

Entwicklung einer Projektdokumentationssoftware auf Basis von Geodaten

Projektarbeit Modul T2000
über die Praxissemester 3 und 4

im Studiengang
Nachrichtentechnik/Kommunikationstechnik für Verkehrssysteme

an der DHBW Ravensburg
Campus Friedrichshafen

vorgelegt von

John Nitzsche

Bearbeitungszeitraum:	01.05.2017 - 31.08.2017
Abgabetermin:	11.09.2017
Matrikelnummer/Kurs:	3976172, TEK-15
Partnerunternehmen:	DB Kommunikationstechnik GmbH Chemnitzer Str. 48 01187 Dresden
Betreuer:	Dipl.-Ing. (FH) Martin Schneider

Erklärung

gemäß § 5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel

Entwicklung einer Projektdokumentationssoftware auf Basis von Geodaten

selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt, keine anderen als die angegebenen Hilfsmittel benutzt und wörtliche sowie sinngemäße Zitate als solche gekennzeichnet habe.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Dresden, den 31.08.2017

John Nitzsche

Danksagung

Der Autor möchte an dieser Stelle allen Unterstützern dieser Arbeit danken. Dank gilt Stephan Hartmann für die programmiertechnische Ideenfindung, Michael Reifenhäuser und Constantin Müller für die Beratung für die geografische Verarbeitung von Daten in Theorie und Praxis, sowie Martin Schneider für die Herstellung zu hilfreichen Kontakten und sonstige unterstützende Maßnahmen. Besonderen Dank gilt Alice Nancy Nitzsche, für die sprachliche Expertise und Korrektur.

Abstract

Deutsche Bahn operates a complex infrastructure with various technologies. The telecommunication network is part of the whole infrastructure. Due to various factors, it is difficult to maintain an overview of any ongoing or planned projects of this telecommunication infrastructure. The aim of this thesis was to develop software which supports the relevant groups of people and improves the overall overview.

Before the development began, a catalog of requirements was created, that contains all claims for the software to be developed.

With the help of several libraries, a web platform has been created, in which running projects can be inserted with georeferenced data. Data storage and management was implemented by a database system. A web map interface was used to visualize the geodata. The software was additionally developed in a user-oriented manner, which allowed introducing a management of rights.

For the visualization and evaluation also open geodata of DB Netz AG were included. These data enabled the display of positions and details of important infrastructure elements within the map. Furthermore, an additional tool has been developed which can evaluate these geodata and display all relevant objects stored in the geodata of a railway section ordered by the course of the railway section.

Inhaltsverzeichnis

Anlagenverzeichnis	III
Abbildungsverzeichnis	IV
Tabellenverzeichnis	V
1 Einleitung	1
2 Problemstellung und Ziel der Arbeit	2
3 Anforderungskatalog nach BSI IT-Grundschutz-Katalogen	2
3.1 Funktionale Anforderungen	3
3.2 IT-Einsatzumgebung	3
3.3 Kompatibilitätsanforderungen	4
3.4 Performanceanforderungen	4
3.5 Interoperabilitätsanforderungen	4
3.6 Zuverlässigkeitsanforderungen	4
3.7 Konformität zu Standards	4
3.8 Einhaltung von internen Regelungen und gesetzlichen Vorschriften	4
3.9 Anforderungen an die Benutzerfreundlichkeit	4
3.10 Anforderungen an die Wartbarkeit	4
3.11 Anforderungen an die Dokumentation	5
3.12 Anforderungen an die Softwarequalität	5
3.13 Sicherheitsanforderungen	5
4 Voruntersuchung: Einsatz von Geodaten und interaktiven Karten bei der Deutschen Bahn	5
5 Festlegung der Software-Plattform und Bibliotheken	8
5.1 Geografische Verarbeitung	8

5.1.1 Analyse verfügbarer Software-Plattformen	8
5.1.1.1 Installierbare Desktop-Anwendungen	8
5.1.1.2 Webbasierte Anwendungen	9
5.1.2 Auswahl der Software-Plattform	10
5.1.3 Verfügbare webbasierte anpassbare GIS-Anwendungen	12
5.1.3.1 OpenStreetMap	12
5.1.3.2 umap	13
5.1.3.3 Google Maps API	15
5.1.3.4 Leaflet und OpenLayers	16
5.2 Datenspeicherung, -verarbeitung und Benutzeroberfläche	21
6 Programmierung der Software	23
6.1 Details zur Programmierung der geografischen Anwendungen	24
6.1.1 Grundaufbau der Karte	25
6.1.2 Einfügen der Geodaten der DB Netz AG	29
6.1.3 Einbindung der Projektdaten	35
6.2 Details zur Benutzeroberfläche	36
6.3 Sicherheitstechnische Betrachtung der Software	42
7 Installation und Inbetriebnahme der Software	45
7.1 Anforderungen an die Informationstechnik	45
7.2 Installation der Software auf dem Server	45
8 Fazit	47
8.1 Erfüllung der vorgegebenen Anforderungen	47
8.2 Ausblick	49
9 Literaturverzeichnis	50

Anlagenverzeichnis

Anlage 1: Beispielprojekte mit von der DB AG Datensätzen und Schnittstellen	I
Anlage 2: Dresden Hauptbahnhof Gleis 19 in OSM exportiert als XML (Stand 25. August 2017)	VII
Anlage 3: Dresden Hauptbahnhof Gleis 19 in OSM konvertiert zu GeoJSON (Stand: 25. August 2017)	VIII
Anlage 4: Verweise auf den Quellcode und Demonstrationsexemplar dieser Arbeit	IX

Abbildungsverzeichnis

Abbildung 1: erstellte umap-Karte der RDZ-Linie 11B im Abschnitt Cottbus-Guben (verfügbar auf https://umap.openstreetmap.fr/de/map/rdz-gorlitz-guben_149763).....	14
Abbildung 2: Architektur der benötigten bzw. verwendeten Hardwareinfrastruktur.....	22
Abbildung 3: Layerkontrollsystem mit Standardmitteln aus Leaflet.....	27
Abbildung 4: Mit dem Plugin leaflet-groupedlayercontrol erstelltes Layerkontrollsystem.....	28
Abbildung 5: Vergleich zwischen dem standardmäßigen Verarbeiten (links) und dem geclusterten Verarbeiten der Betriebsstellen-GeoJSON-Datei	32
Abbildung 6: Vier übereinanderliegende Betriebsstellen in Dresden-Neustadt, angezeigt als Gruppe.....	33
Abbildung 7: Die vier übereinanderliegenden Betriebsstellen in Dresden-Neustadt, angezeigt als einzelne Marker	34
Abbildung 8: Fertiggestellter Kartenausschnitt Dresden Hbf mit geclusterten Geodaten der DB Netz AG (Datensatz: Betriebsstellen)	35
Abbildung 9: Verzeichnisbaum der Software, Stand: 30. Juni 2017	37
Abbildung 10: Ein Beispielprojekt mit Karte angezeigt in der projectdetail.php (links: PC-Monitor, rechts: Mobiltelefon)	40

Tabellenverzeichnis

Tabelle 1: Vor- und Nachteile von Desktop- und webbasierten Anwendungen im Sinne der Anforderungen.....	11
Tabelle 2: Vergleich JavaScript-Code von OpenLayers und Leaflet.....	20
Tabelle 3: Verwendete Tilelayer in der Karte.....	26
Tabelle 4: Beschreibung der gelieferten Geodaten (DB Netz AG) vom 5. Mai 2017	30
Tabelle 5: Konfigurationsmöglichkeiten der globalconfig.php	46
Tabelle 6: Erfüllen der in Punkt 3 gestellten Anforderungen	48

Abkürzungsverzeichnis

API.....	application programming interface (dt.: Programmierschnittstelle)
BSI.....	Bundesamt für Sicherheit in der Informationstechnik
CAD	computer-aided design (dt.: computerunterstütztes Entwerfen)
DB AG	Deutsche Bahn
GIS	Geoinformationssystem-Software
Hbf.....	Hauptbahnhof
HTML.....	Hypertext Markup Language
ORM	OpenRailwayMap
OSM	OpenStreetMap
PHP	Hypertext Preprocessor
POI	Point of Interest
s/w	schwarz/weiß
URL	Uniform Resource Locator

1 Einleitung

Die Digitalisierung ist ein wichtiger Bestandteil des heutigen Zeitalters. Durch die fortschreitende Entwicklung neuer Technologien steigt die Priorisierung der Digitalisierung weiter an. Im Alltag, in Wirtschaft und Industrie, sowie in der Forschung fallen deswegen mehr Daten an, die es zu übertragen gilt. Das gilt auch für die Infrastruktur der Deutschen Bahn AG (DB AG). Video-Anlagen, Zugfunk, technische Überwachungssysteme der Leit- und Sicherungstechnik und weitere Netzwerke erzeugen Daten, die übertragen werden müssen. Die Deutsche Bahn betreibt dafür eine großflächige Infrastruktur für die Telekommunikation. Diese Infrastruktur enthält verschiedene technische Systeme, die Daten werden über verschiedene Medien (Funk, Kupferkabel, Lichtwellenleiter) übertragen. Die große Anzahl an verschiedenen Systemtypen macht die Infrastruktur komplex. Da diese komplexe Telekommunikationsinfrastruktur fortwährend umgebaut, erweitert und modernisiert wird, ist es schwierig einen Gesamtüberblick über alle laufenden Projekte an dieser Infrastruktur zu behalten. Verstärkt wird dies dadurch, dass verschiedene Firmen, sowohl konzernintern als auch extern, an der Infrastruktur der Deutschen Bahn arbeiten und planen.

Um einen besseren Überblick über verschiedenste Systeme - auch abseits des technischen Bereiches - zu erhalten, werden im Allgemeinen oft Karten benutzt. Im Alltag werden Karten bei der Navigation und Orientierung verwendet. Meteorologen nutzen Karten zur Wetterauswertung und -vorhersage. Gesellschaftswissenschaftler tragen zudem auch ihre Ergebnisse in Karten ein und können beispielsweise so Veränderungen bei der sozialen Entwicklung eines Landes erforschen. Auch in der Wirtschaft helfen Karten, den Überblick zu behalten. Viele Stadtwerke verwalten Ihre Stationen für Wasser, Fernwärme und Datenanbindung über Karten, da die Disposition im Störungs- oder Wartungsfall viel effizienter durchgeführt werden kann. Eine Software, die auf einer Oberfläche laufende Projekte darstellt, scheint deswegen ein guter Weg zu sein, die Projektdokumentation der Telekommunikationsinfrastruktur der Deutschen Bahn zu vereinfachen.

2 Problemstellung und Ziel der Arbeit

Die Deutsche Bahn AG betreibt eine große Telekommunikationsinfrastruktur, an welcher fortwährend geplant, gebaut und gearbeitet wird. Die Vielzahl der verschiedenen technischen Telekommunikationssysteme, sowie die großflächige Verbreitung dieser, erschweren das Management der parallel laufenden und die Berücksichtigung vergangener Infrastrukturprojekte. Auf Grund des nicht bzw. nur eingeschränkt vorhandenen Gesamtüberblickes der Verantwortlichen und des fehlenden Wissens über parallel laufende Projekte besteht die Gefahr von redundanten Planungen und Arbeiten.

Ziel der Arbeit ist, eine auf Geodaten basierte Software zu entwickeln, welche die eben genannten Probleme minimiert. Auf einer Karte sollen alle laufenden Projekte der Telekommunikationsarchitektur übersichtlich dargestellt werden. Zudem sollen in dem Tool Ansprechpartner, Laufzeit und der aktuelle Status der einzelnen Projekte sichtbar werden. Dadurch soll der Gesamtüberblick über laufende Projekte vereinfacht und die Kommunikation zwischen den Verantwortlichen gefördert werden. Ebenso ist eine Historienübersicht für laufende Projekte und Planungen hilfreich.

3 Anforderungskatalog nach BSI IT-Grundschutz-Katalogen

Vor der Planung und Projektierung der zu erstellenden Software wurde ein Anforderungskatalog angefertigt. Dieser Anforderungskatalog wurde in Anlehnung an die IT-Grundschutz-Kataloge des Bundesamtes für Sicherheit in der Informationstechnik (BSI)¹ erstellt und enthält alle Zielsetzungen, die an das Software-Projekt gestellt werden. Da dieser Arbeit keine Machbarkeitsstudie vorangegangen ist, ist zunächst nicht klar, ob bestimmte Anforderungen an die Software im Zeitraum des Projektes erfüllbar sind, bzw. ob diese technisch realisierbar sind. Deswegen sind zum Projektstart einige Anforderungen vorweg als optionale Anforderungen markiert worden. Eine Erfüllung dieser optionalen Anforderungen an die Software ist wünschenswert, jedoch nicht zielsetzend in dieser Arbeit.

¹ (Bundesamt für Sicherheit in der Informationstechnik, 2016)

3.1 Funktionale Anforderungen

Auf einer digitalen Karte sollen alle laufenden Projekte zur Telekommunikation der DB Kommunikationstechnik zu sehen sein.

In einer Datenbank sollen Kerninformationen über alle laufenden Projekte hinterlegt sein.

Durch Popups oder andere Lösungen sollen Details zu den Projekten in der Karte sichtbar gemacht werden.

Es soll ein Verwaltungssystem existieren, in dem Projekte und seine Details verwaltet (neu erstellt, bearbeitet, gelöscht) werden können.

Die Verwaltung soll benutzerorientiert gestaltet sein. Es soll eine Benutzerhierarchie (Administratoren, Mitwirkende, Betrachter) existieren. Administratoren sollen Vollzugriff auf alle Projekte und zusätzlich eine Benutzerverwaltung besitzen. Mitwirkende sollen nur eigene Projekte bearbeiten dürfen, jedoch volle Leserechte. Betrachter sollen nur Leserechte besitzen.

optional: Es soll zusätzlich zu den Projekten eine Historie hinterlegt und abrufbar sein.

optional: Es sollen offizielle Geodaten der DB Netz AG bei der Karte eingesetzt werden.

3.2 IT-Einsatzumgebung

Die Software soll Mitarbeiter der Deutschen Bahn AG unterstützen. Die Deutsche Bahn benutzt das sogenannte BKU-System¹, ein zentral verwaltetes Windows-Betriebssystem, was zum Stand der Erstellung dieser Arbeit auf Windows 7 Enterprise (64-Bit) basiert. Die Hardware entspricht dem markttypischen Personal-Computer- und Notebooksortiment mit x86-Prozessor. Auf dem BKU-System ist Standardsoftware wie Microsoft Office, Google Chrome und andere vorinstalliert.

Die bei der Deutschen Bahn benutzten Firmenmobiletelefone benutzen größtenteils die Betriebssysteme Android von Google oder iOS von Apple. Es sind auch andere Mobil-Plattformen verfügbar, welche nicht durch Mobilapplikationen erweitert werden können. Das Betriebssystem Blackberry OS wird nicht mehr bei neu angeschafften Geräten ausgeliefert.

¹ BKU: Bürokommunikation Unternehmensweit

Bestimmte Mitarbeitergruppen werden durch Tablet-Computer unterstützt. Diese haben die Displaydiagonalen von sieben bis zwölf Zoll und werden entweder mit dem Betriebssystem Android von Google, iOS von Apple oder Windows von Microsoft ausgeliefert.

3.3 Kompatibilitätsanforderungen

optional: Bei Verwendung offizieller Geodaten der DB Netz AG muss auf eine Kompatibilität mit den von der DB Netz AG verwendeten Dateiformaten geachtet werden.

3.4 Performanceanforderungen

Die Ausgabe von Ergebnissen soll bei der Benutzung eine Zeitverzögerung von fünf Sekunden in keiner Situation überschreiten.

3.5 Interoperabilitätsanforderungen

Die Software soll plattformunabhängig gestaltet sein, oder es sollen verschiedene Versionen für die bei der Deutschen Bahn eingesetzten Plattformen (vgl. 3.2) existieren.

3.6 Zuverlässigkeitsanforderungen

Durch den Benutzer durchgeführte ungültige Eingaben sollen abgefangen werden, die Software soll zu einer erneuten Eingabe auffordern, die Stabilität des Programms soll bei ungültigen Eingaben gewährleistet bleiben.

3.7 Konformität zu Standards

entfällt

3.8 Einhaltung von internen Regelungen und gesetzlichen Vorschriften

entfällt

3.9 Anforderungen an die Benutzerfreundlichkeit

Die Benutzeroberfläche soll einfach gehalten sein. Überflüssige Optionen und Informationen (Datenbank-IDs, Debug-Informationen) sollen für den normalen Benutzer nicht oder nur in einer erweiterten Ansicht sichtbar sein.

3.10 Anforderungen an die Wartbarkeit

Es werden keine Anforderungen an die Wartbarkeit gestellt.

