-Paketen.

Ein Paketmanager ist eine Software, die es ermöglicht, sogenannte Pakete herunterzuladen und in ein Projekt zu importieren. Unter einem Paket versteht man ein Projekt, das von EntwicklerInnen bereitgestellt wurde, um von anderen EntwicklerInnenn in deren Projekten eingebunden zu werden. Auf diese Weise können EntwicklerInnen ihre Arbeit miteinander teilen und Zeit sparen, indem sie bereits vorhandene Projekte verwenden, anstatt alles von Grund auf neu entwickeln zu müssen.

Für jeden Paketmanager gibt es eine Paketzentrale oder auch Registry genannt. In der Registry von NPM sind über 800.000 Pakete/Projekte zu finden. NPM ermöglicht außerdem, eine bestimmte Version eines Pakets in ein Projekt zu importieren. Diese Funktion erlaubt es EntwicklerInnen, selbst mit veralteten Versionen eines Projekts weiterzuarbeiten. NPM ist nur einer von vielen Paketmanagern, die es gibt, dennoch hat sich NPM im JavaScript-Umfeld als das Standardwerkzeug für die Paketverwaltung herausgebildet.[43]

3.4.2 Vite

Vite ist ein modernes Build-Tool, das speziell für die Entwicklung von Webanwendungen mit modernen Technologien wie Vue, React und TypeScript entwickelt wurde. Es wurde entworfen, um die Entwicklung zu beschleunigen, indem es schnelle Builds und eine effiziente Entwicklungsumgebung bereitstellt. [40]

Merkmale von Vite

1. **Schnelle Entwicklungsumgebung:** Vite bietet eine extrem schnelle Entwicklungszeit, indem es eine schnelle Startzeit und inkrementelle Builds ermöglicht. Dies erleichtert die Entwicklung und Tests während des Schreibens von Code.
2. **Effizientes HMR (Hot Module Replacement):** Mit HMR können Änderungen im Code sofort angewendet werden, ohne dass die gesamte Anwendung neu geladen werden muss. Dies beschleunigt den Entwicklungsprozess erheblich.
3. **Unterstützung für verschiedene Frameworks:** Vite unterstützt verschiedene Frontend-Frameworks wie Vue, React und sogar Vanilla JavaScript, was es flexibel für verschiedene Projekte macht.
4. **Eingebautes Dev-Server:** Der integrierte Entwicklungsserver von Vite bietet eine optimale Erfahrung während der Entwicklung, einschließlich Fehlermeldungen im Browser, automatischem Öffnen des Browsers und mehr.
5. **Effizientes Bundling:** Vite nutzt ESM (ECMAScript Module) und bietet ein effizientes Bundling für die Produktion, was zu kleinen und schnellen Build-Dateien führt.

3.4.3 Material Design Bibliothek

Material Design ist ein Designsystem, das von Google entwickelt wurde, um eine konsistente und ansprechende Benutzeroberfläche für Web- und Mobilanwendungen zu schaffen. Es basiert auf den Prinzipien von Material Design, die auf visueller Hierarchie, Bewegung und Interaktion basieren. Material Design bietet eine Reihe von Komponenten, Icons und Stilen, die EntwicklerInnenn helfen, moderne und benutzerfreundliche Benutzeroberflächen zu erstellen. [23]

Da Material Design Richtlinien von Google sind, entwickeln Dritte Bibliotheken, die diese Richtlinien umsetzen. Dabei gibt es jeweils Bibliotheken für die unterschiedlichen Frontend-Frameworks. Für Vue gibt es z. B. Vuetify, für React gibt es Material-UI und für Angular gibt es Angular Material. Da die vorherige CO2-Runter-App auf React.js basierte, wurde dort die Material-UI Bibliothek verwendet. Um ein äquivalentes Design zu erreichen, wird in der Weiterentwicklung der CO2-Runter-App die Vuetify-Bibliothek verwendet, die speziell für Vue.js entwickelt wurde, aber dennoch die Material Design Richtlinien von Google umsetzt, wie auch die Material-UI (MUI) Bibliothek.

Beide Bibliotheken bieten eine Vielzahl von Komponenten, zum Beispiel Knöpfe, Textfelder etc., die EntwicklerInnen ermöglichen, schnell und einfach eine schöne und moderne Benutzeroberfläche zu erstellen. Material UI (MUI) oder auch Vuetify sind dabei UI-Komponenten-Bibliotheken, die auf React.js oder Vue basieren und die Material Design Richtlinien von Google anwenden, um ein einheitliches, schönes modulares Design zu schaffen, und somit die Entwicklung von Webanwendungen oder Prototypen zu vereinfachen.[24, 42]

3.4.4 NGINX

Für die Entwicklung von Webanwendungen ist ein Webserver unerlässlich. engine x (NGINX) ist ein Open-Source-Webserver, der für seine Leistung, Skalierbarkeit und Zuverlässigkeit bekannt ist. Er wird häufig als Reverse-Proxy-Server, Lastausgleichs-Server und HTTP-Cache verwendet. NGINX bietet eine Vielzahl von Funktionen, die es zu einem beliebten Webserver für die Entwicklung und Bereitstellung von Webanwendungen machen.

NGINX fungiert als Reverse-Proxy-Server, der eingehende Anfragen an verschiedene Backend-Server weiterleitet. Diese Funktion ermöglicht eine flexible Konfiguration von Routen und eine effiziente Verteilung des Datenverkehrs. Zum Beispiel kann NGINX den Datenverkehr basierend auf bestimmten Kriterien wie IP-Adresse, Header oder URL auf verschiedene Backend-Server verteilen. Dadurch können Anwendungen skalierbarer gestaltet werden, da NGINX den Datenverkehr je nach Bedarf auf zusätzliche Server umleiten kann, um die Last zu verteilen und die Ausfallsicherheit zu erhöhen.

Ein weiterer wichtiger Aspekt von NGINX ist sein HTTP-Caching. Durch den Einsatz von NGINX als HTTP-Cache können statische Inhalte wie Bilder, CSS-Dateien und JavaScript-Dateien zwischengespeichert werden. Dies reduziert die Last auf Backend-Servern und beschleunigt die Ladezeiten für BenutzerInnen, da häufig angeforderte Ressourcen direkt aus dem Cache bereitgestellt werden können. Darüber hinaus kann NGINX auch als Reverse-Proxy-Cache dienen, indem es die Antworten von Backend-Servern zwischenspeichert, um wiederholte Anfragen zu beschleunigen und die Serverlast zu verringern.

NGINX ermöglicht es uns, die Datenbank, die API und das Frontend zu verbinden. Es ist ein wichtiger Bestandteil der Infrastruktur, da es die Kommunikation zwischen den verschiedenen Komponenten der Anwendung ermöglicht. NGINX bietet eine leistungsstarke und flexible Plattform für die Bereitstellung von Webanwendungen und ist ein unverzichtbares Werkzeug für die Entwicklung und Bereitstellung von modernen Webanwendungen. [31]

3.4.5 Docker

Docker ist eine Open-Source-Plattform, die es EntwicklerInnenn ermöglicht, Anwendungen in Containern zu erstellen, zu testen und zu bereitstellen. Container sind eine Art von Virtualisierung, die es ermöglichen, Anwendungen und ihre Abhängigkeiten in einer isolierten Umgebung auszuführen. Docker bietet eine Vielzahl von Funktionen, die die Entwicklung und Bereitstellung von Anwendungen erleichtern, darunter die Möglichkeit, Anwendungen in Containern zu verpacken, Container zu verwalten und Container in der Cloud zu bereitstellen.

Eine der leistungsstarken Funktionen von Docker ist die Möglichkeit, Anwendungen über verschiedene Umgebungen hinweg zu skalieren und bereitzustellen. Durch die Verwendung von Docker-Containern können EntwicklerInnen sicherstellen, dass ihre Anwendungen konsistent und deterministisch in verschiedenen Umgebungen ausgeführt werden, von lokalen Entwicklungsumgebungen bis hin zu Produktionsumgebungen in der Cloud.

Darüber hinaus bietet Docker eine einfache Möglichkeit, Anwendungen horizontal zu skalieren, indem mehrere Instanzen von Containern gestartet und verwaltet werden können. Dies ermöglicht eine effiziente Nutzung von Ressourcen und verbessert die Skalierbarkeit von Anwendungen, insbesondere in hoch belasteten Umgebungen.

Ein weiterer Vorteil von Docker ist die Portabilität von Containern. Docker-Container können leicht zwischen verschiedenen Umgebungen verschoben werden, da sie alle ihre Abhängigkeiten und Konfigurationen in einem Paket enthalten. Dies erleichtert die Bereitstellung von Anwendungen auf verschiedenen Infrastrukturen, von lokalen Rechnern bis hin zu öffentlichen Cloud-Plattformen wie AWS, Azure oder Google Cloud.

Docker ermöglicht es uns, die Anwendung in einer isolierten Umgebung zu testen und zu entwickeln, ohne dass die Umgebung der EntwicklerInnen beeinträchtigt wird. Es bietet eine konsistente und zuverlässige Plattform für die Entwicklung und Bereitstellung von Anwendungen und ist ein unverzichtbares Werkzeug für die Entwicklung moderner Webanwendungen. [5]

Nachdem die Grundlagen der nutzerzentrierten Design-Methoden und der Webentwicklung erläutert wurden, wird im nächsten Kapitel auf die Nutzerzentrierte Umfrage eingegangen. Im folgenden Kapitel wird beschrieben, wie eine nutzerzentrierte Umfrage aufgebaut wird und welche verschiedenen Arten einer Umfrage es gibt. Anschließend wird erläutert, welche Faktoren bei der Auswahl der Zielgruppe entscheidend sind und wie die tatsächliche Durchführung der nutzerzentrierten Umfrage verlief. Es werden die Ergebnisse der im Rahmen dieser Studienarbeit durchgeführten Umfrage offengelegt und analysiert. Abschließend wird das Thema Personas angesprochen.

4 Nutzerzentrierte Umfrage

Während dieser Studienarbeit steht die Verbesserung und Restrukturierung der CO2-Runter Webseite im Vordergrund. Dabei soll besonders auf die Nutzergruppe eingegangen werden, um mit dem bereits vorgestellten Prinzip des nutzerzentrierten Designs die Webseite für EndnutzerInnen ansprechender und interessanter zu gestalten. Durch dieses Vorgehen soll die Webseite sowohl in den Punkten Design und Nutzerfreundlichkeit, als auch im Allgemeinen verbessert werden. Das Ziel dahinter ist, dass mehr potenzielle NutzerInnen gefunden und angesprochen werden, um so zu einem klimabewussteren Denken anzuregen.

Da der/die NutzerIn im Zentrum des Vorgehens steht, werden selbstverständlich Ideen, Kritik und Einflüsse von potenziellen NutzerInnen der Webseite, aber auch von Menschen im Allgemeinen, über die bisherige Webseite benötigt. Um diese Kritik und Einflüsse zu sammeln wird im Rahmen dieser Studienarbeit eine nutzerzentrierte Umfrage durchgeführt. Mithilfe der Umfrage sollen Meinungen, Kritik und Ideen von unparteiischen Personen gesammelt werden, sodass diese im Anschluss ausgewertet werden können und dem Entwicklerteam entscheidende Hinweise darüber bieten, welche Aspekte der aktuellen Webseite größeren Sanierungsbedarf haben bzw. welche Aspekte bereits ihren Zweck und Wert erfüllen. In den folgenden Unterkapiteln wird das Vorgehen von der Sammlung der möglichen Fragen der Umfrage, bis hin zur Auswertung der Umfrageergebnisse dokumentiert.

4.1 Methodik der Umfrage

Die Methodik der Umfrage bildet das methodische Grundgerüst, dass die systematische Erfassung und Analyse von Daten ermöglicht, um präzise Einblicke in die untersuchte Thematik zu gewinnen. In diesem Kapitel werden die spezifischen Vorgehensweisen und Verfahren detailliert erläutert, die im Rahmen der durchgeführten Umfrage angewendet wurden. Von der Auswahl der Stichprobe über die Gestaltung des Fragebogens bis hin zur Datenerhebung und -auswertung werden die methodischen Schritte transparent dargestellt.

4.1.1 Planung der Umfrage

Zu Beginn jeder Umfrage steht die Planung der Umfrage. Die Planung ist einer der wichtigsten, wenn nicht sogar der wichtigste, Schritt einer Umfrage, da auf Basis der in der Planung festgelegten Konzepte etc. die finale Umfrage aufbaut und schlußendlich auch durchgeführt wird. Dazu wird im Buch *Umfrage* beschrieben, dass jedes Forschungsprojekt grundsätzlich mit einer Planungsphase beginne. Darin werden unter anderem die theoretische Fundierung der Forschungsfrage als auch die daraus resultierende Art der Froschung festgelegt.[1]

Die Forschungsfrage wird am Anfang eines Forschungsprojektes bestimmt und steuert das weitere Vorgehen des Projektes. [1] In unserem Fall bearbeiten wir die Forschungsfragen: "Wie kann man mehr Menschen dazu motivieren, die CO2-Runter Webseite zu nutzen?" und "Wie kann man klimabewussteres Verhalten im Allgemeinen erzeugen?".

Mit der Forschungsfrage ergibt sich zumeist automatisch auch die Art der Forschung. Dabei unterscheidet man im Allgemeinen grundsätzlich vier Arten von Untersuchungen:

- **Explorative Untersuchung:** Untersucht man einen Bereich mit seiner Untersuchung bzw. Umfrage, in welchem bisher nur sehr wenige bzw. keine Informationen und Erkenntnisse vorliegen, so handelt es sich um eine explorative Untersuchung. Das Ziel einer explorativen Untersuchung ist es, einen ersten Überblick über den Untersuchungsbereich zu gewinnen und Grundlagenforschung durchzuführen. Häufig wird diese Art der Untersuchung als Umfragenforschungsprojekt als vorbereitende Forschung verwendet. [1]
- **Deskriptive Untersuchung:** Bei einer deskriptiven Untersuchung gibt es bereits ein relativ großes Vorwissen. Das primäre Ziel ist hierbei Detailinformationen zu einem Thema zu erlangen. Zusätzlich im Zentrum des Interesses stehen hier Fragen nach der Verteilung bestimmter Merkmale und Merkmalskombinationen, aber auch nach Veränderungen z.B. im Zeitverlauf. [1] Ergebnisse einer deskriptiven Untersuchung werden häufig in Tabellen präsentiert. Bekannt sind diese aus Printmedien und/oder dem Fernsehen.[1]
- **hypothesentestende bzw. kausalanalytische Forschung:** Sollte man bei einer Untersuchung bereits vor der empirischen Untersuchung Vermutungen über die Ausprägung bestimmter Verteilungen angestellte werden, spricht man von einer hypothesentestenden Untersuchung.[1]

Bei unserer Umfrage handelt es sich um eine kausale Untersuchung. Das liegt daran, dass mit der Umfrage eine Ursache-Wirkung-Beziehung zwischen den Aspekten der Webseite und der Nutzerfreundlichkeit herauszufinden. Durch die Sammlung der Daten, welche Aspekte der aktuellen Webseite gut oder schlecht bewertet wurden, kann man Hypothesen

aufstellen, welche Elemente der Webseite zu einer positiven bzw. negativen Nutzererfahrung führen.

Bevor die eigentliche Umfrage erstellt wurde, wurden in einem Dokument Fragen gesammelt um einen Fragenkatalog zu erstellen. Der Fragenkatalog dient zur Sammlung möglicher Fragen, damit diese nach der Sammelphase erneut überarbeitet und gewichtet werden können. Der Fragenkatalog besteht aus insgesamt **31** Fragen. Die Fragen wurden in einer Brainstorming-Aktion gesammelt und ohne Ordnung notiert. Nach der Brainstorming-Aktion wurden die Fragen kategorisiert. Dabei wurden die folgenden Kategorien erstellt und für die Einordnung der Fragen genutzt:

- CO2 Fußabdruck
- Motiviation Nutzung
- Verbesserung Webseite
- aktueller Stand der Webseite

Im nächsten Schritt wurden die kategorisierten Fragen bewertet und nach ihrer Wichtigkeit eingeteilt. Für die Bewertung wurden Ganzzahlen im Interval von 0 (= nicht so wichtig) bis 5 (= muss vorhanden sein) bewertet. Alle Fragen die mit einer 5 gekennzeichnet wurden, wurden später automatisch mit in die Umfrage aufgenommen. Von allen 31 Fragen wurden insgesamt 26 mit einer 5 gekennzeichnet und wurden somit 1:1 bzw. in einer ähnlichen Form in die Umfrage aufgenommen. Selbstverständlich wurden während der Erstellung der Umfrage weitere Fragen hinzugefügt, da der Fragenkatalog lediglich als Orientierung und ersten roten Faden verwendet wurde.

Nun wurde mit der eigentlichen Erstellung der Umfrage begonnen. Die Umfrage wurde mithilfe von Google Formulare erstellt. Da Google mit Google Formulare eine kostenlose Möglichkeit zum Erstellen von Umfragen mit vielen Features bietet, wurde auf diese Software bei der Erstellung der Umfrage gesetzt.

Die Umfrage enthält insgesamt 5 Abschnitte. Die Abschnitte sind hierbei mit Kategorien zu vergleichen, da darauf geachtet wurde, dass ähnliche Fragen innerhalb eines Abschnitts präsentiert werden. Die Umfrage startet mit einem kleinen Einführungsabschnitt. Darin werden die TeilnehmerInnen begrüßt und der Grund der Umfrage wird genauer erläutert. Es wird ebenfalls darauf hingewiesen, dass es von Vorteil ist, die Originalwebseite aufzurufen. Dies hat zum einen den Vorteil, dass die TeilnehmerInnen die Webseite live erleben können. Aber es bietet selbstverständlich auch eine erste Werbung. Abschließend werden die TeilnehmerInnen in diesem ersten Abschnitt darauf aufmerksam gemacht, dass die Umfrage ungefähr 30min dauern kann und diese anonym durchgeführt wird. Es wird

also weder eine Email, noch andere persönliche Daten für die Teilnahme an der Umfrage benötigt.

Der zweite Abschnitt thematisiert das vorhandene Wissen der TeilnehmerInnen über den CO2-Fußabdruck im Allgemeinen. Das Ziel dieses Abschnitts ist es, zu erfahren, inwieweit sich die TeilnehmerInnen bereits mit den Themen CO2-Fußabdruck und Klimaschutz beschätigt haben. Beispielhafte Fragen aus diesem Abschnitt sind zum Beispiel:

1. Kennen Sie den Begriff CO2-Fußabdruck?
2. Haben Sie sich schon einmal Gedanken über ihren CO2-Fußabdruck gemacht?
3. Wie bewusst sind Sie sich im Allgemeinen über ihren CO2-Fußabdruck?

Im Falle, dass die erste Frage mit *Nein* beantwortet wurde, besteht die Möglichkeit, innerhalb der Umfrage ein zweiminütiges Erklärvideo zum Thema CO2-Fußabdruck anzuschauen.

Abschnitt drei behandelt das Thema *Motivation*. Innerhalb des Abschnitts steht die Frage im Fokus, wie man Menschen zu klimafreundlicherem Verhalten motivieren und anregen kann. Zusätzlich soll auch ermittelt werden, was sich NutzerInnen eines CO2-Rechners von der Webseite erwarten bzw. erhoffen. In diesem Teil der Umfrage sind zudem häufig Textfelder unter den Antwortmöglichkeiten, damit die TeilnehmerInnen ihre eigenen Anregungen/Ideen/Meinungen zu den Themen kommunizieren können. Ziel dieses Abschnitts ist es, herauszufinden, was die NutzerInnen und vor allem (noch) potenzielle NutzerInnen von der Webseite erwarten und fordern. Durch das hier erhaltene Feedback kann bei der späteren Implementierung der neuen Features darauf geachtet werden, den NutzerInnen gerecht zu werden. Der Abschnitt beinhaltet zum einen Fragen, die mit *Ja* oder *Nein* beantwortet werden, wie z.B.:

1. Sind Sie bereits aktiv dabei, unser Klima zu schützen?
2. Motiviert es Sie, wenn Sie ihren Fußabdruck mit anderen NutzerInnenn vergleichen können?
3. Würde ihnen ein Punktesystem helfen, ihren CO2-Fußabdruck häufiger zu berechnen?

Jedoch befinden sich auch Fragen, die mehrere Auswahlmöglichkeiten bieten, um auf die Frage zu antworten. Beispielhafte Fragen sind hier:

1. Welche Anreize könnten Sie dazu motivieren, die Webseite häufiger zu verwenden?
2. Welche Möglichkeiten müssten eine Webseite ihnen bieten, um unser Klima (noch) aktiver zu schützen?

Dabei stehen sowohl vordefinierte Auswahlmöglichkeiten zur Verfügung, als auch ein Feld, bei welchem die TeilnehmerInnen eigene Antwortmöglichkeiten eingeben können. Am Ende des Abschnitts besteht für die TeilnehmerInnen die Option, eigene Ideen einzubringen, wie klimafreundlicheres Verhalten erzeugt werden kann.

