Universität Zürich[UZH]

Bachelorarbeit am Institut für Informatik

Optimierung einer Cap-and-Trade-Plattform für den Handel von CO_2-Zertifikaten innerhalb von Organisationen

Vital Arnold

Luzern, Schweiz

11-738-499

Betreuender Professor: Prof. Dr. Lorenz Hilty

Betreuer: Yuliyan Maksimov

Abgabe der Arbeit: 31. März 2017

Zusammenfassung

Der Treibhauseffekt und die damit verbundene Erwärmung des weltweiten Klimas sind wichtige Themen der heutigen Gesellschaft. Ein Ansatz, der längerfristig Treibhausgas-Emissionen senken soll, ist der Emissionshandel, der nach dem Cap-and-Trade Prinzip funktioniert. In dieser Arbeit wird eine Web-Applikation, die Emissionshandel innerhalb einer Organisation ermöglicht, optimiert. Die Web-Applikation ermöglicht einerseits den Handel von beschränkten Ressourcen, zum Beispiel von CO_2-Zertifikaten. Andererseits können Mitglieder der Organisation ihre Dienstreisen und deren CO_2-Emissionen erfassen.

Um den zukünftigen Einsatz der Web-Applikation zu ermöglichen, wurde die Web-Applikation und deren Schnittstellen analysiert und der ursprüngliche Funktionsumfang wiederhergestellt. Dazu wurde eine neue Berechnungsmethode für CO_2-Emissionen von Flugreisen implementiert. Um die Nutzung der Web-Applikation auf mobilen Geräte zu verbessern, werden verschiedene Ansätze vorgestellt. Mit dem Responsive Design konnte die Web-Applikation für mobile Geräte optimiert werden.

Abstract

One approach to decrease greenhouse gas emissions over a long-term period is emission trading with the cap-and-trade system. This bachelor thesis has the goal to optimize a web application that allows emission trading within an organisation. With the web application members of the organisation can trade limited resources, e.g. CO_2-certificates. They can also record their business trips including the resulting CO_2 emissions.

Within the scope of this bachelor thesis, the web application was analysed and its original functional range could be restored. To calculate the CO_2 emissions from an air trip, a new computation was implemented. Additionally, the concept of Responsive Design was applied to the web application. With the Responsive Design, the optimization for the use on mobile devices could be realized.

Inhaltsverzeichnis

1 Einleitung 1

1.1 Ziel 1

1.2 Hintergrund 1

1.3 Begriffserklärung 3

1.3.1 Web-Applikation 3

1.3.2 Web-Technologien 3

1.3.2.1 HTML (Hypertext Markup Language) 3

1.3.2.2 CSS (Cascading Style Sheet) 4

1.3.2.3 JavaScript 4

2 Usability von Webseiten 5

2.1 Mobile Usability 6

2.1.1 Mobile Geräte 6

2.1.1.1 Bildschirmgrösse 6

2.1.1.2 Steuerung 6

2.1.1.3 Downloadzeiten 7

2.1.2 Mobilstrategie 7

2.1.2.1 Native Apps 7

2.1.2.2 Hybride Apps 8

2.1.2.3 Mobile Webseite 8

2.1.2.4 Responsive Design 8

3 Analyse der bestehenden Web-Applikation 11

3.1 Darstellung 11

3.2 Framework 13

3.3 Schnittstellen 14

3.3.1 Google Maps 14

3.3.1.1 GWT-Maps 14
3.3.1.2 Google Maps JavaScript API 14
3.3.2 Flightstats APIs 15
3.3.2.1 Airports API 15
3.3.2.2 Connections API 16
3.3.2.3 Flight Track API 16
3.4 Berechnung der CO_2-Emissionen 16
3.4.1 Auto 16
3.4.2 Zug 17
3.4.3 Flugzeug 17
4 Optimierungsmöglichkeiten der Web-Applikation 18
4.1 Optimierung für mobile Geräte 18
4.2 Browser Verlauf 19
4.3 Framework und Java-Bibliothek GWT-Maps 20
4.4 Berechnung der CO_2-Emissionen durch Flugreisen 20
4.4.1 Berechnung mit Daten von Flightstats 21
4.4.2 Berechnung mit der Great Circle Distance 21
5 Umsetzung 23
5.1 Anpassung an mobile Geräte 23
5.1.1 Responsive Design 23
5.1.2 Guidelines 27
5.2 Browser Verlauf 29
5.3 Berechnung der CO_2-Emissionen bei Flugreisen 30
6 Fazit und Ausblick 32
6.1 Fazit 32
6.2 Ausblick 32
7 Literaturverzeichnis 33
Anhang A: Parameter für die Berechnung des CO_2-Äquivalents 36

Abbildungsverzeichnis

Abbildung 1: Beispiel des Responsive Design im Desktop-Browser 9

Abbildung 2: Beispiel des Responsive Design auf einem Tablet 10

Abbildung 3: Beispiel des Responsive Design auf einem Smartphone 10

Abbildung 4: Anmeldeseite in verkleinertem Browserfenster 12

Abbildung 5: Anmeldeseite im Browser des Testgeräts 13

Abbildung 6: Beispiel einer CSS Media Query 23

Abbildung 7: Ansicht der Erfassungsmaske einer Dienstreise bei einer Breite von 1400 Pixeln 24

Abbildung 8: Ansicht der Erfassungsmaske einer Dienstreise bei einer Breite von 1000 Pixeln 25

Abbildung 9: Ansicht der Erfassungsmaske einer Dienstreise auf dem Testgerät 26

Abbildung 10: Startseite der Web-Applikation 27

Abbildung 11: Vorschau einer Dienstreise bei einer Breite von 200 Pixeln 28

Abbildung 12: Anmeldeseite der Web-Applikation 29

Abbildung 13: Eingabemaske für eine Dienstreise mit Flugzeug 30

Abbildung 14: Vorschau einer Dienstreise mit Flugzeug 31

1 Einleitung

1.1 Ziel

Das Ziel dieser Arbeit ist es, die bestehende Web-Applikation für intra-organisationalen Emissionshandel zu analysieren und zu optimieren. In den folgenden Kapiteln werden die Planung und die Umsetzung der Optimierungen aufgezeigt:

- Um die Applikation zu verbessern, gehe ich im Kapitel 2 den folgenden Fragen nach: Was sind die Anforderungen an eine moderne Web-Applikation im Bereich der Usability? Inwiefern muss dabei die Verbreitung an mobilen Endgeräten berücksichtigt werden?
- Um die Optimierungen für die Web-Applikation festzulegen, habe ich mich mit der Web-Applikation und den darin genutzten Technologien auseinandergesetzt. Im Kapitel 3 stelle ich Erkenntnisse aus der Untersuchung der Web-Applikation vor.
- Mit den Befunden aus Kapitel 2 und 3 werden die Anforderungen an die Optimierungen festgelegt. Diese beschreibe ich in Kapitel 4.
- Im Kapitel 5 beschreibe ich die Umsetzung der Optimierungsmassnahmen.

1.2 Hintergrund

Der Klimawandel und das damit verbundene Thema der Treibhausgas-Emissionen ist heutzutage ein wichtiges Thema unserer Gesellschaft. Zwischen 1864 und 2012 ist die Temperatur in der Schweiz um 1,8 °C gestiegen und der weltweite Temperaturanstieg in der Zeit von 1880 bis 2012 beträgt 0,85 °C. Klimaszenarien prognostizieren auch für die Zukunft eine weitere Erwärmung. Gegen Ende des 21. Jahrhunderts wird in der Schweiz eine Erwärmung von 2,7 bis 4,8 °C prognostiziert, falls keine globalen Klimaschutzmassnahmen umgesetzt werden. Aber auch mit globalen Massnahmen ist mit einer zwar etwas geringeren Erwärmung von 1,2 bis 1,8 °C zu rechnen. Der Grund für diese Erwärmung liegt mit grösster Wahrscheinlichkeit am erhöhten Treibhausgas-Ausstoss seit der Industrialisierung (Bundesamt für Umwelt, 2015a).

Um diesem Problem entgegenzuwirken existieren diverse Ansätze. Ein Ansatz, der auf internationaler und nationaler Ebene praktiziert wird, ist der Emissionshandel, der nach dem Cap-and-Trade System funktioniert. Der Emissionshandel hat das Ziel, die "Treibhausgasemissionen da zu reduzieren, wo die Kosten am tiefsten sind" (Bundesamt für Umwelt, 2015b).

Beim Emissionshandel nach dem Cap-and-Trade System wird eine systemweite Obergrenze (Cap) an Emissionen festgesetzt, die nicht überschritten werden soll. Diese Gesamtmenge an Emissionen wird in handelbare Zertifikate unterteilt. Die Zertifikate werden an die Marktteilnehmer verteilt oder verkauft und erlauben dem Besitzer den Ausstoss einer gewissen Menge an Treibhausgasen. Durch den Handel dieser Zertifikate (Trade) soll die Reduzierung der Emissionen auf die festgelegte Obergrenze zu gesamtwirtschaftlich möglichst tiefen Kosten erfolgen. Denn ein Marktteilnehmer, dessen Kosten für die Verringerung seiner Emissionen tiefer als der Zertifikatpreis sind, ist interessiert daran, die Emissionen einzusparen und seine Zertifikate zu verkaufen. Mit der stetigen Senkung der Obergrenze soll über längere Zeit auch der Gesamtaustoss an Treibhausgasen gesenkt werden (Center for American Progress, 2008).

In der EU und in der Schweiz wurde jeweils ein solches Emissionshandelssystem für Emissionen des Treibhausgases CO_2 installiert (Bundesamt für Umwelt, 2015b).

Im Rahmen zweier Masterarbeiten wurde eine Web-Applikation entwickelt, die einen solchen Emissionshandel innerhalb einer Organisation ermöglicht. Die beiden Masterarbeiten wurden an der Universität Zürich im April 2013 eingereicht.

Die Masterarbeit von David Oertle (2013) trägt den Titel "Plattform für den Handel von Ressourcen innerhalb eines Unternehmens am Beispiel von CO_2-Zertifikaten für Dienstreisen". Die Arbeit untersucht, wie ein Cap-and-Trade System innerhalb einer Organisation angewendet werden könnte, um beschränkte Ressourcen effizient zu verteilen. Für den Handel mit diesen beschränkten Ressourcen werden in der Arbeit mehrere Auktionsformen als Marktmechanismen vorgestellt. Als Resultat entstand eine Web-Applikation, die den Handel von beschränkten Ressourcen, wie beispielsweise Emissionszertifikate oder Parkplätze, ermöglicht. Die Applikation ist für den Handel innerhalb einer Organisation konzipiert.