3.11 Anforderungen an die Dokumentation

Es soll eine Dokumentation existieren, die die Funktionen und Bedienung der Software beschreibt.

Die Dokumentation muss selbsterklärend sein, sie soll ohne weitere Dokumente für den Benutzer verständlich und umsetzbar sein.

Als Dokumentation zur Software gilt diese Arbeit.

3.12 Anforderungen an die Softwarequalität

Es werden keine Anforderungen an die Softwarequalität gestellt.

3.13 Sicherheitsanforderungen

Die Software verarbeitet interne Daten der Deutschen Bahn AG, welche vor unberechtigten Aufrufen und Bearbeitungen geschützt werden müssen. Die Software soll, auch lesend, nur berechtigte Benutzer in das System lassen. Es soll eine Autorisierung mit Benutzername und Passwort existieren.

Benutzerpasswörter sollen verschlüsselt (gehashed) in einer Datei oder Datenbank hinterlegt werden.

optional: Es soll automatisiert eine Dokumentation (Log-File) über Bearbeitung von Daten erstellt werden.

4 Voruntersuchung: Einsatz von Geodaten und interaktiven Karten bei der Deutschen Bahn

Die Verarbeitung von Geoinformationen ist bei der Deutschen Bahn keine Neuheit. Durch den konzernweiten Kurs, die fortschreitende Digitalisierung zur Verbesserung der internen Prozesse zu nutzen, öffnete die Deutsche Bahn sich auch für die Themen Startup-Unternehmen und Open-Data. Open-Data beschreibt die Veröffentlichung von einst internen digitalen Daten, die bei der Deutschen Bahn als statische Datensätze oder dynamische Echtzeit-Schnittstellen (APIs¹) zur Verfügung gestellt werden. Dafür gibt es ein eigenes Online-Portal². Die offenen Daten sollen interne als auch externe

¹ API: Application Programming Interface

² <http://data.deutschebahn.com>

Entwickler inspirieren und unterstützen. Projekte auf Basis der offenen Daten werden diskutiert und ausgetauscht. Da die Daten meist unter einer Creative-Commons-Lizenz, einer Lizenz die das Verarbeiten und Teilen der Daten unter Namensnennung erlaubt, veröffentlicht werden, gibt es eine Vielzahl von Softwareprojekten in der Gemeinschaft der Entwickler.

Unter den Datensätzen im Open-Data-Portal der DB AG befinden sich auch statische Geodaten des Schienennetzes der DB Netz AG, die in verschiedenen Formaten und Ständen veröffentlicht sind. Neben dem Schienennetz sind auch Bahnübergänge, Betriebsstellen, Brücken, Tunnel und weitere Bauwerke und Punkte erfasst worden. Die Daten liegen unter anderem als Excel-Format, GeoJSON und XML vor, auch das System MapInfo des gleichnamigen US-amerikanischen Unternehmens wird benutzt, welches eine professionelle Geoinformationssystem-Software (GIS) liefert.

Im sogenannten ShowCase (deutsch: Schaukasten) werden Projekte der Gemeinschaft vorgestellt, die unter anderem auch aus Entwicklern der DB Systel, der Dienstleisterin für Informations- und Telekommunikationstechnik der Deutschen Bahn, besteht.¹ Beispielprojekte, die im Sinne dieser Arbeit interessant sind, sind in Anlage 1, Beispiele 1 bis 3 vorgestellt. Auffällig an diesen Beispielen ist die Gemeinsamkeit, dass die Benutzeroberflächen der Karten allesamt mit Hintergrundbildern des OpenStreetMap-Projektes (OSM) ausgestattet wurden. OSM ist eine offene Weltkarte, an der jeder Mensch Bearbeitungen durchführen kann. Das fördert die Aktualität und Genauigkeit der Karte. Weiterhin fällt auf, dass jegliche GIS-Anwendungen mit der quellcodeoffenen JavaScript-Bibliothek Leaflet vollzogen worden.

Neben den durch die Community angefertigten Projekten aus den offenen Daten, werden anderswo auch digitale Kartensysteme im Konzern der DB AG verwendet. Schon im August 2013 startete die Deutsche Bahn mit dem „Zugradar“ eine Betaversion eines Projektes, welche Positionen von Reisezügen der Deutschen Bahn mit Hilfe des Fahrplans berechnet und auf einer Karte anzeigt. Es werden betriebliche Zuglaufmeldungen zur Fehlerkorrektur verwendet (Ihlenfeld, 2013). Eine genaue Ortung der Züge, beispielsweise durch satellitengestützte Ortungsverfahren, wird bis heute nicht angewandt. Darunter leidet die Qualität der Positionsdaten, da Abweichungen vom

¹ Vgl. Github-Präsenz von DB Systel <https://github.com/dbsystel>

Fahrplan in aller Regel nur durch die nächste Zuglaufmeldung erkannt und verarbeitet werden können. Einen Einblick gibt der Screenshot in Anlage 1, Beispiel 4 dieser Arbeit. Anders als die im vorigen Abschnitt vorgestellten Projekte, wird bei dem DB Zugradar die proprietäre Google Maps API verwendet, um die benötigten GIS-Anwendungen anzeigen zu können. Auch ist der Hintergrund die Straßenkarte von Google Maps und nicht OSM.

Ein weiteres jüngeres Projekt stellt „RI Maps“ des Teams „Projekt Reisendeninformation“ (Anlage 1, Beispiel 5) dar. Auf einer Karte werden Übersichtspläne von bestimmten größeren Bahnhöfen gezeigt. Es werden Gleisabschnitte eingetragen, sowie Geschäfte und andere POIs (Points of Interest), wie Geldautomaten und Taxisstellen. Eine Besonderheit ist, dass man zwischen verschiedenen Levels wechseln kann. Die Levels spiegeln verschiedene Höhenniveaus, ähnlich von Stockwerken in Gebäuden, wieder. Als Hintergrund wird auch hier OSM benutzt, statt Leaflet benutzt das Team „Projekt Reisendeninformation“ jedoch die JavaScript-Bibliothek OpenLayers. Das Projekt „RI Maps“ ist derzeit (Stand: Mai 2017) keine öffentlich zugängliche Plattform und steht nur Mitarbeitern der Deutschen Bahn zur Verfügung.

Die externe Firma Geo++ betreibt im Auftrag der Deutschen Bahn eine Plattform, mit der beliebige Stellen im Streckennetz von DB Netz AG relativ zur Streckennummer und -kilometrierung angezeigt werden können. Andersherum kann durch die Eingabe von Streckennummer- und kilometer die jeweilige Stelle in einer Karte angezeigt werden. Das Produkt wird GNRaiLNav genannt und wird als Android-App und Browseranwendung (Anlage 1, Beispiel 6) zur Verfügung gestellt. Die Browseranwendung bedient sich ebenfalls der OpenLayers-Bibliothek. Als Hintergrundlayer kann man unter den von Google bereitgestellten Satelliten-, Karten- und Hybridansichten wählen. Im Quelltext wird auch die Google Maps API eingebunden. Die erforderlichen Daten entstammen aus dem Datenpool von DB Netz.

Nicht öffentlich einsehbar ist das von der DB Netz AG, Abteilung Fahrweg betriebene Tool GeoViewer. In diesem werden verschiedene Infrastrukturelemente und bahnrelevante Bereiche und Informationen zusammengetragen. Für die Benutzung sind

nur freigeschaltete Mitarbeiter der Deutschen Bahn AG zugelassen. Es werden Lizenzgebühren verlangt.

5 Festlegung der Software-Plattform und Bibliotheken

5.1 Geografische Verarbeitung

Ziel der Arbeit ist nicht eine grundlegend neue Software zu entwickeln, die ohne das Zutun von externem Quellcode die in Kapitel 3 erläuterten Anforderungen erfüllt. Es soll durch existierende Plattformen, Laufzeitumgebungen und Bibliotheken eine Software geschaffen werden, die den Anforderungen entspricht. Es gibt verschiedene Ansätze eine GIS-Software zu nutzen, zu entwickeln oder abzuändern, damit sie die Anforderungen dieser Arbeit erfüllt.

5.1.1 Analyse verfügbarer Software-Plattformen

5.1.1.1 Installierbare Desktop-Anwendungen

Es gibt verschiedene Softwaresysteme, die GIS-Anwendungen beinhalten oder nur für diese veröffentlicht wurden. Bei der Deutschen Bahn wird beispielsweise die proprietäre Software MapInfo verwendet. Das Dokumentationstool für Telekommunikationsnetze NeDocS, welches bei der Deutschen Bahn eingesetzt wird, erlaubt die Verknüpfung zwischen den in NeDocS hinterlegten Daten und MapInfo. In NeDocs werden für verschiedene Netzelemente und Standorte georeferenzierte Koordinaten hinterlegt, die in MapInfo verarbeitet werden können. MapInfo bringt jedoch den Nachteil mit, dass es durch die proprietäre Verbreitung keine offene Software ist, also nicht beliebig verändert werden kann. Zudem ist der Preis für diese Software vergleichsweise hoch.

Bestimmte andere GIS-Software-Plattformen für Windows heben diesen Nachteil durch ihre Quelloffenheit auf. Die Software QGIS veröffentlicht ihren Quellcode und die kompilierten ausführbaren Dateien unter der Lizenz GPL 2.0. Diese Lizenz erlaubt die freie Nutzung der Software auch für kommerzielle Zwecke und die Modifizierung der Software. QGIS ist durch eine Vielzahl von Plug-Ins auch erweiterbar und bringt auch eine Applikation für Android-Systeme mit. Nichtsdestotrotz bleibt die in Punkt 3.1 geforderte Benutzerorientierung und Rechteverwaltung schwer umsetzbar.

Windows-Software bietet generell eine bessere Performance als Webapplikationen (nächster Abschnitt). Die Erweiterung (auch von quelloffener und durch Plug-Ins

erweiterbarer) Windows-Software benötigt tiefe Programmierkenntnisse. Zudem leidet die Plattformunabhängigkeit (vgl. 3.5) durch die obligatorische Desktop-Plattform, auch wenn beispielsweise QGIS-Versionen für Android, Windows, Linux und andere Betriebssysteme veröffentlicht sind. Denn durch verschiedene Plattformen ändern sich sowohl die Handhabung als auch die Erweiterungsmöglichkeiten durch Plug-Ins und Quellcodeerweiterungen. Der Programmieraufwand ist durch die verschiedenen Plattformen höher. Hinzu kommt bei der Deutschen Bahn das Problem, dass eine PC-Software durch ein Zentrales Softwaremanagement-System verteilt werden muss, der normale BKU-Nutzer hat keine Möglichkeit Windows-Software auf seinem PC zu installieren. Das erhöht den Verwaltungsaufwand und verringert die Benutzerfreundlichkeit. Es besteht die Möglichkeit eine Software portabel zu installieren, bei dem bereits genannten Dokumentationstool NeDocS ist dies der normale Installationsweg. Softwareaktualisierungen müssen dann händisch durch den Benutzer bei jeder neuen Version durchgeführt werden.

Ein weiteres Problem stellt sich in der notwendigen Zentralisierung der Daten. Jede Instanz der Software muss seine Daten aus einem gemeinsamen Datenpool wie eine zentral abgelegte Datenbank greifen. Diese Datenbankplattform muss weiterhin programmiert und eingebunden werden. Eine Datenbankverbindung verschlechtert jedoch die Performance im Vergleich zur Verarbeitung lokaler Dateien durch die Übertragungswege und das gleichzeitige Abrufen und Bearbeiten von Daten von verschiedenen Instanzen.

5.1.1.2 Webbasierte Anwendungen

Eine weitere Möglichkeit besteht in der Entwicklung einer browserbasierten Anwendung. Wird diese zentral auf einem Server abgelegt, spricht man von einer Webanwendung. Webanwendungen haben den Vorteil, dass nur die Installation an einer zentralen Instanz, einem Webserver, erfolgen muss. Die Benutzer der Software greifen dann über einen Browser auf die Software zu, der bei allen aktuellen Betriebssystemen vorinstalliert ist. Webanwendungen können durch sich anpassendes Design (in der Webprogrammierung „responsive design“ genannt) auch für mobile Plattformen brauchbar gemacht werden, ohne dass die gesonderte Programmierung einer mobilen Version notwendig ist. Sich anpassende Webanwendungen sind deshalb im Sinne der Anforderungen nach Punkt 3.5 plattformunabhängig.

Die Anforderungen an die benötigten GIS-Werkzeuge beschränken sich für den normalen Benutzer in der geforderten Software lediglich auf das Setzen von Markern und das Auslesen von Informationen. Durch die wenigen Optionen kann eine webbasierte Lösung übersichtlicher gestaltet werden als die Anpassung einer menügeführten GIS-Software für Desktop-Systeme. Die Benutzerfreundlichkeit ist dadurch hoch. Jedoch müssen statt der Menüführung andere Wege gefunden werden, Objekte zu bearbeiten und Details zu Objekten anzuzeigen.

Für GIS-Systeme können hier verschiedene Plattformen und Bibliotheken im Web-Bereich verwendet werden. Darunter zählt die weit verbreitete Google Maps-Plattform, die durch eigene Schnittstellen auch fremde Daten und Funktionen einbinden kann. Weiterhin gibt es mit OpenLayers und Leaflet quelloffene Bibliotheken, die Werkzeuge für GIS-Anwendungen mitbringen.

Webanwendungen sind weniger performant als Desktop-Anwendungen, da ihnen clientseitig weniger Ressourcen des Prozessors zugeteilt wird und Code ineffizienter verarbeitet werden kann. Gerade bei der Verarbeitung von vielen Geodaten zur selben Zeit haben Webanwendungen einen entscheidenden Nachteil. Bei einer Verwendung einer serverorientierten Lösung (Mapserver-Software wie MapServer oder Geoserver) entsteht insbesondere bei der Verarbeitung vieler Geodaten durch viele Benutzer ein hoher Ressourcenverbrauch. JavaScript-Bibliotheken, wie OpenLayers und Leaflet, führen Funktionen lokal auf dem Endgerät des Benutzers aus. Dadurch sinkt die Auslastung des Webserver, da die Rechenprozesse auf das jeweilige Endgerät umgelagert werden.

Die Speicherung der Daten muss auch bei webbasierten Anwendungen zentralisiert erfolgen. Da die Software jedoch selbst an einem zentralen Ort abgerufen wird, ist bei webbasierten Anwendungen in der Regel bereits die Infrastruktur für eine entsprechende Datenbank vorhanden und muss nicht zusätzlich geschaffen werden.

5.1.2 Auswahl der Software-Plattform

Sowohl Desktop-Software als auch webbasierte Anwendungen bringen Vor- und Nachteile mit. Zudem gibt es verschiedene Varianten und Plattformen, welche benutzt werden können. Es soll festgelegt werden, ob eine Desktop-Software oder eine webbasierte Anwendung die Anforderungen am besten erfüllbar machen. Dafür soll

zunächst kein bestimmtes Produkt oder keine bestimmte Bibliothek ausgeschlossen oder als Lösung markiert werden. Die Vor- und Nachteile der Plattformen, wie sie bereits in Punkt 5.1.1 dargelegt wurden, sind richtig abzuwägen.

	Vorteile	Nachteile
Desktop-Anwendungen	<ul style="list-style-type: none"> hohe Performance 	<ul style="list-style-type: none"> keine Plattformunabhängigkeit hoher Programmieraufwand hoher Verwaltungsaufwand innerhalb des DB-Konzerns zusätzliche Infrastruktur für die zentrale Speicherung von Daten notwendig
webbasierte Anwendungen	<ul style="list-style-type: none"> keine Installation von Software beim Benutzer notwendig übersichtlichere Benutzeroberfläche möglich plattformunabhängig einfachere Umsetzung 	<ul style="list-style-type: none"> unter Umständen hoher Ressourcenverbrauch schlechte Performance zusätzliche Lösung zur Anzeige von Details notwendig

Tabelle 1: Vor- und Nachteile von Desktop- und webbasierten Anwendungen im Sinne der Anforderungen

In Tabelle 1 wurden die Vor- und Nachteile nochmals übersichtlich dargelegt. Es muss betont werden, dass die Vor- und Nachteile der Plattformen im Sinne dieser Arbeit und nicht allgemein für GIS-Anwendungen gelten. Beispielsweise ist der Vorteil der Desktop-Anwendungen, dass sie ein eingehenderes Portfolio an Werkzeugen, Optionen und Ausgabe- und Eingabemöglichkeiten haben, nicht relevant für die Erstellung und Handhabung der geforderten Software in dieser Arbeit.

Nach Abwägung der in Tabelle 1 gezeigten Informationen, entschied der Autor, sich auf Lösungen für webbasierte Anwendungen zu beschränken. Da es wiederum verschiedene Produkte für webbasierte GIS-Anwendungen gibt, gilt es nun eine geeignete Lösung für das Projekt zu finden.