Innerhalb des nächsten Abschnitts, Abschnitt 4, wird das Thema *Nutzerfreundlichkeit* behandelt. Hier werden die TeilnehmerInnen bezüglich der Nutzerfreundlichkeit der aktuellen Webseiten befragt. Zudem soll ermittelt werden, wie die Nutzerfreundlichkeit und das Zurechtfinden auf der Webseite besser gewährleistet werden kann. Das gewünschte Resultat ist hierbei, dass ein generelles Bild über die Nutzerfreundlichkeit der aktuellen CO2-Runter Webseite gewonnen wird. Zusätzlich sollen durch das Feedback der TeilnehmerInnen wichtige Punkte gewonnen werden, die im späteren Designprozess der neuen CO2-Runter Webseite zum Thema Nutzerfreundlichkeit mit eingearbeitet werden können. Die Mehrheit der Fragen des Abschnitts sind so gestaltet, dass den TeilnehmerInnenn Screenshots aus der aktuellen Webseite präsentiert werden. Auf dem Screenshot sind die im Fokus stehenden Aspekte der Webseite mit roten Boxen hinterlegt. Abbildung 4.1 zeigt ein solches Beispiel:

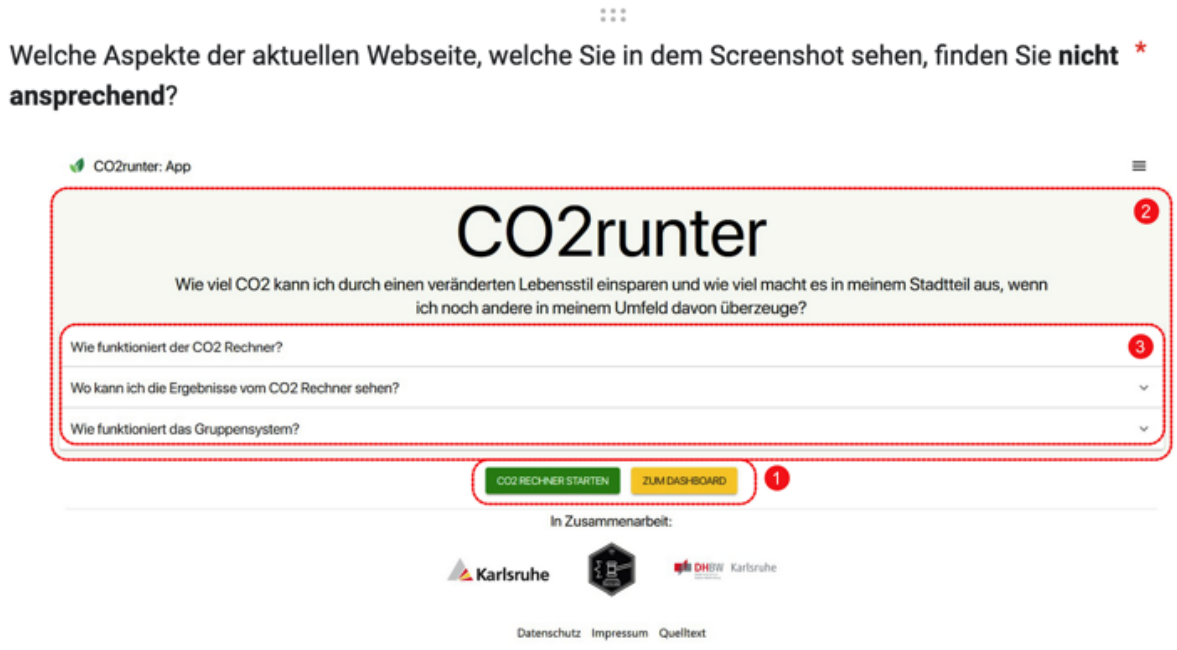

Abbildung 4.1: Frage aus Abschnitt 4, rote Boxen zur Erkennung des Fokus

Anschließend werden den TeilnehmerInnen Ankreuzmöglichkeiten anhand der Zahlenwerte geboten, um die Frage zu beantworten. Es gibt auch jedes Mal die Antwortmöglichkeit, die alle Optionen verneint bzw. bejaht. Sowie die Möglichkeit, eine freie Antwort in Form eines Kurztextes zu verfassen. Da es innerhalb des CO2-Rechners der Webseite auch die

Möglichkeit gibt, bei Fragen eine detaillierte Ansicht zu aktivieren, wird die Notwendigkeit und Meinung nach solch einem Feature abgefragt. Die detaillierte Ansicht wandelt hierbei eine Frage in mehrere Unterfragen um, bei der detailliertere Fragen zu einem bestimmten Themenbereich gestellt werden.

Der letzte Abschnitt behandelt das Thema *Design*. Dabei wird sowohl auf das aktuelle Design der CO2-Runter Webseite eingegangen, als auch auf neu erstellte Designprototypen. Die TeilnehmerInnen werden aufgefordert, dass alte Design der Webseite zu bewerten. Im Gegenzug bekommen sie auch Vorschläge zu neuen Designmöglichkeiten, die mit der Software Figma erstellt wurden. Das Ziel des letzten Abschnitts ist es, herauszufinden, in welchen Teilen der Webseite ein verbessertes Design erwünscht ist. Dadurch soll gleichzeitig auch die Nutzerfreundlichkeit und die allgemeine Nachfrage nach der CO2-Runter Webseite steigen. Es gibt dabei zwei grundsätzliche Fragetypen innerhalb des Abschnitts: Bewertungsfragen und Ja/Nein/Neutral-Fragen. Bei den Bewertungsfragen wird den TeilnehmerInnen ein Screenshot des relevanten Themas/Bereichs gezeigt. Anschließend soll der Screenshot bewertet werden. Ein Beispiel einer solchen Frage zeigt Abbildung 4.2:

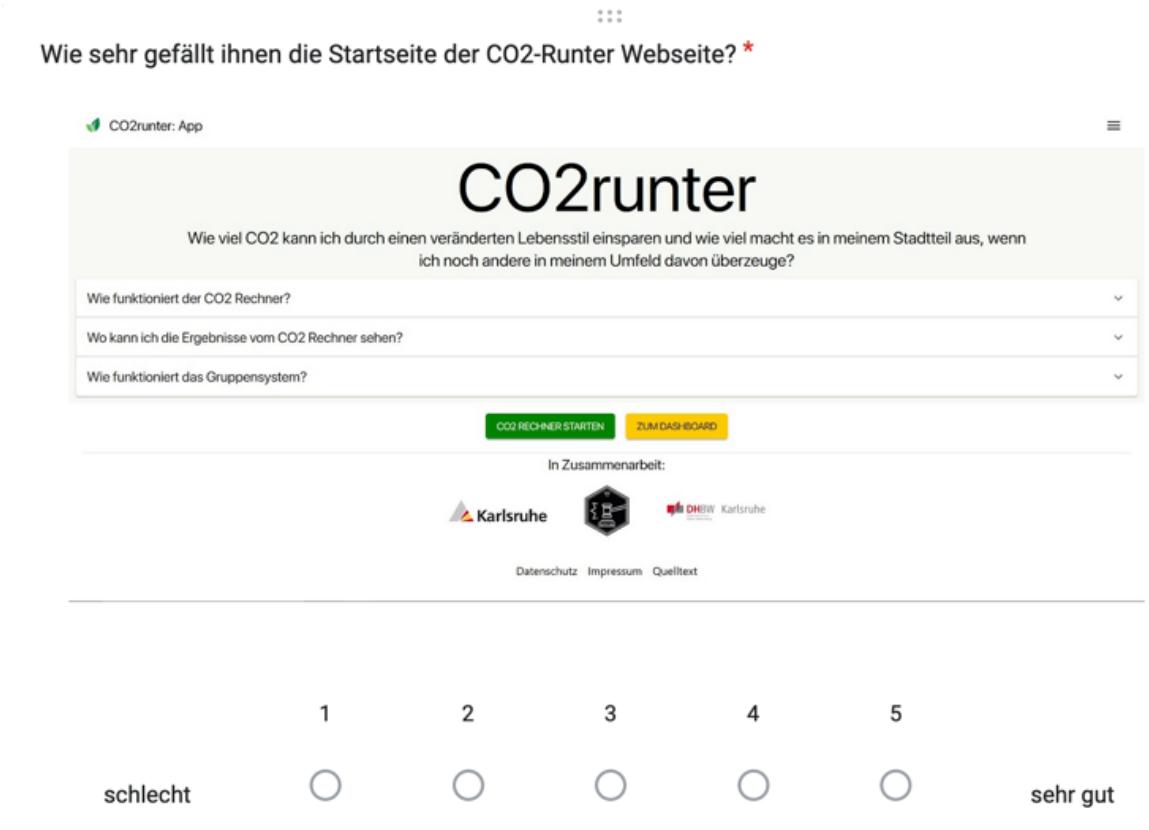

Abbildung 4.2: Bewertungsfrage aus Abschnitt 5

Der andere Fragetyp soll die Meinung der TeilnehmerInnen zu möglichen neuen Features erfragen. Dabei werden die TeilnehmerInnen gebeten, eine der folgenden Antwortmöglich-

keiten zu nutzen: Ja, Nein, vielleicht. Eine beispielhafte Darstellung dieses Fragetyps zeigt Abbildung 4.3:

Im Anschluss an die erfolgreiche Erstellung der Umfrage wurde diese noch einmal kontrolliert. Daraufhin wurde sich um eine geeignete Zielgruppe informiert und mit dem Projektteam abgesprochen. Informationen über den Auswahlvorgang und die angesprochene Personengruppe werden in Kapitel 4.1.3 (nächste Seite) dokumentiert. Zu guter Letzt wurde die Umfrage gestartet.

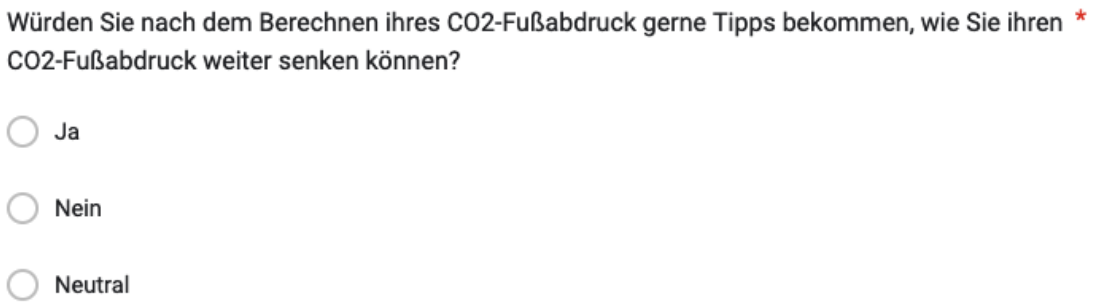

Abbildung 4.3: Ja/Nein/Vielleicht-Frage aus Abschnitt 5

4.1.2 Fokusdurchführung der Umfrage

Der Schwerpunkt der Umfrage liegt darin, die Wünsche und Anforderungen der TeilnehmerInnen zu sammeln, sodass diese im Nachhinein eingearbeitet werden können. Zu den Wünschen und Schwerpunkten zählen hierbei sowohl Features, die den potenziellen NutzerInnen fehlen und für wichtig erachtet werden. Zusätzlich sind aber auch Designs und die allgemeine Frage nach dem Design abzufragen und zu sammeln. Durch die Ergebnisse der Umfrage können dann neue Features eingearbeitet werden, veraltet oder uninteressante Features abgeschafft werden und veraltetes Design überarbeitet werden. Dadurch kann nach der Durchführung der Implementierung die CO2-Runter Webseite in neuem Glanz erscheinen.

4.1.3 Auswahl der Zielgruppe

Bevor die Umfrage veröffentlicht wurde, wurde über eine möglichst geeignete Zielgruppe diskutiert. Im Fokus stand dabei, dass man innerhalb der Zielgruppe möglichst viele Menschen neu für das Thema Klimaschutz gewinnen und überzeugen kann. Andererseits möchte man mit der Zielgruppe auch erfahrene oder bereits interessierte Menschen ansprechen, sodass man Antworten von Menschen mit Erfahrungen zu diesem Thema erhält.

Dadurch ist zu erwarten, dass sowohl unerfahrene Menschen, als auch Menschen, die sich mit dem Thema Klimaschutz und CO2-Fußabdruck auseinandergesetzt haben, an der Umfrage teilnehmen.

Aus den oben genannten Gründen wurde entschieden, eine Personengruppe mit einer Altersspanne von 12 bis 25 zu wählen. Diese Zielgruppe verbindet die gewünschten Charateristiken, dass Menschen dieser Personengruppe sowohl Neulinge als auch erfahrene Personen im Bereich Klimaschutz sind. Zudem ist diese Zielgruppe von Vorteil, da der Klimaschutz und die Reduktion der Ausmaße des Klimawandels ein langjähriger Prozess sein wird. Dieser Prozess wird vorausschauend hauptsächlich von den jungen Heranwachsenden dieser Zielgruppe getragen.

Um die Zielgruppe möglichst optimal zu treffen, wurde die Umfrage zum größten Teil am Gymnasium Schönau im Schwarzwald durchgeführt. Den SchülerInnenn wurde eine kleine Einführungspräsentation zu der Studienarbeit gegeben. Anschließend wurde die Umfrage schulweit verbreitet. Die Umfrage wurde aber auch auf anderen Kommunikationswegen für andere interessierte Personen verbreitet.

4.2 Erhebung von Nutzerfeedback

Die Umfrage wurde am 12. Dezember 2023 veröffentlicht und stand ab diesem Zeitpunkt online zur Verfügung. Beendet wurde die Umfrage am 22. Februar 2024. Somit wurde die Befragung zur CO2-Runter Webseite in einem Zeitraum von ungefähr 10 Wochen durchgeführt. Innerhalb dieses Zeitraums nahmen 67 Personen aktiv an der Umfrage teil.

In diesem Abschnitt sollen einige der Ergebnisse der Befragung präsentiert und offengelegt werden. Das Feedback und die Antworten der TeilnehmerInnen wird teilweise in Tabellen aufbereitet und nach den Fragekategorien geordnet. Sollten die Rückmeldungen nicht innerhalb von Tabellen einfach dargestellt werden können, werden die Ergebnisse innerhalb des Fließtextes präsentiert.

Beim ersten Frageabschnitts handelt es sich um den CO2-Fußabdruck. Innerhalb des Abschnitts wurden fünf Fragen gestellt, die den allgemeinen Wissensstand der TeilnehmerInnen rund um den CO2-Fußabdruck abzufragen. Vier der fünf Fragen waren sogenannte Ja/Nein-Fragen. Die Ergebnisse sowie die genau Fragestellung sind in Tabelle 4.1 aufgelistet:

Fragestellung	**Ja**	**Nein**
Kennen Sie den Begriff CO2-Fußabdruck?	98,5%	1,5%
Haben Sie sich schon einmal Gedanken über ihren CO2-Fußabdruck gemacht?	83,6%	16,4%
Haben Sie ihren CO2-Fußabdruck schon einmal berechnet/berechnen lassen?	58,2%	41,8%
Kennen Sie den Klima-Buddy und haben Sie diesen evtl. schon benutzt?	6,0%	94,0%

Tabelle 4.1: Ja/Nein-Fragen aus dem Frageabschnitt CO2-Fußabdruck

Des Weiteren wurde die Frage gestellt, wie bewusst sich die TeilnehmerInnen im Allgemeinen über ihren CO2-Fußabdruck sind. Als Antwortmöglichkeiten gab es die Zahlen *1* (kein Bewusstsein) bis *5* (volles Bewusstsein). Insgesamt befinden sich 86,5% der Befragten im Zahlenraum 2-4. Lediglich drei der befragten Personen behaupten, vollstes Bewusstsein über ihren CO2-Fußabdruck zu haben. Auf der anderen Seite besitzen sechs Personen unter den 67 TeilnehmerInnen gar kein Bewusstsein über ihren Fußabdruck.

Im nächsten Frageabschnitt wurde thematisiert, aus welcher Motivation die TeilnehmerInnen die CO2-Runter Webseite benutzen würden. Insgesamt wurden den Befragten in diesem Abschnitt zehn Fragen gestellt, wovon eine optional war. Bei vier der neun Pflichtfragen handelt es sich erneut um Ja/Nein-Fragen, die aber eine weitere neutral bzw. unschlüssige Antwortmöglichkeit ermöglichen. Diese sind mit deren Antwortverteilungen in Tabelle 4.2 aufgelistet:

Fragestellung	**Ja**	**Nein**	**Neutral**
Sind Sie bereits aktiv dabei, unser Klima zu schützen?	61,2%	38,8%	-
Motiviert es Sie, wenn Sie ihren Fußabdruck mit anderen Nutzern vergleichen können?	56,7%	25,4%	17,9%
Würde ihnen ein Punktesystem helfen, ihren CO2-Fußabdruck häufiger zu berechnen?	59,7%	19,4%	20,9%
Würden Sie ihren Fortschritt bzw. ihren CO2-Fußabdruck gerne mit anderen Leuten teilen können?	26,9%	26,9%	46,3%

Tabelle 4.2: Ja/Nein-Fragen aus dem Frageabschnitt Motivation

Zusätzlich wurde die folgenden fünf Multiple-Choice-Fragen innerhalb des Motivationsabschnitts gestellt:

- **Warum würden Sie die CO2-Runter Webseite nutzen?**
  - aus Interesse (43,4%)
  - um mir meinen aktuellen CO2-Fußabdruck berechnen zu lassen (58,2%)
  - um Tipps zum Thema Klimaschutz zu erhalten (28,4%)
  - aus Neugier (52,2%)
  - durch Arbeitskollegen (1,5%)
- **Was erhoffen Sie sich von einer Webseite, welche den Klimaschutz unterstützt?**
  - Aufklärung über den Klimaschutz und den CO2-Fußabdruck (71,6%)
  - Tipps, wie ich mein Verhalten zum Schutz unseres Klimas verbessere (73,1%)
  - Keinen Vorteil (10,4%)
  - Sonstige (3%)
- **Welche Möglichkeiten müsste eine Webseite ihnen bieten, um unser Klima (noch) aktiver zu schützen?**
  - Mehr Tipps, wie ich das Klima schützen kann (38,8%)
  - Aufklärung, damit ich sehe wie ich dem Klima schade (61,2%)
  - Verlinkung zu Klimaschutzgruppierungen (10,4%)
  - Tipps, um den Klimaschutz in den Alltag zu integrieren (80,6%)
  - Sonstiges (4,5%)
- **Welche Anreize könnten Sie dazu motivieren, die Webseite häufiger zu verwenden?**
  - Belohnungen (56,7%)
  - Wettbewerbe (25,4%)

  - soziale Interaktion (23,9%)
  - Vergleiche mit anderen (46,3%)
  - Keine (3%)
  - Sonstiges (6%)

- **Welche Konzepte finden Sie sinnvoll für eine Webseite, um klimafreundliches Verhalten zu fördern?**
  - Gamifizierung (59,4%)
  - Soziale Interaktion (31,3%)
  - Persönliche Herausforderungen (54,7%)
  - Bildung und Bewusstseinsbildung (35,9%)
  - Belohnungssystem (53,1%)
  - Wettbewerbe (20,3%)

Bei der optionalen Frage wurden die TeilnehmerInnen gebeten, eigene Ideen und Anregungen zu nennen, die sie haben, um klimafreundlicheres Verhalten erzeugen oder unterstützen zu können. Insgesamt haben neun Personen auf die optionale Frage geantwortet.