Badertscher (2013) beschreibt in seiner Arbeit die Entstehung des Cap-and-Trade Systems im Bereich der Umweltpolitik und sucht mögliche Anwendungsgebiete dafür. Im praktischen Teil wurde ein Plugin für die Web-Applikation der erstgenannten Arbeit erstellt. Dieses Plugin ermöglicht die Erfassung von CO_2-Emissionen durch Dienstreisen. Für die Dienstreisen stehen die Alternativen Auto, Zug und Flugzeug zur Verfügung. Jedes Mitglied der Organisation oder dessen Abteilung bekommt eine gewisse Menge an CO_2-Zertifikaten zugeschrieben. Diese Zertifikate können aufgebraucht oder gehandelt werden.

1.3 Begriffserklärung

1.3.1 Web-Applikation

Eine Web-Applikation wird normalerweise wie eine normale Webseite in einem Browser aufgerufen. Der Unterschied zu einer herkömmlichen Webseite liegt darin, dass Web-Applikationen mehr interaktive Funktionen bieten. Der Funktionsumfang gleicht eher der einem lokal installierten Programm. Im Gegensatz zur lokalen Software werden aber die Daten bei der Web-Applikation auf dem Server verarbeitet. Somit kann von verschiedenen Geräten über den Browser auf die Applikation zugegriffen werden, ohne dass eine Installation notwendig ist. Webseiten dagegen sind eher darauf ausgelegt, Informationen zur Verfügung zu stellen und bieten wenig Interaktionen an. Für Webseiten, wie auch für Web-Applikationen können die gleichen Technologien eingesetzt werden (HTML, CSS, JavaScript) (Warndorf, 2013).

1.3.2 Web-Technologien

Im Folgenden möchte ich kurz auf die drei genannten Technologien des Web-Bereiches eingehen. Diese drei Technologien werden auch in der hier vorgestellten Web-Applikation genutzt.

1.3.2.1 HTML (Hypertext Markup Language)

HTML ist zuständig für die Struktur einer Webseite oder einer Web-Applikation. Eine Webseite oder Web-Applikation besteht aus HTML-Elementen, die ineinander

geschachtelt werden können. Diese Elemente beinhalten schlussendlich auch den Inhalt der Seite (bspw. Text oder ein Bild) (W3C, o. J.).

1.3.2.2 CSS (Cascading Style Sheet)

CSS ermöglicht die Gestaltung der Webseite oder Web-Applikation. Mit dieser Sprache können die HTML-Elemente formatiert werden. So können die Grösse, das Aussehen und die Anordnung der Elemente festgelegt werden. Auch Farben, Schriftarten und -grössen können mit CSS formatiert werden. Zudem erlaubt CSS die Anpassung der Darstellung an verschiedene Geräte. Dadurch kann die Darstellung der Webseite oder Web-Applikation an verschiedene Bildschirmgrössen angepasst werden (W3C, o. J.).

1.3.2.3 JavaScript

Mit der Programmiersprache JavaScript ist es möglich eine Webseite mit komplexeren Interaktionsmöglichkeiten auszustatten. Durch diese Interaktionsmöglichkeiten unterscheidet sich eine Web-Applikation von einer Webseite. Eine interaktive Karte, wie sie in der Web-Applikation für intra-organisationalen Cap-and-Trade Handel genutzt wird, ist ein Beispiel für den Einsatz von JavaScript (Mozilla Developer Network, 2017).

2 Usability von Webseiten

Die Web-Applikation, die im Zentrum dieser Arbeit steht, wird wie eine normale Webseite über den Browser aufgerufen. Zudem werden die klassischen Web-Technologien HTML, CSS und JavaScript verwendet. Aus diesen Gründen nutze ich hier Design-Vorschläge, die für Webseiten vorgeschlagen werden. Die Usability einer Webseite ist stark vom Anzeigegerät abhängig. Deshalb müssen Webseiten an diese Geräte angepasst werden. Ursprünglich wurden Webseiten zur Anzeige an Desktop-Computern und Laptops entwickelt. Doch heute wird auch mit mobilen Geräten wie Smartphones und Tablets darauf zugegriffen. Deshalb muss man sich als Entwickler dieser Webseiten auch mit der Frage auseinandersetzen, wie die Webseite beim Aufruf mit einem mobilen Gerät reagieren soll.

Badertscher (2013, S. 80-85) hat in seiner Arbeit die Nutzerfreundlichkeit der Web-Applikation bereits evaluiert: Eine Testgruppe konnte die Web-Applikation testen und einen Fragebogen ausfüllen. Zur Evaluation wurde das AttrakDiff-Framework verwendet. Dabei konnten die Benutzer beurteilen, inwiefern sie eine gewisse Eigenschaft mit der Web-Applikation verbinden. In den Resultaten werden die Eigenschaften auf einer Skala von 1 bis 7 vorgestellt, wobei höhere Werte eher mit der Applikation in Verbindung gesetzt wird. Hohe Werte (grösser als 5) erhielten die Wörter "übersichtlich", "direkt", "fesselnd", "kreativ", "vorzeigbar", "wertvoll", "schön", "einladend" und "anziehend". Tiefe Werte (kleiner als 3) erhielten die Wörter "praktisch", "voraussagbar", "kompliziert", "konventionell", "herkömmlich", "konservativ", "entmutigend", "stillos", "laienhaft", "unangenehm", "unsympathisch", "schlecht" und "entmutigend".

Somit verbinden die Testpersonen vorwiegend positive Eigenschaften mit der Applikation und auch Badertscher (2013, S. 85) schreibt: "Abschliessend lässt sich jedoch bereits erkennen, dass eine hohe Chance besteht, dass das Programm von den Benutzern akzeptiert wird."

Ich gehe davon aus, dass für diese Evaluation Desktop-Computer oder Laptops jedoch keine mobilen Geräte verwendet wurden. Da in der Evaluation die Usability der Desktop-Version bereits untersucht wurde, konzentriere ich mich im folgenden Kapitel auf die Usability bei mobilen Geräten.

2.1 Mobile Usability

2.1.1 Mobile Geräte

Die Mobilität, die mit diesen Geräten einhergeht, sollte auch für die hier behandelte Web-Applikation nicht ausser Acht gelassen werden. Insbesondere weil die Applikation es ermöglicht Dienstreisen zu erfassen. Somit ist das Szenario, dass ein Benutzer die Dienstreise gleich während der Dienstreise auf einem mobilen Gerät erfasst, sehr plausibel. Doch gelten für die Darstellung auf einem mobilen Gerät andere Anforderungen als bei der Gestaltung für herkömmliche Anzeigegeräte. Im Folgenden möchte ich drei Punkte aufzeigen, in denen sich mobile Geräte von herkömmlichen Desktop-Computern unterscheiden.

2.1.1.1 Bildschirmgrösse

Um die Mobilität zu gewährleisten, müssen mobile Geräte handlich und portabel sein. Das führt dazu, dass auch die Bildschirmgrösse relativ klein ist und somit weniger Platz für das Anzeigen von Inhalten zur Verfügung steht.

2.1.1.2 Steuerung

Bei der Gestaltung der Inhalte für verschiedene Geräte ist die Steuerung ein weiterer wichtiger Punkt. Während am Desktop-Computer Tastatur und Maus verwendet werden, ist bei den mobilen Geräten heutzutage meist eine Touch-Steuerung vorhanden.

Ein Unterschied ist die Genauigkeit bei der Eingabe. Mit der Maus ist eine höhere Genauigkeit möglich als bei der Touch-Steuerung (Nielsen & Budiu, 2013). Diese niedrigere Genauigkeit sollte bei der Gestaltung von Webseiten für mobile Geräte berücksichtigt werden. Shitkova et al. (2015) präsentieren in ihrer Arbeit eine Sammlung von Usability Guidelines für mobile Webseiten und mobile Apps. Sie schlagen vor, dass Buttons eine Grösse zwischen 7 und 10 mm haben sollten und die Webseite oder Applikation für unpräzise Eingaben optimiert werden sollte.

2.1.1.3 Downloadzeiten

Mobile Geräte nutzen grundsätzlich drahtlose Internetverbindungen wie WLAN oder das Mobilfunknetz. Gerade beim Mobilfunknetz sind enorme Differenzen in der Übertragungsgeschwindigkeit möglich: Das Mobilfunknetz mit der EDGE Technologie (Enhanced Data Rates for GSM Evolution) bietet unter optimalen Bedingungen eine Maximalgeschwindigkeit von 256 kBit/s. Die zur Zeit modernste Technologie LTE (Long Term Evolution) dagegen bietet eine Maximalgeschwindigkeit von bis zu 150 Mbit/s. (Swisscom, o. J.).

Um dem Nutzer unnötige Wartezeiten zu ersparen, sollte deshalb die mögliche Einschränkung durch eine langsame Internetverbindung auch bei der Erstellung von Webinhalten berücksichtigt werden. Nielsen und Budiu (2013, S. 100f) schlagen deshalb folgende Punkte vor:

- Die Anzahl Seitendownloads soll möglichst tief gehalten werden. Die Serverabfragen sollten möglichst gebündelt stattfinden. Das heisst beispielsweise, dass bei einer Eingabemaske mit mehreren Eingabefeldern nicht jedes einzelne Eingabefeld eine Serveranfrage starten sollte. Alle getätigten Eingaben sollten mit einer Serveranfrage überprüft werden.
- Die Seiteninhalte sollten sich auf die wichtigsten Informationen beschränken.
- Es sollten nicht zu viele Bilder eingebaut werden, da Bilddateien in der Regel eine grosse Datenmenge haben und somit die Downloadzeit verlängern.
- Den Nutzern sollte der Ladevorgang eines Downloads angezeigt werden, zum Beispiel in Form eines Ladebalkens.

2.1.2 Mobilstrategie

Um die Funktionen einer Webseite auch auf mobilen Geräten sinnvoll zur Verfügung zu stellen, gibt es mehrere Ansätze, die ich hier vorstellen will.