5.1.3 Verfügbare webbasierte anpassbare GIS-Anwendungen

Neben den folgenden Softwareprojekten gibt es weitere Plattformen und Programmierschnittstellen, welche die Anforderungen dieser Arbeit erfüllen können. Es folgt jedoch nur eine Auswahl der bekanntesten Anwendungen.

5.1.3.1 OpenStreetMap

Prinzipiell gibt es zwei verschiedene Wege eine webbasierte GIS-Anwendung zu schaffen. Der erste Weg wäre eine Infrastrukturlandschaft zu schaffen, die jegliche Geodaten beinhaltet, diese rendert, also zu sichtbaren Kacheln zusammenfügt, und Schnittstellen zu Änderungen bereitstellt. OpenStreetMap (OSM) hat eine solche Landschaft und ist ein gutes Beispiel, welches die Komplexität eines solchen Unterfangens darstellt. OSM hat eine breite Serverinfrastruktur, die benötigt wird, um aus den rohen Geodaten eine anschauliche Webkarte zu generieren.

Die rohen Geodaten in OSM liegen in einer Datenbank vor. Genutzt wird die PostgreSQL-Plattform. In der Datenbank liegen alle Objekte als Punkte, Linien oder Relationen¹ vor. Durch das Einfügen von Attributen können diese Objekte genauer beschrieben werden. Als Beispiel soll „Anlage 2: Dresden Hauptbahnhof Gleis 19 in OSM exportiert als XML (Stand 25. August 2017)“ dienen. Zu sehen sind der Typ des Objektes (way: engl. für Linie), und Eigenschaften dazu, beispielsweise die vorhandene Elektrifizierung mit 15000 Volt und 16,7 Hertz, die Streckennummer 6240 oder die maximale Geschwindigkeit von 80 km/h. Die mit den `<nd ref>`-Tag markierten Nummern sind Relations-IDs. Hinter diesen verbergen sich Relationen, wie z.B. alle geografischen Linien, die zur DB-Strecke 6240 gehören, oder alle geografischen Linien, die von der S-Bahn-Linie S2 befahren werden.

Die Rohdaten können in verschiedenen grafischen Oberflächen angezeigt und bearbeitet werden, sie eignen sich bisher aber noch nicht dazu, grafisch ansprechend angezeigt zu werden. Aus den Rohdaten werden, bevor man OSM standardmäßig besucht, sog. Kacheln im bekannten PNG-Format gerendert. Diese Kacheln werden Tiles genannt. Je nach Eigenschaften der Elemente und Konfiguration der Tile-Server sehen die Tiles anders aus. Es gibt verschiedene Projekte, die sich verschiedenen Tile-

¹ Eine Relation bezeichnet eine Gruppe von Linien, Punkten und anderen Relationen und wird bspw. auch benutzt, um Flächen abzubilden.

Varianten widmen: Es gibt beispielsweise Wander-Karten, topografische Karten und Eisenbahnkarten.¹

Die verschiedenen Kacheln werden nach dem Rendern abgelegt und beim Aufruf auf dem Browserbildschirm ausgegeben. Mehrere Kacheln ergeben dann insgesamt die Ansicht die man im Browser sieht.

Eine Infrastruktur wie OpenStreetMap aufzubauen ist komplex. Allein für die Webseite wird eine Linux-Distribution verlangt, mit PostgreSQL, ImageMagick, Ruby und anderen Zusatzprogrammen.² Daneben birgt das Management der Geodaten einen großen Administrationsaufwand, da eigene Geodaten verwendet werden. Es ist möglich und auch erlaubt die Datenbank von OSM zu kopieren, jedoch würde es beim Einbinden eigener Bearbeitungen der Geodaten nicht mehr möglich sein, aktuellere Geodaten zu übernehmen ohne die eigenen Bearbeitungen zu verlieren.

Da in dieser Arbeit keine neue Weltkarte geschaffen oder adaptiert werden soll, ist es nicht sinnvoll die Soft- und Hardware-Infrastruktur von OSM zu übernehmen.

5.1.3.2 umap

umap ist ein Projekt der OpenStreetMap-Gemeinschaft in Frankreich, welches auf das eigentliche OpenStreetMap aufbaut. Die benutzerorientierte Oberfläche³ bietet jedem Anwender die Möglichkeit, eigene Karten zu erstellen. Es wird das Layer-Prinzip verwendet, wie es aus vielen Bildbearbeitungsprogrammen, aber auch CAD-Programmen bekannt ist. Es wird prinzipiell eine Hintergrundkarte gewählt, der Tilelayer, in dem prinzipielle Informationen dargestellt werden. Das kann die Standardkarte von OSM sein, andere Tile-Server⁴ oder lizenzfreie Luftbilder. Darüber können Layer gelegt werden, die aus Markern für Punkte oder Linien bestehen können. Diese sind frei anpassbar. Die Karte kann am Ende geteilt werden und je nach Einstellung können bestimmte oder alle Nutzer die Karte weiter bearbeiten.

¹ (OpenStreetMap Wiki contributors, 2017)

² (OpenStreetMap Software contributors, 2017)

³ Eine Registrierung auf umap ist nicht zwingend erforderlich.

⁴ (OpenStreetMap Wiki contributors, 2017)

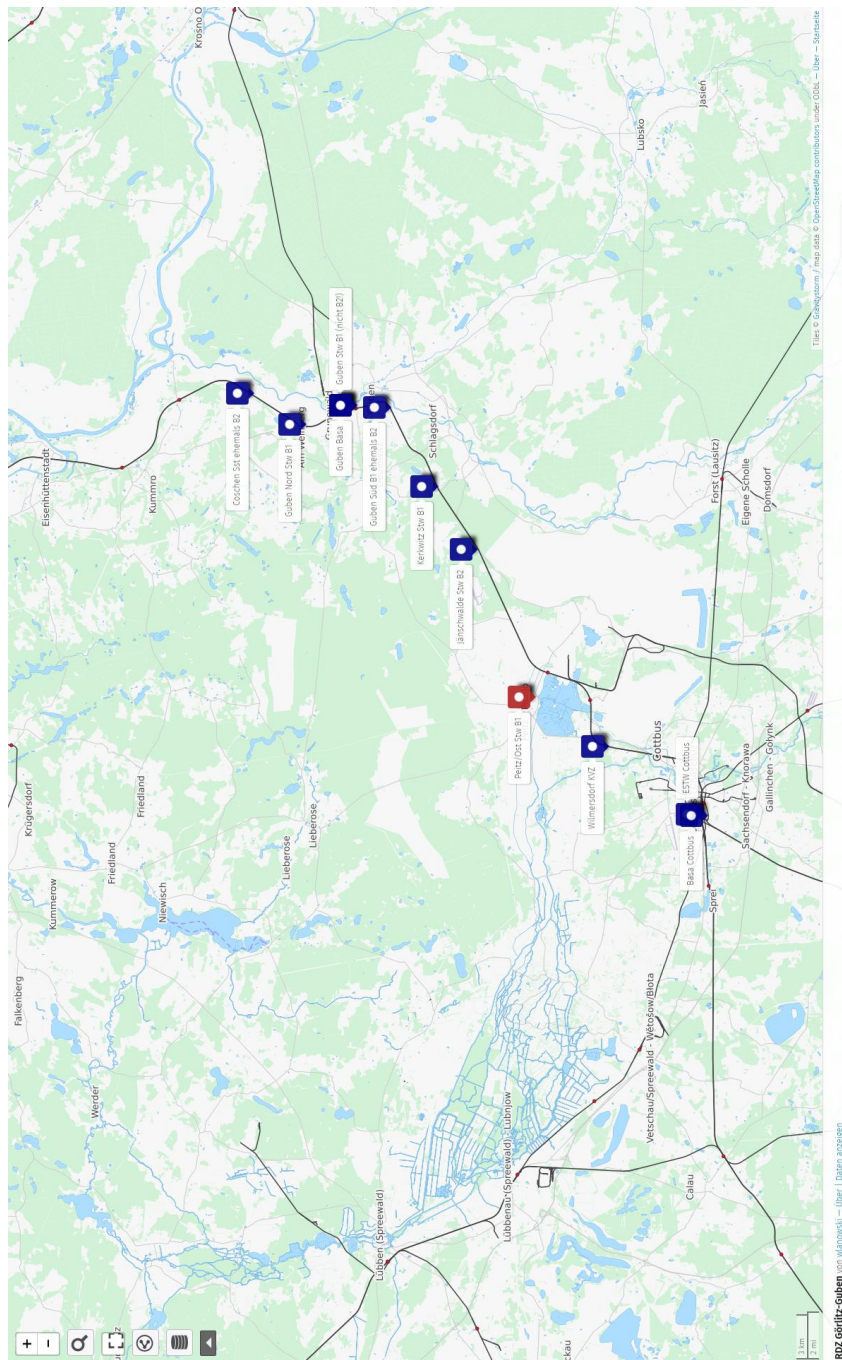


Abbildung 1: erstellte umap-Karte der RDZ-Linie 11B im Abschnitt Cottbus-Guben (verfügbar auf https://umap.openstreetmap.fr/de/map/rdz-gorlitz-guben_149763)

Abbildung 1 zeigt eine solche Karte. Aufgabe war es alle Stationen einer RDZ¹-Linie in einem gegebenen Abschnitt übersichtlich auf einer Karte darzustellen. Es wurde umap

¹ RDZ: Rechnergestützte Dispatcherzentrale, ein bei der Deutschen Reichsbahn eingeführtes System zur Zuglaufverfolgung.

als Plattform genutzt, die Koordinaten entstammen aus NeDocS. Der Bearbeiter der Karte ist Autor dieser Arbeit.

Umap ist schnell erlernbar und einfach zu bedienen, Kenntnisse im Umgang mit GIS-Systemen erleichtern jedoch die Arbeit mit umap. Die hohe Anpassbarkeit der Ansichten ohne Programmierkenntnisse bedeutet einen schnellen zielführenden Erfolg. Der Nachteil bei Umap besteht darin, dass die Benutzerorientierung von umap einen hohen Administrationsaufwand mit sich bringt. Die Nutzung des Dienstes an sich ist zudem nicht mit den Regeln der Geheimhaltung von Unternehmensdaten vereinbar. Die Schaffung einer gleichen Infrastruktur ist dank Quelloffenheit und Dokumentation möglich, jedoch nur unter vergleichsweise schwierig erfüllbaren Bedingungen, wie das Nutzen eines speziellen Datenbankformates. Geodaten können zudem nur statisch eingebunden werden, das Einbinden von dynamischen Daten, beispielsweise aus einer Datenbank, ist mit umap nur mit einer tiefgründigen Anpassung der Software möglich. Außerdem ist es mit umap nicht möglich, mehrere Tilelayer einzufügen. Das Einbinden der OpenRailWayMap würde, da sie ein Overlay-Tile zurückgibt, nur auf einem grauen Hintergrund zu sehen sein.

5.1.3.3 Google Maps API

Die Google Maps API ist eine proprietäre Schnittstelle für GIS-Anwendungen die in verschiedenen Zweigen angeboten wird. Die Zweige enthalten Schnittstellen für die gängigen Plattformen. Die Entwicklung mobiler Applikationen für Android und iOS würde durch das Nutzen der Google Maps API vereinfacht werden. Der Autor entschied sich im vorherigen Abschnitt für eine webbasierte Plattform, auf der verschiedene Daten visualisiert werden. Die Google Maps JavaScript API¹ ist am besten für die Erfüllung der Anforderungen in Kapitel 2 dieser Arbeit einsetzbar. Die gewerbliche Nutzung, die in diesem Fall gegeben ist, ist anders als in den bisher beschriebenen Lösungen nicht kostenfrei. Der Preis für die Standardlizenz variiert abhängig von der Anzahl an Anforderungen an die Schnittstelle und reicht bis zu 1.125 US-Dollar für 100.000 Aufrufe pro Monat². Es wird zudem eine Premiumlizenz angeboten, Preise für diese

¹ (Google Inc.)

² <https://developers.google.com/maps/pricing-and-plans/?hl=de> [Stand und Abrufdatum: 07. Juni 2017]

werden nur auf Anfrage genannt. In der Premium-Variante ist ein technischer Support enthalten.

Google Maps genießt insbesondere im deutschsprachigen Raum einen guten Ruf und wird von vielen Internetnutzern genutzt. Die Einführung einer GIS-Anwendung mit der bekannten und gewohnten Google Maps-Karte würde eine hohe Akzeptanz finden. Durch das Einbinden weiterer APIs von Google Maps kann die GIS-Anwendung weitere Aufgaben übernehmen, beispielsweise das Konvertieren von Postadressen in Geokoordinaten.

5.1.3.4 Leaflet und OpenLayers

Leaflet und OpenLayers sind Bibliotheken für GIS-Anwendungen, die wie die Google Maps JavaScript API auf JavaScript basieren. Der maßgebliche Unterschied liegt in der Quelltextoffenheit und der Möglichkeit der freien Benutzung auch für kommerzielle Zwecke. Es wird jedoch keine Haftung übernommen und es gibt keine Garantie.¹

Die Funktionsweise der beiden Bibliotheken ist gleich. Durch auf dem Clientsystem ausgeführten Code werden über verschiedene Schichten (Layer) Geo-Informationen ein- oder ausgeblendet. Ein Unterschied zwischen OpenLayers und Leaflet liegt in der Anzahl der mitgebrachten Funktionen. Während Leaflet den Anspruch hat, möglichst klein und minimalistisch dazustehen, hat OpenLayers den Anspruch möglichst viele Funktionen in die Bibliothek einzubringen, was die Größe der Bibliothek erhöht. Die Anzahl an verfügbaren Plug-Ins von Drittanbietern zeigt, dass OpenLayers von den Funktionen lebt, welches es sich selbst in seiner Bibliothek mitbringt, während die Funktionen von Leaflet durch eine Vielzahl von Plug-Ins erweitert werden können.

OpenLayers gilt als detaillierter was GIS-Anwendungen angeht.² Da die Bibliotheken sich ansonsten sehr ähnlich sind, soll nun die Schwierigkeit der Erlernbarkeit der Syntax verglichen werden. Eine selbst gestellte Beispielaufgabe soll den Programmieraufwand mit Leaflet und OpenLayers vergleichen:

¹ siehe Inhalt der „BSD 2-clause License“ unter <https://opensource.org/licenses/BSD-2-Clause> [zuletzt aufgerufen 07.06.2017] (englisch). Diese Lizenz gilt für Leaflet und OpenLayers.

² (diverse Autoren)

Erstellen Sie eine Webkarte mit folgenden Anforderungen:

- Die OSM Transport Map soll als Hintergrund dienen.
- Die Karte soll auf die Dezimalkoordinaten 51.16997107448923, 10.521424594879116 Zoomlevel 6 initialisiert werden.
- Es soll ein Marker auf die Koordinaten 51.3766 / 12.4783 gesetzt werden.

Da der Autor zum Zeitpunkt dieser Aufgabenstellung keinerlei Kenntnisse über die Programmierung mit den genannten Bibliotheken besitzt, soll auch die Dokumentation der Bibliotheken verglichen werden, weswegen zur Erfüllung der Aufgabe nur die offiziellen Webpräsentationen der Bibliotheken konsultiert wurden. Auf das Zurückgreifen auf Suchmaschinen und anderen Webseite oder Literatur wurde bewusst verzichtet. Es wurden die aktuellsten Versionen von OpenLayers und Leaflet (Stand: 08. Juni 2017) verwendet.

Die JavaScript-Codes in Tabelle 2 (Seite 20) zeigen, wie die oben genannte Aufgabe mit Hilfe der Bibliotheken von OpenLayers und Leaflet umgesetzt werden kann. Auf das Einfügen der benötigten HTML und CSS-Codes wird hier verzichtet. Im Code fällt zunächst auf, dass die benötigte Zeilenanzahl in Leaflet wesentlich geringer ist, als in OpenLayers. Analog dazu war die benötigte Zeit zum Erfüllen der gestellten Beispielaufgabe auch kürzer. Neben den benötigten Zeilen liegt das aber auch an den Dokumentationen. Die OpenLayers-Dokumentation ist sehr ausführlich, zudem werden Anfängern wesentlich mehr Beispiele gezeigt. Nichtsdestotrotz dauerte das Finden einer passenden Lösung lang. Erschwert wurde dies unter anderem auch dadurch, dass die genannten Beispiele an einigen Stellen eine ältere Version der OpenLayers-Bibliotheken benutzten, was nicht sofort offensichtlich war. Die Codes für die älteren Bibliotheken sind nicht mit der aktuellen Bibliotheksversion kompatibel.