Im nächsten Abschnitt wurden den TeilnehmerInnenn 13 Fragen zur Nutzerfreundlichkeit der CO2-Runter Webseite gestellt. Unter den 13 Fragen befinden sich drei optionale Fragen, vier Multiple-Choice-Fragen und sechs Ja/Nein-Fragen. Die Ja/Nein und die dazugehörigen Ergebnisse sind in Tabelle4.3 aufgelistet:

Fragestellung	**Ja**	**Nein**	**Neutral**
Ist ihnen auf dem vorherigen Screenshot der Abschnitt Ïhr aktueller CO2-Fußabdruckäufgefallen?	46,3%	53,7%	-
Wünschen Sie sich, besser durch den CO2-Rechner geführt zu werden?	23,9%	20,9%	55,2%
Finden Sie es sinnvoll, Zahlenwerte wie in dieser Abbildung zu schätzen?	44,8%	14,9%	40,3%
Wünschen Sie sich, dass der aktuelle CO2-Fußabdruck stärker in den Vordergrund rückt?	61,2%	19,4%	19,4%

Die Fortsetzung erfolgt auf der nachfolgenden Seite

Fragestellung	**Ja**	**Nein**	**Neutral**
Würden ihnen Richtwerte helfen, die Kurzantworten besser zu kategorisieren?	85,1%	4,5%	10,4%
Würden Sie die detaillierte Ansicht bei Fragen nutzen?	70,1%	10,4%	19,4%

Tabelle 4.3: Ja/Nein-Fragen aus dem Frageabschnitt Nutzerfreundlichkeit

Bei den Multiple-Choice-Fragen ging es vor allem darum, herauszufinden, welche Aspekte der zu sehenden Webseite besonders nutzerfreundlich und welche eher nicht nutzerfreundlich gestaltet sind. Dabei wurde sich in der Umfrage auf die Startseite der CO2-Runter Webseite und auf den Fußabdruckrechner fokussiert. Die Antworten auf die Multiple-Choice-Fragen sind folgendermaßen ausgefallen:

- **Welche Aspekte der aktuellen Startseite finden Sie besonders ansprechend oder benutzerfreundlich?**
  - Buttons (47,8%)
  - Webseitenbanner (9,0%)
  - Infokarten (29,9%)
  - Alles (16,4%)
  - Gar nichts (19,%)
- **Welche Aspekte der aktuellen Webseite, welche Sie im Screenshot sehen, finden Sie nicht ansprechend?**
  - Buttons (11,9%)
  - Webseitenbanner (49,3%)
  - Infokarten (19,4%)
  - Ich finde alles ansprechend (19,4%)
  - Ich finde nichts ansprechend (19,4%)
  - Sonstiges (6,0%)
- **Welche Aspekte des CO2-Rechners finden Sie besonders ansprechend oder benutzerfreundlich?**

  - Themenbalken (55,2%)
  - Fragebogen (44,8%)
  - Anzeige für den aktuellen CO2-Fußabdruck (55,2%)
  - Gar nichts (9,0%)
  - Sonstiges (3,0%)

- **Welche Aspekte des CO2-Rechners finden Sie nicht ansprechend?**
  - Themenbalken (19,4%)
  - Fragebogen (20,9%)
  - Anzeige für aktuellen CO2-Fußabdruck (22,4%)
  - Ich finde alles prima! (43,3%)
  - Farbdesign/Optik (4,5%)
  - Sonstiges (3,0%)

Des Weiteren beschäftigen sich drei optionale Fragen mit weiteren Verbesserungsvorschlägen der TeilnehmerInnen. Pro Frage wurden durchschnittlich 15 Antworten eingereicht. Im Folgenden sind einige beispielhaft aufgelistet:

- *„aktueller CO2 Fußabdruck“ hervorheben* (9x)
- *Kennzeichnung durch Farbe* (13x)
- *CO2 Fußabdruck Balken farbig hinterlegen*
- *Sieht altmodisch aus* (4x)
- *Attention-Grabber wie ein Video für die Hauptseite*
- *Spielerischere Ansicht, zum erreichen des jüngeren Publikums*

Abschließend haben sich die TeilnehmerInnen im letzten Abschnitt mit dem Thema Design befasst. Im Designabschnitt werden die TeilnehmerInnen dazu aufgefordert, das Design der aktuellen CO2-Runter Webseite zu bewerten. Zusätzlich werden neue Designvorschläge präsentiert, die ebenfalls bewertet werden sollen. Innerhalb des Abschnitts werden den TeilnehmerInnen neun Fragen gestellt. Davon sind drei Ja/Nein-Fragen, drei Bewertungsfragen, zwei optionale Fragen und ein Designvergleich. Die Bewertungsfragen befolgen erneut einem Bewertungsschema von den Zahlen *1 bis 5*, wobei 1 die schlechteste Bewertung und

5 die beste Bewertung ist. Im Folgenden finden Sie die Bewertungsfragen zusammen mit den entsprechenden Antworten:

- **Wie sehr gefällt Ihnen die Startseite der CO2-Runter Webseite?**
  - 1, sehr schlecht (20,9%)
  - 2, schlecht (19,4%)
  - 3, neutral (35,8%)
  - 4, gut (19,4%)
  - 5, sehr gut (4,5%)
- **Wie sehr gefällt Ihnen das Design des CO2-Rechners?**
  - 1, sehr schlecht (7,5%)
  - 2, schlecht (20,9%)
  - 3, neutral (44,8%)
  - 4, gut (25,4%)
  - 5, sehr gut (1,5%)
- **Wie übersichtlich finden Sie den CO2-Rechner?**
  - 1, sehr unübersichtlich (0,0%)
  - 2, unübersichtlich (10,4%)
  - 3, neutral (31,3%)
  - 4, übersichtlich (43,3%)
  - 5, sehr übersichtlich (14,9%)

Die Ja/Nein-Fragen und ihre Antworten sind in Tabelle 4.4 dokumentiert:

Fragestellung	**Ja**	**Nein**	**Neutral**
Denken Sie, ein FAQ wäre sinnvoll?	71,6%	7,5%	20,9
Würde eine einfachere Benutzeroberfläche Sie eher dazu anregen die Webseite zu nutzen?	64,2%	10,4%	25,4%

Würden Sie nach dem Berechnen ihres CO2-Fußabdruck gerne Tipps bekommen,wie Sie ihren CO2-Fußabdruck weiter senken können?	79,1%	3,0%	17,9%

Tabelle 4.4: Ja/Nein-Fragen aus dem Frageabschnitt Design

Beim Designvergleich wurde das alte Design mit einem Prototyp verglichen. Laut der Umfrage finden 82,1% der Befragten den Prototyp besser. Lediglich 9% finden das alte Design ansprechender. Die restlichen 9% finden weder das alte Design, noch das neue Design besser oder enthalten sich. Die optionalen Fragen beschäftigen sich erneut mit weiteren Verbesserungsvorschlägen rund um das Design der CO2-Runter Webseite. Anbei sind einige Antworten der TeilnehmerInnen:

- *Cliparts passend zu Thema (Bäume, Weltkugel, Tiere, Menschen, etc.)*
- *Moderner gestallten (eyecatcher)*
- *Mehr Farbe, mehr Symbole, ansprechendere Schriftart, „jüngeres“ Design*
- *Einpflanzung eines Eyecatchers zu dem Thema, wie bei dem bsp. des neuen Designs*
- *Mehr Effekte und Farbe, damit es lebhafter wirkt.*

4.2.1 Erstellung von Personas anhand der Umfrageergebnisse

Um die Bedürfnisse und Anforderungen der NutzerInnen besser zu verstehen, haben wir die Ergebnisse unserer Umfrage analysiert und Personas erstellt, die typische Nutzergruppen repräsentieren. Dafür ist es typischerweise möglich, diese durch Clustering und andere Methoden zu erstellen. In unserem Fall haben wir uns auf die Analyse der Umfrageergebnisse konzentriert und die Personas basierend auf den Ergebnissen erstellt, weil die Teilnehmeranzahl der Umfrage zu gering war, um eine Clusteranalyse durchzuführen.

Somit haben wir die Umfrageergebnisse analysiert und die wichtigsten Erkenntnisse zusammengefasst. Für die Analyse haben wir die Antworten auf die Fragen zu CO2-Fußabdruck, Motivation, Nutzerfreundlichkeit und Design ausgewertet und durch Mind-Mapping und Brainstorming die wichtigsten Erkenntnisse zusammengefasst. Die Ergebnisse der Umfrage zeigen, dass die TeilnehmerInnen unterschiedliche Kenntnisse und Einstellungen zum Thema Klimaschutz und CO2-Fußabdruck haben.

Zusammenfassend haben wir durch unsere Analyse Personas erstellt, welche typische Nutzergruppen repräsentieren und diese uns helfen, das Verhalten, die Bedürfnisse und die Ziele der Zielgruppe besser zu verstehen. Basierend auf den Ergebnissen unserer Umfrage haben wir die folgenden Personas entwickelt:

1. **Umweltbewusste Person**: Diese Persona ist motiviert, ihren CO2-Fußabdruck zu reduzieren und sich aktiv am Klimaschutz zu beteiligen. Sie ist offen für neue Informationen und Tipps, um ihren Lebensstil nachhaltiger zu gestalten. Diese Persona sucht nach detaillierten Informationen über den Klimawandel und konkreten Anleitungen zur Umsetzung von Maßnahmen.

2. **Interessierte Person mit begrenztem Wissen**: Diese Persona ist neugierig darüber, mehr über ihren CO2-Fußabdruck zu erfahren und sucht nach einfachen Möglichkeiten zur Reduzierung. Sie ist offen für neue Ideen und Tipps, benötigt jedoch leicht verständliche Informationen und Anleitungen, um ihren Beitrag zum Klimaschutz zu leisten.

3. **Skeptische Person**: Diese Persona steht dem Thema Klimaschutz skeptisch gegenüber und benötigt fundierte Informationen und Beweise, bevor sie ihr Verhalten ändert. Sie ist nicht bereit, große Veränderungen im Lebensstil vorzunehmen, und sucht nach praktischen Möglichkeiten zur Reduzierung ihres CO2-Fußabdrucks, ohne bevormundet zu werden.

Durch die Konzentration auf diese Personas und das Verständnis ihrer Bedürfnisse und Motivationen ist es uns möglich geworden, die Webseite aus der Perspektive der NutzerInnen zu betrachten und die Benutzererfahrung zu verbessern. Dadurch konnten wir spezifische Anforderungen definieren und die Webseite so gestalten, dass sie die Bedürfnisse und Ziele der Zielgruppe besser erfüllt. Die Anforderungen werden im nächsten Kapitel vorgestellt und behandelt. Es wird erklärt, welche Art von Anforderungen definiert wurden und wie die konkreten Anforderungen aussehen.

5 Anforderungen an die neue Software

Zur Ermittlung der Anforderungen an die neue Webanwendung wurden im Rahmen eines Brainstormings bestehende und weitere Aspekte zusammengetragen, die in der alten Webanwendung nicht oder nur unzureichend umgesetzt wurden. Aber auch Grundfunktionen, die die neue Webanwendung haben muss, wurden gesammelt. Die Ergebnisse wurden anschließend in einer Liste zusammengefasst. Die insgesamt **19** Anforderungen wurden zusätzlich in folgende Kategorien eingeteilt: Muss-, Soll- und Kann-Anforderungen.

- **Muss-Anforderungen** beschreiben essentielle Anforderungen an die neue Webanwendung, die unbedingt erfüllt werden müssen. Bei diesen Anforderungen handelt es sich um Kernfunktionalitäten und wichtige Merkmale. Die Erfüllung dieser Anforderungen ist unerlässlich und stellt dadurch den Basisumfang der Software dar.
- **Soll-Anforderungen** beschreiben Anforderungen, die für die neue Webanwendung wünschenswert, aber nicht zwingend notwendig sind. Bei diesen Anforderungen handelt es sich um Funktionalitäten, die den Basisumfang der **Muss-Anforderungen** der Software erweitern und einen relevanten Mehrwert bieten.
- **Kann-Anforderungen** beschreiben optionale Anforderungen an die neue Webanwendung, deren Erfüllung weitere Funktionalitäten ermöglicht, die aber von untergeordneter Bedeutung sind.

Im Folgenden werden die Anforderungen an die neue Webanwendung spezifiziert. Hierbei werden zunächst die funktionalen Anforderungen an die Webanwendung beschrieben, gefolgt von den nicht-funktionalen Anforderungen. Abschließend werden alle Anforderungen in Form eines tabellarischen Überblicks zusammengefasst. Die Begriffe *System*, *Software* und *Webanwendung* werden im Folgenden synonym verwendet, um über die zu entwickelnde Webanwendung zu sprechen. Jede Anforderung wird mit einer eindeutigen ID versehen, die sich aus dem Buchstaben **R** für **Requirement** und einer fortlaufenden Nummer zusammensetzt. Die Anforderungen werden in der Form [**RXX**] beschrieben, wobei **XX** für die fortlaufende Nummer steht.

5.1 Funktionale Anforderungen

Im Folgenden werden die funktionalen Anforderungen spezifiziert sowie die Services, die das System bereitstellen soll.

5.1.1 Ereignisverarbeitung

[R01] Abrufen der Bestandsdaten von der Datenbank

Die neue Webanwendung soll in der Lage sein, Bestandsdaten aus der Datenbank abzurufen, um diese für die weitere Verarbeitung innerhalb der Webanwendung zu nutzen. Das beinhaltet das Abfragen der Informationen aus der Datenbank ohne Zwischenfälle und Schwierigkeiten. Deshalb muss das Abrufen der Daten zuverlässig und sicher sein, und die aktuellen Daten müssen korrekt und vollständig geladen werden. - (Muss-Anforderung).

[R02] Berechnung des CO2-Fußabdrucks

Der CO2-Rechner muss in der Lage sein, die eingegebenen Daten der BenutzerInnen zu verarbeiten und den entsprechenden CO2-Fußabdruck zu berechnen. Dabei sollte der Rechner benutzerfreundlich gestaltet sein, leicht verständlich und logisch aufgebaut sein, um den BenutzerInnen ein angenehmes Erlebnis zu bieten. Die Berechnungen müssen genau sein und auf relevanten wissenschaftlichen Grundlagen basieren. - (Muss-Anforderung).

[R03] Hochladen von CO2-Daten

BenutzerInnen müssen die Möglichkeit haben, die berechneten CO2-Daten hochzuladen, um sie für ihre individuellen Fälle zu speichern und später darauf zugreifen zu können. Das Hochladen der Daten sollte einfach und intuitiv sein, um den BenutzerInnen eine reibungslose Erfahrung zu bieten. - (Muss-Anforderung).

5.1.2 Systemfunktionen

[R04] Homepage

Die Homepage der Webanwendung muss ansprechend gestaltet sein und die BenutzerInnen dazu animieren, sich über ihren CO2-Fußabdruck zu informieren und aktiv zu werden, um diesen zu reduzieren. Neben einer ästhetischen Gestaltung sollten auf der Homepage auch

hilfreiche Tipps, Informationsquellen und Verbesserungsvorschläge präsentiert werden, um die BenutzerInnen zu unterstützen.

[R05] Dashboard

Ein Dashboard soll wichtige Informationen zum CO2-Fußabdruck auf einen Blick darstellen. Dabei sollen relevante Kennzahlen, Grafiken und Vergleiche präsentiert werden, um den BenutzerInnen eine schnelle und übersichtliche Einsicht in ihren aktuellen CO2-Verbrauch zu ermöglichen. - (Muss-Anforderung).

[R06] Gruppensystemintegration

Das Gruppensystem soll es den NutzerInnen ermöglichen, sich mit anderen NutzerInnen zu vernetzen und gemeinsam als Gruppe ihre CO2-Einsparungen zu verfolgen, sowie die Möglichkeit bieten, sich mit anderen Gruppen zu vergleichen. - (Soll-Anforderung).

[R07] FAQ- und Literaturlisten-Seite

Als zusätzliche Informationsquelle soll eine FAQ- und Literaturlisten-Seite bereitgestellt werden, um den NutzerInnen weiterführende Informationen und Quellen zum Thema CO2-Fußabdruck zur Verfügung zu stellen. - (Soll-Anforderung).

5.1.3 CO2-Rechner

[R08] Tipps und Tricks zur Reduzierung des CO2-Fußabdrucks

Der CO2-Rechner soll den NutzerInnen basierend auf den eingegebenen Daten Tipps und Tricks zur Reduzierung ihres CO2-Fußabdrucks geben. Diese Empfehlungen sollen den NutzerInnen praktische Anleitungen bieten, wie sie ihren Lebensstil anpassen können, um ihren ökologischen Fußabdruck zu verringern. - (Kann-Anforderung).

[R09] Durchführung des CO2-Rechner-Workflows

Die NutzerInnen sollen durch den CO2-Rechner geleitet werden, um die CO2-Daten hochzuladen und zu speichern. Dabei soll der Workflow klar strukturiert sein und den NutzerInnen schrittweise Anleitungen bieten, um die erforderlichen Informationen korrekt einzugeben und den Prozess reibungslos abzuschließen. - (Kann-Anforderung).

[R10] Integration von Grafiken

Die Integration von ansprechenden Stockfotos und Grafiken auf der Webseite soll die Nutzererfahrung verbessern und die Inhalte visuell ansprechender gestalten. Durch die Verwendung von hochwertigen Bildern können komplexe Konzepte einfacher vermittelt und die Aufmerksamkeit der NutzerInnen besser auf bestimmte Inhalte gelenkt werden. - (Kann-Anforderung).

[R11] Visualisierung und Vergleich des CO2-Fußabdrucks

Das System soll das Ergebnis des CO2-Fußabdrucks mit relevanten Vergleichsdaten visualisieren und vergleichen, um den NutzerInnen eine sinnvolle Perspektive zu bieten. Grafische Darstellungen und Vergleichsdiagramme können den NutzerInnen dabei helfen, ihren CO2-Verbrauch besser zu verstehen und geeignete Maßnahmen zur Reduzierung zu identifizieren. - (Kann-Anforderung).

[R12] Modernisierung des Designs

Die Homepage und der CO2-Rechner sollen ein modernes Design erhalten, dass die Nutzererfahrung verbessert und wichtige Informationen hervorhebt. Durch eine klare visuelle Gestaltung und die gezielte Platzierung relevanter Inhalte kann die Nutzerführung optimiert und die Nutzung der Webanwendung erleichtert werden. - (Muss-Anforderung).

[R13] Anzeige von weiteren Quellen/Infos im CO2-Rechner

Zusätzliche Quellen und Informationen sollen im CO2-Rechner sowie auf einer separaten Seite angezeigt werden, um den NutzerInnen weitere Einsichten zu ermöglichen und ihr Verständnis für das Thema zu vertiefen. Durch die Bereitstellung von weiterführenden Informationen kann die NutzerIn ihr Wissen erweitern und gezieltere Maßnahmen in Betracht ziehen. - (Soll-Anforderung).

[R14] Anpassung von Question.json

Die Fragestellung im CO2-Rechner soll durch die Anpassung der Question.json-Datei verändert werden, sodass diese einerseits verständlicher und andererseits aussagekräftiger wird. - (Kann-Anforderung).

5.2 Nichtfunktionale Anforderungen

Die nachfolgenden nicht-funktionalen Anforderungen beschreiben Einschränkungen und Qualitätsmerkmale, die für die Entwicklung und den Betrieb des Systems gelten.

5.2.1 [R15] Benutzerfreundlichkeit

Die neue CO2-Webanwendung muss eine hohe Benutzerfreundlichkeit aufweisen, um eine breite Nutzerbasis anzusprechen und die Nutzererfahrung im Vergleich zur alten Webseite zu verbessern. Dafür sollte die Benutzeroberfläche modernisiert werden und dennoch viele wichtige Informationen für die NutzerInnen bereitstellen. Durch eine benutzerzentrierte Gestaltung soll das Interesse der NutzerInnen geweckt und ihre Motivation gesteigert werden, sich aktiv mit ihrem CO2-Fußabdruck auseinanderzusetzen. Dafür sollen Personas angewendet werden, um die passenden NutzerInnen anzusprechen und den Anforderungen der NutzerInnen gerecht zu werden. - (Muss-Anforderung).

5.2.2 [R16] Modernes Design

Bei der Webanwendung wird versucht, interessierte NutzerInnen durch ein modernes und attraktives Design anzusprechen. Dadurch soll die Nutzererfahrung verbessert und die Interaktion mit der Seite gefördert werden. Durch die Integration von hochwertigen Bildern und Eyecatchern wird die visuelle Attraktivität der Webseite erhöht und das Interesse der NutzerInnen geweckt. Ein ästhetisches Design trägt dazu bei, das Markenimage zu stärken und das Vertrauen der NutzerInnen in die Webanwendung zu stärken. - (Muss-Anforderung).

5.2.3 [R17] Zuverlässigkeit

Eine moderne Webanwendung sollte eine hohe Verfügbarkeit aufweisen und den NutzerInnen eine reibungslose Nutzung ermöglichen. Durch die Verwendung von zuverlässigen Technologien und die Einhaltung bewährter Entwicklungspraktiken soll die Stabilität und Leistungsfähigkeit des Systems sichergestellt werden. - (Muss-Anforderung).

5.2.4 [R18] Wartbarkeit und Erweiterbarkeit

Die Wartbarkeit und Erweiterbarkeit des Systems ist ein wichtiger Aspekt für die langfristige Entwicklung und Pflege der Webanwendung. Durch eine klare und strukturierte Codierung sowie die Einhaltung von bewährten Entwicklungspraktiken wird die Wartbarkeit des Systems verbessert. Eine umfassende Dokumentation und die Einhaltung eines Styleguides erleichtern zudem die Zusammenarbeit im Entwicklerteam und ermöglichen es, das System effizient zu erweitern und anzupassen. - (Muss-Anforderung).

5.2.5 [R19] Codequalität und Styleguide-Einhaltung

Die Qualität des Codes einer Software ist essentiell und sorgt für die Stabilität, Sicherheit und Skalierbarkeit des Systems. Um dies zu erreichen, ist das Einhalten von festgelegten Coding-Standards und Best Practices notwendig. Durch die Einhaltung von festgelegten Coding-Standards und Best Practices wird sichergestellt, dass der Code lesbar, wartbar und fehlerfrei ist. Einheitliche Namenskonventionen, Kommentare und eine konsistente Codeformatierung erleichtern die Zusammenarbeit im Entwicklerteam und tragen zur langfristigen Qualitätssicherung des Systems bei. - (Muss-Anforderung).

5.3 Übersicht der Anforderungen

Im Folgenden ist eine Tabelle, welche die funktionalen (f) und nicht-funktionalen (nf) Anforderungen aus den vorherigen Abschnitten (3.1, 3.2), übersichtlich in Muss-, Soll- und Kann-Anforderungen gruppiert und jeweils priorisiert anhand der Reihenfolge. Darüber hinaus hat jede Anforderung einen Verweis auf das Kapitel, in dem das Konzept und die Implementierung beschrieben sind.

ID	Anforderung	Art	Impl.
Muss-Anforderungen			
R01	Abrufen der Bestandsdaten von der Datenbank	f	6.7
R02	Berechnung des CO2-Fußabdrucks	f	6.6
R03	Hochladen von CO2-Daten	f	6.6
R04	Homepage	f	6.4
Die Fortsetzung erfolgt auf der nachfolgenden Seite			

ID	Anforderung	Art	Impl.
R05	Dashboard	f	6.6
R12	Modernisierung des Designs	f	6
R15	Benutzerfreundlichkeit	nf	6
R16	Modernes Design	nf	6
R17	Zuverlässigkeit	nf	6
R18	Wartbarkeit und Erweiterbarkeit	nf	6
R19	Codequalität und Styleguide-Einhaltung	nf	6
Soll-Anforderungen			
R06	Gruppensystemintegration	f	6.7
R07	FAQ- und Literaturlisten-Seite	f	6.8
R13	Anzeige von weiteren Quellen/Infos im CO2-Rechner	f	-
Kann-Anforderungen			
R08	Tipps und Tricks zur Reduzierung des CO2-Fußabdrucks	f	-
R09	Durchführung des CO2-Rechner-Workflows	f	6.6
R10	Integration von Grafiken	f	6
R11	Visualisierung und Vergleich des CO2-Fußabdrucks	f	6.6.3
R14	Anpassung von Question.json	f	6.4

Tabelle 5.1: Übersicht der funktionalen und nicht-funktionalen Anforderungen

Nachdem die Anforderungen für die neue CO2-Runter-Webseite definiert und in Muss-, Soll- und Kann-Anforderungen unterteilt wurden, kann mit der Neuerstellung der Webseite gestartet werden. Das nächste Kapitel präsentiert die Ergebnisse der Implementierungsphase. Dabei wird sowohl das Enddesign, dass der NutzerIn auf der Webseite begegnet, gezeigt, als auch auf den Code eingegangen, mit dem das Frontend erstellt und implementiert wurde. Zusätzlich wird auf den Framework-Wechsel und die Restrukturierung des Projekts eingegangen.