2.1.2.1 Native Apps

Native Apps gibt es zusätzlich oder anstelle einer Webseite. Diese Apps werden direkt auf dem Endgerät installiert und sind nur für eine bestimmte Plattform entwickelt (z.B. Android oder iOS). Die Entwicklung für mehrere Plattformen ist mit hohem Aufwand verbunden. Die Vorteile dieser Apps sind, dass sie auf Funktionen

des Geräts wie GPS zugreifen können, und dass sie auch offline funktionieren. Diese Apps werden im Normalfall über sogenannte App Stores installiert. Da sie plattformspezifisch entwickelt werden, sind sie meist auch der Benutzeroberfläche der Plattform angepasst (Nielsen & Budiu, 2013).

2.1.2.2 Hybride Apps

Neben den nativen Apps gibt es noch hybride Apps. Diese werden wie die nativen Apps über den App Store installiert. Allerdings sind sie meist nicht viel mehr als ein Webbrowser, der eine Seite darstellt, die mit den drei vorgestellten Web-Technologien erstellt werden kann. Dadurch sind die Entwicklungskosten einer solchen App relativ gering gegenüber einer nativen App. Die Erscheinung gleicht jedoch sehr derjenigen einer nativen App und ermöglicht die Präsenz im App Store (Nielsen & Budiu, 2013).

2.1.2.3 Mobile Webseite

Mobile Webseiten sind speziell für mobile Endgeräte entwickelt worden und existieren meist neben der normalen Desktop-Version. Zu diesen Seiten gehören auch die Web-Apps. Sie werden wie normale Webseiten über den Browser aufgerufen und müssen nicht über einen App Store geladen werden. Das Erscheinungsbild von Web-Apps ist aber möglichst nahe an einer nativen App. Die Web-App kann als Lesezeichen auf dem Startbildschirm von mobilen Geräten gespeichert werden. Somit kann sie wie eine native App geöffnet werden (Budiu, 2013).

2.1.2.4 Responsive Design

Beim *Responsive Design* wird die Webseite so gestaltet, dass sie sich an die Bildschirmgrösse und Ausrichtung des Anzeigegeräts anpasst. Somit sind keine zusätzliche App und keine separate mobile Webseite nötig. Durch die Veränderung der Darstellung ist es auch möglich, dass gewisse Funktionen bei kleinerer Bildschirmgrösse nicht mehr angezeigt werden. So kann auch der Funktionsumfang der Webseite an verschiedene Geräte angepasst werden (Schade, 2014).

Beim Responsive Design wird der Inhalt der Webseite in mehrere Spalten eingeteilt. Die Breite des Anzeigegeräts, bzw. des Browsers bestimmt dann, wie diese Spalten dargestellt werden. Je nach Displaygrösse werden die Spalten dann nebeneinander oder übereinander dargestellt (Schade, 2014).

In den Abbildungen 1 bis 3 ist zu sehen, wie sich der Inhalt einer Webseite mit Responsive Design bei verschiedenen Bildschirmgrössen verhalten kann. Durch die Darstellung der Spalten übereinander, wird der vertikale Scroll-Balken ersichtlich (siehe Abbildungen 2 und 3). Das horizontale Scrollen wird aber durch die Anpassung der Spalten verhindert. In Abbildung 3 ist zu sehen, dass auch das Design funktional angepasst wird. Die einzelnen Navigationspunkte sind nicht mehr sichtbar, stattdessen muss die Navigationsleiste zuerst über den Button *Menu* geöffnet werden.

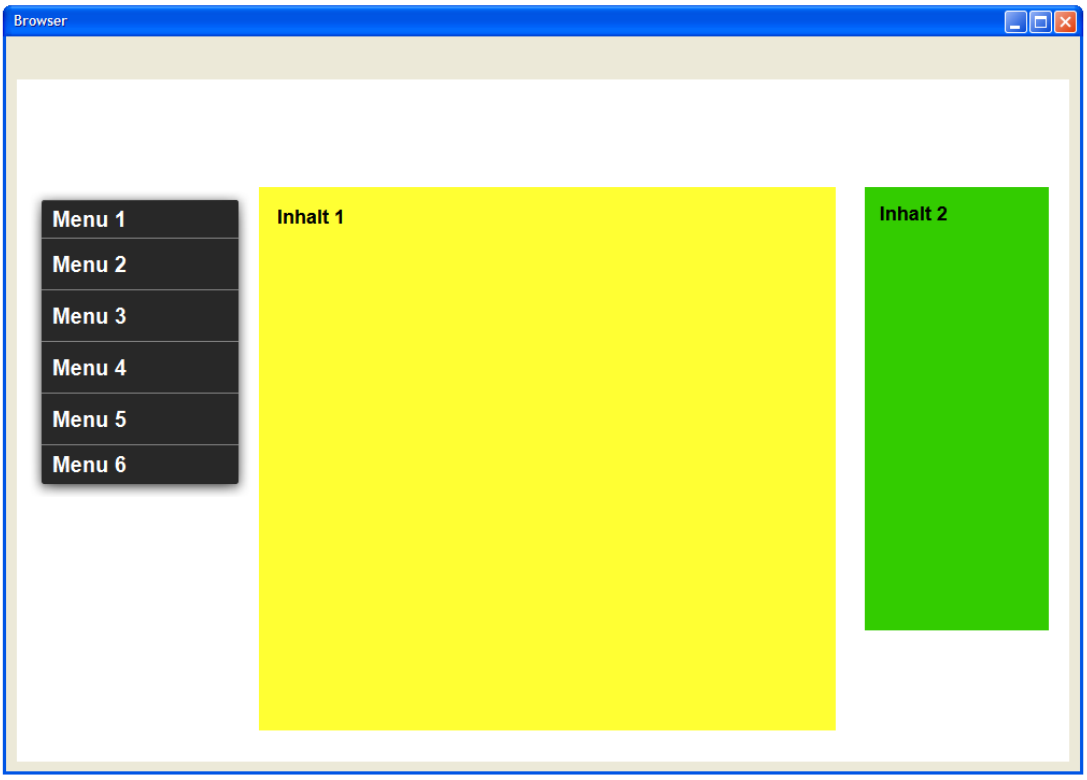

Abbildung 1: Beispiel des Responsive Design im Desktop-Browser (Eigene Darstellung)

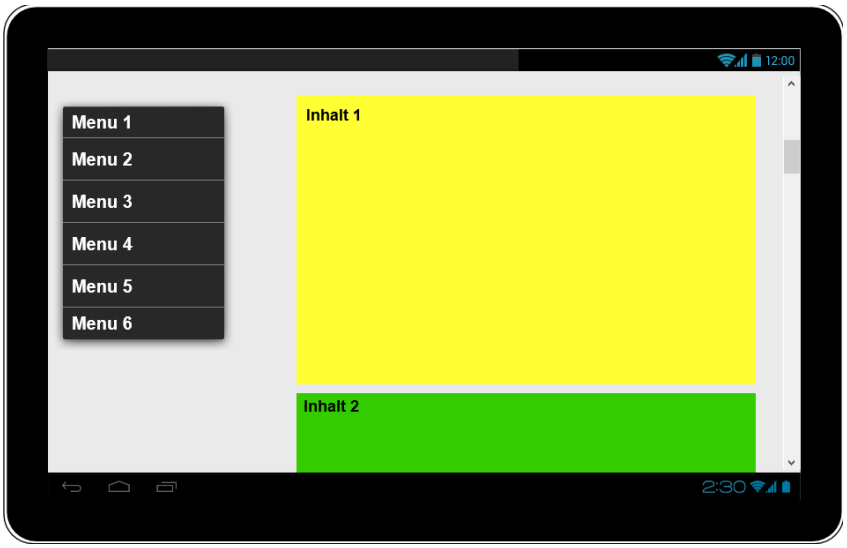

Abbildung 2: Beispiel des Responsive Design auf einem Tablet (Eigene Darstellung)

Abbildung 3: Beispiel des Responsive Design auf einem Smartphone (Eigene Darstellung)

3 Analyse der bestehenden Web-Applikation

Nachdem ich den Unterschied der nativen und hybriden App beschrieben habe und auf das Responsive Design eingegangen bin, stelle ich im Folgenden die wichtigsten Technologien und Funktionen der bestehenden Web-Applikation vor. Zudem zeige ich die Ursachen auf, welche verantwortlich sind für das Nicht-Funktionieren einiger Bereiche der Web-Applikation. Eine Diskussion der hier vorgestellten Befunde, wie auch die daraus resultierenden Optimierungsvorschläge, werden in einem weiteren Kapitel behandelt.

3.1 Darstellung

Die Web-Applikation ist nicht auf die Nutzung durch mobile Geräte optimiert. Die Grössen der einzelnen HTML-Elemente sind als feste Pixelgrössen hinterlegt. Das heisst die Applikation passt sich nicht an die Bildschirmgrösse an. Die Abbildungen 4 und 5 verdeutlichen diese Problematik. In Abbildung 4 ist zu sehen, wie im Browser-Fenster ein Teil der Seite abgeschnitten wird und der Scroll-Balken für horizontales Scrollen erscheint.

