

Besondere Lernleistung

**Webanwendung zum Finden eines optimalen Standortes
durch Distanzminimierung mithilfe von
Online-Kartendiensten**

Gernot Zacharias

15. Dezember 2019

Inhaltsverzeichnis

| | |
|---|-----------|
| Zusammenfassung | 2 |
| Abkürzungsverzeichnis | 3 |
| 1 Einleitung | 4 |
| 1.1 Gegenstand und Motivation | 4 |
| 1.1.1 Gegenstand | 4 |
| 1.1.2 Problematik | 4 |
| 1.2 Motivation | 4 |
| 1.3 Problemstellung | 4 |
| 1.4 Zielsetzung | 5 |
| 1.5 Aufgabenstellung | 5 |
| 1.6 Existierende Lösungen | 5 |
| 2 Grundlagen | 7 |
| 2.1 World Wide Web | 7 |
| 2.2 HTML | 7 |
| 2.3 JavaScript | 7 |
| 2.4 Google Maps | 8 |
| 2.5 Open Street Map | 8 |
| 3 Algorithmus | 9 |
| 4 Implementierung | 10 |
| 5 Ergebnis | 15 |
| 6 Diskussion | 17 |
| 6.1 Auswertung | 17 |
| 6.2 Ausblick | 17 |

Zusammenfassung

Durch den aufstrebenden Online-Versandhandel, wie zum Beispiel durch Online-Versandapotheke, entstehen neue Probleme. Die Versandhändler benötigen Lagerhäuser für ihre Waren und die Fahrtwege vom Lager zum Kunden sollten möglichst minimal gehalten werden, um Zeit und Geld zu sparen. Durch die Verkürzung der Fahrwege wird auch die Gesundheit und die Natur weniger belastet. Auch Privatanwender profitieren von kurzen Fahrtwegen. Dieses Problem lässt sich durch eine Anwendung lösen, die aus der Häufigkeit der angesteuerten Orte eine optimale Position mit möglichst kurzen Fahrwegen bestimmt. Diese Anwendung kann zum Beispiel auf Basis der Kartendienste von „Google Maps“ oder „Open Street Map“ realisiert werden. Dies liegt sich am einfachsten durch eine Webanwendung lösen, da diese meistens plattformunabhängig sind. Somit ist keine Installation von zusätzlicher Software, außer einem aktuellen Webbrower, nötig. Des weiteren wurde der freie Kartendienst von *Open Street Map* gewählt, da dieser für nicht kommerzielle Zwecke kostenlos ist. Auch gibt es für Open Street Map eine größere Auswahl an Nutzerschnittstellen. Die Anwendung funktioniert sehr gut. Somit lässt sich auch der Treibstoffverbrauch reduzieren, was dem Geldbeutel und der Umwelt zu Gute kommt. Es gab auch positives Feedback von anderen Personen, die das Programm getestet haben. Allerdings bietet die Anwendung auch noch viel Spielraum für individuelle Erweiterungen, Verbesserungen und Änderungen.

Abkürzungsverzeichnis

| Begriff | Erklärung des Begriffs |
|---------|------------------------------------|
| API | Application Programming Interface |
| CSS | Cascading Style Sheets |
| GIS | Geographic Information System |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| JS | JavaScript |
| PNG | Portable Network Graphic |
| WWW | World Wide Web |

1 Einleitung

1.1 Gegenstand und Motivation

1.1.1 Gegenstand

Versandhändler suchen eine möglichst kurze Strecke zum Kunden und zur Produktionsstätte. Auch Normalverbraucher wünschen sich einen möglichst kurzen Weg zu ihrer Arbeit, zum Kindergarten, Arzt, oder zu ihrem (Sport-)Verein und sucht deshalb eine Wohnung von der Er diese Orte auf dem kürzesten Weg erreicht.

1.1.2 Problematik

Durch lange Fahrtwege erhöht sich die Dauer der Fahrt bis zum Ziel und somit der Energieverbrauch sowie die Schadstoffbelastung. Auch die Kosten erhöhen sich bei langen Fahrten, da der Fahrer und/oder der Treibstoff bezahlt werden muss. Die Kraftfahrer werden durch längere Strecken stärker belastet und das Risiko für Unfälle aufgrund von Übermüdung steigt.[3]

1.2 Motivation

Durch die Reduktion der Distanz zum Ziel wird der Energieverbrauch und die Zeit verringert, dadurch kann Geld gespart und die Umwelt geschont werden. Auch wird der Fahrer durch eine geringere Fahrzeit weniger stark belastet.