6 Implementierung und Verbesserung der Webseite

In den vorherigen Kapiteln wurde zunächst die bisherige Webseite analysiert, um ihre Stärken und Schwächen zu identifizieren. Anschließend wurde eine Nutzerumfrage durchgeführt, um die Zufriedenheit der NutzerInnen mit der Webseite zu ermitteln. Darauf aufbauend wurden aus den Ergebnissen der Nutzerumfrage Personas erstellt und Anforderungen abgeleitet. Diese Anforderungen wurden im Kapitel 5 spezifiziert und in die Kategorien Muss-, Soll- und Kann-Anforderungen eingeteilt.

Diese Erkenntnisse sollen nun genutzt werden, um eine verbesserte Version der Webseite zu entwickeln. Bevor dies geschehen kann, muss jedoch die bisherige Codebasis auf ein neues Frontend-Framework migriert werden, da die Zufriedenheit mit dem vorherigen Framework nicht sehr hoch war. Nach der Migration werden trotzdem die Ergebnisse der Nutzerumfrage genutzt, um die Webseite zu optimieren. Dabei wird auf die Ergebnisse der Umfrage eingegangen, und die Webseite wird entsprechend angepasst, um die Nutzererfahrung zu verbessern. Außerdem werden neue Features und Funktionen hinzugefügt, die die Nutzererfahrung bereichern und die NutzerInnen dazu anregen sollen, die Webseite vermehrt zu nutzen.

Bei jedem relevanten Kapitel werden die entsprechenden Anforderungen, welche dadurch erfüllt werden, referenziert. Da einige Anforderungen keine explizite Umsetzung erfordern, da diese in jedem Implementierungsschritt beachtet werden sollten, werden diese durch das Projekt hinweg konstant betrachtet. Folgende Anforderungen werden konstant beachtet und angewendet:

1. **[R10] Integration von Grafiken**
2. **[R12] Modernisierung des Designs**
3. **[R15] Benutzerfreundlichkeit**
4. **[R16] Modernes Design**
5. **[R17] Zuverlässigkeit**
6. **[R18] Wartbarkeit und Erweiterbarkeit**
7. **[R19] Code Qualität und Styleguide Einhaltung**

6.1 Frameworkwechsel und Verbesserungen

6.1.1 Motivation für den Frameworkwechsel

Die Webseite wurde bisher unter Verwendung des Frontend-Frameworks **React.js** entwickelt, das aufgrund seiner Beliebtheit bei vielen EntwicklerInnenn weit verbreitet ist. Dennoch wurde die Entscheidung getroffen, das Framework zu wechseln, bedingt durch verschiedene Herausforderungen und auch aufgrund von Kompetenzüberlegungen der Entwickler. Durch den Übergang zu einem anderen Framework soll die Webentwicklung vereinfacht und die Nutzererfahrung verbessert werden. Insbesondere sollen bekannte Probleme wie fehlerhaftes State Management vermieden werden.

Wie bereits im Grundlagenkapitel erläutert, wird als neues Framework **Vue** eingeführt, das ebenfalls weit verbreitet ist und von vielen EntwicklerInnenn genutzt wird.

6.1.2 Umsetzung des Frameworkwechsels

Die Umsetzung ist dabei zunächst nicht zu kompliziert, da das gesamte Projekt in verschiedene Abschnitte unterteilt war, wie zum Beispiel das Frontend der Webseite, das Backend und die Datenbank. Somit musste lediglich der Ordner, der das Frontend beinhaltet, ausgetauscht werden. Das Backend und die Datenbank bleiben unverändert. Um das Frontend mit dem neuen Framework (**Vue**) auszustatten, muss zunächst der Inhalt des bisherigen Ordners, der das Frontend beinhaltet, gelöscht werden. Daraufhin konnte über das Werkzeug **Vite** ein **Vue**-Projekt erstellt werden, das auch direkt **TypeScript** unterstützt. Dies hilft beim Entwickeln schon frühzeitig Probleme zu entdecken und zu lösen. Im folgenden Code-Snippet ist zu sehen, wie das Projekt mit **Vite** erstellt wurde.

```
1 npm create vite@latest
```

Listing 6.1: Initialisierung des Vue Projektes mit Vite [39]

Nachdem das Projekt erstellt wurde, konnte der Inhalt des Projekts in den alten und leeren Ordner kopiert werden. Danach mussten lediglich noch Konfigurationen angepasst werden, ebenso wie Dockerfiles und Docker Compose. Die Konfigurationen betreffen das **TypeScript**- und **Vue**-Projekt, während die Dockerfiles und Docker Compose für die Containerisierung der Webseite zuständig sind. Die entsprechenden Konfigurationen und Dockerfiles sind im folgenden Code-Snippet zu sehen.

```
# Use the official Node.js image as the base image
FROM node:16

# Set the working directory inside the container
WORKDIR /usr/src/app

# Copy only the package.json and package-lock.json files to leverage
    Docker cache
COPY package.json ./
COPY package-lock.json ./

# Install dependencies
RUN npm install

# Copy the entire project files to the container
COPY ./ ./

# Expose the port that the Vue app will run on (change this if your
    app uses a different port)
EXPOSE 3000

# Build the Vue project
RUN npm run build

# Command to start the Vue app
CMD ["npm", "run", "start"]
```

Listing 6.2: Dockerfile für das Vue Projekt

In der Docker Compose-Datei wird der Container für das Vue-Projekt erstellt und konfiguriert. Hierbei muss dann noch explizit der Port angegeben werden, auf dem die Webseite laufen soll.

```
version: "3.8"

client:
stdin_open: true
environment:
    - WDS_SOCKET_PORT=3050
    - CHOKIDAR_USEPOLLING=true
    - WATCHPACK_POLLING=true
build:
    dockerfile: Dockerfile
    context: ./client
volumes:
    - /app/node_modules
```

```
14      - ./client:/app
15  ports:
16      - "3000:3000"
```

Listing 6.3: Docker Compose für das Vue Projekt

Das alles funktioniert reibungslos. Außerdem muss in der *vite.config.ts*-Datei noch der Port angepasst werden, auf dem die Webseite laufen soll. Dieser muss mit dem Port in der Docker-Compose-Datei übereinstimmen.

```
1  export default defineConfig({
2      ...
3      server: {
4      host: '0.0.0.0',
5      port: 3000,
6  },
7  });
```

Listing 6.4: Port Konfiguration für das Vue Projekt

Nach diesem Schritt ist das Projekt erfolgreich migriert und kann nun mit dem neuen Framework weiterentwickelt werden. Es ist auch möglich, alles über die Docker Compose Datei zu starten und zu stoppen.

6.2 Unterstützung von PWA für Mobile NutzerInnen

Zu Beginn dieser Arbeit wurde in Kapitel 2 erwähnt, dass die Webseite trotz ihrer webbasierten Natur als App bezeichnet wird. Um die Nutzererfahrung zu optimieren und die Benutzerfreundlichkeit der Webseite zu erhöhen, wird die Integration von Progressive Web App (PWA) angestrebt. Eine PWA ist eine Technologie, die es ermöglicht, Webseiten ähnlich wie native Mobile Apps zu nutzen. Dies bedeutet, dass die Webseite auch offline zugänglich ist und auf dem Startbildschirm eines Smartphones installiert werden kann. Dadurch wird die Interaktion mit der Webseite komfortabler und die Nutzererfahrung insgesamt verbessert. [27]

Die Implementierung von PWA gestaltet sich vergleichsweise unkompliziert und erfordert nur wenige Schritte. Zunächst wird ein Plugin zu Vite hinzugefügt, das sich um die PWA-Funktionalität kümmert. Dies kann einfach über die Konsole durch die Installation des entsprechenden Plugins erfolgen.

```
npm i vite-plugin-pwa -D
```

Listing 6.5: Installation des PWA Plugins

Danach müssen wir die Vite-Konfiguration anpassen, um das Plugin zu aktivieren und zu konfigurieren. Dies umfasst das Hinzufügen eines Manifests, eines Service Workers sowie verschiedener Grafiken für mobile Geräte. Im folgenden Code-Snippet wird dargestellt, wie die Vite-Konfiguration angepasst wurde, um die Unterstützung für PWA zu integrieren.

```
...
import { VitePWA } from 'vite-plugin-pwa';

export default defineConfig({
    plugins: [
        ...
        VitePWA({
            registerType: 'autoUpdate',
            devOptions: {
                enabled: true,
            },
            includeAssets: [
                'favicon.ico',
                'apple-touch-icon.png',
                'mask-icon.svg',
            ],
            manifest: {
                name: 'CO2Runter',
                short_name: 'CO2Runter',
                description:
                    'CO2 Runter Webseite hilft bei der Reduktion von
                        CO2',
                theme_color: '#ffffff',
                icons: [
                    {
                        src: '/pwa-192x192.png',
                        sizes: '192x192',
                        type: 'image/png',
                    },
                    {
                        src: '/pwa-512x512.png',
                        sizes: '512x512',
                        type: 'image/png',
                    },
                    {
                        src: '/pwa-512x512.png',
```

```
                        sizes: '512x512',
                        type: 'image/png',
                        purpose: 'any maskable',
                    },
                ],
            },
        }),
    ],
    ...
});
```

Listing 6.6: Angepasste Vite Konfiguration für PWA

Ebenfalls erfordert es die Anpassung der *index.html*-Datei, um die im Manifest definierten Grafiken zu laden.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale
        =1.0" />
    <title>CO2 Runter</title>
    <meta name="description" content="CO2 Runter Webseite hilft bei
        der Reduktion von CO2">

    <link rel="icon" href="/favicon.ico">
    <link rel="apple-touch-icon" href="/apple-touch-icon.png" sizes="
        180x180">
    <link rel="mask-icon" href="/mask-icon.svg" color="#FFFFFF">
    <meta name="theme-color" content="#ffffff">
</head>

<body>
    <div id="app"></div>
    <script type="module" src="/src/main.ts"></script>
</body>

</html>
```

Listing 6.7: Anpassungen an der index.html Datei

Das Manifest ist eine JavaScript Object Notation (JSON)-Datei, die im Hintergrund von Vite generiert wird und Informationen über die Webseite enthält, wie den Namen, die Beschreibung, das Icon und die Farbe. Der Service Worker ist ein Skript, das im

Hintergrund ausgeführt wird und die Offline-Funktionalität der App ermöglicht. Nun ist die Webseite als PWA konfiguriert und kann auf dem Homescreen des Smartphones installiert werden. Dadurch wird die Nutzererfahrung verbessert und die Webseite kann auch offline genutzt werden, ähnlich wie eine native App. [2]

6.3 Projekt aufräumen und Strukturierung

Bisher wurden essenzielle Veränderungen am Framework vorgenommen. Bevor es weitergeht, muss das Projekt aufgeräumt und strukturiert werden. Dies beinhaltet das Entfernen von ungenutzten Dateien und das Umbenennen von Dateien, um eine konsistente Namenskonvention zu gewährleisten. Außerdem müssen die Ordnerstruktur und die Dateistruktur überarbeitet werden, um eine bessere Übersichtlichkeit und Wartbarkeit des Projekts zu gewährleisten. Dies beinhaltet das Erstellen von Ordnern für Komponenten, Seiten, Services und andere wichtige Dateien. In Abbildung 6.1 ist die neue Ordnerstruktur zu erkennen. Der gesamte Code des Frontends befindet sich im Ordner *client* auf der Root-Ebene. Innerhalb dessen befinden sich weitere Ordner. Der *src*-Ordner ist hierbei der interessanteste und wichtigste Ordner des client-Ordners. *src* ist eine Abkürzung für *source* und beinhaltet sämtlichen Code für das Frontend-Framework Vue.js, also Definitionen von Komponenten, Logik innerhalb der Komponenten sowie Konstanten, den Router und vieles mehr. In der folgenden Auflistung wird erklärt, was genau die einzelnen Unterordner beinhalten und welchen Part in der neuen UI sie dabei einnehmen:

- **assets**: Der *assets*-Ordner beinhaltet jene Dinge, die nicht durch Quellcode angegeben werden können. Dazu zählen unter anderem Bilder, wofür der *assets*-Ordner im Falle der CO2-Runter-Webseite hauptsächlich genutzt wurde. Die Bilder werden hier an einem zentralen Ort gespeichert, damit sie innerhalb der Komponenten, die die Bilder beispielsweise implementieren, verwendet werden können und nicht überall im Quellcode verteilt sind.

- **components**: Das Herzstück des Projekts ist der *components*-Ordner. In diesem Ordner liegen die Komponenten, aus welchen die finale Webseite später zusammengebaut wird. Man kann sich den *components*-Ordner wie eine Art Baukasten vorstellen, aus dem die verschiedenen Seiten der CO2-Runter-Webseite schlussendlich zusammengebaut werden. Wie bereits in Kapitel 6.6 erwähnt, sind Beispiele für Komponenten unter anderem die QuestionStepper- oder CurrentCO2-Komponente.

- **composables**: Im Kontext von Vue-Anwendungen ist ein Composable eine Funktion, um zustandsbehaftete Logik zu kapseln und wiederzuverwenden. Somit befinden sich im *composables*-Ordner alle Hilfsfunktionen, die helfen, solche Logik zu kapseln und

damit wiederverwendbar zu machen. Ein Beispiel im Sinne der CO2-Runter-Webseite wäre das Composable, das den aktuellen CO2-Wert innerhalb des Rechners nach dem Auswählen einer Antwortmöglichkeit neu berechnet und an das Frontend schickt.

- **constants**: Der *constants*-Ordner ist der Ablageort für Konstanten, die innerhalb der Applikation immer wieder verwendet werden oder Quellcode, der mithilfe von Konstanten einfacher bzw. kompakter dargestellt werden konnte. So werden zum Beispiel die Literaturquellen des FAQs in Konstanten gespeichert.

- **layout**: Wie der Name bereits verrät, werden in diesem Ordner Layouts für die verschiedenen Seiten der Webseite definiert. Da im Kontext der CO2-Runter-Webseite nur ein Layout verwendet wird, ist dieser dementsprechend nur mit einer *Default.vue*-Datei befüllt.

- **plugins**: Plugins sind in sich geschlossene Codes, die in der Regel Funktionen auf Anwendungsebene zu Vue hinzufügen. In unserem Fall musste das Plugin *Vuetify* hinzugefügt werden, da über diese Anwendung Komponenten durch bereits vorher definierten Code aufgebaut wurden. Aus diesem Grund wurde diese Plugininstallationsdatei im *plugins*-Ordner abgelegt.

- **router**: Der router-Ordner beinhaltet die Logik für den sogenannten Router. Ein Router ist dafür zuständig, für die unterschiedlichen Webseiten und teilweise auch für einzelne Abschnitte von Webseiten, spezielle URLs zu definieren. So kann z.B. über die URL ẗhttps://co2runter.karlsruhe.de/rechneräuf den CO2-Rechner zugegriffen werden. Die Datei *routes.ts* speichert hierbei jegliche Routen ab und liegt innerhalb des *router*-Ordners.

- **store**: Ein Store ist ein zentraler Ort, an dem Daten gespeichert werden, die über alle Anwendungskomponenten hinweg verfügbar sind und sein müssen. Jegliche Daten dieser Form werden im *stores*-Ordner untergebracht. Beispiele für einen Store innerhalb der Anwendung sind das Loggen und Abspeichern des CO2-Werts.

- **types**: Wie der Name eventuell bereits verrät, werden im Ordner *types* Objekte, sogenannte Typen, definiert. Die Typen sind hierbei als exportierbare Interfaces implementiert, die später vom Frontend-Code für Logik oder andere Dinge wiederverwendet werden können. Ein Beispiel für einen solchen Typen ist eine Literaturquelle (*LiteratureSource*), die jeweils immer aus fünf Strings besteht: einem Titel, einem Autor, einem Veröffentlicher, einem Veröffentlichungsjahr und einer URL zur Literaturquelle.

- **views**: Innerhalb des *views*-Ordners sind alle erreichbaren Seiten definiert und zu finden. Eine View setzt sich dabei aus den verschiedenen Komponenten zusammen

und sind somit die auf der Webseite zu sehenden Seiten, die durch die verschiedenen URLs (=Routen) aufgerufen werden können.

Zuguterletzt sind noch drei abschließende Dateien im *src*-Ordner: *App.vue* ist die oberste Schicht der UI, auf welcher die ganze Views durch den Router implementiert werden. *main.ts* ist eine weitere Datei, die die Plugins, welche im *plugins*-Ordner liegen, auf der Anwendungsebene registriert. Abschließend implementiert *vite-env.d.ts* die Vite-Umgebung.

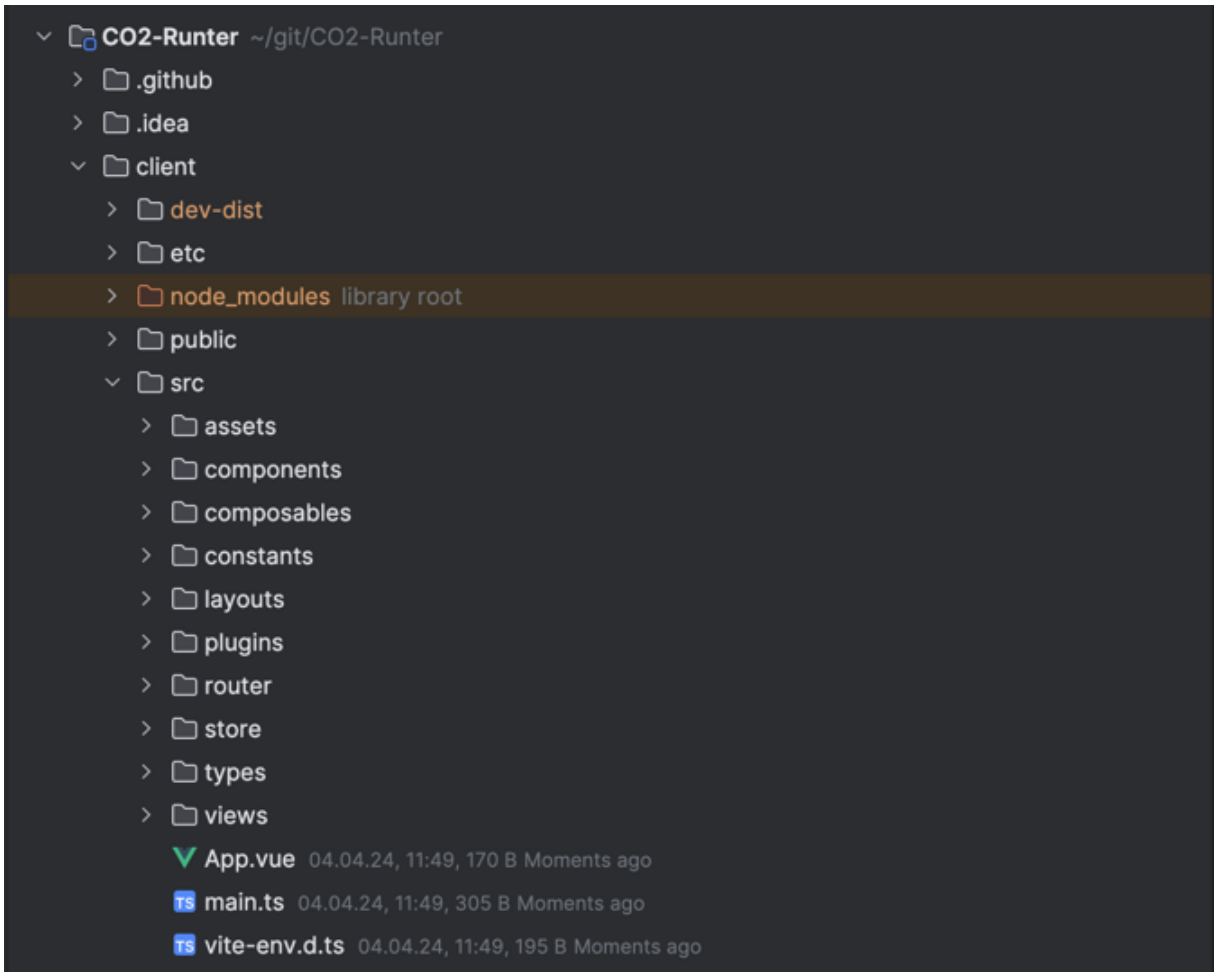

Abbildung 6.1: Neue Ordnerstruktur der CO2-Runter-Webseite

6.4 Anpassung der questions.json

Nachdem nun die Ordnerstruktur des Projekts aufgeräumt und neu strukturiert wurde, wird nun die *questions.json* angegangen. Da die bisherige *questions.json* sehr unübersichtlich und tief verschachtelt war, wurde diese neu aufgebaut und neu strukturiert. Die alte Struktur der *questions.json* wird in Listing 6.9 vereinfacht dargestellt. Dort ist zu beobachten, dass die JSON die Kategorien bereits besitzt, wie sie später im CO2-Rechner angezeigt

werden. Jede Kategorie beinhaltet den Namen der Kategorie sowie eine Liste von Fragen. Innerhalb der Frageliste werden die Fragen erneut über das Attribut *name* in mögliche Unterkategorien unterteilt. Anschließend erkennt man ab Zeile 8 und Zeile 16 der alten questions.json, dass sich hier ein *quick*- und ein *detailed*-Objekt eröffnen. Diese beinhalten jeweils die eigentlichen Fragen, die im CO2-Rechner verwendet werden. Standardmäßig werden die Fragen und Antworten aus dem *quick*-Objekt geladen. Schaltet die NutzerIn jedoch die Frage auf detailliert um, wie es beim alten CO2-Rechner beispielsweise in Abbildung 2.2 zu sehen ist, werden die Fragen aus dem *detailed*-Objekt geladen.