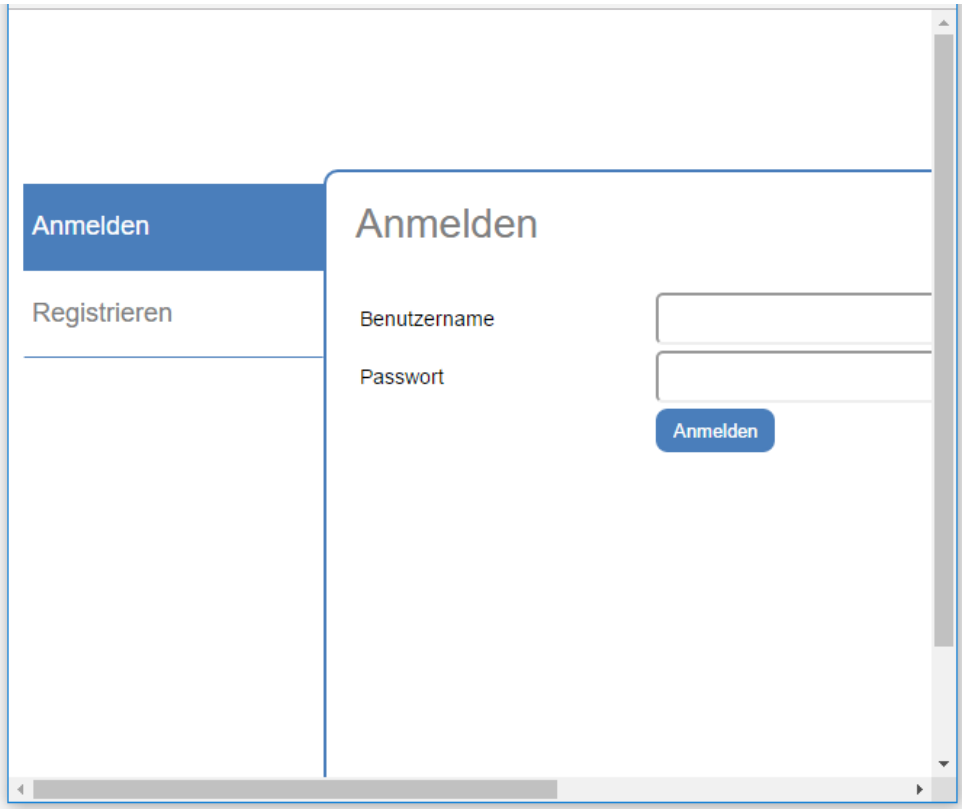

Abbildung 4: Anmeldeseite in verkleinertem Browserfenster (Eigene Darstellung)

Die Darstellung wurde auch auf einem Smartphone getestet. Als Testgerät diente ein Sony Xperia Z5 Compact. Die Darstellung der Web-Applikation auf dem Testgerät mit dem Browser Chrome ist in Abbildung 5 zu sehen. Die Applikation wird durch den Browser automatisch verkleinert, sodass kein horizontales Scrollen nötig ist. Dadurch werden auch die Schrift und die Bedienungsflächen verkleinert. Diese Darstellung ist für die Bedienung mit dem Finger zu klein. Die Seite muss zuerst durch Zoomen vergrössert werden, um sie bedienen zu können.

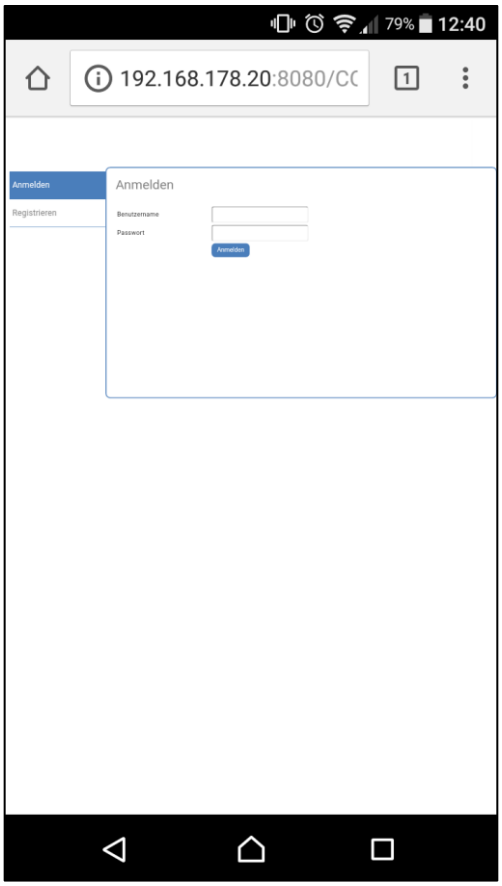

Abbildung 5: Anmeldeseite im Browser des Testgeräts (Eigene Darstellung)

3.2 Framework

Im Anschluss an die Ansichten für den Benutzer, möchte ich nun auf die technischen Aspekte eingehen. Für die Entwicklung der Web-Applikation wurde das Framework Google Web Toolkit (GWT) verwendet. Mit GWT können Web-Applikationen mit der Programmiersprache Java erstellt werden. Der GWT-Compiler kompiliert den client-seitigen Code zu JavaScript. Dadurch ist die Applikation in allen Browsern, die die Ausführung von JavaScript ermöglichen, lauffähig. Alle gängigen Browser (Safari, Firefox, Chrome, Microsoft Edge) wie auch die mobilen Browser für Android und iOS ermöglichen die Ausführung von JavaScript (GWT Project, o. J.-a).

Die bestehende Web-Applikation verwendet die Version 2.5 von GWT. Diese Version wurde am 24. Oktober 2012 veröffentlicht. Mittlerweile sind weitere Versionen verfügbar (GWT Project, o. J.-b).

3.3 Schnittstellen

Für die Berechnung der CO_2-Emissionen der Dienstreisen nutzt die Applikation externe Datenquellen. Einerseits werden Distanzangaben für die Reisen benötigt. Andererseits soll die Reise auf einer Google Maps Karte visualisiert werden. Um diese Daten zu erlangen, wird auf bereitgestellte APIs (Application Programming Interface) zugegriffen. Es wird eine Anfrage an einen externen Server geschickt, um die gewünschten Informationen zu erhalten. Meist werden innerhalb einer Dokumentation die Abfragemöglichkeiten einer API erklärt.

In den nächsten Abschnitten gehe ich auf diese Schnittstellen ein, die in der Web-Applikation verwendet werden.

3.3.1 Google Maps

Für die Visualisierung der Dienstreise wird auf die Google Maps JavaScript API zugegriffen. Zudem liefert diese API auch Distanzangaben. Diese API ist aber für den Gebrauch mit JavaScript ausgelegt. Damit aber die Anfrage an die API in Java geschrieben werden kann, ist die Java-Bibliothek GWT-Maps in der Applikation integriert.

3.3.1.1 GWT-Maps

Diese Java-Bibliothek ermöglicht, dass der Zugriff auf die Google Maps JavaScript API in Java geschrieben werden kann. Mithilfe dieser Bibliothek wird somit beim Kompilieren JavaScript-Code erstellt, welcher eine korrekte Abfrage an die API darstellt. Diese Bibliothek wird nicht mehr unterhalten oder weiterentwickelt (Google, o. J.).

3.3.1.2 Google Maps JavaScript API

Mit der beschriebenen Java-Bibliothek lässt sich der API Aufruf in Java schreiben. Für diesen Aufruf sind folgende Parameter wichtig:

- Schlüssel: Für den Aufruf der Google Maps JavaScript API ist zur Authentifizierung ein Schlüssel nötig (Google, 2016a). Mittels dieses Schlüssels werden die API-Aufrufe dokumentiert. Mit einem Standard-Schlüssel sind bis zu

25`000 Aufrufe innerhalb von 24 Stunden kostenlos. Weitere Aufrufe sind dann mit Kosten verbunden (Google, 2016b). Beim Testen der Web-Applikation funktionierte der Aufruf dieser Schnittstelle nicht, da bei der Anfrage kein gültiger Schlüssel angegeben wurde. Dadurch wurde die Reiseroute nicht visualisiert.

- Version: Hier wird angegeben, welche Version der Google Maps JavaScript API geladen wird. Im Code der untersuchten Web-Applikation wird die Version 2 der API geladen. Diese Version 2 wird aber seit November 2013 nicht mehr unterstützt. Es werden aber weiterhin Karten der Version 2 mittels eines JavaScript-Wrappers zur Verfügung gestellt. Für einfache Karten sollte somit dieser API-Aufruf noch funktionieren (Hoetmer, 2013).

3.3.2 Flightstats APIs

Eine weitere Schnittstelle sind die APIs von Flightstats. Die Flightstats APIs stellen Daten zu verschiedenen Bereichen im Flugwesen bereit. Zum Beispiel zu geplanten Flügen, zu Flughäfen und zu Airlines. Die Nutzung dieser APIs ist kostenpflichtig. Zum Testen der Web-Applikation konnte ich einen Test-Account einrichten. Dieser ist aber zeitlich beschränkt. Flightstats stellt mehrere APIs zur Verfügung. Jede API bietet Informationen zu einem bestimmten Bereich. Im Folgenden stelle ich die APIs vor, die in der Web-Applikation aufgerufen werden.

Für die Anfragen an die Flightstats APIs gibt es verschiedene Methoden. In der Web-Applikation wird die JSONP-Technik (JSON with Padding) benutzt. Mit dieser Methode lässt sich das Problem der Same Origin Policy vermeiden. Die Same Origin Policy verhindert den Zugriff auf einen anderen Server als den eigenen (Badertscher, 2013).

3.3.2.1 Airports API

Anhand der eingegebenen Start- und Zielflughäfen werden die Daten der Flughäfen mittels der Airports API ermittelt. Darunter ist auch das Flughafenkürzel, welches für weitere Abfragen benötigt wird (Flightstats, o. J.-a).

3.3.2.2 Connections API

Die Connections API wird in der Web-Applikation in der Version 1 aufgerufen. In der Dokumentation von Flightstats ist aber nur noch die Version 2 vorgestellt (Flightstats, o. J.-b). Deshalb funktioniert das Erfassen der Dienstreise per Flugzeug nicht. Ausserdem wird die Distanz der Flugreise, die zur Berechnung der CO_2-Emissionen benötigt wird, den Daten dieser API entnommen. In der Version 2 ist diese Information dagegen nicht mehr enthalten. Zudem werden die Antworten der Version 2 im JSON- oder XML-Format bereitgestellt (Flightstats, o. J.-b).

3.3.2.3 Flight Track API

Für das Darstellen der genauen Flugroute greift die Web-Applikation auf die Flight Track API zu. Allerdings bietet diese API nur Informationen zu aktiven Flügen an (Flightstats, o. J.-c).

Auf der Webseite von Flightstats konnte ich die Anfragen an die API direkt testen. Es sind nur Anfragen zulässig, die weniger als drei Tage in die Zukunft oder einige Tage in die Vergangenheit weisen. Das heisst die Informationen zur Route des geplanten Fluges können maximal drei Tage vor Abflug abgefragt werden. Die Positionsdaten der effektiv geflogenen Route sind somit nur wenige Tage über diese API abrufbar.

3.4 Berechnung der CO_2-Emissionen

Für die Berechnung der CO_2-Emission der Dienstreisen werden verschiedene Faktoren benutzt. Diese Emissionsfaktoren werden mit der Distanz der Reise multipliziert.