1.3 Problemstellung

Der Online Versandhandel ist ein zurzeit stark wachsender Sektor der Wirtschaft.[4] Deshalb verwundert es auch nicht das es heutzutage Es gibt auch Online-Versandapothenen, welche Medikamente an die Haustür liefern. Dies ist vor

allem für bewegungseingeschränkte Personen interessant, da diese sich nun nicht mehr zu nächsten Apotheke bewegen müssen. Auch ermöglicht es Personen die nicht in der Nähe einer Apotheke wohnen, sich die benötigten Medikamente zu besorgen. Für die Anbieter dieser Versandapothen ist es nun interessant herauszufinden, wie man Zeit und Geld beim Versand einsparen kann. Dazu kann man zum Beispiel die Warenlager dort plazieren, wo die Nachfrage am größten ist.

1.4 Zielsetzung

Das Ziel ist es eine Anwendung zu kreieren, welche aus der Häufigkeit des Ansteuerns von bestimmten Orten, einen Ort bestimmt, von welchem aus die Gesamtdistanz, in Abhängigkeit von der Häufigkeit, zu den gegebenen Orten, möglichst klein ist.

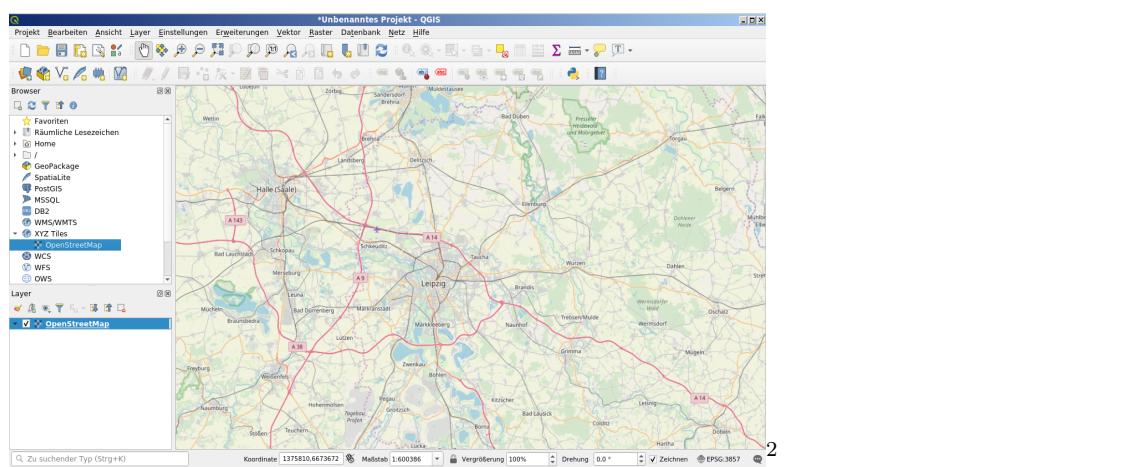
1.5 Aufgabenstellung

Zur Lösung der oben genannten Probleme soll eine Webanwendung erstellt werden, welche eine Karte anzeigt, auf der der Nutzer/-in Standorte auswählt, die er mit einer bestimmten Häufigkeit, die er/sie/es selbst angibt, aufsucht.

1.6 Existierende Lösungen

Die WIGeoGIS¹ Softwareerstellungs- und Handelsgesellschaft m.b.H. bietet verschiedene Services für Kunden aus dem Industriesektor an, unter anderem WebGIS zur Datenanalyse, die über verschiedene Schnittstellen eingebunden werden können, auf digitalen Landkarten. Auch bieten sie eine kostenpflichtige Erweiterung des OpenSource Programms QGIS an, um Kunden Daten zur Marktlage oder Bevölkerung zur Verfügung zu stellen. Das als Opensource verfügbare Programm QGIS ist für Laien recht unübersichtlich und hat einen großen Funktionsumfang. Bei der Anwendung handelt es sich auch nicht um eine Webanwendung, wodurch der Nutzer zusätzlich Software herunterladen muss.