OpenLayers wirkt im Code als auch in den Beispielen als eingehenderes Tool für GIS-Anwendungen als Leaflet. Leaflet wirkt dagegen sowohl im Code, als auch in der Dokumentation und in den Beispielen als weniger komplex. Die vom Autor dieser Arbeit gesammelten Eindrücke werden auch von anderen Programmierern geteilt. Der

spanische Entwickler für Indoor-Karten Iván Sánchez Ortega beschreibt in einer Bildschirmpräsentation¹ zusammenfassend folgende objektive Vor- und Nachteile:

- Die Anfängerfreundlichkeit ist bei Leaflet gegeben, während OpenLayers sich an GIS-Spezialisten wendet (Kapitel 3, Seite 6).
- Leaflet bringt weniger Funktionen selbst mit, eröffnet aber durch über 200 Plugins Erweiterungsmöglichkeiten. OpenLayers bringt viele Funktionen mit, bietet jedoch kaum Erweiterungen (Kapitel 4, Seite 1).
- Die Dokumentation von Leaflet zeigt wenige Beispiele, dafür eine präzise Dokumentation der API. OpenLayers bietet Beispiele, jedoch hat Ortega auch die Erfahrung mit nicht funktionierenden Tutorials gemacht. Die Dokumentation der API beschreibt er als zu ausführlich. Die Qualität der Dokumentation ist bei OpenLayers besser.
- Bestimmte Funktionen, wie eine Rotation der Karte, sind mit Leaflet nicht nur mit Einschränkungen oder extremen Programmieraufwand möglich.

Ortega gibt am Ende die Empfehlung ab, Leaflet zu benutzen, sollte man eine Webseite gestalten. GIS-orientierte, professionelle Anwender sollten zu OpenLayers greifen.

Der Autor dieser Arbeit entscheidet sich durch die gemachten Erfahrungen mit der Programmierung beider Bibliotheken dazu, die GIS-Anwendung mit der Leaflet-Bibliothek zu programmieren.

¹ (Ortega, 2015)

JavaScript zur Erfüllung mit OpenLayers 3

```
// Karte initialisieren
var map = new ol.Map({
  target: 'map',
  layers: [
    // Start:Tile-Layer hinzufügen
    new ol.layer.Tile({
      source: new ol.source.XYZ({
        url: 'http://{a-
c}.tile2.opencyclemap.org/transport/{z}/{x}/{y}.png'
      })
    })
    // Ende:Tile-Layer hinzufügen
  ],
  view: new ol.View({
    center: ol.proj.fromLonLat([10.521424594879116,
51.16997107448923]),
    zoom: 6
  })
});

// Marker hinzufügen
var markers = new ol.layer.Vector({
  source: new ol.source.Vector({
    features: [
      new ol.Feature({
        geometry: new ol.geom.Point(ol.proj.fromLonLat([12.4783,
51.3766])),
      })
    ]
  }),
  style: new ol.style.Style({
    image: new ol.style.Icon({
      src: '//openlayers.org/en/v3.12.1/examples/data/icon.png',
      anchor: [0.5, 1]
    })
  })
});
map.addLayer(markers);
```

JavaScript zur Erfüllung mit Leaflet

```
//Karte initialisieren
var map = L.map('map').setView([51.16997107448923, 10.521424594879116],
6);

//Tile-Layer hinzufügen
L.tileLayer('http://{s}.tile2.opencyclemap.org/transport/{z}/{x}/{y}.png', {
  attribution: '&copy; <a
href="http://osm.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);

//Marker hinzufügen
L.marker([51.3766, 12.4783]).addTo(map);
```

Tabelle 2: Vergleich JavaScript-Code von OpenLayers und Leaflet

5.2 Datenspeicherung, -verarbeitung und Benutzeroberfläche

In Punkt 5.1 wurde eine webbasierte JavaScript-Bibliothek für GIS-Anwendungen gewählt. Diese bringt nicht alle im Abschnitt 3.1 geforderten Funktionen mit, weswegen zusätzliche Software-Komponenten notwendig sind. Aufgrund der Auswahl einer Browser-Anwendung für die geografischen Verarbeitungen, bietet sich zunächst der Einsatz eines Webserver an. Verbreitet sind PHP¹- und MySQL²-Plattformen. Sie erlauben das dynamische Erzeugen von Webseiten mittels PHP und eine hochperformante Speicherung von Daten mittels MySQL. PHP- und MySQL sind, anders als andere Webskriptsprachen und Datenbankplattformen, auf den meisten Webhostern vorhanden und verfügbar. Aufgrund bereits vorhandener Programmier- und Administrationskenntnisse mit diesen Systemen, sowie der bereits vorhandenen Infrastruktur, legt sich der Autor auf eine PHP-/MySQL-basierte Verarbeitung und Erzeugung des Inhaltes fest. Auf andere Plattformen soll hier nicht eingegangen werden. Die ausgewählten Plattformen lassen auf eine Architektur schließen, wie sie mit dem UML-Diagramm in Abbildung 2 gezeigt wird.

¹ PHP: Hypertext Preprocessor. Skriptsprache zum dynamischen Erstellen von Webseiten

² MySQL: Datenbankplattform für Web- und sonstige Anwendungen

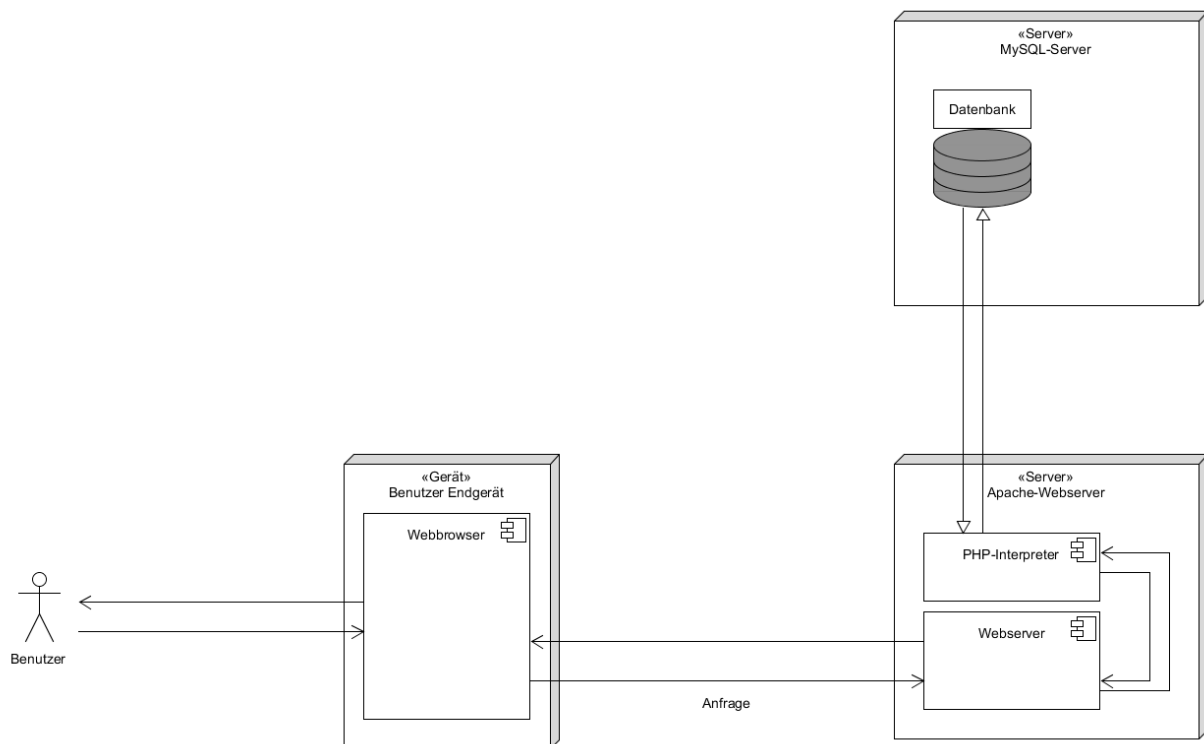


Abbildung 2: Architektur der benötigten bzw. verwendeten Hardwareinfrastruktur

Zur Vereinfachung und Zeitersparnis soll ein Framework¹ benutzt werden. Ein Framework schafft einen Rahmen und verringert somit den Programmieraufwand. Eine wichtige Anforderung an das Webframework in dieser Arbeit liegt im responsive Design. Weiterhin sollen eine Administrationsoberfläche und die dafür notwendigen Elemente mitgeliefert werden. Wichtig ist zudem, dass die Benutzung des Frameworks auch für kommerzielle Zwecke ohne schwerwiegende Einschränkungen rechtlich möglich ist. Nach Recherche und Versuchen an Frameworks verschiedener Art entschied der Autor sich für das Bootstrap²-Framework „gentelella“ von Colorlib. Es wurde unter der MIT-Lizenz veröffentlicht, welches das Kopieren, Bearbeiten und Nutzen des Frameworks unter Namensnennung erlaubt.

¹ Framework: zu Deutsch: Rahmen, Gerüst

² ein von Twitter Inc. entwickeltes Framework auf Grundlage von HTML und CSS

6 Programmierung der Software

Da die Ausführung von PHP-Skripten, sowie das Hosten von MySQL-Datenbanken eine Webserversoftware benötigt, wurde die Softwaresammlung XAMPP in einer portablen Version herangezogen. Programmiert wurde mit der Entwicklungsumgebung PhpStorm. Für die Datenbankadministration wurde zunächst das in XAMPP mitgelieferte Webtool phpmyadmin benutzt. Aufgrund der Masse an durchgeführten Verarbeitungen und Abrufe, sowie wegen der wenig performanten Oberfläche wurde im Laufe der Entwicklung auf die Datenbankverwaltungssoftware HeidiSQL zurückgegriffen. Weiterhin wurde bei dem Git-Bereitsteller github.com ein Repository angelegt. Dies erlaubt das Programmieren an verschiedenen Standorten sowie eine Historie über vergangene Änderungen. Der Quelltext der Webanwendung, sowie der Karte ist frei verfügbar¹. Im Laufe der Entwicklung wurde bei der Konvertierung und Verarbeitung von Geodaten verschiedene andere Software benutzt. Auf diese wird an der jeweiligen Stelle gesondert eingegangen.

Die Software an sich lässt sich in verschiedene Module eingliedern, die im zusammengeführten Gesamtbild die fertige Software ergeben. Im Groben sind das zunächst das Programmieren der Karte und weiterhin die Benutzeroberfläche. Die Datenbank fungiert als Schnittstelle zwischen beiden Teilen, da in dieser die Projektdetails inklusive der geografischen Standorte hinterlegt sind. Weiterhin liegen da die Daten der Benutzerkonten, weswegen eine zusätzliche Interaktion zwischen Datenbank und Weboberfläche stattfindet. Der normale Benutzer hat dabei nur Interaktionsmöglichkeiten mit der Weboberfläche in seinem Browser (vgl. dazu auch Abbildung 2).

Nachfolgend werden Details zur Ideenfindung und -durchsetzung bei der Programmierung beschrieben. Bei der Beschreibung werden keine grundlegenden Punkte für die Programmierung mit HTML, JavaScript, PHP und MySQL aufgezeigt.

¹ vgl. Anlage 4: Verweise auf den Quellcode und Demonstrationsexemplar dieser Arbeit

6.1 Details zur Programmierung der geografischen Anwendungen

Zunächst soll die Grundlage für die geografische Verarbeitung genutzt werden. Wie in Abschnitt 5 erläutert, soll die JavaScript-Bibliothek Leaflet zum Einsatz kommen. Eine bloße Webkarte zu erstellen, um sie im Webbrowser anzeigen zu können, bedarf dabei nur wenige Zeilen, wie in Tabelle 2 bewiesen. Der darin gezeigte JavaScript-Code wird in den jeweiligen HTML-Header inkludiert. Der Code erzeugt dann einen HTML-Container, welcher wiederum im Body des HTML-Dokumentes eingebaut wird, damit dieser sich anzeigt. Benötigte weitere Skripte, wie die Leaflet-Bibliothek, sowie verwendete Plugins, werden in den Header des HTML-Codes inkludiert bzw. eingefügt. Nachfolgend sieht man das Grundgerüst der entsprechenden HTML-Datei, wie man sie im Browser aufruft:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Leaflet Beispiel </title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" href="leaflet/leaflet.css">
    <script src="leaflet/leaflet.js"></script>

  </head>
  <body>
    <div id="map" style="width: 100%; height: 400px;"></div>
    <script src="leaflet_beispiel.js"></script>
  </body>
</html>
```

Der zweite Meta-Tag macht die Karte responsiv. Mit dem eingebundenen Stylesheet `leaflet.css` wird das Aussehen verschiedener Kartenelemente, wie die Mitwirkendenen-Fußzeile bestimmt. Die eingebundene `leaflet.js`-Datei entspricht der Bibliothek. Im `div`-Tag im Body des HTML-Dokumentes wird unsere Karte angezeigt, die ID „map“ wird mit dem JavaScript beschrieben, der unsere Karte beschreibt, eingebunden als `leaflet_beispiel.js`.

6.1.1 Grundaufbau der Karte

Zunächst wird im JavaScript die Karte an sich definiert, damit diese in den HTML-Container eingefügt werden kann. Das wird durch den Code

```
var mymap = L.map('mapid', {  
  center: [51.679, 9.866],  
  zoom: 6,  
  layers: [base_cartolight]  
});
```

bewerkstelligt. 'mapid' entspricht dabei der ID des HTML-Layers. In den folgenden drei Zeilen werden die Standardeinstellungen der Karte definiert, also das was zu sehen sein wird, wenn die Karte neu geladen wird. Die Karte wird auf die in center definierten Koordinaten mit dem Zoomlevel 6 zentriert. Als Layer ist der Basislayer Carto Light (Minimal (s/w) standardmäßig aktiviert. Es ist bereits mit diesem Code möglich, durch die Karte zu scrollen und zu zoomen, wenn der Layer base_cartolight definiert ist. Weitere Funktionen werden erst eingebaut.

Die Karte wird durch verschiedene Daten und Layer, sowie durch Funktionen und Plugins erweitert. Zunächst sollen weitere Hintergründe definiert und auswählbar gemacht werden. Alle verwendeten Hintergründe werden als Tiles zur Verfügung gestellt, welche dem Schema folgen:

```
http://{s}.beispielserver.domain/{z}/{x}/{y}.png
```

Die Variable s wird im Laufe des JavaScripts fortlaufend durch die Buchstaben a, b und c ersetzt. Dadurch wird die Eigenschaft in modernen Browsern umgangen, dass nur ein paralleler Bilddownload zur selben Zeit pro Domain erlaubt ist. Die Variable z steht für den Zoomlevel, x und y für die Koordinaten Dezimalschreibweise. Durch den Leafletbefehl

```
var layername =  
L.tileLayer('http://{s}.beispielserver.domain/{z}/{x}/{y}.png'  
, {  
  attribution: 'Quellenangabe zum Layer'  
});
```

kann ein einer Tile-Layer definiert werden. Eine Besonderheit bei den Tilelayern gibt es noch zu beachten: Man unterscheidet zwischen Base-Tilelayer und Overlay-Tilelayer.

Beide geben nach der Anforderung png-Dateien wieder, wobei Base-Tilelayer keine Transparenz und Overlay-Layer transparente Kacheln wiedergeben. Der Hintergrund ist, dass mit Overlay-Layern bestimmte Details hervorgehoben auf einem bestimmten Hintergrund gezeigt werden können. Die OpenRailwayMap bietet solche Overlay-Layer. Die in der Tabelle 3 aufgeführten Tilelayer wurden in die Karte über den oben genannten Befehl definiert und eingearbeitet.

Name des Layers in dieser Arbeit Variablenname	Urheber	Beschreibung	Typ
Google Satellit base_Google	GeoBasis-DE/BKG, Google	Satellitenansicht von Google Maps ohne Beschriftungen	Base
Minimal (s/w) base_cartolight	Carto	Graustufenkarte mit Beschriftungen, eignet sich gut als Hintergrund für Daten und s/w-Ausdrucke	Base
OpenStreetmap base_OSM	OpenStreetMap	Straßenkarte mit vielen Farben und POIs	Base
TransportMap base_TPM	Thunderforest, OpenCycleMap	Für den Transport wichtige Hervorhebungen, beispielsweise Eisenbahnstrecken und Autobahnen	Base
OpenRailwayMap: Infrastruktur ORM_INFRA	OpenRailwayMap	bloße Linien für Eisenbahnstrecken mit Hervorhebung je nach Streckenart. Anzeige von Streckennummern und Betriebsstellen	overlay
OpenRailwayMap: Maximale Geschwindigkeiten ORM_SPEED	OpenRailwayMap	bloße Linien für Eisenbahnstrecken, Farbe je nach höchster zugelassener Geschwindigkeit	overlay
OpenRailwayMap: Signalisierung ORM_SIGNAL	OpenRailwayMap	Zugsicherungsarten und Standorte von Eisenbahnsignalen inkl. Bezeichnungen	overlay

Tabelle 3: Verwendete Tilelayer in der Karte

Der Layer Minimal (s/w) ist mit der Kartendefinition bereits standardmäßig aktiviert. Um andere Layer aus- und anschalten zu können, muss ein Kontrollsystem eingebaut werden. Das von Leaflet mitgebrachte Layerkontrollsystem funktioniert dabei gruppenbasiert. Eine Gruppe wird dabei ganz normal über ein JavaScript-Array definiert. Beispielsweise sieht die Gruppendefinition der Hintergrundlayer so aus:

```
var baseLayers = {  
  "Google Satellit": base_Google,  
  "Minimal (s/w)": base_cartolight,  
  "OpenStreetMap": base_OSM,  
  "TransportMap": base_TPM  
};
```

Durch die Gruppierung und die Initialisierung des Kontrollsystems mit

```
L.control.layers(baseMaps, ORM_overlays1).addTo(mymap);
```

wird das folgende Kontrollsystem in der Karte angezeigt (Abbildung 3):



Abbildung 3: Layerkontrollsystem mit Standardmitteln aus Leaflet.

Die Hintergrundkarten lassen sich nun ab- und anwählen, wobei die Auswahl hier exklusiv ist. Das heißt, dass nur ein Hintergrund ausgewählt werden kann, die Wahl eines anderen deaktiviert den zuvor gewählten. Die OpenRailwayMap-Overlays lassen sich auch ab- und anwählen, jedoch ist die Auswahl nicht exklusiv. Die OpenRailwayMap zeigt auf verschiedenen Layern verschiedene Details zu den gleichen Standorten. Eine nicht exklusive Auswahlmöglichkeit macht wegen der daraus

¹ ORM_overlays entspricht der Gruppe der OpenRailwayMap-Tile-Overlays.

resultierenden Überlappung mit den verwandten Layern keinen Sinn. Die Standardfunktionen von Leaflet erlauben nur einer Gruppe die exklusive Auswahl, weswegen die Funktion durch das Plugin `leaflet-groupedlayercontrol`¹ erweitert wird. Zunächst wird durch das Definieren eines leeren Tilelayers ein auswählbarer Layer erstellt. Er wird als Ausschalter verwendet. Mithilfe der Definition der Optionen

```
exclusiveGroups: ["OpenRailwayMap"],  
groupCheckboxes: true
```

wird die vorher definierte und mit "OpenRailwayMap" bezeichnete Gruppe exklusiv gemacht. Der neu erstellte leere Layer wird in die Initialisierungsoptionen der Karte mit aufgenommen, damit der Markierungspunkt im Kontrollsystem aktiv ist. Durch das Ersetzen des alten Kontrollsystems mit

```
var layerControl = L.control.groupedLayers(baseLayers,  
groupedoverlays, options, null, {  
    collapsed: true,  
    autoZIndex: false  
});
```

wird das neue durch das Plugin definierte Layerkontrollsystem aktiviert. Das Ergebnis ist das folgende gewollte Kontrollsystem (Abbildung 4):



Abbildung 4: Mit dem Plugin `leaflet-groupedlayercontrol` erstelltes Layerkontrollsystem

¹ Eine Zusammenstellung aller verwendeten Leaflet-Erweiterungen wird beim Aufruf des Punktes „Über“ in der Software angezeigt.

6.1.2 Einfügen der Geodaten der DB Netz AG

Wie in Kapitel 4 dieser Arbeit bereits erwähnt, veröffentlicht die Deutsche Bahn Datensätze und Programmierschnittstellen rund um die Arbeitsbereiche des eigenen Konzerns. Darunter sind auch Geodaten von verschiedenen Erfassungsdaten zu finden. Die zum Zeitpunkt der Erstellung dieser Arbeit aktuellsten Geodaten tragen das Erfassungsdatum von Januar 2017 und wurden im Mai 2017 veröffentlicht. Die Geodaten sind untergliedert in die in Tabelle 4 genannten Bereiche. Aus der Tabelle ist ersichtlich, welche Dateiformate zur Verarbeitung bereitstehen. Bei der Untersuchung der Dateien wurde festgestellt, dass insbesondere bei den Excellisten und .csv-Dateien zur geografischen Verarbeitung wichtige Details nicht zur Verfügung stehen, weshalb sich diese für die Aufbereitung der Karte nicht eignen. Leaflet bringt keine Funktionen mit, das proprietäre MapInfo-Format, sowie die Shapefiles aufbereiten zu können. Mithilfe des Tools ogr2ogr, welches die Konvertierung von Geodaten ermöglicht, wurden die MapInfo-Dateien in das offene GeoJSON¹-Format gewandelt.

Datensatz		verfügbare Formate	Typ der hinterlegten Objekte
Strecken		.xlsx, .csv, MapInfo Relationen, Shapefiles, Datenbeschreibungen zur Exceltabelle und den MapInfo Relationen	Polylinien
Bahnübergänge		.xlsx, .csv, MapInfo Relationen, Shapefiles, Datenbeschreibungen zur Exceltabelle und den MapInfo Relationen	Punkte
Betriebsstellen		.xlsx, .csv, MapInfo Relationen, Shapefiles, Datenbeschreibungen zur Exceltabelle und den MapInfo Relationen	Punkte
Brücken	gesamt	.xlsx, .csv, Datenbeschreibungen zur Exceltabelle	-
	klein	MapInfo Relationen, Shapefiles, Datenbeschreibung zu den MapInfo Relationen	Punkte

¹ Die typische Syntax eines GeoJSON-Objektes ist in Anlage 3: Dresden Hauptbahnhof Gleis 19 in OSM konvertiert zu GeoJSON (Stand: 25. August 2017) ersichtlich.

	groß	MapInfo Relationen, Shapefiles, Datenbeschreibung zu den MapInfo Relationen	Polylinien
Kilometer		MapInfo Relationen, Shapefiles, Datenbeschreibung zu den MapInfo Relationen	Punkte
Tunnel		MapInfo Relationen, Shapefiles, Datenbeschreibung zu den MapInfo Relationen	Polylinien

Tabelle 4: Beschreibung der gelieferten Geodaten (DB Netz AG) vom 5. Mai 2017

Für die Auswertung von Geodaten mittels PHP wurden die Geodaten in eine MySQL-Datenbank überführt. Dafür wurden, falls vorhanden, die mitgelieferten CSV-Dateien mittels der Software HeidiSQL in die Datenbank importiert. Im Fall der Brücken konnte die CSV-Datei nicht benutzt werden, da sie keine Informationen über die Polyline-Daten der großen Brücken enthielt. Bei den Tunneln wurde keine CSV-Datei mitgeliefert. In beiden Fällen wurden die in den ShapeFiles enthalten DBF-Datenbanken mit OpenOfficeCalc in eine CSV-Datei konvertiert. Die Konvertierung mit Microsoft Excel 2010 war auf Grund von Kodierungsproblemen¹ nicht möglich.

Bei dem Formular für die Projekterstellung (siehe 6.2) gibt es die Möglichkeit, durch die Eingabe von Bahnstreckennummer und Bahnkilometer, Koordinaten zu erhalten. Dafür mussten die Kilometer mit Koordinaten in die Datenbank eingefügt werden. Da keine CSV-Datei vorhanden ist, und die vorliegende DBF-Datei keine georeferenzierbaren Daten beinhaltete (keine Geokoordinaten), wurden die Shapefiles mit ogr2ogr in eine durch PostGIS erweiterte PostgreSQL-Datenbank eingearbeitet. In dieser waren die georeferenzierten Daten eingebunden, die Geometrie lag im „wkb_geometry“-Format vor. Da die Kilometer-Daten ausschließlich punktuelle Geoinformationen beinhalten, konnte mit der PostGIS-Erweiterung dieses Format in eine X- und Y-Koordinate umgewandelt werden. Die Koordinaten wurden in neue Spalten der Tabelle in der Datenbank eingetragen, die Tabelle konnte dann in eine CSV-Datei exportiert werden, diese wiederum konnte in die MySQL-Datenbank dieser Arbeit importiert werden.

¹ Die mit ISO 8859-1 kodierte DBF-Datei konnte nicht mit Microsoft Excel in das von der Datenbank geforderte UTF-8 konvertiert werden.

Die GeoJSON-Dateien wurden konvertiert auf den Webserver geladen, diese werden mit AJAX¹ in das JavaScript geladen und von der Leaflet-Bibliothek verarbeitet. Der Aufbau eines GeoJSON-Datensatzes entspricht annäherungsweise dem eines bekannten JSON-Datensatzes. Der folgende Code beschreibt den Bahnhof Dresden Hauptbahnhof in GeoJSON:

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [
          13.73118873502,
          51.039933336314
        ]
      },
      "properties": {
        "id": "SNode-1492253",
        "formOfNode": "railwayStop",
        "railwayStationCode": "DH",
        "geographicalName": "Dresden Hbf",
      }
    }
  ]
}
```

Zunächst wird deklariert, dass es sich in der Datei um eine FeatureCollection, also sinngemäß um eine Sammlung von Objekten handelt. Das GeoJSON-Format schreibt dann vor, dass zunächst der Typ festgelegt werden soll, im Beispiel "type": "Feature". Danach folgt die geometrische Beschreibung des Objektes. Am oben beschriebenen Beispiel handelt es sich um einen Punkt an den Koordinaten 13,73118873502; 51,039933336314. Die darauffolgend beschriebenen properties können das Objekt optional und zusätzlich beschreiben, wobei die darin genutzten Schlüssel, also id,

¹ Akronym: Asynchronous JavaScript and XML, eine Schnittstelle zur Datenübertragung zwischen Client und Server mittels JavaScript.

formOfNode usw. frei wählbar sind. Die darin enthaltenen Informationen können dann zusätzlich verarbeitet werden.

Die einfachste Möglichkeit eine geoJSON-Datei über Ajax einzubinden ist über das Leaflet-Plugin leaflet-ajax. Die einzige benötigte Codezeile, die einen eigenen Layer mit den in der angegebenen GeoJSON-Datei Daten erstellt gleicht der folgenden:

```
var geojsonLayer =  
L.geoJson.ajax("../geodata/konvertiert/bst/bst.geojson");
```

Bei einem Test mit dem Datensatz aus dem Bereich der Betriebsstellen, fiel eine sehr lange und störende Ladezeit der Daten auf. Ein Verschieben der Karte empfand sich als sehr unangenehm. Schuld daran ist vermutlich die enorme Datenmenge die es zu verarbeiten und anzuzeigen gilt. In dem Betriebsstellen-Datensatz gibt es 9829 Dateneinträge. Als Abhilfe wurden die Punkte und Linien mittels des Plugins leaflet-markercluster gruppiert. Die Performance hat sich dadurch merklich erhöht. Die Funktionsweise liegt darin, dass die heruntergeladenen Marker je nach Zoomstufe gruppiert werden und statt jeden Punktes, nur die Gruppen angezeigt werden. Die folgende Abbildung (Abbildung 5) zeigt, wie die Gruppen dann angezeigt werden.

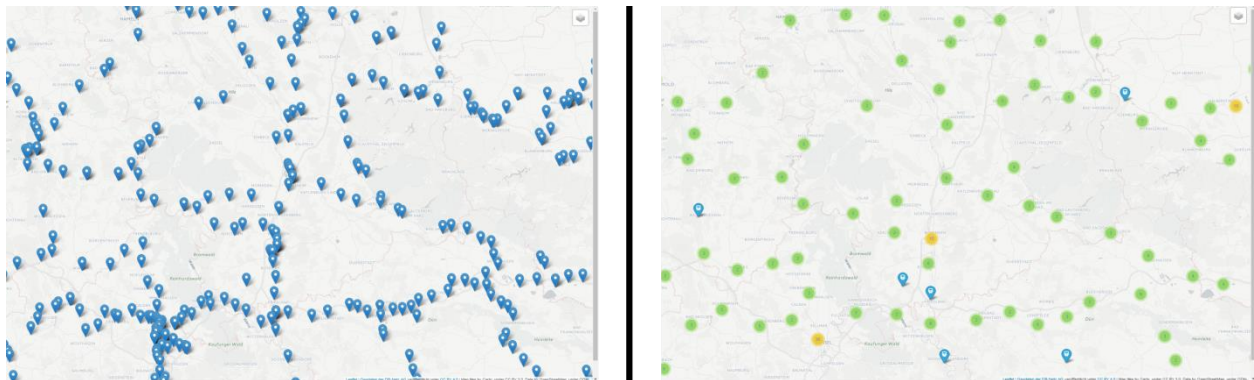


Abbildung 5: Vergleich zwischen dem standardmäßigen Verarbeiten (links) und dem geclusterten Verarbeiten der Betriebsstellen-GeoJSON-Datei

Das Aussehen der Marker wurde bei der rechten Variante bereits angepasst. Auf die Ladegeschwindigkeit hat diese Veränderung keinen fühlbaren Einfluss.

Ein positiver Nebeneffekt der Clusterung ist die deutliche Verbesserung der Übersichtlichkeit. Die Möglichkeit der Clusterung funktioniert jedoch nur bei Objekten, die als Punkt angezeigt werden.

Auch eine Clusterung der Projekte (nächster Abschnitt dieser Arbeit) wurde vorgenommen. Der Hintergrund ist hier jedoch nicht die Verbesserung der Performance, sondern das praktische Feature des genutzten Plugins, Objekte die auf genau denselben Koordinaten abgelegt sind trotzdem erkennbar zu machen. Das wird am folgenden Beispiel deutlich:

Der Bahnhof Dresden-Neustadt ist ein Bahnknotenpunkt, in dem mehrere Eisenbahnstrecken zusammenlaufen. Ohne auf die bahnbetrieblichen Hintergründe einzugehen, verlaufen durch den besagten Bahnhof die folgenden Strecken:

- 6239 Pirna - Coswig
- 6241 Dresden Hbf - Dresden-Neustadt
- 6246 Dresden-Pieschen - Dresden-Neustadt
- 6363 Leipzig Hbf - Dresden-Neustadt

Die zweigleisige Strecke 6241 kommt von Südwesten und endet im Bahnhof Dresden Neustadt. Die ebenfalls zweigleisige Strecke 6363 kommt von Nordosten und endet auch im Bahnhof Dresden-Neustadt. Die beiden Strecken enden geografisch auf demselben Punkt. Die Betriebsstellen für diese Strecken liegen deswegen auch auf den geografisch selben Koordinaten. Es liegen in Dresden-Neustadt also vier Betriebsstellen „übereinander“ (siehe Abbildung 6). Das besagte Leaflet-Plug-In leaflet-markercluster löst dieses Problem auf, in dem es bei einem Klick die Marker sternförmig um den Punkt anzeigt und durch ein eingezeichnetes Linienkreuz die wahre Position der versetzten Marker zeigt (s. Abbildung 7).

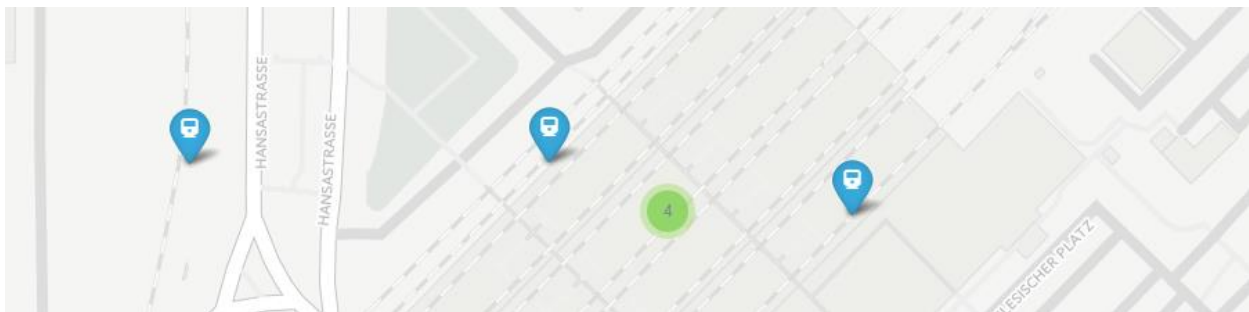


Abbildung 6: Vier übereinanderliegende Betriebsstellen in Dresden-Neustadt, angezeigt als Gruppe



Abbildung 7: Die vier übereinanderliegenden Betriebsstellen in Dresden-Neustadt, angezeigt als einzelne Marker

Aufgrund immer noch großer Ladezeiten der Daten, trotz Clustering, wurde die Möglichkeit, mehrere Datensätzen zur selben Zeit in der Karte anzuzeigen gezielt deaktiviert. Dass nur ein Datensatz von DB-Netz wählbar ist, ist unpraktisch, jedoch wurde die Bedienbarkeit hier vorrangig behandelt. Trotz der Deaktivierung lädt insbesondere beim ersten Aufruf der Seite, die Seite relativ lang. Schätzungsweise sechs bis sieben Sekunden vergingen¹, bis die Karte mit allen Punkten erfolgreich geladen wurde. Während dieser Zeit ist die gesamte Webseite nur mit Verzögerungen bedienbar, außerdem erhält man kein Feedback, dass gerade Daten geladen werden. Dies genügt nicht, um die in Punkt 3.4 gestellte Anforderung zu erfüllen. Zur Fertigstellung der Arbeit wurde noch keine bessere Vorgehensweise gefunden.