3.4.1 Auto

Bei einer Dienstreise mit dem Auto ist es möglich zwischen verschiedenen Fahrzeugtypen zu wählen. Für die verschiedenen Fahrzeugtypen sind auch verschiedene Emissionsfaktoren hinterlegt. Der Emissionsfaktor gibt an, wieviel Kilogramm CO_2 pro Kilometer ausgestossen werden. Die Distanz für die Reise mit dem Auto wird durch Daten der Google Maps JavaScript API berechnet.

3.4.2 Zug

Für die Zugreise ist im Quellcode der Emissionsfaktor 0.052 hinterlegt. Badertscher (2013, S. 78) verweist in seiner Arbeit darauf, dass die Distanzangaben für Zugreisen durch die Google Maps JavaScript API nur Näherungswerte sind.

3.4.3 Flugzeug

Bei der Flugreise stehen zwei verschiedene Klassen zur Auswahl: Business oder Economy. Für die beiden Klassen werden verschiedenen CO_2-Faktoren genutzt. Diese Faktoren werden mit der Distanz verrechnet. Die Distanz wird mittels Daten, welche über die Flightstats APIs geladen werden sollen, berechnet. Da aber die Connections API auf eine neue Version geändert hat, ist diese API-Anfrage in dieser Form nicht mehr möglich. Beim Testen funktioniert deshalb die Erfassung einer Dienstreise per Flugzeug nicht. Im Kapitel 4 beschreibe ich deshalb eine Alternative für die Berechnung der Dienstreise per Flugzeug.

4 Optimierungsmöglichkeiten der Web-Applikation

Mit den Informationen aus den vorangehenden Kapiteln möchte ich in diesem Kapitel die Optimierungsmassnahmen für die Web-Applikation ausführen.

4.1 Optimierung für mobile Geräte

Um die Nutzung der Web-Applikation auch für mobile Geräte zu verbessern, habe ich mich entschieden die Web-Applikation nach dem Responsive Design zu gestalten. Dafür sprechen folgende Gründe:

- Die Web-Applikation hat keine tiefe Navigationshierarchie. Eine solche wäre für mobile Geräte nicht geeignet (Nielsen & Budiu, 2013). Da aber diese nicht vorhanden ist, ist es sinnvoller die gleiche Hierarchie für alle Geräte zu behalten. Weil anzunehmen ist, dass die Nutzer die Web-Applikation auf verschiedenen Geräten nutzen, kann durch das Responsive Design ein einheitliches Layout und die gleichen Funktionen für alle Geräte geboten werden.
- Der Aufwand für die Entwicklung einer nativen App ist sehr hoch, insbesondere, wenn sie für verschiedene Plattformen bereitgestellt werden soll. Auf der anderen Seite sind die Vorteile einer nativen App für die Applikation nicht relevant: Die Applikation muss beispielsweise nicht auf tiefere Funktionen des Geräts zurückgreifen.

Um das Responsive Design umzusetzen soll die Darstellung mit CSS gesteuert werden. CSS erlaubt zudem den Einsatz von Media Queries. Mit Media Queries kann die Darstellung an verschieden grosse Displays angepasst werden.

Aus den Guidelines der Arbeit von Shitkova et al. (2015) habe ich eine Auswahl getroffen, die in der Web-Applikation angewendet werden sollen. Sie beschränkt sich auf die für die Web-Applikation relevanten Guidelines. Diverse Guidelines werden nicht aufgeführt, da die Applikation weder eine komplexe noch tiefe Navigationsstruktur hat. Auch die Inhalte sind relativ einfach: Es gibt keine langen Texte und keine Bilder. Die folgende Auswahl aus den Guidelines von

Shitkova et al. (2015, S. 1607) soll bei der Anpassung der Web-Applikation berücksichtigt werden:

1. *"Place content in the central part of the screen"*
2. *"Avoid horizontal scrolling"*
3. *"Arrange content vertically, avoid using tabs"*
4. *"Position buttons in the middle or at the end of the dialogue"*
5. *"Position buttons on the right side of the screen"*
6. *"Use button size between 7mm and 10mm"*
7. *"Provide automatic suggestions within the application (e.g. search autocomplete)"*
8. *"Prevent data loss by reminding users of unsaved changes"*
9. *"Optimize interface to correct imprecise touch control"*

4.2 Browser-Navigation

Heutige Browser ermöglichen mit den Zurück- und Vorwärts-Tasten das Navigieren zur vorherigen bzw. zur nächsten Seite. In einer GWT Applikation wird die Darstellung und die Navigation meist durch JavaScript bestimmt. Das heisst, sie besteht nicht aus mehreren HTML-Seiten wie eine klassische Webseite. Bei einer klassischen Webseite kann mit der Browser-Navigation durch die verschiedenen HTML-Seiten navigiert werden. Um die Browser-Navigation innerhalb des JavaScript Codes zu ermöglichen, kann die History-Funktion von GWT genutzt werden. Mithilfe dieses Mechanismus kann die Web-Applikation die Navigation durch den Browser unterstützen (GWT Project, o. J.-c).

Diese History-Funktion soll in der Web-Applikation implementiert werden, sodass die Browser-Navigation genutzt werden kann.

4.3 Framework und Java-Bibliothek GWT-Maps

Wie in Kapitel 3 beschrieben wird die Java-Bibliothek GWT-Maps nicht mehr weiterentwickelt und greift zudem auf eine ältere Version der Google Maps JavaScript API zurück. Dennoch ist die Bibliothek durch die bereitgestellten Wrapperklassen der Google Maps JavaScript API immer noch funktionsfähig.

Als Alternative wäre es möglich eigene Klassen zu schreiben, welche den Zugriff auf die Google Maps JavaScript API ermöglichen. Aber auch die Google Maps JavaScript API wird laufend weiterentwickelt, sodass eine solche Klasse ständig unterhalten werden müsste.

Es stellt sich die Frage, ob GWT die geeignete Entwicklungsumgebung für eine solche Applikation ist. Bei beiden externen Schnittstellen ist man auf zusätzlichen Code angewiesen. Bei der Google Maps JavaScript API wird die zusätzliche Java-Bibliothek benötigt. Bei den Flightstats APIs werden die Overlay-Klassen benötigt, die Badertscher (2013, S. 70-78) in seiner Arbeit beschreibt. Somit wäre eine weitere Alternative, die Applikation in einem JavaScript Framework wie AngularJS neu zu erstellen. Dadurch könnte mit JavaScript direkt auf die API zugegriffen werden. Die Unterhaltung der API-Aufrufe wäre dadurch sicher einfacher, da die Overlay-Klassen und die Java-Bibliotheken nicht gebraucht und somit nicht unterhalten werden müssten.

Da beide Alternativen im Rahmen dieser Arbeit zu aufwendig sind, wird mit dem bestehenden Framework und der funktionierenden Java-Bibliothek weitergearbeitet. Als Framework soll aber eine neuere Version von GWT verwendet werden.

4.4 Berechnung der CO_2-Emissionen durch Flugreisen

Im Kapitel zur Schnittstelle von Flightstats beschrieb ich, weshalb die Berechnung der CO_2-Emissionen für Flugreisen nicht mehr funktioniert. Deshalb ist es hier nötig, die Berechnung anzupassen, bzw. eventuell auf andere Datenquellen zurückzugreifen. Folgende Möglichkeiten habe ich evaluiert:

4.4.1 Berechnung mit Daten von Flightstats

Der Vorteil von Flightstats sehe ich darin, dass diese API die Daten der genauen Flugroute liefern kann. Dieser Vorteil ist nicht unerheblich, da er massgeblich zur Genauigkeit der Flugdistanz beiträgt. Gegen die Benutzung dieser API sprechen folgende Gründe:

Die aktuelle Version der Connections API bietet neu keine Informationen mehr zur Distanz oder zu den Wegpunkten (Flightstats, o. J.-b). Die genauen Flugdaten sind nur über die Flight Track API abrufbar und nur für einen sehr begrenzten Zeitraum. Meiner Meinung nach sollte aber die Berechnung der CO_2-Emissionen bereits bei der Planung der Dienstreise möglich sein, sodass ein Nutzer die berechneten CO_2-Emissionen in seine Planung einfliessen lassen kann. Ein Beispiel wäre eine innereuropäische Dienstreise, bei der die Wahl zwischen Bahn und Flugzeug besteht. Der Nutzer sollte nun bei der Planung, die mehrere Wochen vor Antritt der Reise sein kann, bereits die möglichen CO_2-Emissionen berechnen können.

Zudem stellt die Connections API in der Version 2 laut Flightstats (o. J.-b) nur die Abfrageformate JSON und XML zur Verfügung. Somit wäre die bisherige Abfrage mit JSONP nicht möglich und es müsste eine neue Lösung für die Same Origin Policy gefunden werden.

4.4.2 Berechnung mit der Great Circle Distance

Die Organisation myclimate bietet auf ihrer Webseite einen CO_2-Rechner an. Dieser Rechner berechnet das CO_2-Äquivalent für eine Flugreise. Da verschiedene Treibhausgase unterschiedliche Auswirkungen auf die Erderwärmung haben, können diese in CO_2-Äquivalente umgerechnet werden, um damit die Auswirkungen aller Treibhausgase vereinheitlicht darzustellen. Somit können die gesamten Treibhausgas-Emissionen zusammengefasst werden (myclimate, o. J.).