¹<https://www.wigeogis.com> [02.04.2019 10:58]



²Screenshot von QGIS 3.10 mit geladener Openstreetmap Karte

2 Grundlagen

2.1 World Wide Web

In der heutigen Zeit gewinnt das World Wide Web (WWW) immer mehr an Bedeutung und ist fast schon nicht mehr wegzudenken. Das World Wide Web ist der Teil des Internets, in dem die Daten mithilfe des HTTP/HTTPS -Protokolls ausgetauscht werden. Dabei sendet ein Computer(Server), auf dem die entsprechenden Daten gespeichert sind, auf Anfrage(Request) durch einen anderen Computer(Client), diesem diese Daten schickt. Das WWW umfasst also fast alle öffentlich zugänglichen Webseiten.

2.2 HTML

Die Hyper Text Markup Language (HTML) ist die benutzte Sprache im WWW und beschreibt eine Textdatei oder ein Textstream der vom Server an den Client geschickt wird und zwischengespeichert wird. Diese Textdatei wird dann von einem Browser analysiert und zeigt dem Endnutzer die enthaltenen Informationen mit entsprechender Formatierung an. Reine HTML Webseiten besitzen wenig dynamischen Elemente und bieten dem Nutzer weniger Interaktionsmöglichkeiten als Seiten mit integrierten Skripten.

2.3 JavaScript

JavaScript ist eine Scriptsprache, welche direkt in eine HTML Datei über das `<script>...</script>`-tag eingebunden werden kann. Dieser Bereich wird meist von einem Unterprogramm des Browsers ausgeführt und lädt beispielsweise Bilder oder andere Daten nach. Da JavaScript eine Programmiersprache ist, können auch Berechnungen ausgeführt und die Webseite nachträglich verändert werden, ohne die Seite neu laden zu müssen.[2]

2.4 Google Maps

Google Maps¹ ist der bekannteste Online Kartendienst und bietet viele Möglichkeiten zum erstellen von dynamischen Kartendarstellungen auf der eigenen Webseite. Allerdings benötigt man zur Nutzung der Google Maps API² einen benutzerbezogenen Key, welchen man nur gegen Angabe von Kreditkarten Informationen erstellen kann.

2.5 Open Street Map

Open Street Maps³ spielt im Vergleich mit Google Maps eine untergeordnete Rolle. Dafür bietet Open Street Maps freies Kartenmaterial an und es steht eine große Menge an freien APIs. In meinem Prototypen setze ich auf Leaflet⁴, welches die Kartendaten über eine externe Webseite bezieht. Die Anwendung nutzt einen Server von OpenStreetMap. Für intensive Nutzung sollte die Nutzung eines anderen Servers in Betracht gezogen werden, da die Server Spenden finanziert sind und nicht für übermäßige Nutzung ausgelegt sind.⁵

¹<https://www.google.com/maps/> [18.06.2019 11:18]

²<https://cloud.google.com/maps-platform/> [18.06.2019 15:42]

³<https://www.openstreetmap.org/about> [02.04.2019 11:01]

⁴<https://leafletjs.com/> [18.06.2019 15:07]

⁵https://wiki.openstreetmap.org/wiki/DE:Tile_usage_policy [13.12.2019 21:10]

3 Algorithmus

Der Mittelpunkt wird mit folgender Formel aus den auf der Karte gesetzten Markern berechnet:

$$\lambda = \left(\sum_{i=1}^n \lambda_i \cdot w_i \right) \cdot \left(\sum_{i=1}^n w_i \right)^{-1} \quad (3.1)$$

Diese Formel wird jeweils für die x- und die y-Koordinate des Markers auf der Karte genutzt. Dabei wird λ jeweils mit x und y substituiert. Dafür wird zuerst das Produkt aus der Koordinate λ_i mit der Wichtung(Priorität) w_i multipliziert. Dies wird für alle vorhandenen Marker, das heißt n -mal, durchgeführt und die Summe aus den ermittelten Werten wird gebildet. Die Koordinate für den gewichteten Mittelpunkt erhält man danach durch die Division mit der Summe aus allen Wichtungen, die ebenfalls n -mal vorhanden sind, da n der Anzahl der Marker entspricht.