Die eingefügten Marker wurden angepasst. Leaflet bietet mit der Methode `setIcon` eine einfache Möglichkeit, die Icons seinen Bedürfnissen anzupassen:

```
layer.setIcon(L.AwesomeMarkers.icon({  
    icon: 'train',  
    prefix: 'fa',  
    markerColor: 'blue',  
    iconColor: 'white'  
}));
```

Hier wurde mithilfe des Plugins Leaflet-AwesomeMarkers ein Marker geschaffen, welches dem in der Abbildung 8 entspricht. Um die zusätzlich in `properties` angegebenen Eigenschaften visuell darstellen zu können wurde ein Popup über jedes Objekt gelegt. Es öffnet sich, wenn man auf das jeweilige Objekt klickt. Das Aussehen des Popups wird mit Hilfe gewöhnlicher HTML-Codes angepasst.

¹ gemessen mit den Entwicklerooptionen in Google Chrome 58, BKU-Notebook (Windows 7)

```
layer.bindPopup("
  <b>" + feature.properties.bezeichnung + "</b><br>
  Kürzel: <tab id=t1>" + feature.properties.kuerzel + "<br>
  Strecke: <tab to=t1>" + feature.properties.streckennu + "<br>
  Bkm: <tab to=t1>" + feature.properties.km_1
);
```

Nach der Anpassung der Marker und dem Einfügen des Popups, sieht das Ergebnis für den Betriebsstellendatensatz aus, wie es in Abbildung 8 zu sehen ist.

Für die anderen Datensätze wurden andere Stile für das Icon und ein anderer Aufbau der Popups gewählt. Auf diese wird hier nicht eingegangen.

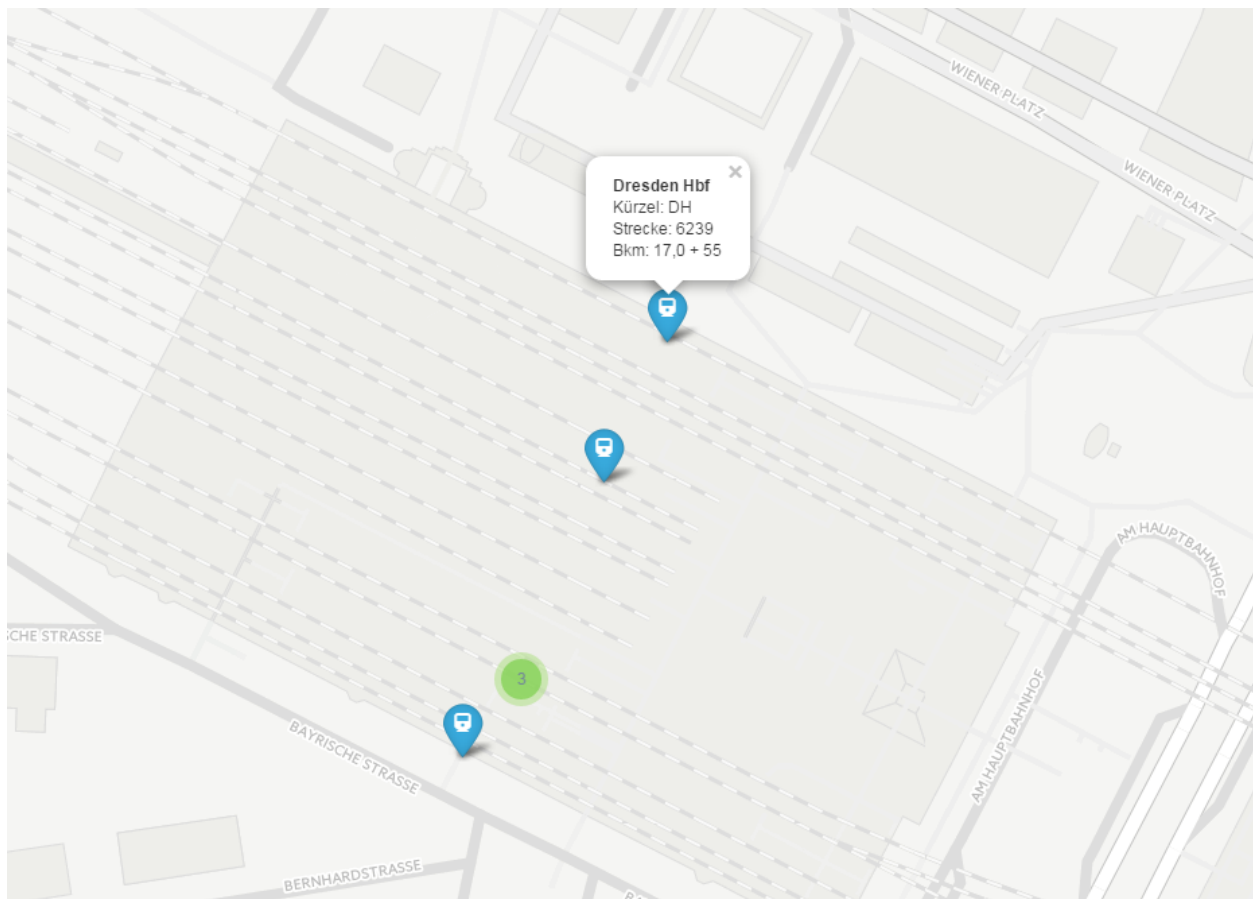


Abbildung 8: Fertiggestellter Kartenausschnitt Dresden Hbf mit geclusterten Geodaten der DB Netz AG (Datensatz: Betriebsstellen)

6.1.3 Einbindung der Projektdaten

Nach Punkt 3.1 dieser Arbeit sollen die Projektdaten zentral in einer Datenbank hinterlegt werden. In Punkt 5.2 wurde sich auf eine MySQL-Datenbank festgelegt. Es soll nun eine Möglichkeit gefunden werden, einen Datensatz, welcher Koordinaten

hinterlegt hat, in die Leaflet-Karte einzufügen. Da es sich hierbei, anders als die Geodaten von DB Netz, um dynamische, sich veränderte Daten handelt, muss eine Möglichkeit gefunden werden mittels einer Schnittstelle die MySQL-Daten in ein Leaflet-fähiges Format zu konvertieren. Das muss automatisiert werden und auf Abfrage funktionieren können. Ein Leaflet-fähiges Format haben wir mit den GeoJSON-Dateien im vorherigen Absatz bereits kennengelernt. In Anlehnung an McBride¹ wurde ein PHP-Skript benutzt, um die in der Datenbank hinterlegten Punkte dynamisch in GeoJSON zu wandeln. Das Skript tätigt eine klassische PHP-Abfrage und schreibt ein Array mit den aus der Tabelle entnommenen Daten. Die Koordinaten werden vorher an die Koordinatenposition eines typischen GeoJSON-Satzes kopiert und danach entfernt. Dann werden alle verbliebenen Einträge dem Array als Feature angehängen. Das wird für jeden Datensatz der Tabelle getan. Nach der Ausgabe entsteht eine dynamisch erstellte GeoJSON-Datei, die wie die Geodaten von DB-Netz mittels Ajax in das Skript eingebunden wird. Auch für die eingebundenen Projektdaten wurden die im vorherigen Abschnitt beschriebenen Anpassungen der Icons und des Popups vorgenommen.

6.2 Details zur Benutzeroberfläche

Neben der Karte gilt es noch eine allgemeine Benutzeroberfläche zu schaffen. Wie in Abschnitt 5.2 dieser Arbeit beschrieben, soll das Framework Bootstrap benutzt werden. Das responsive Design ist ein ausschlaggebender Grund dafür. Mithilfe des Bootstrap-typischen Gridsystems lassen sich Seiten gestalten, die das Layout dynamisch an die Displaygröße des Endgerätes anpassen. Breitere Geräte (Computermonitore) können so mehr Inhalte in der x-Achse anzeigen und generell ist durch die hohe Auflösung die Anzahl an anzeigbaren Informationen pro Fensterinhalt groß. Schmalere Geräte, wie Mobiltelefone, können durch ihre Größe weniger Informationen anzeigen, die Bedienung über Toucheingaben erfordern zudem ein möglichst großes Layout mit Buttons. Das Gridsystem von Bootstrap erlaubt es abhängig von der Gerätegröße eine Seite in verschieden viele Spalten aufzuteilen. Ein Beispiel, an dem man das responsive Design gut erkennen kann, findet sich mit der Abbildung 10 (Seite 40).

Das gewählte Derivat „gentelella“ bringt bereits viele Grundfunktionen mit und beschreibt die meisten Funktionen mittels Beispielseiten, deren HTML-Quellcode offen

¹ (McBride, 2015)

vorliegt¹. Der HTML-Quellcode der zu erstellenden Software soll nicht statisch für jede Seite geschrieben werden, sondern dynamisch mit PHP erzeugt. Das vermindert die Dateigröße auf dem Server. Generelle Änderungen an allen Seiten der Software können dann zentral vorgenommen werden. Zudem wird PHP verwendet, um Daten aus der MySQL-Datenbank zu entnehmen und zu verarbeiten. Unterstützt wird PHP durch JavaScript, im vorherigen Abschnitt wurde beispielsweise die Programmierung der GIS-Anwendung in JavaScript vorgenommen. Zum Zeitpunkt kurz vor Fertigstellung dieser Arbeit, war die Software folgendermaßen auf dem Server abgelegt: (Abbildung 9)

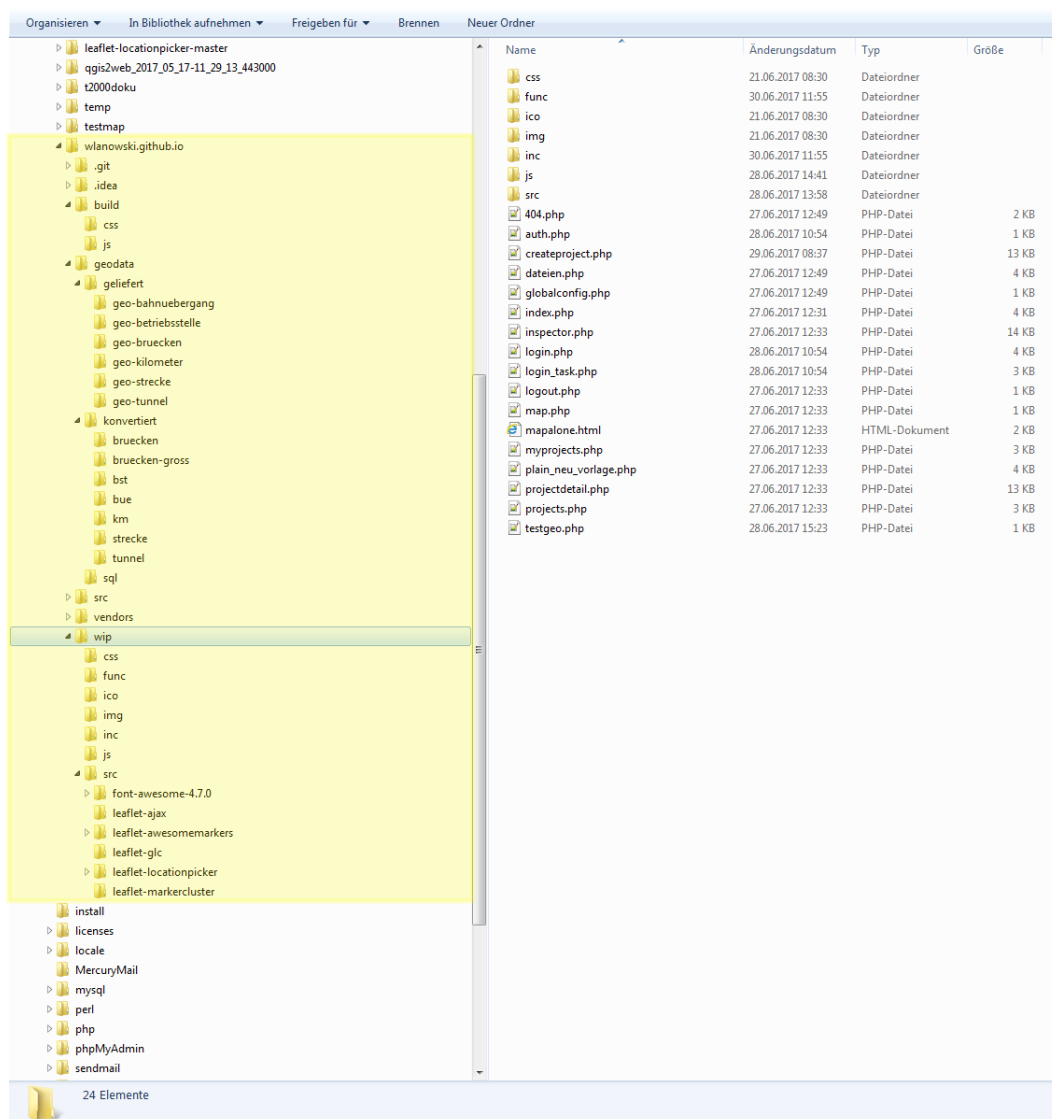


Abbildung 9: Verzeichnisbaum der Software, Stand: 30. Juni 2017

¹ (colorlib)

Gelb hervorgehoben ist in der Abbildung der Verzeichnisbaum des Projektes ohne die enthaltenen Dateien. Die Ordner `.git` und `.idea` sind vom Gitsystem, sowie von der benutzen Entwicklerumgebung PhpStorm automatisch angelegte interne Verzeichnisse. In den Ordnern `build`, `src` und `vendors` sind vom Framework verwendete Skripte und Stylesheets enthalten. Im Ordner `geodata` befinden sich ausschließlich die Geodaten der DB Netz AG im Auslieferungszustand und in einem Extraordner in konvertieren Dateiformaten. Um eine eventuelle Neuinstallation bewerkstelligen zu können, befindet sich im Ordner `SQL` ein gleichnamiges Skript, welches angewandt auf einen entsprechenden Datenbankserver alle verwertbaren Geodaten in eine Datenbank einträgt. Das Verzeichnis `wip` ist das Arbeitsverzeichnis und enthält die zum Seitenaufbau benötigten PHP-Skripte.

Im Folgenden werden grundlegende Bestandteile der Benutzeroberfläche beschrieben:

Mehrfach verwendete Codezeilen werden in PHP-Skripten ausgelagert und an entsprechender Stelle durch `require()`-Funktionen eingefügt. Beispiele dafür sind der immer gleichbleibende HTML-Kopf (`head`). Auch global benötigte Variablen, beispielsweise die Zugangsinformationen zum MySQL-Server werden zentral in einer Datei abgespeichert und an entsprechender Stelle mit eingebunden. Das erlaubt im Falle eines Serverwechsels (zum Beispiel beim Wechsel von der lokalen Entwicklungsumgebung in das Internet bzw. Internet) einen unkomplizierten Wechsel des Datenbankservers. Der Webseitenaufbau gleicht prinzipiell dem einer normalen HTML-Seite. Das grundlegende Layout der Arbeit wird dabei wie HTML-Kopf und -Fuß mit PHP ausgelagert und zentralisiert. Die Sidebar, die Kopfzeile und das Design finden sich in der Datei `layout.php` wieder. Da also generell wiederkehrende Elemente und Skripte ausgelagert sind, braucht sich bei der Erstellung einer Seite ausschließlich auf den Hauptinhalt der Seite konzentriert werden.

Zum Stand der Abbildung 9 sind vom Autor dieser Arbeit die in derselben Abbildung in der rechten Hälfte gezeigten Seiten erstellt wurden.

Die Datei `map.php` liefert die im Abschnitt 6.1 beschriebene Karte. Das JavaScript ändert hierbei den in der Datei festgelegten `div`. Die Datei `map.php` dient als Eingangsseite der Software. Die `index.php`, die bei der Eingabe der URL ohne Datei (nur Pfad) aufgerufen wird, leitet auf die Datei `map.php` weiter.