Die Berechnungsmethode beschreibt myclimate in der Dokumentation des Flugrechners (myclimate, 2015). Im Folgenden beschreibe ich diese Berechnung in verkürzter Form:

Um das CO_2-Äquivalent der Flugreise zu berechnen, werden Durchschnittsdaten aus dem Flugwesen benutzt. Für die Distanz wird die Great

Circle Distance verwendet. Diese stellt die kürzeste Distanz zwischen zwei Flughäfen dar. Da diese Distanz von der Distanz der effektiven Flugroute abweicht, wird eine Distanzkorrektur dazugerechnet. Um Näherungswerte für den Treibstoffverbrauch zu berechnen, wird folgende quadratische Gleichung genutzt:

$$f(x) + LTO = ax^2 + bx + c \qquad \text{(myclimate, 2015, S. 3)}$$

Wobei x die Distanz inklusive Distanzkorrektur ist und LTO der zusätzliche Treibstoffverbrauch für Start und Landung. Der Treibstoffverbrauch wird dann auf den einzelnen Passagier berechnet. Dazu wird die durchschnittliche Anzahl von Sitzplätzen (S) und die durchschnittliche Auslastung der Flugzeuge (PLF) hinzugezogen. Da neben Passagieren auch Frachtgüter transportiert werden, wird ein Frachtfaktor (CF) miteinberechnet. Für die verschiedenen Klassen gibt es ebenfalls einen Faktor (CW), da Business- oder First-Class Sitze mehr Platz im Flugzeug einnehmen. Dementsprechend werden diesen Klassen mehr Emissionen angerechnet. Der Treibstoffverbrauch eines Passagiers wird dann mit dem CO_2-Emissionsfaktor von Kerosin (EF) verrechnet. Da schlussendlich das CO_2-Äquivalent berechnet wird, wird ein zusätzlicher Faktor hinzugezogen, der den Ausstoss weiterer Treibhausgase berücksichtigt (M). Zusätzlich werden auch die Emissionen zur Herstellung von Kerosin miteinberechnet (P). Daraus ergibt sich folgende Formel:

$$E = \frac{ax^2+bx+c}{S \times PLF} \times (1 - CF) \times CW \times (EF \times M + P) \qquad \text{(myclimate, 2015, S. 4)}$$

Bei der Berechnung wird zwischen Kurz- und Langstreckenflügen unterschieden. Die Werte, welche für die Faktoren genutzt werden, sind in Anhang A aufgeführt.

In dieser Form der Berechnung sehe ich folgende Vorteile: Die CO_2-Emissionen können immer berechnet werden und sind nicht abhängig davon, ob die Flugdaten vorhanden sind. Zudem kann die Great Circle Distance mit den Daten der Google Maps JavaScript API berechnet werden. Somit fallen weitere API-Anfragen an Flightstats weg. Dadurch müssen weniger Daten übertragen werden und es können Ladezeiten, vor allem beim mobilen Gebrauch, vermieden werden. Deshalb werde ich diese Methode zur Berechnung der CO_2-Emissionen für Flugreisen nutzen.

5 Umsetzung

5.1 Anpassung an mobile Geräte

5.1.1 Responsive Design

Um die Web-Applikation dem Responsive Design anzupassen, habe ich zuerst die Darstellungsbedingungen aus dem Java Code entfernt. Die Darstellung soll durch CSS-Regeln bestimmt werden. CSS ermöglicht zudem die Anpassung der Darstellung an das Ausgabegerät. In GWT wird die Darstellung durch verschiedene Panels aufgebaut. Diese Panels werden beim Kompilieren in HTML-Elemente umgewandelt. Auf diese HTML-Elemente werden die CSS-Regeln angewendet. In der Applikation wurden viele Panels (bspw. HorizontalPanel) implementiert, die zum HTML-Element *Table* kompiliert werden. Dieses Element ist aber für die Gestaltung der Seite nicht geeignet, da es eine starre Tabelle ist. Dadurch ist das flexible Anpassen an verschiedene Darstellungsgrössen nicht möglich. Deshalb habe ich den Seitenaufbau mit FlowPanels gestaltet. Diese werden zum HTML-Element *Div* umgewandelt.

In der CSS-Datei habe ich die Darstellungsregeln angepasst und neue erstellt. Damit die Darstellung auf verschieden grosse Ausgabegeräte angepasst ist, habe ich Media Queries implementiert. Diese Media Queries erlauben, bestimmte CSS-Regeln auf bestimmte Ausgabegeräte anzupassen. In Abbildung 6 ist eine solche Media Query abgebildet. Alle CSS-Regeln, die innerhalb dieser Query stehen, gelten somit nur, wenn für die Darstellung der Breite mehr als 399 Pixel zu Verfügung stehen.

```
@media only screen and (min-width: 399px) {
    .businessTripSelectionPanel {
        display: inline-block;
```

Abbildung 6: Beispiel einer CSS Media Query (Eigene Darstellung)

Für das Responsive Design habe ich die Ansicht in zwei Spalten eingeteilt: Navigationsmenu und Inhalt. Der Inhalt wiederum ist nochmals in zwei Spalten aufgeteilt. Dies ist in Abbildung 7 zu sehen: Links wird im Navigationsbereich das

Navigationsfeld *Zurück* angezeigt. Im Inhaltsbereich werden die Eingabefelder und daneben die Google Maps Karte dargestellt. Die Anzeige des Browsers hat bei dieser Einstellung eine Breite von 1400 Pixeln.

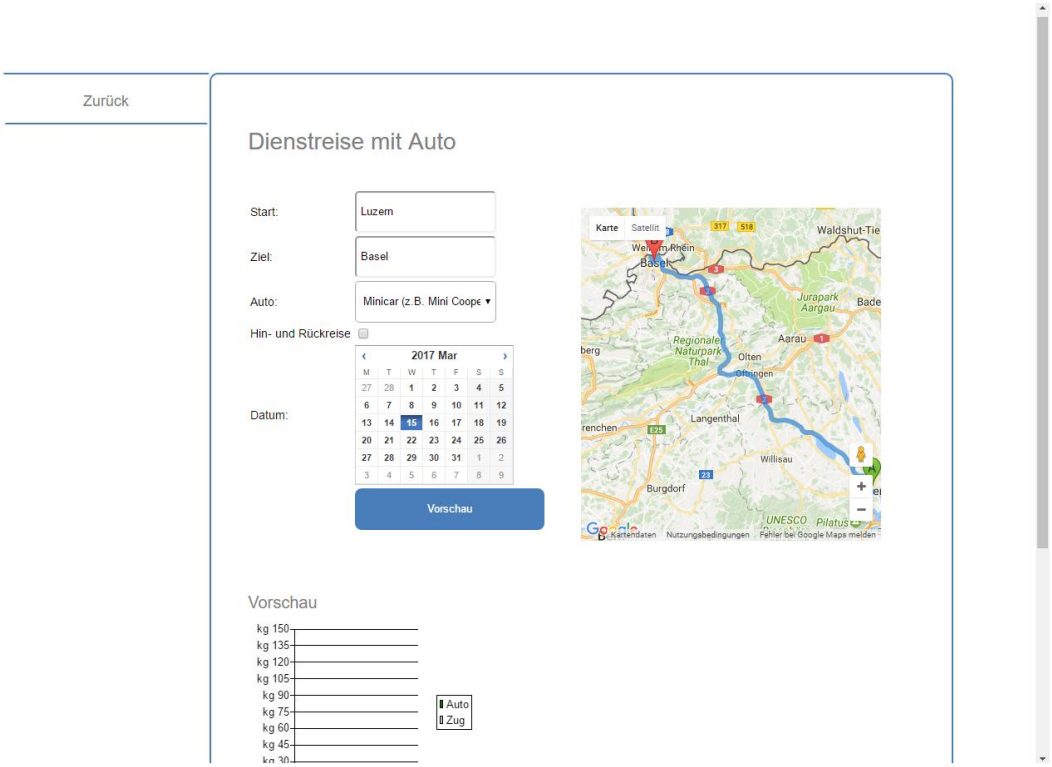

Abbildung 7: Ansicht der Erfassungsmaske einer Dienstreise bei einer Breite von 1400 Pixeln (Eigene Darstellung)

In Abbildung 8 wurde das Browserfenster auf eine Breite von 1000 Pixeln verkleinert. Der Inhaltsbereich wird nur noch in einer Spalte dargestellt. Die Google Maps Karte wird nicht mehr neben, sondern unter den Eingabefeldern angezeigt.

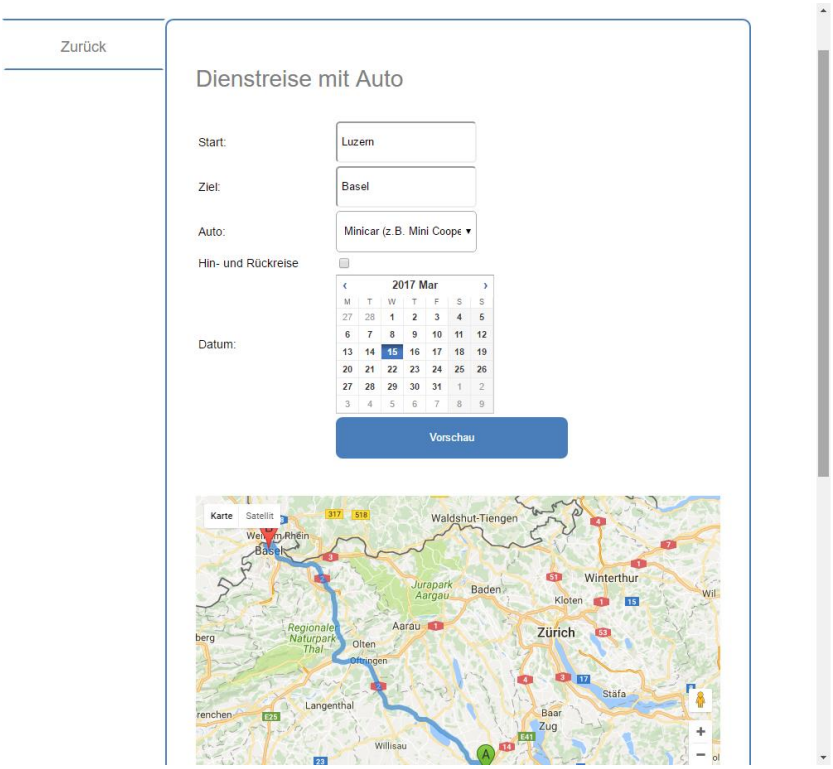

Abbildung 8: Ansicht der Erfassungsmaske einer Dienstreise bei einer Breite von 1000 Pixeln (Eigene Darstellung)

Abbildung 9 zeigt die Darstellung auf einem Smartphone. Bei dieser Breite wird nur noch eine Spalte angezeigt. Der Navigationspunkt *Zurück* ist zuoberst platziert.

Abbildung 9: Ansicht der Erfassungsmaske einer Dienstreise auf dem Testgerät (Eigene Darstellung)

Die Navigationspunkte nehmen bei der Darstellung auf dem Testgerät die gesamte Breite des Displays ein. In Abbildung 9 ist der Navigationspunkt *Zurück* sichtbar. Mit diesem gelangt man zurück zur Startseite. In Abbildung 10 ist das gesamte Navigationsmenu zu sehen. Dieses erscheint auf der Startseite.