4 Implementierung

Für die Kartendarstellung wird Leaflet[1], eine OpenSource Bibliothek für die Darstellung von Kartenmaterial aus verschiedenen Quellen, genutzt. Eine Kartendatenquelle, die diese Anwendung benutzt, ist OpenStreetMap. Für die Webanwendung ist nur ein Webbrower notwendig, der in der Lage ist, JavaScript Anwendungen ausführen zu können. Die Webanwendung besteht aus einem HTML Dokument und einer JavaScript Datei sowie der Leaflet Bibliothek. Die Bibliothek wird aus dem Ordner *leaflet* geladen. In diesem befinden sich auch die benötigten Grafikelemente für die Marker. Das Script *leaflet1.js* befindet sich im Ordner *js* und enthält die Funktionsaufrufe, welche zur Darstellung der Karte benötigt werden, sowie die Funktion zur Berechnung des optimierten Standortes.

```
...
<link rel="stylesheet" href="leaflet/leaflet.css" crossorigin="" />
<script src="leaflet/leaflet.js"></script>
...
<div id="map"></div>
<script src="js/leafscript1.js" type = "text/javascript"></script>
...
```

Das Script *leaflet1.js* erstellt eine neue Variable *mymap*, die dann als Objekt für die Kartendaten dient. Die Karte wird dabei erstellt und hat als Mittelpunkt die Stadt Leipzig (Längengrad: 51,339° Nord, Breitengrad: 12,381° Ost). Die Ansicht ist standardmäßig auf eine Zoomstufe von 12 eingestellt, da man so die gesamte Stadt und das Umland sehen kann. Als nächstes wird dann zur Karte eine neue Ebene hinzugefügt, welche die Bildinformationen von einem OpenStreetMap Server abfragt und darstellt. Außerdem wird festgelegt wie weit man in die Karte hinein- und hinauszoomen kann.

```
const mymap = L.map('map').setView([51.33918, 12.38105], 12);
L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png?lang=de', {
```

```

    maxZoom: 20,
    minZoom: 5,
    attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors',
}).addTo(mymap);

```

Als nächstes wird eine Variable für einen Marker initialisiert, sowie eine Konstante i und ein Feld für die gesetzten Marker. Desweiteren die Kreise, welche den Mittelpunkt anzeigen, definiert.

```

let marker;
const i=0;
let markers = [];
let kreis1;
let kreis2;
let kreis3;

```

Damit der Nutzer Marker auf die Karte setzen kann wird einen neuen Funktion *on_Map_Click(e)* definiert, welche an der angeklickten Position einen Marker setzt. Jeder Marker bekommt eine ID zugewiesen, damit man ihn später wieder entfernen kann. Diesem wird standardmäßig eine Priorität mit dem Wert 5 zugewiesen. Die ID wird aus der Länge des *markers* Feld definiert. Dem Marker wird auch ein Popup zugewiesen. Dieses Popup zeigt die Marker ID, Priorität und ein Button zum entfernen des Markers. Das Popup enthält auch ein Eingabefeld zum ändern der Priorität. Beim Ändern der Priorität wird die Funktion *change_marker()* aufgerufen und über gibt die ID des Markers und die Priorität. Der Button ruft bei einem Klick die Funktion *clear_marker()* auf. Danach wird zur Karte ein neuer Layer hinzugefügt und die Marker die im Array *Markers* enthalten sind, werden dem Layer hinzugefügt. Als letztes wird noch die Funktion *center()* aufgerufen.

```

function on_Map_Click(e){
    let id;
    if (markers.length < 1) {
        id = 0;
    }else {
        id = markers[markers.length - 1]._id + 1;
    }
}

```

```

marker = new L.marker(e.latlng, {draggable:false}).addTo(mymap);
marker._id = id;
marker._prio = 5;
marker.bindPopup('<b>Marker <br>' +
+ marker._id
+ '</b><br>'
+ 'Prioritaet:'
+ '<br><input type="number" value="'
+ marker._prio
+ '" oninput="change_marker('
+ marker._id
+ ', this.value)" placeholder="Prioritaet" min="0" max="1000" />'
+ '<br>=====<br>'
+ '<input type="button" value="Entferne Marker"
    onclick="clear_marker('
+ marker._id
+ ')" />'
);
mymap.addLayer(marker);
markers.push(marker);
center();
}

```

Die nächste wichtige Funktion ist *center()*, da diese für die Berechnung des Mittelpunktes zuständig ist. Der Mittelpunkt wird somit immer bei einer Änderung eines Markers neu berechnet. Dabei wird auch geprüft, ob die drei Kreise um den Mittelpunkt bereits bestehen, und werden dann entsprechend entfernt.

```

function center(){
    if(kreis1&&kreis2&&kreis3){
        mymap.removeLayer(kreis1);
        mymap.removeLayer(kreis2);
        mymap.removeLayer(kreis3);
    }
    ...
}