```
1  {
2      "baseline": number,
3      "category":[
4          {
5              "name": string,
6              "questions": [
7                  "name": string,
8                  "quick": {
9                      "text": string,
10                     "typ": string,
11                     "replies": [],
12                     "values": [],
13                     "defaultValue": number,
14                     "selectedValue": number
15                 },
16                 "detailed": {
17                     "questions": {
18                         "text": string,
19                         "typ": string,
20                         "replies": [],
21                         "values": [],
22                         "defualtValue": number,
23                         "selectedValue": number
24                     },
25                 },
26             ],
27             "formula": string,
28         },
29         {
30             ...
31         }
32     ]
33 }
```

Listing 6.8: Aufbau der alten questions.json

Da im neuen CO2-Rechner keine Möglichkeit mehr geboten wird, zwischen detaillierten und kurzen Fragen umzuschalten, enthält die alte, verschachtelte JSON viele Dinge, die gar nicht mehr gebraucht werden. Aus diesem Grund wurde die *questions.json* neu aufgebaut und strukturiert. Listing 6.10 zeigt den neuen Grundaufbau der *questions.json*, die im neuen CO2-Rechner zukünftig verwendet wird.

```
{
    "baseline": number,
    "category": [
        {
            "name": string,
            "questions": [
                {
                    "text": string,
                    "replies": [
                        {
                            "text": string,
                            "value": number
                        },
                        {
                            ...
                        },
                    ],
                    "selected": {
                        "text": string,
                        "value" number
                    }
                },
            ],
            "formula": string
        },
        {
            ...
        }
    ],
    "endFormula": string
}
```

Listing 6.9: Aufbau der neuen question.json

Die neue JSON zeichnet sich dadurch aus, dass die Antworten einer Frage in Objekten aus einem *text*- und einem *value*-Objekt gespeichert sind. Dadurch kann bei der Berechnung des CO2-Werts der Wert einer Antwortmöglichkeit, die im Frontend ausgewählt wurde, leichter gefunden werden und man muss sich nicht ewig durch die Datenstruktur der JSON quälen. Zusätzlich wurde, wie bereits vorher erwähnt, die Fragenunterteilung in *quick* und

detailed entfernt, sodass man insgesamt eine verständlichere Datenstruktur hat. Somit wurde die Anforderung aus dem Kapitel 5.1.3 - [R14] Question.json anpassen erledigt.

6.5 Erstellung der neuen Landing Page

Die Landing Page ist die erste Seite, die die NutzerInnen sehen, wenn sie die Webseite besuchen. Sie ist somit das Aushängeschild der Webseite und sollte die NutzerInnen dazu animieren, die Webseite zu nutzen. Die Landing Page sollte also ansprechend und informativ sein, um die NutzerInnen zu überzeugen, die Webseite zu nutzen, und gleichzeitig die zuvor definierte Anforderung [**R04**] erfüllen. Die neue Landing Page wird mit **Vuetify** erstellt, einem Material Design Framework für Vue.js. Vuetify bietet viele vorgefertigte Komponenten, die es ermöglichen, schnell und einfach ansprechende Webseiten zu erstellen. Die Landing Page könnte wie folgt aussehen:

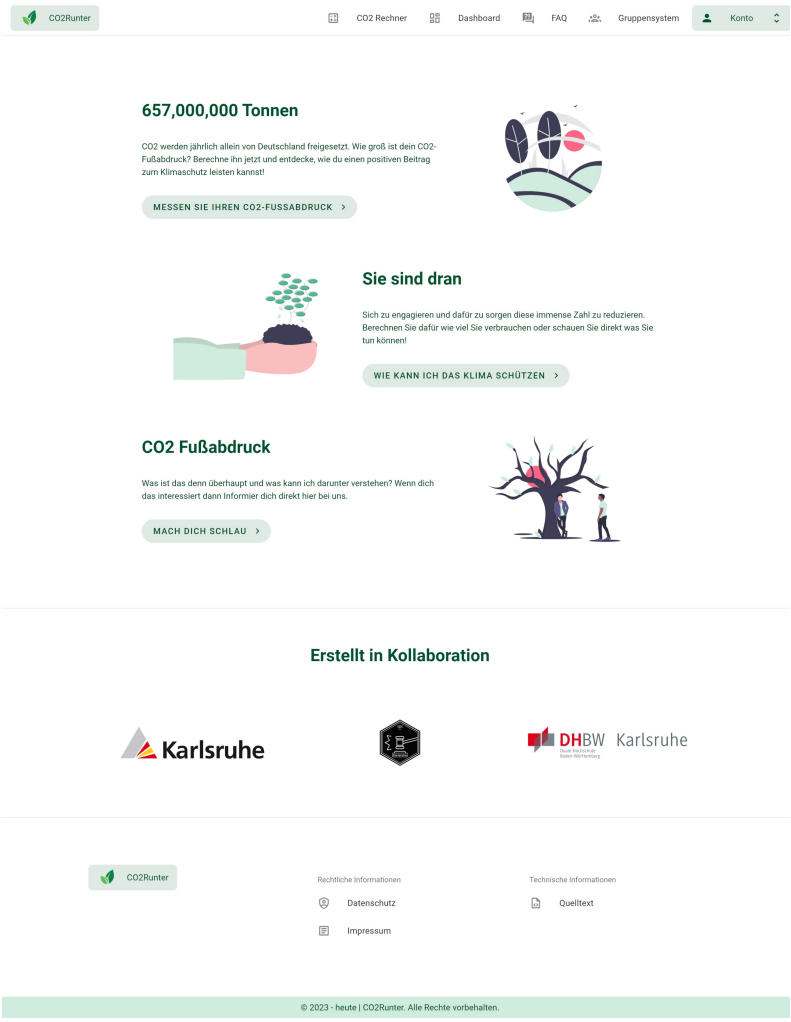

Abbildung 6.2: Die neu erstellte CO2-Runter Landing Page

Das Design, das in der vorheriger Abbildung zu sehen war, wurde anhand der Evaluierung der Umfrage und der erstellten Personas erstellt. Dafür wurde das Tool Figma genutzt und auch direkt mit Vuetify Komponenten gearbeitet, wodurch die Entwicklungsarbeitszeit immens verkürzt wurde. Das ist deshalb der Fall, da das Design und die Komponenten im späteren Implementierungsverlauf 1:1 übernommen werden konnten.

Die neue Homepage kann man in drei Sektionen unterteilen. Einmal die Navigationsleiste, den Kontent der Landing Page und den Footer mit deren relevanten Informationen. Zu beachten ist hierbei, dass die Navigationsleiste, als auch der Footer, auf allen Seiten identisch sein sollen. Einzig der Kontent einer Seite ist dynamisch und soll deshalb veränderbar sein. Daher werden die drei Sektionen im folgenden nacheinander durchgearbeitet. Zuvor aber werden wichtige Änderungen vorgenommen.

6.5.1 Änderungen

Eine wichtige Änderung ist die Anpassung des Farbschemas, um die Webseite ansprechender zu gestalten. Dafür wird das Farbschema von Vuetify angepasst, um eine konsistente Farbgebung zu gewährleisten. Hierfür erstellen wir lediglich nach der Vuetify-Dokumentation ein neues Farbschema und passen die Farben an. Dieses Farbschema wird dann in der Vuetify-Konfiguration eingebunden. Das Farbschema sieht dann wie folgt aus: [41]

```
export default createVuetify({
    theme: {
        themes: {
            dark: false,
            colors: {
                primary: '#D0ECDE',
                'primary-darken-1': '#01502D',
                ...
            },
            variables: {
                ...
            },
        },
    },
});
```

Listing 6.10: Vuetify Farbschema anpassung

Nicht nur die Farben, sondern auch das Layout sollte angepasst werden. Dafür wird ein Layout erstellt, das auf allen Seiten gleich ist und nur der Kontent variabel ist. Dieses

Layout wird dann in jeder Seite durch die Vue Router Funktionalität eingebunden. Das Einbinden sieht wie folgt aus:

```
const routes = [
    {
        path: '/',
        component: () => import('@/layouts/default/Default.vue'),
        children: [
            {
                path: '',
                name: 'Home',
                component: () => import('@/views/HomeView.vue'),
            },
        ],
    },
    ... weitere Routen ...
];
```

Listing 6.11: Einbindung des Layouts in der Vue Router Konfiguration

Damit das Einbinden richtig funktioniert, muss in der *Default.vue* Datei mindestens ein *<router-view />* Tag existieren, damit die jeweilige Seite eingebunden und angezeigt wird. Die angepasste Layout der CO2-Runter-Webseite beinhaltet, dass auf jeder erreichbaren Seite die Navigationsleiste, der Kontent und der Footer angezeigt wird. Dieses Layout wird dann in der *App.vue* Datei wiederum über ein *<router-view />* Tag eingebunden und sieht wie folgt aus:

```
<template>
    <v-app>
        <TheNavigationHeader />
        <RouterView />
        <TheFooter />
    </v-app>
</template>
```

Listing 6.12: Layout Definition

Nachdem diese Schritte befolgt wurden und jede Seite das Layout eingebunden hat, sowie das Farbschema zu jeder Webseite angepasst wurde, kann sich nun um die restliche Komponenten gekümmert werden. Dabei handelt es sich um die Komponenten der Navigationsleiste, des Kontents und des Footers.

6.5.2 Navigationsleiste

Eine Navigationsleiste dient den NutzerInnen einer Webseite dazu, den Überblick über den aktuell zu sehenden Bereich einer Webseite zu behalten und bietet eine schnelle und einfache Navigation über die Webseite. Im Fall der CO2-Runter-App gibt es die Startseite, den CO2-Rechner, das Dashboard, sowie eine FAQ- und Gruppensystemseite. Diese sollen über die Navigationsleiste erreichbar sein.

```
<template>
    <v-app-bar :elevation="0">
        <v-app-bar-nav-icon @click="isDrawerOpen␣=␣!isDrawerOpen" />

        <v-toolbar-title>
            <Co2RunterLogo />
        </v-toolbar-title>

        <div>
            <Navigationsleiste />
        </div>
    </v-app-bar>

    <v-navigation-drawer v-model="isDrawerOpen">
        ...
    </v-navigation-drawer>
    <PWAInstallationDialog />
</template>
```

Listing 6.13: Navigationsleiste für Web als auch Mobile

6.5.3 Footer

Ein Footer ist das Gegenstück zur Navigationsleiste und ist immer ganz am Ende bzw. am unteren Bereich einer Webseite zu finden. Auch der Footer dient dazu, Informationen darzustellen und besser über eine Webseite navigieren zu können. Um den Footer zu erzeugen, gibt es drei Bereiche. Einmal die Logos der Partner, dann die rechtlichen Informationen und die technischen Informationen und am Ende das Copyright.

```
<template>
    <div>
        <h1 class="text-primary-darken-1">Erstellt in Kollaboration</
            h1>
        <v-container class="my-16">
            <Logos />
```

```
6           </v-container>
7           <v-container class="my-16">
8               <Datenschutz-Impressum />
9           </v-container>
10      </div>
11
12      <v-footer>
13          <div class="text-primary-darken-1">
14              2023 - heute | CO2Runter. Alle Rechte vorbehalten.
15          </div>
16      </v-footer>
17  </template>
```

Listing 6.14: Footer Definition

6.5.4 Kontent

Der Hauptpart der neuen Landing Page ist selbstverständlich der Kontent. Hier ist es besonders wichtig, die NutzerInnen direkt anzusprechen und die Seite attraktiv und interessant wirken zu lassen. Aus diesem Grund wurde sich für ein Design entschieden, das die wichtigsten Aspekte der Webseite mit interessanten Einführungstexten in Reihen auf der Landing Page zeigt. Jede Reihe ist unterteilt in einen linken und einen rechten Part, welche jeweils abwechselnd mit Text bzw. mit einem passenden Bild gefüllt sind. Dabei wechselt sich die Position des Textes und des Bilds nach jeder Reihe ab, sodass man auch ohne Trennlinien das Layout gut erkennen kann und dem Auge der NutzerInnen nicht langweilig wird. Der Textpart ist hierbei immer mit einer kleinen Überschrift, einem Einführungstext sowie einem Button gefüllt. Die Überschriften der insgesamt drei Reihen sind dabei so gewählt, dass die NutzerInnen angesprochen werden oder zumindest ein gewisses Grundinteresse geweckt wird. Im Einführungstext unter der Überschrift wird dieses Konzept aufgegriffen und weiter über das angesprochene Thema informiert. Durch den Button am Ende des Einführungstexts gelangen die NutzerInnen direkt auf die angesprochene und verlinkte Webseite und können sich unter anderem ihren CO2-Wert berechnen lassen oder sich weiter über den Klimaschutz auf der FAQ-Seite informieren. Das Bild ist hingegen so gewählt, dass es zur jeweiligen Überschrift bzw. zum Thema des Absatzes passt.

Grundsätzlich ist auch der Kontent in den Hauptfarben der CO2-Runter-Webseite gehalten. Jedoch wurden die Überschriften und Texte innerhalb der Buttons etwas dunkler und auffälliger gestaltet, da diese ja vom Auge der NutzerInnen direkt registriert werden sollen. Dadurch wird der optimale Fokus gelenkt. In der folgenden Abbildung ist der Code für den Kontent zu sehen:

```
<template>
    <v-container justify="center">
        <v-row class="d-flex flex-column-reverse flex-md-row my-16">
            <v-col cols="12" md="7" class="text-center text-md-start"
                >
                <h1 class="text-primary-darken-1">657,000,000 Tonnen
                    </h1>
                <p class="text-secondary my-8">
                    Text1
                </p>
                <v-btn
                    variant="tonal"
                    :rounded="true"
                    color="primary-darken-1"
                    append-icon="mdi-chevron-right"
                    size="large"
                >
                    Aktion 1
                </v-btn>
            </v-col>
            <v-col class="d-flex align-center justify-center" cols="
                12" md="5">
                <v-img
                    width="360px"
                    height="200px"
                    src="../assets/undraw_nature_m5ll.svg"
                />
            </v-col>
        </v-row>
        ... weitere Reihen ...
    </v-container>
</template>
```

Listing 6.15: Home Page Kontent

6.6 Neuerstellung des CO2-Rechners

Der CO2-Rechner ist ein wichtiges Feature der Webseite, da den NutzerInnen dadurch ermöglicht wird, ihren CO2-Fußabdruck zu berechnen. Bei der Neuerstellung hat man sich an der bisherigen Implementierung orientiert und diesen Workflow des CO2-Rechners beibehalten. Dadurch konnte dann die Anforderung **R09** erfüllt werden. Der CO2-Rechner besteht aus mehreren Kategorien, die die NutzerInnen durchlaufen müssen, um ihren CO2-Fußabdruck zu berechnen. Die Kategorien lauten: Mobilität, Ernährung, Wohnen und

Konsum. Die NutzerInnen müssen für jede Kategorie verschiedene Fragen beantworten, um ihren CO2-Fußabdruck zu berechnen. Die Fragen sind so gestaltet, dass die NutzerInnen diese leicht beantworten können. Die Fragen sehen wie folgt aus:

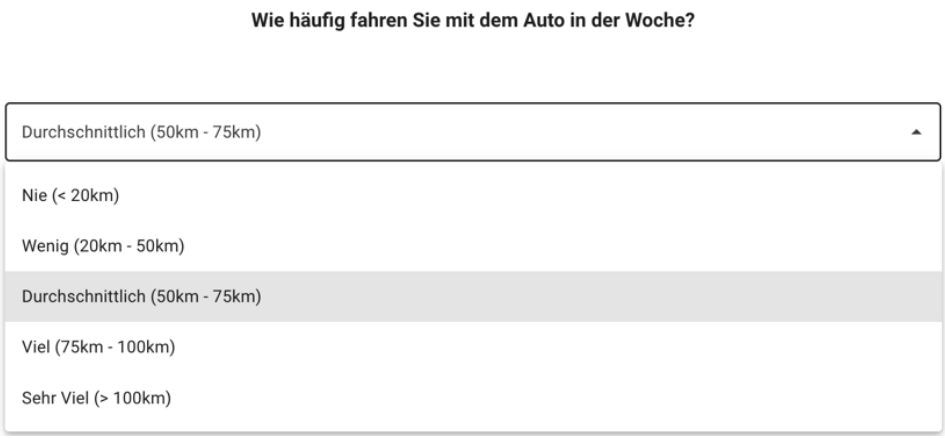

Abbildung 6.3: Design einer Frage des neuen CO2-Rechners

Besonders viel hat sich nicht am Design einer Frage innerhalb des Rechners geändert. Es wurden jedoch mehr Fragen in den Rechner eingebaut, die im alten Design als „detailliert“ beschrieben wurden. Zusätzlich wurde, wie bereits im Abschnitt 6.4 Anpassung der questions.json, der Wert hinter den Anwortmöglichkeiten angepasst sowie die gesamte Struktur des generierten JSON.

Die größte Veränderung erlebt der neue CO2-Rechner in Sachen Allgemeindesign und Nutzerfreundlichkeit. Wie in Abbildung 6.4 zu sehen ist, wurde hier besonders darauf geachtet, der NutzerInnen eine ansprechende Webseite zu präsentieren, die nicht nur als langweiliger und stumpfer Fragebogen durchgeht.

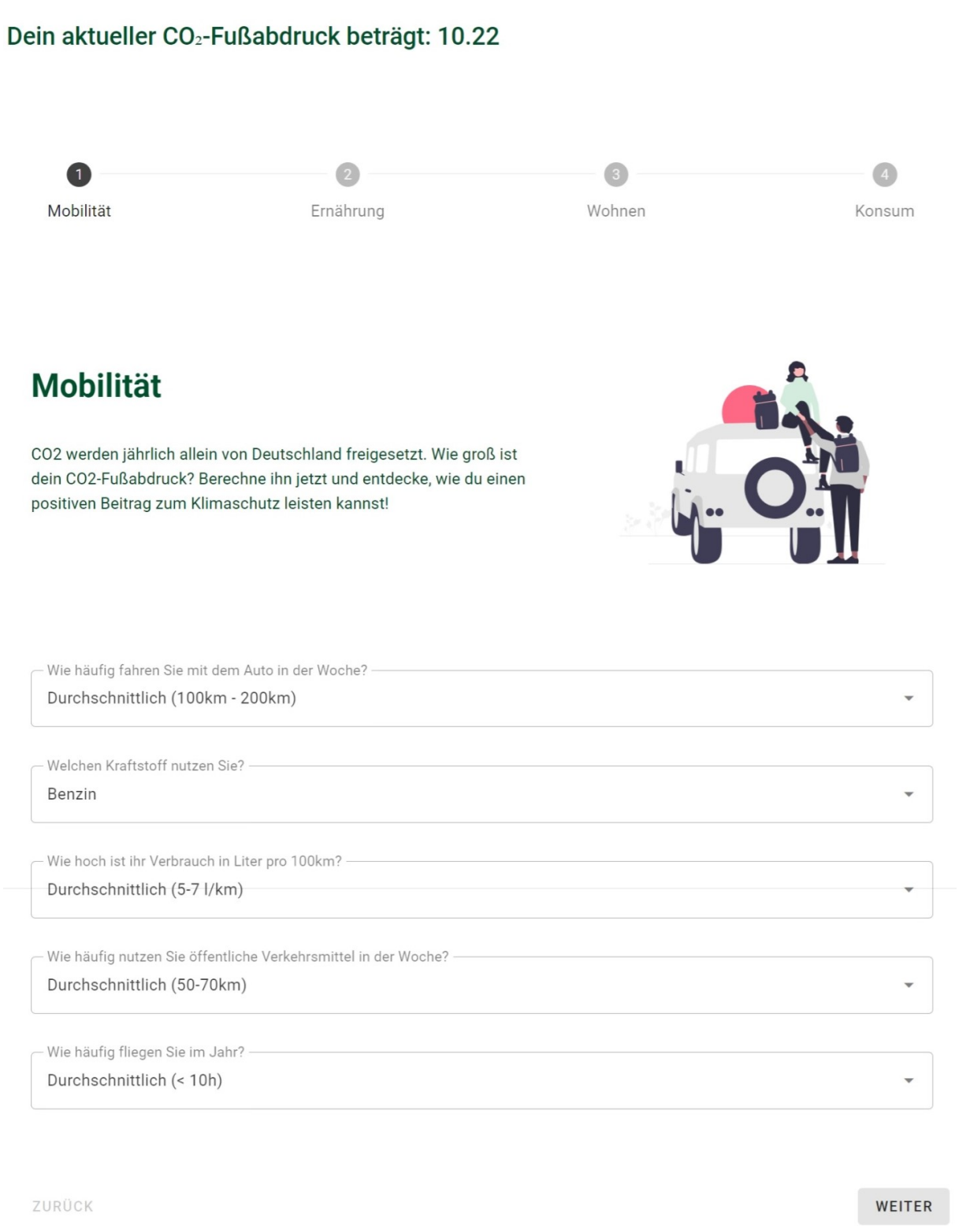

Abbildung 6.4: Design des neuen CO2-Rechners

Der berechnete CO2-Wert aus den gewählten Antwortmöglichkeiten befindet sich nun direkt am Start der Webseite. Um den berechneten Fußabdruck noch deutlicher in den Vordergrund zu setzen, wurde diesem die Primärfarbe verpasst, so dass er in einem dunkelgrün erscheint.

Eine weitere grundlegende Veränderung des neuen CO2-Rechners findet man unmittelbar unterhalb der Kategorienauflistung. Bevor die NutzerInnen auf den eigentlichen Rechner stoßen, wurde für jede Kategorie ein kleiner Einführungsabschnitt geschrieben und ein dazu passendes Bild hinzugefügt. Diese Neuerung soll dafür sorgen, dass sich die NutzerInnen nicht direkt in den Rechner stürzen, sondern über das aktuelle Thema aufgeklärt und informiert werden. Außerdem bringt der Abschnitt mit dem Bild einen positiven Aspekt im Bereich der Nutzerfreundlichkeit.

Für die Implementierung des neuen CO2-Rechner wurde eine neue Komponente namens *CalculatorView.vue* erstellt. Diese bildet die Hauptansicht des Rechners und implementiert weitere, kleinere Komponenten, so genannte *Components* und ist in der folgenden Abbildung zu sehen.