Eine weitere geografische Aufbereitung der Geodaten von DB stellt der Streckeninspektor (`inspector.php`) dar. Die Geodaten von DB Netz wurden in eine Datenbank eingefügt. Es existiert eine Tabelle, in der alle interessanten Objekte (Betriebsstellen, Bahnübergänge, Brücken und Tunnel) eingepflegt sind. Mittels der MySQL-Abfrage

```
SELECT * FROM geo_alles WHERE strecke= STRECKENNUMMER ORDER BY km_i  
ASC;
```

werden alle Datensätze einer eingegebenen Streckennummer geordnet nach Bahnkilometer selektiert und abgerufen. Abhängig von der Art des Objektes erfolgt dann eine grafische Aufbereitung aller interessanten Informationen für jedes Objekt. Durch die Sortierung nach Bahnkilometern kann so die Strecke tabellarisch abgefahren und so für Projekte relevante Orte gefunden werden. Neben den Objekten werden in einer anderen Tabelle Informationen zu Streckenabschnitten hinterlegt. Diese Informationen lassen sich wie die eben genannten Objekte mittels MySQL abrufen.

Über die Datei `projects.php` werden aus der Datenbank alle hinterlegten Projekte abgefragt und in einer Tabelle angezeigt. Diese Tabelle ist mit der Bibliothek `DataTables` erweitert (wird durch `gentellela` mitgeliefert), wodurch diese Tabelle mittels JavaScript durchsuchbar und filterbar ist. Das verringert die Zeit die benötigt wird, um alle für einen bestimmten Fall relevanten Projekte zu finden. Die Titel der Projekte verlinken mittels angehangener GET-Variable auf eine individuell erzeugte Detailseite, die `projectdetail.php`. In der Tabelle nicht sichtbare Informationen werden auf dieser Seite offensichtlich, u.a. wird auch eine kleine Karte eingeblendet, in der die geografischen Punkte des Projektes angezeigt werden. Diese Karte wird wie für diese Arbeit üblich mittels `Leaflet` generiert. In der Abbildung 10 ist die Seite `projectdetail.php` beispielhaft gezeigt. Zu sehen ist ein fiktives Projekt der S-Bahn Dresden.

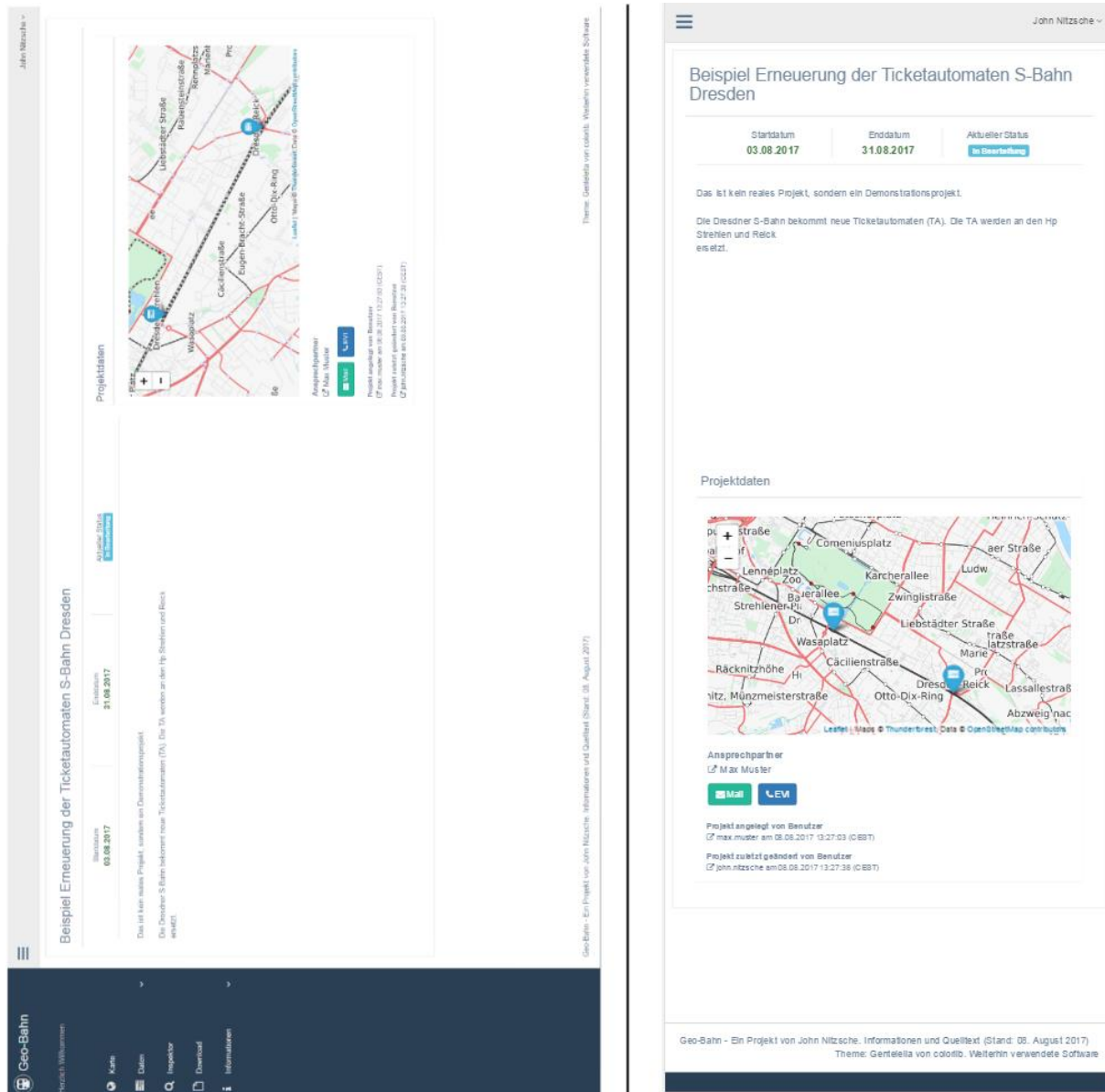


Abbildung 10: Ein Beispielprojekt mit Karte angezeigt in der projectdetail.php (links: PC-Monitor, rechts: Mobiltelefon)

Um Projekte erstellen und bearbeiten zu können, wird die Datei `createproject.php` verwendet. Über ein dynamisches Formular lassen sich mit dieser Datei Projekte erstellen. Durch das JavaScript-Plugin SmartWizard wird das Formular immer abschnittsweise eingeblendet, am Ende folgt eine Zusammenfassung. Eine Umfrage im kleinen Kreise ergab, dass das Einblenden des vollumfänglichen Formulars überfordernd wirke, da viele Informationen abgefragt werden. Gerade das Abfragen der geografischen Informationen wirkte da abschreckend. Das Einsetzen des beschriebenen Plugins verringerte dieses Phänomen. Zur Benutzerfreundlichkeit sollen auch verschiedene Wege beitragen, die geografischen Koordinaten der Projektpunkte einzupflegen. Es wird eine manuelle Dateneingabe ermöglicht, so können beispielsweise über Copy and Paste Koordinaten aus NeDocs eingetragen werden. Weiterhin ist es möglich, Koordinaten von Betriebsstellen und Streckenkilometern abzufragen und einzufügen. Bei der Abfrage der Streckenkilometer können zum Stand der Fertigstellung dieser Arbeit nur die vollen Streckenkilometer berücksichtigt werden, da nur diese geokodiert vorliegen. Eine Interpolation von nicht ganzzahligen Kilometern ist durch das Vorhandensein des Streckennetzes als Polyline prinzipiell möglich, zur Fertigstellung wurde auf diese jedoch relativ komplexe Berechnung verzichtet, da diese Funktion nicht zum Erfolg der Arbeit beiträgt. Das Problem wird als möglicherweise zukünftige Funktion angemerkt. Weiterhin ist es möglich, einen Marker mittels Klicks und Ziehen auf einen bestimmten Punkt auf einer Karte zu setzen. So kann die nicht umgesetzte Interpolation durch den Benutzer händisch durchgeführt werden. Neben den geografischen Informationen werden alle anderen projektrelevanten Informationen abgefragt.

Durch die Übergabe einer GET-Variable kann nach einer Berechtigungsabfrage ein Projekt über die Datei `createproject.php` auch bearbeitet werden. Dafür wird dasselbe Formular verwendet, welches mit Hilfe von PHP und JavaScript mit den jeweiligen Informationen vorausgefüllt wird. Nach Absenden des Formulars wird bei der Erstellung ein Eintrag hinterlegt, von welchem Benutzer zu welcher Zeit dieses Projekt erstellt bzw. zuletzt geändert wurde. Beide Informationen werden in der Projektdetailseite angezeigt. Für das Erschaffen einer Historie von Projekten wurden bereits grundlegende Steine gelegt, zur Fertigstellung dieser Arbeit ist jedoch nur der Zeitpunkt der letzten Bearbeitung, sowie des bearbeitenden Benutzers in der Datenbank hinterlegt.

6.3 Sicherheitstechnische Betrachtung der Software

Die nach Punkt 3.13 gestellten Sicherheitsanforderungen werden durch ein Login-System erfüllt, welches auf PHP-Sessions basiert. Das Benutzen der PHP-Sessions hat den Vorteil, dass jegliche Informationen lokal als Cookie gespeichert werden und diese über eine ID verschlüsselt werden. Ein Auswerten der Cookies auf einem Benutzer-PC ist somit genauso wenig möglich, wie das Manipulieren von Cookieinhalten. In jede aufrufbare PHP-Datei wird zu Beginn die Datei `auth.php` eingebunden. Das darin enthaltene Skript überprüft, ob bestimmte Session-Variablen gesetzt sind. Sollte das nicht der Fall sein, wird ohne jegliche Informationen zu generieren direkt auf die `login.php` weitergeleitet, mit einer Fehlermeldung, dass man nicht angemeldet sei. Die `login.php` ist das Anmeldeformular der Software, hier werden Benutzername und Passwort abgefragt. Die eingegeben Anmeldedaten werden verdeckt über PHP-Post an die Datei `login-task.php` versandt, wo die eingegeben Daten mit der Datenbank verglichen werden. Bei erfolgreicher Eingabe werden die Session-Variablen gesetzt, ein Betreten der Software ist erst ab diesem Moment möglich. Die Passwörter werden nach dem aktuell gängigen und als sicherheitstechnisch sicher bewertbaren Salt-Key-Verfahren verschlüsselt in der Datenbank abgelegt. Eine zusätzliche Maßnahme zur Verbesserung der Sicherheit ist das Einführen des HTTPS-Protokolls, welches die Kommunikation zwischen Server und Client verschlüsselt und so das System für Man-in-the-middle-Attacken immun macht. Diese Maßnahme ist eine administrative Aufgabe des eingesetzten Servers und somit nicht Bestandteil dieser Arbeit. Die vorliegende Testplattform ist über HTTP und HTTPS erreichbar und funktionsfähig. Das Einbinden externer Daten, in diesem Fall sind die Karten-Tiles ein gutes Beispiel, kann zu Problemen mit dem HTTPS-Protokoll führen.

Die Bearbeitung eines Projektes ist nur für drei Benutzergruppen möglich: Dem Administrator, der jegliche Benutzer- und Projektinformationen bearbeiten kann, dem Projektersteller, der immer die eigen erstellten Projekte bearbeiten darf und sich auch nicht ändern lässt, sowie der Benutzergruppe der Bearbeiter, die vom Projektersteller, Administrator oder anderen Bearbeitern berechtigt werden, bestimmte Projekte zu bearbeiten.

Projektersteller und -bearbeiter werden im jeweiligen Datensatz eines Projektes hinterlegt. Über PHP wird beim Aufruf einer Projektbearbeitungsseite abgefragt, ob sich

die in der PHP-Session hinterlegte Benutzer-ID in diesen Datensätzen wiederfindet, oder ob die Benutzerrolle (role), welche ebenfalls in der PHP-Session hinterlegt ist, auf 1 steht, was bedeutet, dass der eingeloggte Benutzer ein Administrator ist. Schlägen alle drei Tests fehl, wird das Skript abgebrochen und es wird die Fehlermeldung angezeigt, dass man nicht berechtigt sei. Geht man davon aus, dass der Datenbankzugriff gesichert ist, kann durch die nicht manipulierbare PHP-Session ein hoher Grad an Sicherheit für die Daten gewährleistet werden.

Datenbankverbindungen, sowie Datenbankparameter werden in der Software öfters über GET- und POST-Variablen angefordert. Das heißt, dass der Benutzer selbst Parameter zur SQL-Abfrage definiert. Sicherheitstechnisch bedenklich ist der Weg, diese Parameter unbearbeitet in die SQL-Abfrage einzuarbeiten. Als Beispiel gelten hier die Anforderungen von Informationen zu einer Strecke im Streckeninspektor:

```
SELECT * FROM " . $db_pref . "_alles WHERE strecke= $_GET["strecke"]  
ORDER BY km_i ASC;
```

Die hervorgehobene Variable zeigt die bedenkliche Variable. Über die Benutzereingabe der Strecke wird automatisch die Seite mit entsprechender GET-Variable aufgerufen. Beispielsweise wird bei der Eingabe der Strecke 6362 auf die Seite

`inspector.php?strecke=6362`

weitergeleitet. Die Datenbankabfrage von oben wird dann den Wert 6362 aus der URL unbearbeitet einbinden. Aus der Abfrage wird dann also

```
SELECT * FROM geo_alles WHERE strecke=6362 ORDER BY km_i ASC; .
```

Es muss sich jedoch immer die Frage gestellt werden, wie ein potentieller Angreifer diese Variante ausnutzen kann. Man stelle sich vor, der Angreifer würde manuell die Seite

`inspector.php?strecke=6362;UPDATE+geo_users+SET+role=1+WHERE+ID=22; .`

aufrufen. Die Folge eines solchen Aufrufes ist die Abänderung der Datenabfrage zu

```
SELECT * FROM geo_alles WHERE  
strecke=6362;UPDATE+geo_users+SET+role=1+WHERE+ID=22; ORDER BY km_i  
ASC; .
```

Der Angreifer kann mit einer solchen Abfrage unberechtigtweise Daten in der Datenbank manipulieren, im genannten Szenario ändert er die Rolle des Benutzers 22 zu einer Admin-Rolle, welche Vollzugriff auf jegliche Daten hat. Um ein solches Risiko zu unterbinden werden sogenannte prepared-Statements benutzt. Die Datenbankabfrage wird abgeändert zu

```
SELECT * FROM " . $db_pref . "_alles WHERE strecke= :u_uebergabe ORDER BY km_i ASC; .
```

Durch das Binden des hervorgehobenen Parameters durch den PHP-Befehl

```
bindParam(':uebergabe', $_GET['strecke']);
```

wird ausschließlich der Parameter mit der GET-Variable gebunden. Eine Abfrage wie im oben beschriebenen Angriffsszenario hat die Folge, dass in der Datenbank nach einer Strecke gesucht wird, welche die Eigenschaft

```
strecke=6362;UPDATE+geo_users+SET+role=1+WHERE+ID=22;
```

besitzt. Dabei wird der hinten angehängte Code nicht ausgeführt. Da keine Strecke mit der genannten Eigenschaft gefunden wird, wird kein Ergebnis angezeigt. Durch diese Vorgehensweise ist das System gegen SQL-Injections geschützt.

7 Installation und Inbetriebnahme der Software

Um die Software ausführen zu können, wird ein Webserver benötigt, der lokal in einem Intranet oder im Internet eingebunden sein kann, oder lokal auf einem PC ausgeführt werden kann. Alle Skripte liegen lokal vor, die Software funktioniert deswegen auch ohne Anbindung an das Internet. Der Client sollte über eine Anbindung zum Internet verfügen, damit dieser die Kacheln der geografischen Anwendungen zu sehen bekommt. Diese werden extern aus dem Internet heruntergeladen.

7.1 Anforderungen an die Informationstechnik

Der Webserver muss die Möglichkeit mitbringen, PHP-Skripte verarbeiten zu können. Beispiele für eine entsprechende Webserver-Software sind nginx und Apache HTTP Server. Getestet wurde die Software auf Apache HTTP Server mit den PHP-Versionen 5.6.31 und 7.1.2. Die Software verbraucht ca. 450 Megabyte Speicherplatz auf dem Webserver.

Neben dem Webserver mit PHP wird auch ein MySQL-Datenbankserver benötigt. Die Datenbank besitzt wegen den Geodaten im neu installierten Status eine Größe von ca. 23,5 Megabyte.

Der Benutzer benötigt einen aktuellen Webbrowser. Cookies und Javascript müssen unterstützt und eingeschaltet werden.

7.2 Installation der Software auf dem Server

Mittels FTP-Zugang oder Direktzugriff auf das Dateisystem muss der vollständige Inhalt des Programmes in das gewünschte Verzeichnis übertragen werden.