```

Darauffolgend wird eine Funktion *add* definiert, welche zwei Werte addiert. Danach wird die Summe aus allen Prioritäten der Marker gebildet. Als nächstes wird die Summe der Längen- und Breitengrade addiert und jeweils durch die Summe der Prioritäten geteilt.

```
...
const add = (a,b)=>a+b;
const prioSum = markers.map(m => m._prio).reduce(add,0);
const lat = markers.map(m => m._latlong.lat *
    m._prio).reduce(add,0)/prioSum;
const lng = markers.map(m => m._latlong.lng *
    m._prio).reduce(add,0)/prioSum;
...
```

Der Mittelpunkt wird durch drei Kreise, mit jeweils anderen Farben, dargestellt. Der erste Kreis hat die Farbe grün und stellt den äußeren Kreis dar. Kreis zwei stellt den mittleren Bereich dar und hat die Farbe gelb und Kreis drei hat den kleinsten Radius und besitzt die Farbe rot. Die Kreise haben besitzen den gewichteten Mittelpunkt als Mittelpunkt.

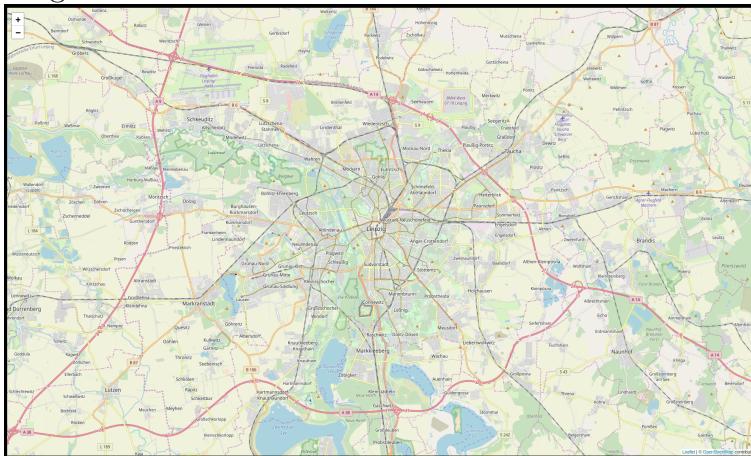
```
...
kreis1 = L.circle([lat, lng], {
    color: 'green',
    fillColor: '#00ff00',
    fillOpacity: 0.2,
    radius: 1000,
}).addTo(mymap);
kreis2 = L.circle([lat, lng], {
    color: 'yellow',
    fillColor: '#ffff00',
    fillOpacity: 0.3,
    radius: 500,
}).addTo(mymap);
kreis3 = L.circle([lat, lng], {
    color: 'red',
    fillColor: '#f03',
    fillOpacity: 0.5,
    radius: 100,
}).addTo(mymap);
```

...

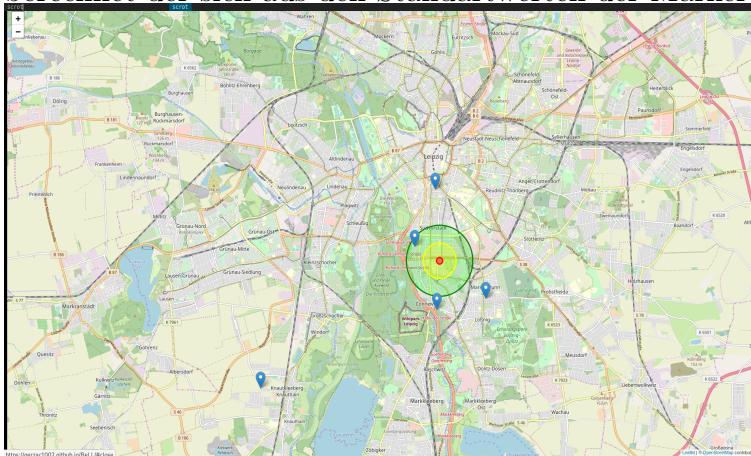
Wenn man nun Marker wieder entfernen möchte Braucht man eine neue Funktion *clear_marker*, welche die gesetzten Marker wieder entfernt.