```
<template>
    <v-container>
        <v-row
        class= ...
        >
            <v-col ...>
                <CurrentCO2 />
            </v-col>
            <v-col ... >
                <v-img
                width="360px"
                height="200px"
                src="../assets/undraw_nature_m5ll.svg"
                />
            </v-col>
        </v-row>

        <v-row ... >
            <v-col>
                <QuestionStepper />
            </v-col>
        </v-row>
    </v-container>
</template>
```

Listing 6.16: Aufbau der CalculatorView.vue Komponente

Wie bereits eingangs erwähnt, implementiert die CalculatorView-Komponente weitere Components. In der Ordnerstruktur des Projekts findet man deshalb innerhalb des Components-Ordner einen Ordner mit dem Namen *Calculator*. In diesem Ordner befinden sich alle weiteren Komponenten, die für die Darstellung des CO2-Rechners benötigt werden. So

auch die Komponenten *QuestionStepper* und *QuestionBlock*, die durch die entsprechenden ".vue" Dateiendungen erzeugt werden und bereits in der vorherigen Abbildung erschienen sind.

Der QuestionStepper ist für die Darstellung des eigentlichen Fragebogens zuständig. Der Fragebogen wird über den *v-stepper* implementiert. Dieser wird von Vuetify geliefert und ist in zwei Teile aufgeteilt: v-stepper-header und v-stepper-window. Der *v-stepper-header* ist für die Kategorienauflistung zuständig und begleitet die NutzerInnen durch den Fragebogen. Durch den *v-stepper-header* wird genau angezeigt, in welcher Kategorie des Fragebogens man sich gerade befindet und wie viele Kategorien es noch zu erledigen gibt. Auf der anderen Seite ist das *v-stepper-window* für die Auflistung der Fragen und somit für den Fragebogen zuständig. Über so genannte *v-stepper-window-items* werden die einzelnen Fragen dargestellt. Für das Laden der Fragen und Antworten ist eine weitere Komponente zuständige, die in QuestionStepper.vue importiert wird: *QuestionBlock*

QuestionBlock ist dafür zuständig, die Fragen und Antworten aus der *questions.json* zu laden und diese so darzustellen, dass QuestionStepper diese nur noch im richtigen Format einfügen muss. Nachfolgend ist der Code für QuestionBlock.vue und QuestionStepper.vue aufgelistet:

```
<v-stepper>
    <v-stepper-header>
        <v-stepper-item title="Mobilitaet" value="1"/>
        <v-stepper-item title="Ernaehrung" value="2"/>
        <v-stepper-item title="Wohnen" value="3"/>
        <v-stepper-item title="Konsum" value="4"/>
    </v-stepper-header>
    <v-stepper-window>
        <v-stepper-window-item>
            ...
            <v-row>
                <v-col>
                    <QuestionsBlock :category-index="QuestionsIndices
                        .MOBILITY"/>
                </v-col>
            </v-row>
            ...
        </v-stepper-window-item>
    </v-stepper-window>
</v-stepper>
```

Listing 6.17: QuestionStepper.vue

```
<div
    v-for="(item, index) in questionStore.category[props.
        categoryIndex].questions"
    :key="index"
>
    <v-select
        v-model="item.selected"
        :items="item.replies"
        item-title="text"
        :label="item.text"
        variant="outlined"
        :return-object="true"
        @update:modelValue="totalCo2EmissionStore.
            calculateCo2ValuesPerCategory()"
    />
</div>
```

Listing 6.18: QuestionBlock.vue

Die Abbildungen sind hierbei vereinfacht dargstellt und sollten detailliert in der Codebase genauer betrachetet werden. Die Berechnung des Fußabdrucks wird nun innerhalb der Datei *totalCo2Emission.ts* berechnet. Diese ist ein Store und berechnet die Werte aus den gewählten Antworten zusammen. Abschließend werden die Werte der Kategorien mit der Baseline addiert, um den endgültigen CO2-Fußabdruck zu berechnen. Wie die Werte innerhalb der Kategorien genau aufgeteilt sind, ist im Kapitel 6.4 genauer beschrieben. Die einzelnen Formeln sind auch hier in der Codebase festgehalten.

Vereinfacht kann man sagen, dass der totalCo2Emission-Store durch die einzelnen Objekte und Werte so lange durchitteriert und die Werte der gewählten Anworten pro Kategorie sammelt. Trifft die Iteration auf den Abschnitt "formula", wird die Formel genutzt um den gesamten CO2-Wert der Kategorie zu berechnen. Abschließend werden die Kategoriewerte in die Variable *totalEmission* geladen und somit der gesamte Wert des CO2-Fußabdrucks gespeichert. Auf diese Weise wurde R02 - Berechnung des CO2-Fußabdrucks umgesetzt. Die folgende Abbildung zeigt einen Ausschnitt aus dem totalCo2Emission-Store, der für die Berechnung der Werte pro Kategorie zuständig ist.

```
calculateCo2ValuesPerCategory() {
        const questionStore = useQuestionStore();
        this.base = questionStore.baseline;

        const categoryTotals = questionStore.category.map((category)
            => {
            const questionValues = category.questions.map((question)
                => {
```

```
7                   if (question.formula) {
8                       let result;
9                       try {
10                          const calculateQuestionValue = eval(
11                              `(${question.formula})`
12                          );
13                          result = calculateQuestionValue(
14                              question.selected.value
15                          );
16                      } catch (e) {...}
17                      return result;
18                  } else {
19                      return question.selected.value;
20                  }
21              });
22
23              const calculateCategoryTotal = eval(`(${category.formula
                    })`);
24              const categoryTotal = calculateCategoryTotal(
                    questionValues);
25              return {
26                  categoryName: category.name,
27                  totalEmission: categoryTotal,
28              };
29          });
```

Listing 6.19: Ausschnitt aus dem totalCo2Emission.ts - Store

Nach dem Beenden des Fragebogens werden die NutzerInnen auf eine Zusammenfassungsseite weitergeleitet. Auf dieser Seite können die NutzerInnen ihre Daten einem Stadteil Karlsruhes zuordnen oder eine Gruppe auswählen, mit der sie ihre Daten teilen möchten. Sollten die NutzerInnen einen Stadtteil auwählen, wird der CO2-Fußabdruck auf diesen berechnet und der Durchschnittswert des Stadtteils geupdatet. Jedoch besteht auch die Möglichkeit, fortzufahren ohne die Daten zu teilen. Die Abfrage nach der Datenverarbeitung sowie die Zuordnung eines Stadtteils sind in der folgenden Abbildung zu sehen:

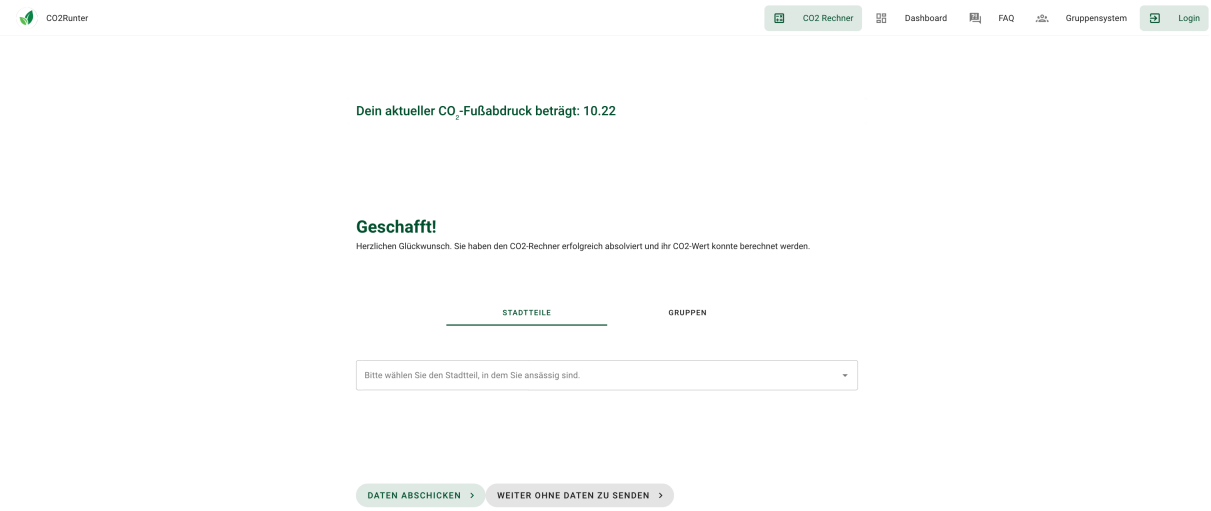

Abbildung 6.5: Datenverarbeitungsabfrage nach Beenden des CO2-Fragebogens

Unabhängig davon, ob die NutzerInnen die Daten teilen oder nicht, wird eine persönliche Zusammenfassungsseite generiert, die die berechneten Werte der einzelnen Kategorien und den Gesamtwert des CO2-Fußabdruck noch einmal präsentiert. Den NutzerInnen werden die Werte der einzelnen Kategorien hierbei in einem Kuckendiagramm aufgezeigt. Eine beispielhafte Zusammenfassungsseite zeigt die nächste Abbildung:

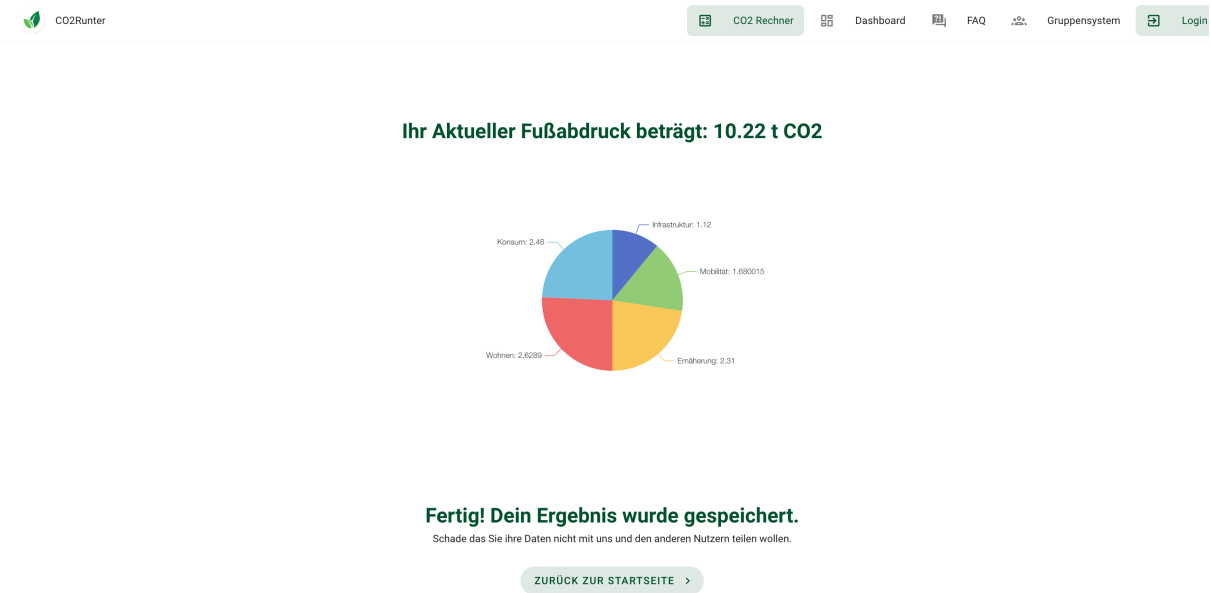

Abbildung 6.6: Persönliche Zusammenfassungsseite

Sollten sich die NutzerInnen dazu entscheiden, die Daten zu teilen, werden diese an den Server gesendet und innerhalb einer Datenbank gespeichert. Dazu müssen die Daten zuerst

einem Stadtteil bzw. einer Gruppe zugeordnet werden. Um die Namen der Stadtteile und der Gruppen anzuzeigen, wird eine Anfrage an den Server geschickt und diese Daten gefetcht. Dies erfolgt mithilfe einer einfachen GET-Anfrage an die Backend-API. Die folgende Abbildung zeigt zwei Methoden, die sich um das fetchen der Daten kümmert.

```
const fetchCityDistricts = async () => {
    isLoadingCityDistricts.value = true;
    try {
        const response = await fetch('/api/districts', {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                co2token: `${localStorage.getItem('CO2Token')}`,
            },
        });

        if (!response.ok) { throw new Error(response.status); }

        const data = await response.json();
        cityDistricts.value = data as Array<CityDistrict>;
    } catch (e) {...}
    isLoadingCityDistricts.value = false;
};

const fetchGroups = async () => {
    isLoadingGroups.value = true;
    try {
        const response = await fetch('/api/groups/member', {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                co2token: `${localStorage.getItem('CO2Token')}`,
            },
        });

        if (!response.ok) { throw new Error(response.status); }

        const data = await response.json();
        groups.value = data as Array<GroupData>;
    } catch (e) {...}
    isLoadingGroups.value = false;
};
```

Listing 6.20: Laden der Stadtteile und Gruppen aus dem Backend

Das Abschicken der Werte des errechneten CO2-Fußabdrucks erfolgt über die Methode *submitCo2EmissionUpload()*. Dort wird zuerst überprüft, ob ein Stadtteil ausgewählt wurde. Wurde ein Stadtteil ausgewählt, wird die ID des Stadtteils gespeichert und später in die gesendeten Daten übergeben, sodass der CO2-Fußabdruck später im Dashboard dem richtigen Stadtteil zugeordnet werden kann. Andernfalls wird als Wert 0 übergeben. Nun wird der eigentliche Part der Methode, nämlich die Anfrage an den Server, aufgebaut. Dazu wird ein Fetch aufgebaut, der als Parameter eine Anfragemethode, einen Anfrageheader und den eigentlichen Body beinhaltet. Als Anfragemethode wird eine POST-Anfrage definiert, die als Header den Content-Type *application/json* besitzt. Im Body der Anfrage werden die eigentlich interessanten Daten gesendet. Diese werden als JSON gesendet und beinhalten die ausgewählten Gruppen, die ID des ausgewählten Stadtteils und die CO2-Werte jeder Kategorie aus dem Rechner. Abschließend werden diese an die URI */api/footprint* gesendet. Die nachfolgende Abbildung zeigt die implementierte Methode:

```
const submitCo2EmissionsUpload = async () => {
    try {
        const districtId = selectedDistricts.value
            ? selectedDistricts.value.district_ID
            : 0;

        const response = await fetch('/api/footprint', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                groups: finalSelectedGroups.value,
                district: districtId,
                data: [
                    categories.mobility,
                    categories.housing,
                    categories.consume,
                    categories.nutrition,
                ],
            }),
        });

        if (!response.ok) { throw new Error(response.status); }
        updateDataSend(true);
        router.push('/rechner/submitted');
    } catch (e) {...}
};
```

Listing 6.21: submitCo2EmissionUpload()-Methode

Die gerade vorgestellten Methoden um Daten aus dem Backend zu Laden und den CO2-Fußabdruck abzusenden sind alle Teil der QuestionStepper-Komponente. Mithilfe der Methoden wurde erfolgreich die Anforderung R03 in den neuen Rechner implementiert.

6.7 Übernahme und Anpassung des Dashboards

Das Dashboard spielt eine entscheidende Rolle auf der Webseite, indem es den NutzerInnen ermöglicht, CO2-Emissionen zu verfolgen, zu analysieren und zu reduzieren. Es präsentiert den NutzerInnen eine Vielzahl von Informationen über die Stadtteile von Karlsruhe. Insbesondere bietet das Dashboard eine interaktive Karte, die die Stadtteile von Karlsruhe und den durchschnittlichen CO2-Verbrauch gemäß den Angaben der NutzerInnen, die ihren Stadtteilen zugeordnet sind, darstellt. Um die **Anforderung** [**R05**] zu erfüllen, wird das Dashboard für die neue Webseite neu entwickelt und an das Framework sowie an die Benutzeroberfläche angepasst. Da dieses Feature bereits in der vorherigen Webseite implementiert war, kann darauf aufgebaut werden, um die **Anforderung** [**R05**] zu erfüllen. Das Dashboard wird in mehrere Komponenten unterteilt, die jeweils für einen bestimmten Teil des Dashboards verantwortlich sind.

Die Herausforderung besteht darin, die Funktionsweise des React-Codes in den neuen Vue-Code zu überführen und sicherzustellen, dass alles einwandfrei funktioniert. Die Bibliothek ECharts, die für die Grafiken genutzt wurde, kann auch problemlos in Vue verwendet werden, wodurch die gleichen Grafikoptionen wie zuvor verfügbar sind. Es müssen lediglich die Daten aus der Datenbank abgerufen und entsprechend angepasst werden, um die **Anforderung** [**R01**] zu erfüllen. Anschließend müssen verschiedene Anpassungen vorgenommen werden, um sicherzustellen, dass alle Funktionalitäten problemlos funktionieren. Die Umsetzung ist weniger kompliziert, als es zunächst erscheinen mag, erfordert jedoch Zeit, um die Integration in das neue Framework zu vollenden und den bestehenden React-Code zu verstehen.

6.7.1 React Echarts alternative für Vue.js

Der erste Schritt vor der eigentlichen Implementierung besteht darin, eine alternative Bibliothek zu finden, die ähnliche Grafiken wie ECharts bzw. React-Echarts erstellen kann, die in der alten Webseite genutzt wurden. Da ECharts so populär ist, gibt es glücklicherweise auch eine Vue-ECharts Variante, was die Arbeit erleichterte, da die essentiellen Konfigurationen für die dargestellten Grafiken gleich blieben. Jetzt muss lediglich der Installationsprozess der Bibliothek durchgeführt werden, indem man die Bibliothek installiert und in die Vue-Komponenten einbindet.

```
npm i echarts vue-echarts
```

Listing 6.22: Installation von Vue-ECharts

Danach kann die Bibliothek in den Vue-Komponenten eingebunden werden und die Grafiken erstellt werden. [3]

6.7.2 Integration von Echtzeitdaten

Wie bereits bei der Implementierung des neuen CO2-Rechners sind Echtzeitdaten aus der Datenbank relevant. Dafür muss sich wie zuvor am bisherigen Code orientiert und die Datenbankabfragen angepasst werden. Zum Beispiel für die Fußabdrücke der NutzerInnen könnte dies wie folgt aussehen:

```
const fetchFootprints = async () => {
    const response = await fetch('/api/dashboard/footprints');
    const data: FootprintResponse = await response.json();
    return data;
};
```

Listing 6.23: Laden der Fußabdrücke der NutzerInnen

Hier wird eine Funktion erstellt, die die Fußabdrücke der NutzerInnen aus der Datenbank abruft. Die Funktion kann dann in einer **onMounted** Funktion aufgerufen werden, damit die Daten beim Laden der Komponente angezeigt werden.