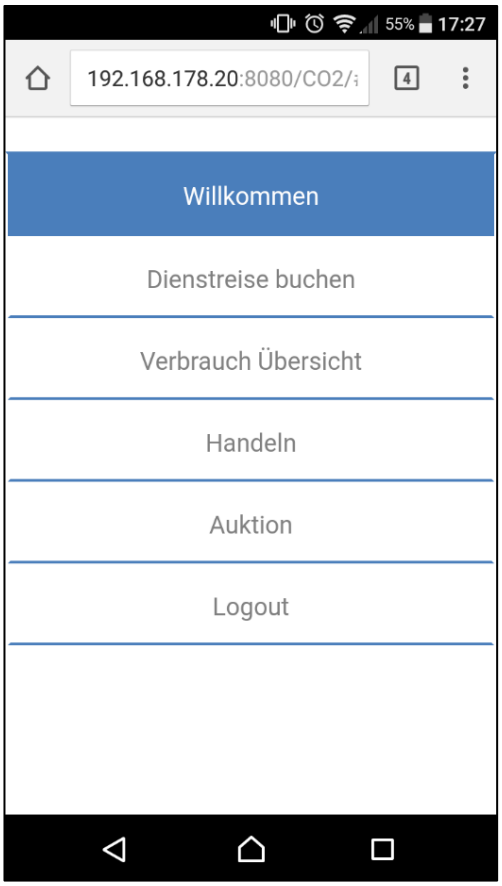

Abbildung 10: Startseite der Web-Applikation (Eigene Darstellung)

5.1.2 Guidelines

In den Abbildungen 9 und 10 ist ersichtlich wie der Inhalt zentral und in vertikaler Reihenfolge platziert ist. So wie es in den Guidelines 1 und 3 vorgeschlagen wird. Durch die Anpassung an die jeweilige Breite des Anzeigegeräts wird auch das horizontale Scrollen soweit wie möglich verhindert, wie dies Guideline 2 verlangt. Allerdings kann das horizontale Scrollen nur bis zu einem gewissen Punkt verhindert werden, da bestimmte Elemente doch noch eine feste Grösse haben und nicht beliebig verkleinert werden können. In Abbildung 11 ist das Balkendiagramm für den Vergleich der CO_2-Emissionen sichtbar. Es hat eine feste Grösse und deshalb erscheint hier der horizontale Scroll-Balken. In dieser Abbildung wurde das Browserfenster auf 200 Pixel verkleinert.

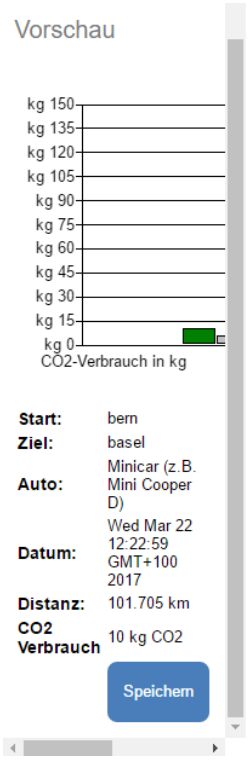

Abbildung 11: Vorschau einer Dienstreise bei einer Breite von 200 Pixeln (Eigene Darstellung)

Die Guidelines betreffend den Button (4 bis 6) sind am Beispiel der Abbildung 12 zu sehen. Der Anmeldebutton ist am Ende der Eingabemaske und auf der rechten Seite platziert. Für die Grösse des Buttons habe ich Mindesthöhen implementiert, sodass die vorgeschlagenen 7 Millimeter nicht unterschritten werden. Ansonsten passt sich der Button ebenfalls dem Anzeigegerät an.

Guideline 9 habe ich insofern umgesetzt, dass auch weitere Felder, die Touch-Eingaben verlangen, vergrössert wurden. Als Beispiel sind in Abbildung 12 die Eingabefelder für den Benutzernamen und das Passwort zu sehen. Diese Felder, wie auch die Navigationsleiste, habe ich an die vorgeschlagene Grösse für Buttons angepasst.

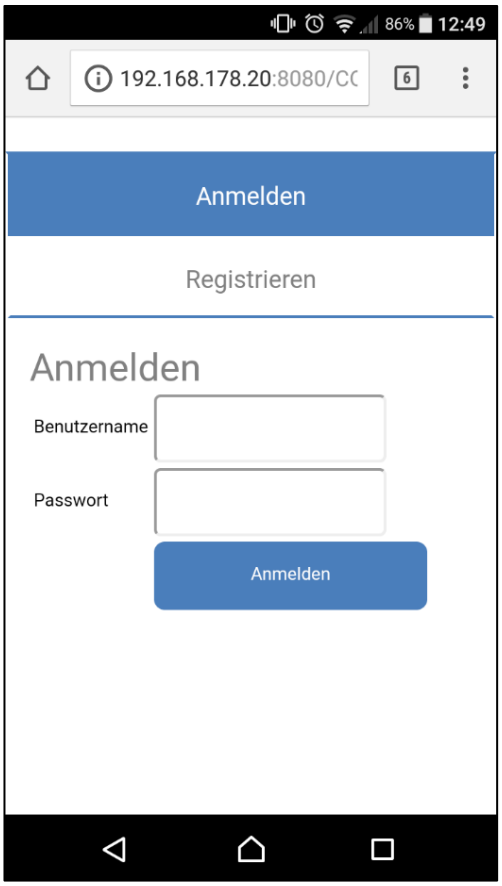

Abbildung 12: Anmeldeseite der Web-Applikation (Eigene Darstellung)

Die Guidelines 7 und 8 konnte ich nicht implementieren. Guideline 7 besagt, dass dem Nutzer bei Eingaben automatisch Vorschläge präsentiert werden sollten. Es wäre gut vorstellbar, dass beim Eintragen einer Dienstreise Vorschläge für die gesuchten Orte angezeigt werden. Dadurch wären aber weitere API-Anfragen nötig und es stellt sich die Frage, ob dadurch nicht lästige Unterbrüche entstehen können. Die Guideline 8 könnte in einem weiteren Schritt ebenfalls umgesetzt werden: Beim Verlassen einer Dienstreise-Erfassung könnte der Benutzer daran erinnert werden, dass er seine Daten noch nicht gespeichert hat.

5.2 Browser-Navigation

Mit dem History-Mechanismus von GWT konnte ich die Navigation durch den Browser ermöglichen. Bei jedem Wechsel der Ansicht wird der jeweilige Name der Ansicht in die URL-Adresse geschrieben. Somit kann mit der Browser-Navigation zwischen diesen verschiedenen URL-Adressen navigiert werden.

5.3 Berechnung der CO_2-Emissionen bei Flugreisen

Für die Berechnung der CO_2-Emissionen für Flugreisen habe ich eine neue Ansicht erstellt. Die neue Ansicht ist entsprechend den Ansichten für Zug- und Autoreisen gestaltet. In Abbildung 13 sind die Eingabefelder für die Flugreise zu sehen. Der Benutzer kann dort den Start- und Zielflughafen eingeben. Mit dem Button *Vorschau* werden dem Benutzer die Daten der geplanten Reise angezeigt (siehe Abbildung 14). Gleichzeitig können die Angaben immer noch angepasst werden. Die CO_2-Emissionen werden gemäss dem Kapitel 4.4.2 berechnet.

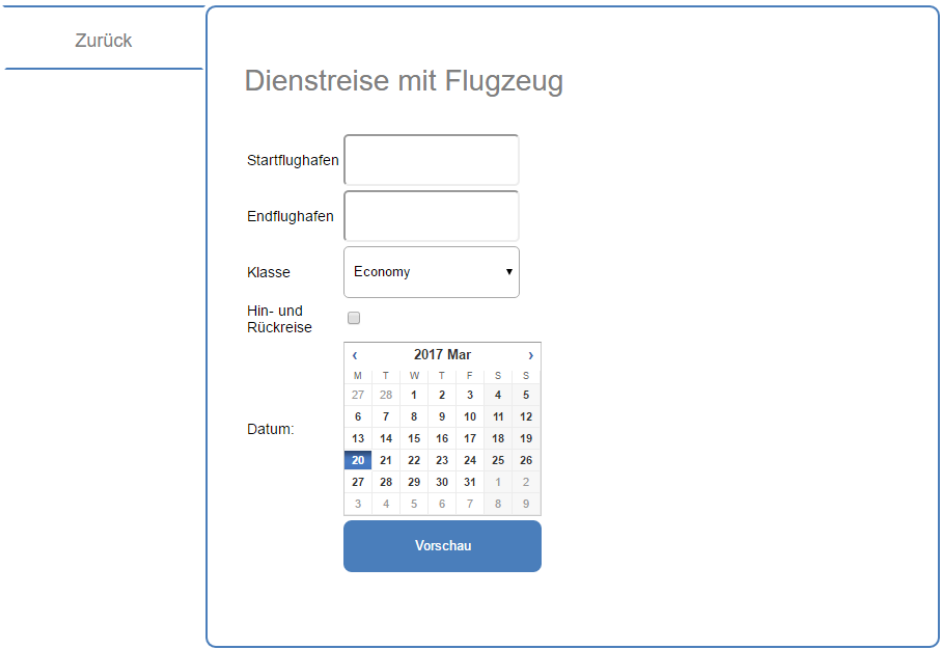

Abbildung 13: Eingabemaske für eine Dienstreise mit Flugzeug (Eigene Darstellung)