5 Ergebnis

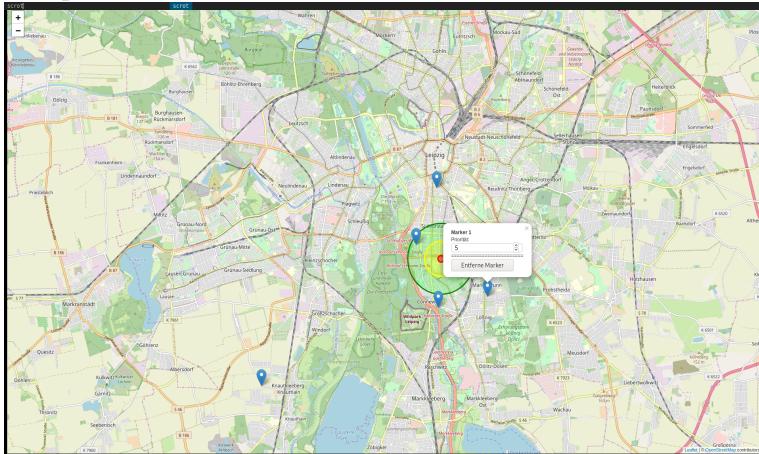
Die Webanwendung zeigt, wie im nachfolgenden Bild zusehen ist, eine Karte mit den Bilddaten von OpenStreetMap. Als Zentrum wurde die Stadt Leipzig ausgewählt.



Auf dieser Karte können nun Orte angeklickt werden, woraufhin ein Marker an der entsprechenden Position gesetzt wird. Auch wird direkt ein Mittelpunkt berechnet der sich aus den Standartwerten der Marker berechnen lässt.



Um die Priorität eines Markers zu ändern muss man diesen anklicken und dort die Priorität im Eingabefeld ändern. Auch kann man in dem Popup einen Knopf zum Entfernen des Markers drücken, falls man ihn nicht mehr braucht.



6 Diskussion

6.1 Auswertung

Die Anwendung ist vor allem für Personen/Dienstleister geeignet, die keine besonderen Ansprüche an den Standort haben. Vorallem Wohnvorlieben wie das Leben auf dem Land oder in der Nähe der Familie werden nicht beachtet, da dies sich nicht mit kurzen Fahrtwegen deckt. Auch für Pendler ist diese Anwendung uninteressant, da bei diesen ihr Arbeitsort als neuer Wohnsitz vorgeschlagen wird. Interessanter ist die Anwendung für Personen die noch bei ihren Eltern oder in einer WG wohnen, die einen Umzug planen und bereits eine feste Arbeitsstelle haben.

6.2 Ausblick

Momentan berechnet die Anwendung die Entfernung nur per Luftlinie und lässt Straßen und natürliche Hindernisse, wie zum Beispiel Gewässer und Wälder, außer Acht. Somit besteht noch viel Spielraum für die Erweiterung der Anwendung, zum Beispiel Integration von einem Wegfindealgorithmus, einer veränderten Distanzrechnung(beispielsweise über die gebrauchte Zeit, ...). Die Integration von Mietpreisen könnte auch bei der Entscheidung über den Standort helfen. Auch die Speicherung der durch den Nutzer eingegebenen Daten in Cookies oder auf dem Server würde die Nutzerfreundlichkeit erhöhen. Oder man entwickelt eine Anwendung für Smartphones, die die Standortdaten über einen gewissen Zeitraum sammelt und diese Daten dann verarbeitet.

Literaturverzeichnis

- [1] CRICKARD III, P. : *Leaflet. js Essentials*. Packt Publishing Ltd, 2014
- [2] FLANAGAN, D. : *JavaScript: Das umfassende Referenzwerk*. O'Reilly Verlag, 1997
- [3] HOFFMANN, H. ; SCHNEIDER, F. : Belastung, Leistungsgrenzen und Ermüdung bei Kraftfahrern. In: *Verhandlungen der Deutschen Gesellschaft für Unfallheilkunde Versicherungs-, Versorgungs- und Verkehrsmedizin eV*. Springer, 1966, S. 113–122
- [4] STEPPER, M. : Innenstadt und stationärer Einzelhandel – ein unzertrennliches Paar? Was ändert sich durch den Online-Handel? In: *Raumforschung und Raumordnung* 74 (2016), Apr, Nr. 2, 151–163. <http://dx.doi.org/10.1007/s13147-016-0391-x>. – DOI 10.1007/s13147-016-0391-x. – ISSN 1869-4179