```
onMounted(async () => {
    footprintsData.value = await fetchFootprints();
});
```

Listing 6.24: Bei laden der Komponente die Fußabdrücke der NutzerInnen speichern

6.7.3 Erstellung der Karte mit Vue-ECharts

Um die Karte zu erstellen, wird wie zuvor erklärt die Vue-ECharts Bibliothek verwendet, die es ermöglicht, interaktive Karten zu erstellen. Die Karte soll die Stadtteile von Karlsruhe und den durchschnittlichen CO2-Verbrauch gemäß den Angaben der NutzerInnen, die ihren Stadtteilen zugeordnet sind, anzeigen. Die Karte wird in einer Vue-Komponente erstellt und die Daten aus der Datenbank abgerufen und entsprechend angepasst, um die Karte zu erstellen. Die Karte könnte wie folgt aussehen:

```
<v-chart :option="chartOptions" style="height: 600px; width: 100%"/>
```

Listing 6.25: Vue-ECharts Diagramm Beispiel

Es ist relativ einfach, eine Karte mit Vue-ECharts zu erstellen, aber das Wichtigste ist, dass die Daten in das Diagramm gegeben werden und dass angegeben wird, wie das Diagramm aussehen soll (ob Karte, Balkendiagramm oder viele andere Möglichkeiten). Wichtig ist vor dem Eingehen auf die **chartOptions**, dass das Diagramm eine Höhe und Breite erhält, um korrekt angezeigt zu werden.

Bei den **chartOptions** handelt es sich um ein Objekt welches die Daten für die Grafik enthält. Hierbei wird die Karte erstellt und die Daten aus der Datenbank abgerufen und entsprechend angepasst, um die Karte zu erstellen. Dabei wird zunächst die **onMounted** Funktion erweitert. Sie soll nicht nur die Daten aus der Datenbank laden sondern auch direkt die konkreten Kartendaten erstellen. Danach sieht die Funktion wie folgt aus:

```
onMounted(async () => {
    footprintsData.value = await fetchFootprints();
    chartOptions.value = getData();
});
```

Listing 6.26: Laden der Fußabdrücke und erstellen der Diagramm Daten und Konfiguration

Die **getData** Funktion ist dafür zuständig die Daten aus der Datenbank in ein Format zu bringen welches von der Vue-ECharts Bibliothek verstanden wird. Hierbei wird die Karte erstellt und die Daten aus der Datenbank abgerufen und entsprechend angepasst, um die Karte zu erstellen. Dabei konnten wir die bisherigen Konfigurationen von React-ECharts übernehmen und mussten lediglich die Daten anpassen. Um einen Einblick zu erhalten, was die **getData** beinhaltet, hier ein Beispiel:

```
function getData() {
    return {
        title: {
            text: 'CO2 Emissionen in Karlsruhe pro Stadtteil',
        },
        ...
        series: loading.value // Die Daten
            ? [
                {
                    name: "Mobilitaet",
                    type: 'map',
                    map: 'Karlsruhe',
                    showLegendSymbol: false,
```

```
                        emphasis: {
                            label: {
                                show: true,
                            },
                        },
                        data: footprintsData.value!.mobility,
                    }, // weitere Kategorien folgen
                ] : [],
    };
}
```

Listing 6.27: Beispiel Konfiguration für ECharts Diagramme

Dieses Vorgehen konnte dann bei allen Grafiken und Diagrammen durchgeführt werden, um das Dashboard erfolgreich in die neue Webseite zu integrieren. Im Folgenden ist ein Screenshot des Kartendiagramms in der neuen Webseite zu sehen:

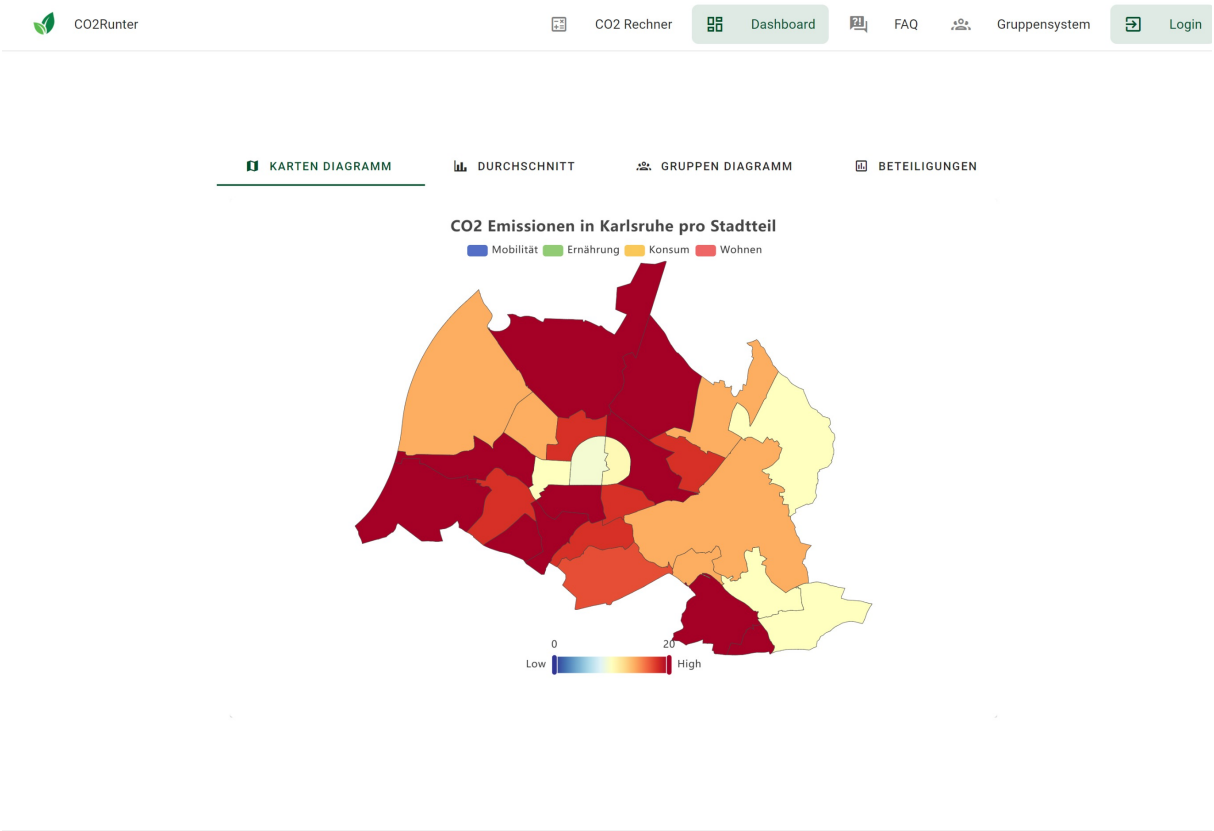

Abbildung 6.7: Kartendiagramm des Dashboards in der neuen Webseite

Dadurch können NutzerInnen nun die CO2-Emissionen in Karlsruhe pro Stadtteil sehen und vergleichen. Dies ist ein wichtiger Aspekt des Dashboards, da es den NutzerInnen ermöglicht, die CO2-Emissionen in Karlsruhe zu visualisieren und zu vergleichen, was in der Anforderung **R11** definiert wurde. Dies ist ein wichtiger Schritt, um das Bewusstsein für

den Klimawandel zu schärfen und die NutzerInnen zu motivieren, ihren CO2-Fußabdruck zu reduzieren. Dazu helfen auch weitere Grafiken wie Balkendiagramme, die den CO2-Fußabdruck der NutzerInnen in verschiedenen Kategorien anzeigen.

6.8 Das neue Gruppensystem

Eine essentielle Funktion der alten Webseite war die Möglichkeit, sich in Gruppen zusammenzuschließen und gemeinsam den CO2-Fußabdruck der Gruppe zu messen und sich gegenseitig zu motivieren, diesen zu reduzieren. Um die **Anforderung [R06]** zu erfüllen, wurde das Gruppensystem in die neue Webseite integriert und verbessert. Das Gruppensystem besteht aus mehreren Komponenten, die jeweils für einen bestimmten Teil des Gruppensystems verantwortlich sind.

Es gibt zunächst eine Seite zum Gruppensystem, auf der NutzerInnen Informationen zum Gruppensystem erhalten und die Möglichkeit haben, einer Gruppe beizutreten oder eine Gruppe zu erstellen. Abbildung 6.6 zeigt diese Seite. Wie zu erkennen ist, kann man im ersten Absatz eine neue Gruppe erstellen. Im letzten Absatz der Seite ist es möglich, über den Gruppencode einer neuen Gruppe beizutreten.

Die Gruppenerstellung ist relativ einfach und besteht lediglich darin, einen Gruppennamen einzugeben, woraufhin ein Gruppencode generiert wird, den andere NutzerInnen nutzen können, um der Gruppe beizutreten. Dieser Gruppencode kann auf den Seiten des Gruppensystems in ein Eingabefeld eingegeben werden, um einer Gruppe beizutreten.

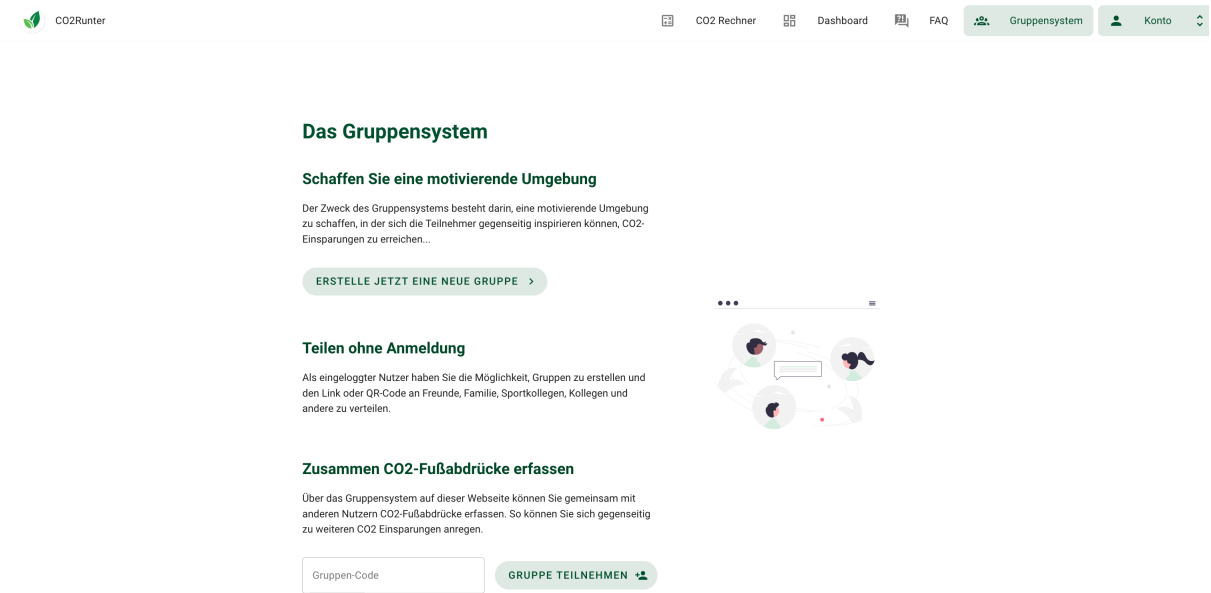

Abbildung 6.8: Design des neuen Gruppensystems

All dies funktioniert natürlich nur, wenn die NutzerInnen eingeloggt sind. Dann können die NutzerInnen über ihr Konto auf eine Gruppeninformationsseite gelangen, wo alle Informationen zu Gruppen dargestellt werden.

Für die Implementierung besteht die Herausforderung hauptsächlich darin, die benötigten Daten und die erforderlichen API-Schnittstellen zu identifizieren.

Im folgenden Screenshot ist einmal die Seite zu erkennen, auf die man weitergeleitet wird, wenn NutzerInnen eine neue Gruppe selbst erstellen möchten. Dort werden die NutzerInnen aufgefordert, einen Namen für ihre Gruppe einzugeben. Sobald die Gruppe erfolgreich erstellt wurde, erscheint der NutzerIn eine Bestätigung mit einem Dashboardlink, Joinlink und dem Gruppencode. Abbildung 6.7 zeigt die Ansicht, in der eine neue Gruppe erstellt werden kann.

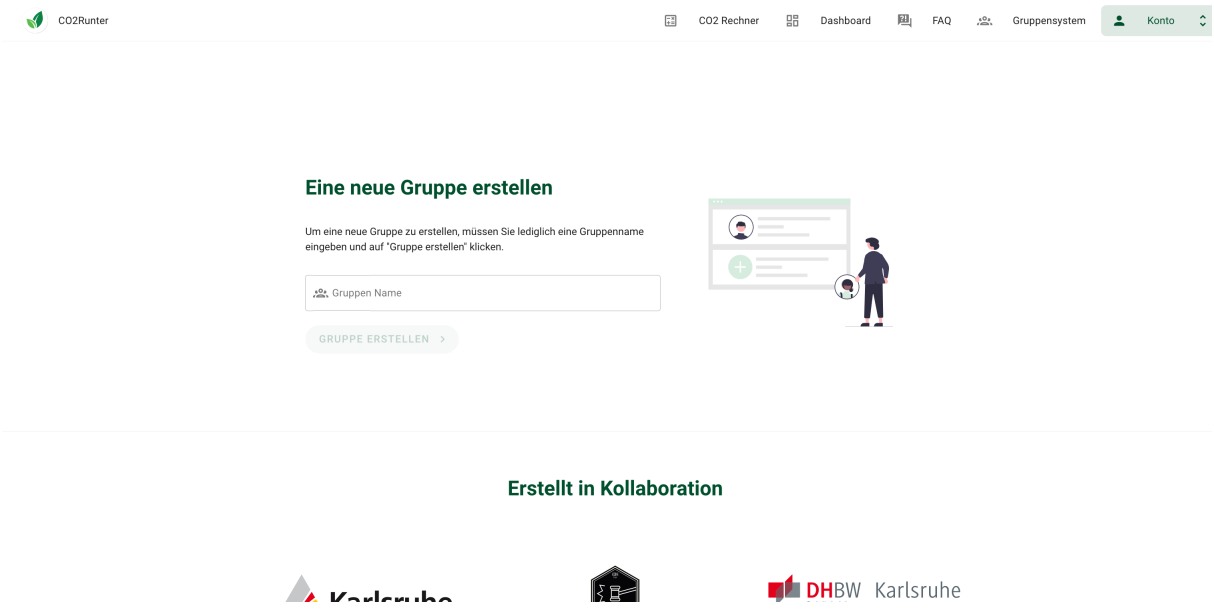

Abbildung 6.9: Ansicht zur Erstellung einer neuen Gruppe

Die nächste Abbildung zeigt die Bestätigungsseite nach der erfolgreichen Erstellung einer Gruppe. Zu beobachten sind zusätzlich die bereits erwähnten Links zur Dashboardansicht der Gruppe und ein Link, um andere Leute in die Gruppe einzuladen. Ebenfalls wird ein QR-Code erstellt, über den andere NutzerInnen der Gruppe beitreten können.

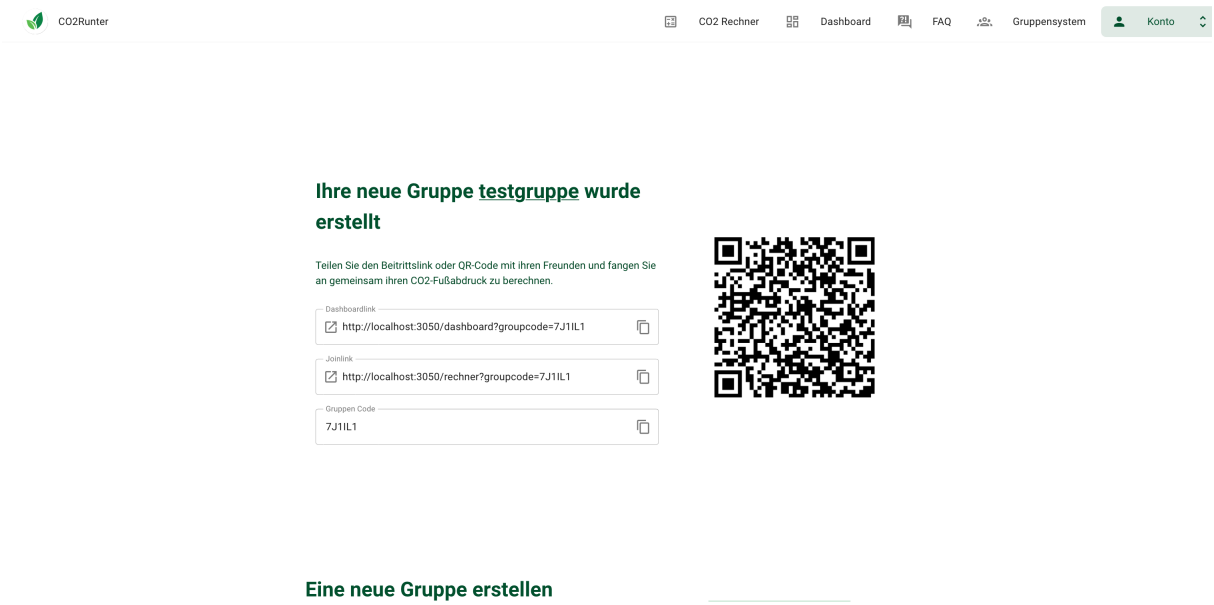

Abbildung 6.10: Bestätigungsansicht nach erfolgreicher Erstellung einer Gruppe

6.9 FAQ-Seite

Und zu guter Letzt wurde noch eine FAQ-Seite auf der neuen CO2-Runter Webseite hinzugefügt. Der Hintergrund hierfür ist, dass es bisher keine Möglichkeit gab, NutzerInnen Informationen zur Verfügung zu stellen, die nicht direkt mit dem CO2-Rechner oder dem Dashboard zusammenhängen.

Die FAQ-Seite soll NutzerInnen die Möglichkeit geben, Antworten auf häufig gestellte Fragen zu finden und zusätzlich eine Literaturliste enthalten, die den NutzerInnen ermöglicht, sich weiter zu informieren.

Die Implementierung dieser Seite ist relativ einfach, da es sich um eine statische Seite handelt. Die Daten für die FAQ-Seite werden aus einer JSON-Datei abgerufen und entsprechend angepasst, um die FAQ-Seite zu erstellen. Die einzige Neuigkeit besteht in der Implementierung einer Literaturliste, welche durch eine Tabelle dargestellt wird. Hierfür wird lediglich die JSON-Datei abgerufen und die Daten in die Tabelle eingefügt. Dies geschieht wie folgt:

```
<v-data-table
    v-model:page="page"
    :headers="headers"
    :items="literatureSources"
    :search="search"
>...</v-data-table>
```

Listing 6.28: Laden der Literaturliste

Die Literaturliste wird in einer Tabelle dargestellt, die es den NutzerInnen ermöglicht, die Literaturquellen zu durchsuchen und auf die Quellen zuzugreifen. Die Tabelle enthält eine Spalte für den Titel der Quelle, eine Spalte für den Autor, eine Spalte für das Erscheinungsjahr und eine Spalte für den Link zur Quelle. Die Tabelle ist durchsuchbar und paginiert, um die Benutzerfreundlichkeit zu verbessern.

Alle Literaturelemente befinden sich in der **literatureSources**-Variable, die in einer separaten TypeScript-Datei definiert wurde. Dabei handelt es sich um ein Array von Objekten, die die Literaturquellen repräsentieren.