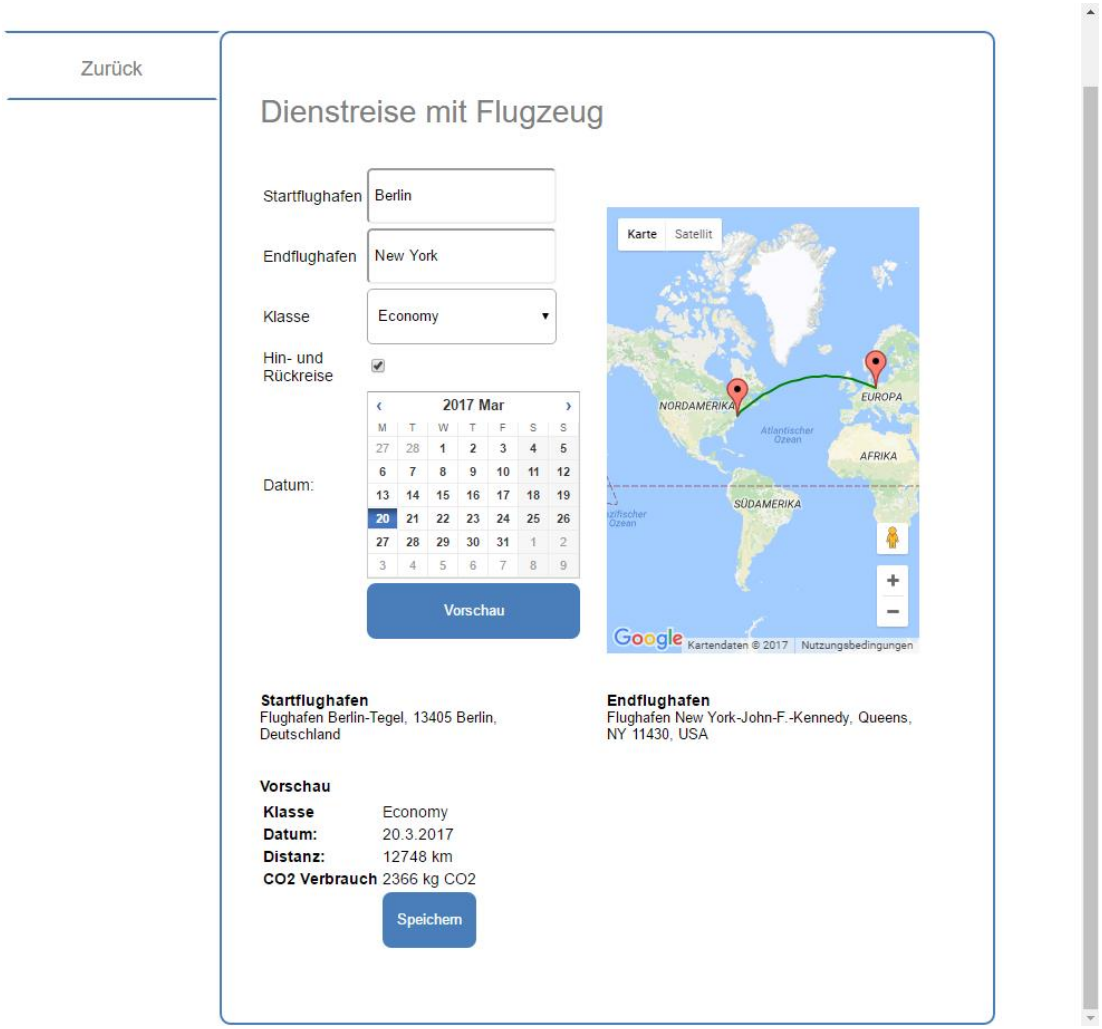

Abbildung 14: Vorschau einer Dienstreise mit Flugzeug (Eigene Darstellung)

6 Fazit und Ausblick

6.1 Fazit

Da die Zugriffe der Web-Applikation auf die Schnittstellen nicht mehr funktioniert haben, konnte ich durch das Implementieren der neuen Berechnung für die CO_2-Emissionen für Flugreisen und Anpassungen an den Aufrufen der Google Maps JavaScript API den ursprünglichen Funktionsumfang der Web-Applikation wiederherstellen. Mit der Anwendung des Responsive Designs und vorgeschlagenen Guidelines für mobile Web-Applikationen konnte ich die Web-Applikation auch für mobile Geräte optimieren. Zudem konnte mit dem History-Mechanismus von GWT die Navigation durch den Browser ermöglicht werden.

6.2 Ausblick

Im Verlauf meiner Arbeit sind verschiedene Fragen aufgetaucht, denen in weiteren Arbeiten nachgegangen werden könnte. Einerseits wurde mir bewusst, wie schnell sich die Technologien im Web-Bereich verändern. Demnach wäre es spannend, der Frage nachzugehen, inwiefern es möglich ist in einem solchen Umfeld nachhaltigen Code zu schreiben, um Unterhaltsarbeiten zu minimieren oder verhindern. Im Falle der Web-Applikation kann überprüft werden, ob das Framework GWT immer noch eine gute Lösung ist oder ob andere Lösungen eine mögliche Weiterarbeit vereinfachen könnten. Andererseits gibt es noch andere Aspekte der Web-Applikation, die vertieft untersucht werden könnten, bspw. der Aspekt der Datensicherheit.

Zudem könnte man sich in weiteren Arbeiten vertiefter mit der Berechnung von CO_2-Emissionen auseinandersetzen. Dadurch könnte eventuell auch wieder auf Daten der Flightstats APIs zurückgegriffen werden. Weiter wäre es spannend, die Web-Applikation in einer Organisation oder Gruppe einzuführen und zu testen.

7 Literaturverzeichnis

Badertscher, S. (13. April 2013). Generalisierung des intra-organisationalen Cap and Trade. (Masterarbeit Universität Zürich)

Budiu, R. (14. September 2013). *Mobile: Native Apps, Web Apps, and Hybrid Apps*. Abgerufen am 3. März 2017 von https://www.nngroup.com/articles/mobile-native-apps/

Bundesamt für Umwelt. (7. Januar 2015a). *Klimawandel*. Abgerufen am 15. März 2017 von https://www.bafu.admin.ch/bafu/de/home/themen/klima/fachinformationen/klimawandel.html

Bundesamt für Umwelt. (24. April 2015b). *Emissionshandel*. Abgerufen am 9. 3 2017 von https://www.bafu.admin.ch/bafu/de/home/themen/klima/fachinformationen/klimapolitik/emissionshandel.html

Center for American Progress. (16. Januar 2008). *Cap and Trade 101*. Abgerufen am 13. März 2017 von https://www.americanprogress.org/issues/green/news/2008/01/16/3816/cap-and-trade-101/

Flightstats. (o. J.-a). *Airports API*. Abgerufen am 3. März 2017 von https://developer.flightstats.com/api-docs/airports/v1

Flightstats. (o. J.-b). *Connections*. Abgerufen am 3. März 2017 von https://developer.flightstats.com/api-docs/connections/v2

Flightstats. (o. J.-c). *Flight Status & Track by Flight*. Abgerufen am 6. März 2017 von https://developer.flightstats.com/api-docs/flightstatus/v2/flight

Google. (12. Oktober 2016a). *Get a Key/Authentication*. Abgerufen am 21. Dezember 2016 von https://developers.google.com/maps/documentation/javascript/get-api-key?hl=de

Google. (12. Oktober 2016b). *JavaScript API Usage Limits*. Abgerufen am 21. Dezember 2016 von https://developers.google.com/maps/documentation/javascript/usage?hl=de

Google. (o. J.). *gwt-google-apis*. Abgerufen am 15. Dezember 2016 von https://github.com/googlearchive/gwt-google-apis

GWT Project. (o. J.-a). *Overview*. Abgerufen am 15. Dezember 2016 von http://www.gwtproject.org/overview.html

GWT Project. (o. J.-b). *Versions*. Abgerufen am 15. Dezember 2016 von http://www.gwtproject.org/versions.html

GWT Project. (o. J.-c). *History*. Abgerufen am 6. März 2017 von http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsHistory.html

Hoetmer, K. (29. April 2013). *An update on the JavaScript Maps API v2 deprecation*. Abgerufen am 21. Dezember 2016 von https://maps-apis.googleblog.com/2013/04/an-update-on-javascript-maps-api-v2.html

Mozilla Developer Network. (4. Februar 2017). *What is JavaScript?* Abgerufen am 21. März 2017 von https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

myclimate. (3. September 2015). *THE MYCLIMATE FLIGHT EMISSION CALCULATOR.* Abgerufen am 21. März 2017 von https://www.myclimate.org/fileadmin/myc/ueber-uns/downloads/myclimate-flight-calculator-documentation_EN.pdf

myclimate. (o. J.). *Was sind CO2-Äquivalente?* Abgerufen am 21. März 2017 von https://www.myclimate.org/de/website/faq/detail/was-sind-co2-aequivalente/

Nielsen, J., & Budiu, R. (2013). *Mobile Usability: Für iPhone, iPad, Android, Kindle.* (mitp Verlag, Übers.) Frechen: mitp Verlag.

Oertle, D. (12. April 2013). Plattform für den Handel von Ressourcen innerhalb eines Unternehmens am Beispiel von CO2-Zertifikaten für Dienstreisen. (Masterarbeit Universität Zürich)

Schade, A. (4. Mai 2014). *Responsive Web Design (RWD) and User Experience*. Abgerufen am 11. März 2017 von https://www.nngroup.com/articles/responsive-web-design-definition/

Shitkova, M., Holler, J., Heide, T., Clever, N., & Becker, J. (2015). Towards Usability Guidelines for Mobile Websites and Applications. In O. Thomas, & F. Teuteberg (Hrsg.), *Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik (WI 2015)*, (S. 1603-1617). Osnabrück.

Swisscom. (o. J.). Abgerufen am 8. März 2017 von https://www.swisscom.ch/de/privatkunden/mobile/mobilnetz.html

W3C. (o. J.). *HTML & CSS*. Abgerufen am 21. März 2017 von https://www.w3.org/standards/webdesign/htmlcss#whathtml

Warndorf, D. (22. Oktober 2013). *Was ist eigentlich eine Webapplikation?* Abgerufen am 14. März 2017 von http://www.zweiblog.com/2013/10/was-ist-eigentlich-eine-webapplikation/

Anhang A: Parameter für die Berechnung des CO_2-Äquivalents

Aircraft type	Average seat number (S)	Passenger load factor (PLF)	Detour constant (DC)	1-Cargo factor (1-CF)	Economy class weight (CW)	Business class weight (CW)	First class weight (CW)	Emission factor (EF)	Preproduction (P)	Multiplier (M)
Hybrid short-haul	158.44	0.77	50	0.951	0.96	1.26	2.4	3.15	0.51	2
Hybrid long-haul	280.39	0.77	125	0.951	0.8	1.54	2.4	3.15	0.51	2

Tabelle A1: Faktoren zur Berechnung des CO_2-Äquivalents einer Flugreise (myclimate, 2015, S. 5)

	a	b	c
Generic short-haul	3.87871E-05	2.9866	1263.42
Generic long-haul	0.000134576	6.1798	3446.2

Tabelle A2: Variablen für die Berechnung des Treibstoffverbrauchs (myclimate, 2015, S. 5)