```
export default const literatureSources: LiteratureSource[] = [
    {
        title: 'Test Title of a Book',
        author: 'Test Name',
        publisher: 'Test Publisher',
        publicationYear: '2019',
        url: 'http://example.com/test-title-of-a-book',
    },
];
```

Listing 6.29: Literaturelemente

Das finale Design der FAQ-Seite sieht sehr modern und übersichtlich aus und lädt die NutzerInnen dazu ein, sich weiter zu informieren.

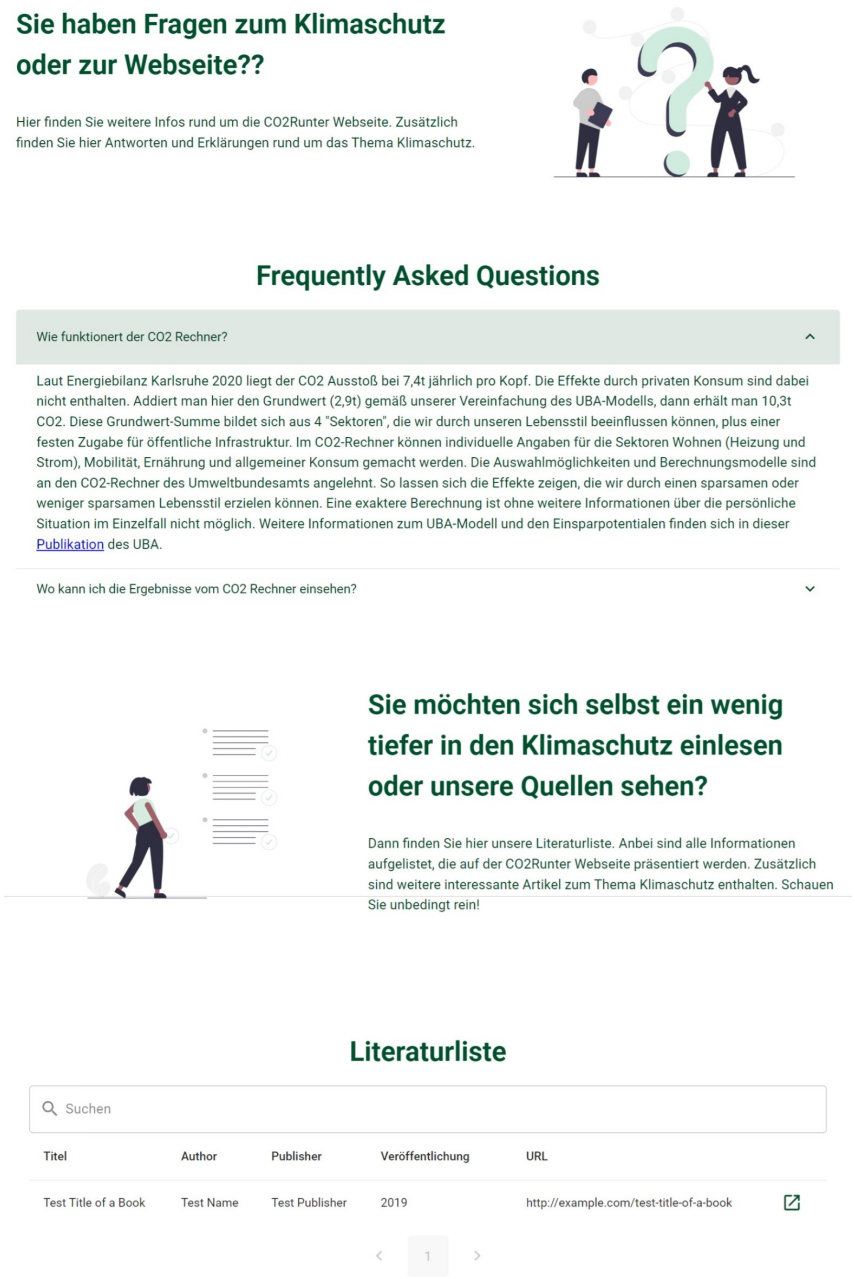

Abbildung 6.11: Design der FAQ-Seite auf der neuen Webseite

Nachdem die neue Webseite erfolgreich implementiert wurde, wird im nächsten Kapitel noch einmal alles zusammengefasst und ein Fazit über die Arbeit und die Ergebnisse gezogen. In dem letzten Kapitel soll noch einmal der gesamte Projektverlauf reflektiert werden. Dazu gehört selbstverständlich eine Zusammenfassung über erreichte Dinge während der Projektlaufzeit, einer Schlussfolgerung sowie einem Ausblick und eine Empfehlung für zukünftige Studienarbeiten, die dieses Projekt als grundlegendes Thema nutzen.

7 Zusammenfassung und Fazit

In diesem Kapitel wird eine Zusammenfassung der Erkenntnisse dieser Studienarbeit präsentiert. Es umfasst einen Rückblick auf die gewonnenen Erkenntnisse und darauf, was letztendlich implementiert wurde, sowie die Ergebnisse der Arbeit. Darüber hinaus werden Schlussfolgerungen gezogen, die nicht nur auf vergangenen Ereignissen und Erkenntnissen basieren, sondern auch einen Ausblick auf die Zukunft der CO2-Runter-App geben und Empfehlungen für deren Weiterentwicklung enthalten.

7.1 Zusammenfassung der Arbeitsergebnisse

Die Arbeitsergebnisse lassen sich grundsätzlich in zwei übergeordnete Kategorien unterteilen: einmal in die nutzerzentrierte Umfrage und dann in die darauf folgende Implementierung der erhobenen Kenntnisse mit der Neuentwicklung der CO2-Runter Webseite. Im folgenden Abschnitt wird auf beide Kategorien eingegangen und die Ergebnisse zusammengefasst.

7.1.1 Nutzerzentrierte Umfrage

Die nutzerzentrierte Umfrage war ein essenzieller Bestandteil der Studienarbeit, um Einblicke in die Bedürfnisse und Präferenzen der potenziellen NutzerInnen der CO2-Runter Webseite zu gewinnen. Die Umfrage wurde durchgeführt, um die Nutzerfreundlichkeit und das Design der Webseite zu bewerten und wertvolles Feedback zu erhalten, das bei der weiteren Entwicklung und Optimierung der Webseite berücksichtigt werden kann. Aus diesen Ergebnissen konnten letztendlich Ziele bzw. Anforderungen für die Implementierung der Webseite abgeleitet werden.

Die Umfrageergebnisse zeigten eine hohe Sensibilisierung der TeilnehmerInnen für den CO2-Fußabdruck sowie ein Interesse an einer Plattform wie CO2-Runter, die ihnen hilft, ihren Fußabdruck zu berechnen und Tipps für den Klimaschutz zu erhalten. Insbesondere wurde das Potenzial von Funktionen wie einem Punktesystem, Möglichkeiten zum Teilen von Fortschritten und Tipps zur Verhaltensänderung positiv bewertet.

Die Bewertung der Nutzerfreundlichkeit und des Designs lieferten wertvolle Einblicke, die bei der weiteren Optimierung der Webseite berücksichtigt werden können. Es wurde

deutlich, dass eine einfache Benutzeroberfläche sowie ansprechende visuelle Elemente wichtige Faktoren für eine positive Nutzererfahrung sind.

Abschließend können die Ergebnisse der nutzerzentrierten Umfrage als Leitfaden genutzt werden, um Personas zu erstellen, die als Grundlage dienten, um die Anforderungen für die Implementierung der Webseite herauszuarbeiten und zu definieren. Diese konnten dann genutzt werden, um bei der Neuentwicklung der Webseite wichtige Aspekte, die für die NutzerInnen wichtig sind, zu berücksichtigen und umzusetzen, um eine bessere Nutzererfahrung zu gewährleisten.

7.1.2 Implementierung der CO2-Runter Webseite

Die Implementierung der CO2-Runter Webseite war der zweite Kernaspekt dieser Studienarbeit. Die Implementierung basierte auf den Ergebnissen, die aus der nutzerzentrierten Umfrage gewonnen wurden, und den definierten Anforderungen, die in Kapitel 5 festgelegt wurden.

Eine effektive Methode, um die Implementierungsleistung und die Ergebnisse der finalen Webseite zu bewerten und zu messen, besteht darin, die in Kapitel 5 definierten Anforderungen zu betrachten. Diese Anforderungen wurden durch verschiedene Kategorien sowie vorangegangene Recherchen und UCD/HCD-Analysen definiert. Auf Basis dieser Erkenntnisse haben wir insgesamt **19** Anforderungen identifiziert.

Ein einfacher Ansatz besteht darin zu ermitteln, wie viele der Anforderungen letztendlich implementiert wurden und welche genau. Von den **19** Anforderungen wurden **17** erfolgreich umgesetzt und implementiert. Die beiden nicht implementierten Anforderungen sind **R08** und **R13**. Diese Quote ist bereits sehr akzeptabel. Zudem ist zu beachten, dass beide nicht erfüllten Anforderungen der Kategorie der **Kann-Anforderungen** angehören, was ihre Nicht-Implementierung als nicht essentiell kennzeichnet. Eine Übersicht der implementierten Anforderungen findet sich in Tabelle 7.1.

ID	Anforderung	Implementiert
R01	Abrufen der Bestandsdaten von der Datenbank	✓
R02	Berechnung des CO2-Fußabdrucks	✓
R03	Hochladen von CO2-Daten	✓
R04	Homepage	✓
Die Fortsetzung erfolgt auf der nachfolgenden Seite		

ID	Anforderung	Implementiert
R05	Dashboard	✓
R06	Gruppensystemintegration	✓
R07	FAQ- und Literaturlisten-Seite	✓
R08	Tipps und Tricks zur Reduzierung des CO2-Fußabdrucks	X
R09	Durchführung des CO2-Rechner-Workflows	✓
R10	Integration von Grafiken	✓
R11	Visualisierung und Vergleich des CO2-Fußabdrucks	✓
R12	Modernisierung des Designs	✓
R13	Anzeige von weiteren Quellen/Infos im CO2-Rechner	X
R14	Anpassung von Question.json	✓
R15	Benutzerfreundlichkeit	✓
R16	Modernes Design	✓
R17	Zuverlässigkeit	✓
R18	Wartbarkeit und Erweiterbarkeit	✓
R19	Codequalität und Styleguide-Einhaltung	✓

Tabelle 7.1: Übersicht der implementierten funktionalen und nicht-funktionalen Anforderungen

7.2 Empfehlung an die Laufzeitumgebung

Das neue Frontend der CO2-Runter-Webseite wurde zu Testzwecken auf einem Webserver deployt und alle Funktionalitäten getestet. Beim laufenden Betriebssystem des Webservers handelt es sich um das Linux-Betriebssystem **Ubuntu 22.04.2 LTS**. Das Betriebssystem wurde 2020 veröffentlicht und wurde als *Long-term support* gekennzeichnet. Das bedeutet, dass es diese Ubuntuversion bis 2025 weiter gewartet und Instand gehalten wird.

Aus diesem Grund empfiehlt das Entwicklerteam, die neue Version der CO2-Runter-Webseite auf einem Webserver mit dem Betriebssystem Ubuntu 22.04 zu deployen.

7.3 Schlussfolgerungen und Ausblick

Da es sich hierbei um eine Studienarbeit mit einem begrenzten Zeitrahmen handelt, war es unser Fokus, die wesentlichen Aspekte der CO2-Runter-App zu entwickeln. Wir haben erfolgreich die Grundfunktionalitäten implementiert, um die alte Webseite durch unsere neu entwickelte zu ersetzen. In dem verfügbaren Zeitrahmen sind wir zu dem Schluss gekommen, dass wir unsere Ziele erreicht haben und sind sehr zufrieden mit dem, was wir erreicht haben und wie viel wir dabei gelernt haben.

Obwohl der Weg nicht immer einfach war und wir auf viele Herausforderungen gestoßen sind, war eine besonders herausfordernde Aufgabe die Implementierung des CO2-Rechners. Wir hatten erwartet, dass wir viele Aspekte aus dem vorherigen Rechner übernehmen könnten, was sich jedoch als schwieriger erwies als gedacht. Insbesondere stellte sich heraus, dass der Rechner im detaillierten Modus nicht ordnungsgemäß funktionierte, einschließlich der Berechnung des Konsums und der damit verbundenen Formeln. Dies zwang uns, umfassende Anpassungen vorzunehmen, was zu einer gründlichen Überprüfung und Analyse führte.

Durch umfangreiche Recherchen und Analysen gelang es uns schließlich, den Rechner zu implementieren und logischere Schlussfolgerungen zu ziehen. Im alten Rechner war es beispielsweise möglich, die Annahme zu treffen, dass bei einer bestimmten Ernährungsweise kein CO2-Ausstoß entsteht, was jedoch unrealistisch ist. Unsere Anpassungen führen zu Ergebnissen, die realistischer sind, obwohl sie aufgrund vieler Mittelwertbildungen nicht zu 100% genau sind, aber sehr nahe am Original liegen.

Wir hoffen, dass unsere Arbeit dazu beiträgt, die Welt ein Stückchen besser zu machen. Das Projekt ist damit jedoch hoffentlich nicht abgeschlossen. Wir hoffen, dass zukünftige StudentInnen an diesem Projekt teilnehmen und es weiterentwickeln und verbessern können, da sicherlich noch viele unentdeckte Möglichkeiten bestehen. Wir haben einige Empfehlungen und Vorschläge für die Zukunft des Projekts.

7.4 Empfehlungen für die Zukunft der CO2-Runter-App

Von Anfang an war es unser Ziel, die CO2-Runter-App so zu gestalten, dass sie kontinuierlich weiterentwickelt und optimiert werden kann. Die Architektur der Webseite wurde bewusst auf Flexibilität und Erweiterbarkeit ausgelegt. Wir haben TypeScript als Hauptentwicklungssprache gewählt und die Anwendungsstruktur entsprechend angepasst, um zukünftige Erweiterungen nahtlos zu ermöglichen. Dadurch besteht definitiv das Potenzial, die Webseite weiterzuentwickeln, indem beispielsweise weitere Funktionen und Features

durch zusätzliche Recherchen und Analysen hinzugefügt werden oder Features, die wir aufgrund von Zeitmangel nicht implementieren konnten.

Bei der Neuentwicklung der Webseite stießen wir auf Herausforderungen im Umgang mit der API. Obwohl sie funktioniert, ist sie ausschließlich in JavaScript geschrieben, was zu Problemen bei der Typisierung und Fehlerbehandlung führte. Zudem war es schwierig zu verstehen, welche Daten der Endpunkt tatsächlich benötigt und zurückgibt. Für das Frontend haben wir deshalb zahlreiche Interfaces erstellt, um die Daten zu typisieren. Eine Verbesserung der API wäre durch eine Neuschreibung in TypeScript möglich. Außerdem ist uns aufgefallen, dass die API keine Möglichkeit bietet, dass NutzerInnen aus einer Gruppe austreten können. Nur der Gruppenadministrator kann NutzerInnen entfernen, indem er die Gruppe auflöst. Dies ist ein weiterer Punkt, der verbessert werden könnte. Dabei sollte auch darüber nachgedacht werden, welche weiteren Funktionen im Gruppensystem gewünscht sind, was ein hohes Potenzial für die EndnutzerInnen der Webseite darstellt. Als letztes gab es noch die konkrete Implementierung der Anforderungen **R08** und **R13**, welche leider nicht umgesetzt werden konnten. Diese Anforderungen können ein hohes Potential bieten und als sehr praktische Erweiterung des CO2-Rechners dienen und kann darauf auf der FAQ- und Literaturlisten-Seite eingebunden werden, welche mit der **R07** bereits implementiert wurde.

Zusammenfassend ist die CO2-Runter-App ein Projekt, das uns sehr am Herzen liegt und das wir gerne weiterentwickeln und verbessern möchten. Wir sind stolz auf das bisher Erreichte und freuen uns auf die Zukunft der Webseite. Unsere Hoffnung ist es, dass die Webseite dazu beiträgt, das Bewusstsein für den CO2-Fußabdruck zu schärfen und Menschen dazu inspiriert, ihren Beitrag zum Klimaschutz zu leisten. Wir sind zuversichtlich, dass die CO2-Runter-App ein nützliches Werkzeug für alle sein wird, die ihren CO2-Fußabdruck reduzieren möchten, und dass sie dazu beitragen wird, die Welt zu einem besseren Ort zu machen.

Literatur

[1] „3 Der Forschungsprozess: Die Planungsphase“. In: *Einführung in die Methoden der Umfrageforschung*. München: Oldenbourg Wissenschaftsverlag, 2011, S. 61–102. ISBN: 9783486710090. DOI: doi:10.1524/9783486710090.61. URL: https://doi.org/10.1524/9783486710090.61.

[2] Anthony Fu. *vite-plugin-pwa*. URL: https://www.npmjs.com/package/vite-plugin-pwa (besucht am 22. 02. 2024).

[3] Baidu EFE team. *vue-echarts - npm*. URL: https://www.npmjs.com/package/vue-echarts (besucht am 18. 04. 2024).

[4] Dmitriy Maschenko. *Vue vs. React vs. Angular - Pros, Cons, Use Cases and Comparison*. URL: https://getdevdone.com/blog/vue-vs-react-vs-angular-pros-cons-use-cases-and-comparison.html (besucht am 02. 11. 2023).

[5] Docker Team. *Docker Docs*. URL: https://docs.docker.com/ (besucht am 22. 02. 2024).

[6] Jr. Emmit A. Scott. *SPA Design and Architecture: Understanding single-page web applications*. Simon und Schuster, 2015.

[7] Even You. *Introduction | Vue.js*. URL: https://vuejs.org/guide/introduction.html (besucht am 31. 10. 2023).

[8] Facebook. *Quickstart - React*. URL: https://react.dev/learn (besucht am 31. 10. 2023).

[9] Facebook. *Writing Markup with JSX - React*. URL: https://react.dev/learn/writing-markup-with-jsx (besucht am 07. 11. 2023).

[10] Facebook. *Writing Markup with JSX - React: Form Component*. URL: https://react.dev/_next/image?url=%2Fimages%2Fdocs%2Fdiagrams%2Fwriting_jsx_form.png&w=750&q=75 (besucht am 07. 11. 2023).

[11] Facebook. *Writing Markup with JSX - React: HTML Image*. URL: https://react.dev/_next/image?url=%2Fimages%2Fdocs%2Fdiagrams%2Fwriting_jsx_html.png&w=750&q=75 (besucht am 07. 11. 2023).

[12] Facebook. *Writing Markup with JSX - React: JavaScript Image*. URL: https://react.dev/_next/image?url=%2Fimages%2Fdocs%2Fdiagrams%2Fwriting_jsx_js.png&w=750&q=75 (besucht am 07. 11. 2023).

[13] Facebook. *Writing Markup with JSX - React: Sidebar Component*. URL: https://react.dev/_next/image?url=%2Fimages%2Fdocs%2Fdiagrams%2Fwriting_jsx_sidebar.png&w=750&q=75 (besucht am 07. 11. 2023).

[14] Google. *Angular - Understanding Angular*. URL: https://angular.io/guide/understanding-angular-overview (besucht am 31. 10. 2023).

[15] hai.hn. *The Battle of Frameworks: Exploring the Pros and Cons of React, Angular, and Vue.js*. URL: https://medium.com/@hai98/the-battle-of-frameworks-exploring-the-pros-and-cons-of-react-angular-and-vue-js-558d5472cdb1 (besucht am 02. 11. 2023).

[16] *Human-Centered-Design for Definition of New Collaborative Scenarios*. 2021. DOI: 10.1007/978-3-030-80415-2_10.

[17] IONOS. *Grundlagen der modernen Webentwicklung*. URL: https://www.ionos.de/digitalguide/websites/web-entwicklung/grundlagen-der-modernen-webentwicklung/ (besucht am 31. 10. 2023).

[18] Kei Hoshi John Waterworth. *The Problems of Design*. 2016. DOI: 10.1007/978-3-319-30334-5_2.

[19] Michael Lewerenz Josefine Lepzien. „Persona-Methode - Eine Methode zur Illustrierung von Bildungsbedarfen“. In: (). URL: https://www.uni-rostock.de/storages/uni-rostock/UniHome/Weiterbildung/KOSMOS/Persona.pdf.

[20] Stadt Karlsruhe. *CO2-Runter-App*. 2023. URL: https://co2runter.karlsruhe.de/.

[21] Lauren Laundry. *What Is Human-Centered Design?* 2020. URL: https://online.hbs.edu/blog/post/what-is-human-centered-design.

[22] Maruti Techlabs. *Advantages of Component-Based Architecture*. URL: https://medium.com/geekculture/an-in-depth-guide-to-component-based-architecture-d023f02f4147 (besucht am 22. 02. 2024).

[23] Material Design - Google. *Material Design*. URL: https://m2.material.io/design (besucht am 22. 02. 2024).

[24] MaterialUI. *Overview - Material UI*. URL: https://mui.com/material-ui/getting-started/ (besucht am 31. 10. 2023).

[25] Tomasz Miaskiewicz und Kenneth A. Kozar. „Personas and user-centered design: How can personas benefit product design processes?“ In: *Design Studies* 32.5 (2011), S. 417–430. ISSN: 0142-694X. DOI: https://doi.org/10.1016/j.destud.2011.03.003. URL: https://www.sciencedirect.com/science/article/pii/S0142694X11000275.

[26] Microsoft. *TypeScript is JavaScript with syntax for types.* URL: https://www.typescriptlang.org/ (besucht am 08. 01. 2024).

[27] Microsoft. *Übersicht über Progressive Web Apps (PWAs).* URL: https://learn.microsoft.com/de-de/microsoft-edge/progressive-web-apps-chromium/ (besucht am 22. 02. 2024).

[28] Mohit Joshi. *Angular vs React vs Vue: Core Differences.* URL: https://www.browserstack.com/guide/angular-vs-react-vs-vue (besucht am 02. 11. 2023).

[29] Mozilla. *JavaScript | MDN.* URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript?retiredLocale=de (besucht am 31. 10. 2023).

[30] Nefe James. *Angular vs. React vs. Vue.js: Comparing performance.* URL: https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/ (besucht am 02. 11. 2023).

[31] NGINX Team. *nginx documentation.* URL: https://nginx.org/en/docs/ (besucht am 22. 02. 2024).

[32] Paul Gillin. *What is Component-Based Architecture?* URL: https://www.mendix.com/blog/what-is-component-based-architecture/ (besucht am 22. 02. 2024).

[33] *Persona Design in Participatory Agile Software Development.* 2020. DOI: 10.1007/978-3-030-60149-2_5.

[34] Frank Ploß. „Usability Engineering in einem Open-Source-Projekt“. Magisterarb. Universität Hamburg, 2007. URL: https://swa.informatik.uni-hamburg.de/files/abschlussarbeiten/Diplomarbeit.pdf.

[35] Sim Van der Ryn. *Human-Centered Design.* 2013. DOI: 10.1007/978-1-61091-505-2_2.

[36] Shylesh S. *A Study of Software Development Life Cycle Process Models.* Techn. Ber. Srinivas Institute of Management Studies, 2017. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2988291.

[37] Sae Schatz u. a. „Learning Engineering Toolkit“. In: 2022. DOI: 10.4324/9781003276579-17.

[38] Heru Susanto u. a. *Human-Centered Design to Enhance the Usability, Human Factors, and User Experience Within Digital Destructive Ecosystems.* Techn. Ber. Harvard Business School Online, 2021. DOI: 10.4018/978-1-7998-4787-8.CH005.

[39] ViteJS. *Getting Started.* URL: https://vitejs.dev/guide/ (besucht am 08. 01. 2024).

[40] ViteJS. *Vite Next Generation Frontend Tooling.* URL: https://www.w3schools.com/whatis/whatis_npm.asp#:~:text=The%20name%20npm%20(Node%20Package,json (besucht am 31. 10. 2023).

[41] Vuetify. *Material color palette - Vuetify*. URL: https://vuetifyjs.com/en/styles/colors/ (besucht am 25. 03. 2024).

[42] Vuetify Team. *Vuetify - Vue Component Framework*. URL: https://vuetifyjs.com/en/ (besucht am 22. 02. 2024).

[43] W3School. *What is npm*. URL: https://vitejs.dev/ (besucht am 08. 01. 2024).

[44] Wikipedia. *Webentwicklung*. URL: https://de.wikipedia.org/wiki/Webentwicklung#:~:text=Als%20Webentwicklung%20(englisch%20Web%20development,dagegen%20meist%20von%20Webdesignern%20%C3%BCbernommen. (besucht am 31. 10. 2023).

Anhang