Nach der Installation müssen die Datenbankstruktur, sowie die Geodaten in die zu verwendende Datenbank eingepflegt werden. Dafür muss die sich im Verzeichnis `geodata\sql\` befindende Datei `zusammengefuehrt.sql` mittels phpMyAdmin, HeidiSQL oder anderen MySQL-Werkzeugen in die Datenbank importiert werden. Die entwickelte Software bietet die Möglichkeit, mehrere Instanzen in einer Datenbank installieren zu können. Um die Installationen unterscheiden zu können, werden Präfixe verwendet. Standardmäßig ist der Präfix „geo“, die Tabellennamen in der Datenbank haben also beispielsweise die Namen „geo_alles“ oder „geo_projekte“. Sind mehrere Installationen gewünscht, kann dieser Präfix durch einen beliebigen anderen Präfix ersetzt werden,

indem man die Tabellen händisch umbenennt und in der Datei `globalconfig.php` (wird im nächsten Abschnitt beschrieben) den Präfix der Software anpasst.

Nach Abschluss der Installation gilt es, einige globale Einstellungen vorzunehmen. In der Datei `globalconfig.php` müssen dafür die Variablen für die Datenbankverbindung angepasst werden. Die anzupassenden Variablen werden in der nachfolgenden Tabelle erläutert.

Variablenname	Beschreibung	Standardwert (XAMPP-Verbindung)
<code>\$projectxname</code>	Bezeichnung der Webseite. Wird in der Kopf- und Fußzeile angezeigt und kann nach Bedürfnissen angepasst werden	"Geo-Bahn"
<code>\$db_host</code>	Server-Adresse zur MySQL-Datenbank	'localhost'
<code>\$db_user</code>	Benutzername für die Datenbankverbindung	'root'
<code>\$db_pass</code>	Passwort für die Datenbankverbindung	' '
<code>\$db_name</code>	Datenbankname	'geobahndb'
<code>\$db_pref</code>	Tabellenpräfix, zu denen Instanz verbinden soll (s.o)	'geo'
<code>\$tf_apikey</code>	API-Key für Thunderforest-Maps	' ' (nicht funktionsfähig)

Tabelle 5: Konfigurationsmöglichkeiten der `globalconfig.php`

Für die Verwendung der Tiles vom Onlinekartendienst Thunderforest (Transportkarte) wird ein spezieller API-Key benötigt. Dieser lässt sich zu unterschiedlichen Konditionen auf der Webseite des Dienstes¹ erwerben oder kostenfrei beantragen.

8 Fazit

Zur Fertigstellung der Arbeit im August 2017 befindet sich die Software in einem ausführbaren Zustand. Die geforderten Funktionen können durchgeführt werden. Es wird jedoch kein Anspruch auf Fehlerfreiheit erhoben, noch kann eine Garantie gegeben werden, dass die Software in jedem Fall funktionstüchtig und sicher ist. Aufgrund der kurzen Zeitspanne musste auf eine Testphase in dieser Dokumentation verzichtet werden. Nichtsdestotrotz kann man die Software auf die Erfüllung der Anforderungen überprüfen.

8.1 Erfüllung der vorgegebenen Anforderungen

Die folgende Tabelle fasst kurz nochmals die in Punkt 3 gestellten Anforderungen zusammen. In der rechten Spalte wird die Erfüllung der Anforderungen bewertet.

¹ <http://thunderforest.com/>

Kurzbeschreibung der Anforderung	Erfüllung der Anforderung
Karte zu laufenden Projekten	erfüllt
Details zu Projekten über Popups o.ä.	erfüllt
Verwaltungssystem	erfüllt
optional: Historie	nicht erfüllt
optional: Geodaten der DB Netz AG	erfüllt
optional: Kompatibilität mit Daten der DB Netz AG	nicht erforderlich
Zeitverzögerung bei einer Anfrage maximal fünf Sekunden	nicht erfüllt (s. 6.1.1)
Plattformunabhängigkeit	erfüllt
Ungültige Eingaben müssen abgefangen werden (Stabilität muss gewährleistet sein)	erfüllt (konnte nicht in jedem Fall überprüft werden)
Standards	
Richtlinien/Vorschriften	
Benutzerfreundlichkeit	erfüllt
Dokumentation	erfüllt
Sicherheitsanforderungen	erfüllt

Tabelle 6: Erfüllen der in Punkt 3 gestellten Anforderungen

Wie die Tabelle 6 zeigt, konnten nicht alle Anforderungen erfüllt werden. Nicht zufriedenstellend ist die Performanceleistung der Karte. Die Zeitverzögerung beim Laden der Karte dauert länger als die geforderte maximale Ladezeit von fünf Sekunden. Zudem ist während dieser Ladezeitverzögerung nicht erkennbar, ob das System gerade Daten errechnet oder lädt. Das führt zu Verwirrungen des Benutzers, der eventuell in diesem Fall von einem Defekt ausgeht. Dieser Effekt hat negative Auswirkungen auf die Benutzerfreundlichkeit.

Das Schaffen einer Historie war eine optionale Anforderung. Durch die erläuterte Datenbankstruktur wurde bereits eine Möglichkeit geschaffen, die letzte Bearbeitung eines Projektes zu erfassen. Die Registrierung sämtlicher Änderungen bedarf deswegen nicht viel Programmieraufwand und gilt als ein nächstes umsetzbares Ziel.

Neben den beiden nicht umgesetzten Anforderungen konnten die verbleibenden Ziele umgesetzt und überprüft werden. Auf die Überprüfung der Stabilität des Systems bei Falscheingaben ist aufgrund der kurzen Bearbeitungszeit verzichtet worden. Durch die Benutzung von prepared-Statements (vgl. 6.3) bei Datenbankverbindungen wurde ein grundlegender Beitrag zur Betriebsstabilität bei Falscheingaben geleistet.

8.2 Ausblick

Das Projekt ist voll funktionsfähig und kann durch die relativ neutralen Daten der DB Netz AG auch in anderen bahnspezifischen Bereichen, als der Telekommunikation eingesetzt werden.

Die bisher nicht umgesetzten Anforderungen in Augenschein zu nehmen, stellt eine weitere Chance für das Projekt dar. Weiterhin wurde oft nach einer tiefergehenden geografischen Verarbeitung gefragt. Beispielsweise würde das Einfügen der dynamischen Projektdaten in den statischen Inspektor eine große Erleichterung nach sich ziehen. Generell kann man neben dem bloßen Setzen der Projektpunkte eine weitergehende geografische Auswertung umsetzen. Vorstellbar wäre das Auswerten der Koordinaten, um relevante Projekte in der Nähe der eigenen Projekte automatisiert anzeigen zu lassen. Ein Benachrichtigungssystem über solche Projekte würde tiefergehend die Vermeidung von Missverständnissen vorantreiben.

Das Einbinden weiterer öffentlicher oder geschlossener Daten wäre zudem vorstell- und machbar. Die Standorte von GSM-R-Funkmasten gelten beispielsweise als sehr interessante Punkte, zu denen es auch nicht öffentliche Daten gibt. Einbindbar wären zudem die Standorte von Ticketautomaten und Entwertertechnik, die von der OSM-Community auch gepflegt werden. Ob die Deutsche Bahn AG entsprechende offizielle Geodaten pflegt ist nicht bekannt.

Ebenfalls nicht umgesetzt wurde die Möglichkeit projektrelevante Dateien in dem System einzupflegen. Die verwendeten Mittel zum Erstellen dieser Software bringen solche Funktionen bereits mit, jedoch ist das Verwalten von Dateien ein relativ hoher Programmier- und Administrationsaufwand. Zudem erlaubt es nicht jeder Webserver, in das Dateisystem einzugreifen. Damit es nicht zu Fehlermeldungen oder Abstürzen kommt, muss ein entsprechender Mechanismus geschaffen werden, der ein solches Dateiverwaltungssystem im Zweifel abstellt

9 Literaturverzeichnis

Bundesamt für Sicherheit in der Informationstechnik. (2016). Erstellung eines Anforderungskatalogs für Standardsoftware. In I. Münch, *IT-Grundschutz-Kataloge, 15. Ergänzungslieferung* (S. 1633-1643). Bonn: Bundesanzeiger-Verlag.

colorlib. (kein Datum). *Gentelella Alela!* Abgerufen am 3. Juli 2017 von colorlib.com: <https://colorlib.com/polygon/gentelella/index.html>

diverse Autoren. (kein Datum). *Should I use OpenLayers or Leaflet?* Abgerufen am 9. August 2017 von Stackexchange: <https://gis.stackexchange.com/questions/33918/should-i-use-openlayers-or-leaflet>

Google Inc. (kein Datum). *Google JavaScript API*. Abgerufen am 7. Juni 2017 von Google Developers: <https://developers.google.com/maps/documentation/javascript/?hl=de>

Ihlenfeld, J. (26. August 2013). *Zugradar zeigt Züge fast in Echtzeit an*. Abgerufen am 16. Mai 2017 von golem.de: <https://www.golem.de/news/deutsche-bahn-zugradar-zeigt-zuege-fast-in-echtzeit-an-1308-101183.html>

McBride, B. R. (1. Juni 2015). *PHP-Database-GeoJSON*. Abgerufen am 19. Juni 2017 von github.com: <https://github.com/bmcbride/PHP-Database-GeoJSON>

OpenStreetMap Software contributors. (3. Januar 2017). *INSTALL.md*. Abgerufen am 2. Juni 2017 von OpenStreetMap on Github.com: <https://github.com/openstreetmap/openstreetmap-website/blob/master/INSTALL.md>

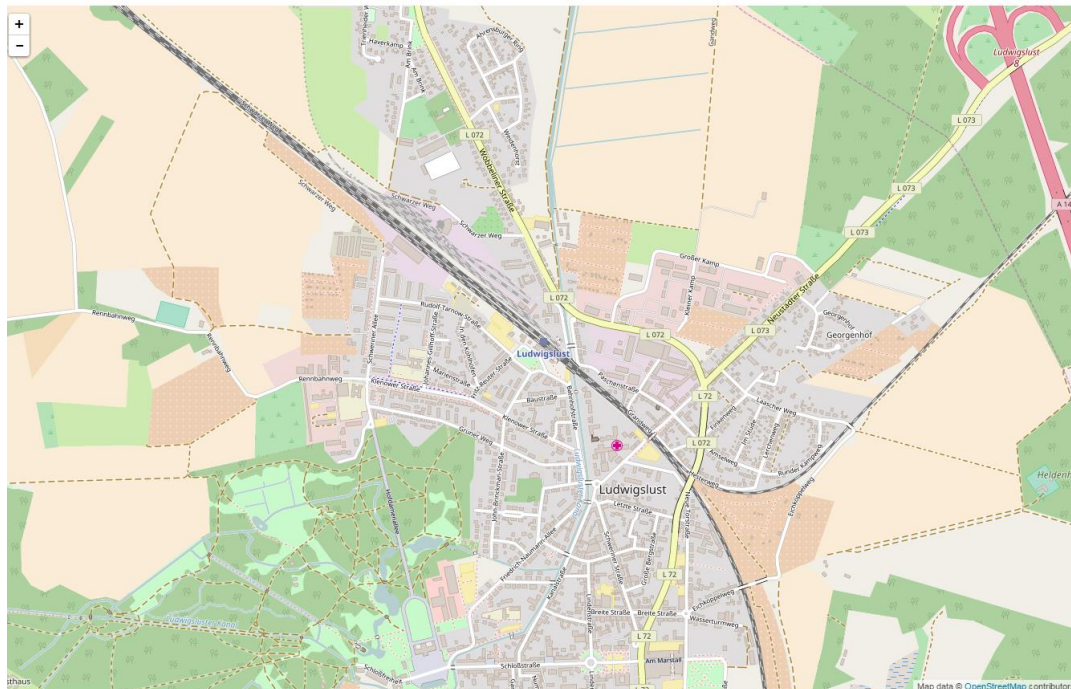
OpenStreetMap Wiki contributors. (26. März 2017). *Tile Servers*. Abgerufen am 2. Juni 2017 von OpenStreetMap Wiki: http://wiki.openstreetmap.org/w/index.php?title=Tile_servers&oldid=1451818

Ortega, I. S. (14. Dezember 2015). *Leaflet vs OL3*. Abgerufen am 8. Juni 2017 von github.io: <http://ivansanchez.github.io/leaflet-vs-openlayers-slides/#/>

Anlagen

Anlage 1: Beispielprojekte mit Datensätzen und Schnittstellen der Deutschen Bahn AG

Beispiel 1: BahnRoulette



BahnRoulette

Das Projekt ist abgeschlossen und das Ergebnis gibt es [hier zum Download](#). Vielen Dank an alle Helfer!

Bahnhofname:
Ludwigslust

Adresse:
**Rudolf-Tarnow-Str. 1
19288 Ludwigslust
Mecklenburg-Vorpommern**

DB100:
WL

Gefundener Kartenausschnitt:
DB ServiceStore, 1, Rudolf-Tarnow-Straße, Georgenhof, Forsthaus, Ludwigslust, Landkreis Ludwigslust-Parchim, Mecklenburg-Vorpommern, 19288, Deutschland
[Ausschnitt in OpenStreetMap öffnen](#)

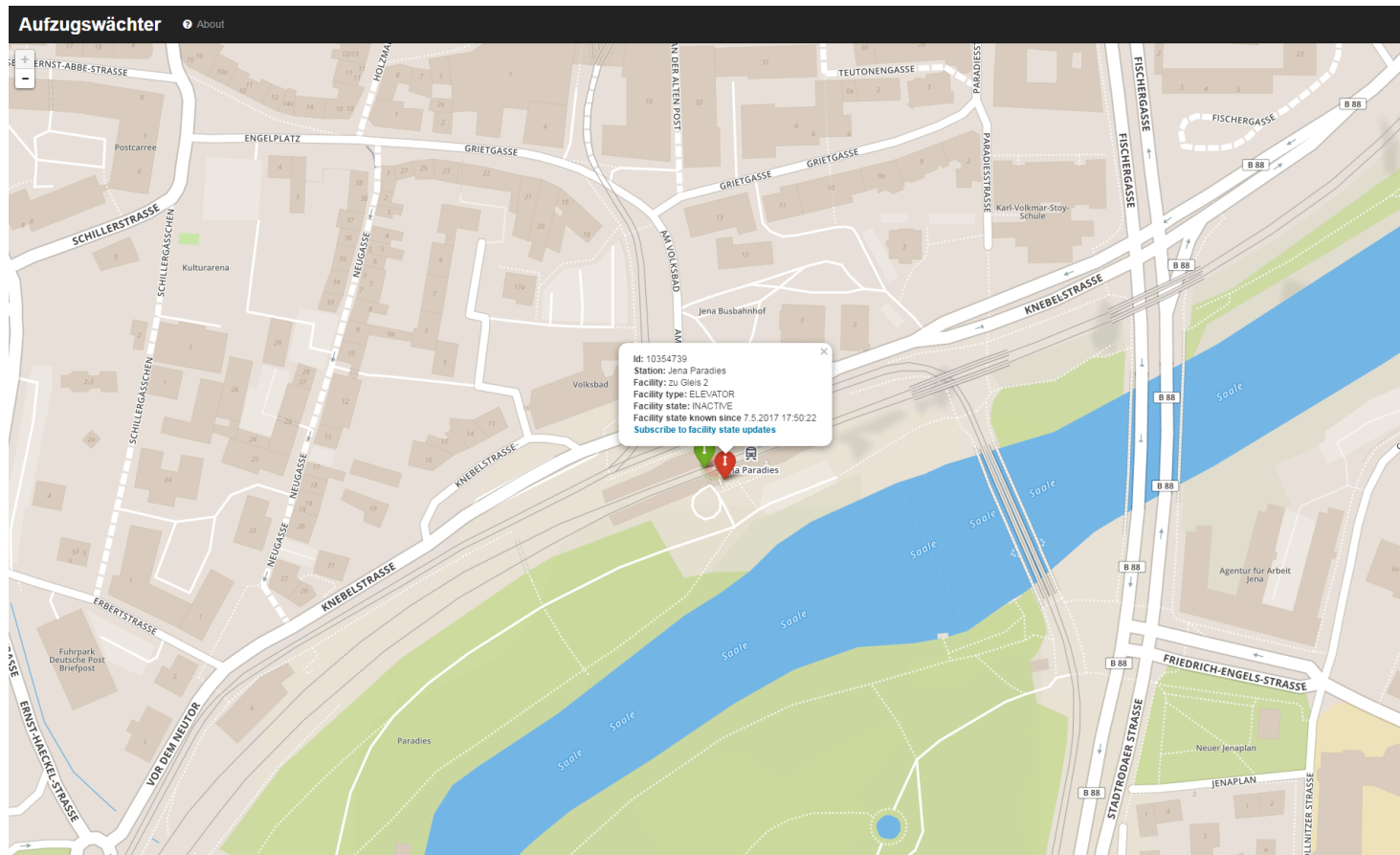
1+ heißt, dass diesen Stationen mindestens einmal eine Koordinate zugeordnet wurde. Um so häufiger eine Zuordnung statt findet, um so Fehleroleranter wird das Endergebnis.

Map data © OpenStreetMap contributors

Details zu BahnRoulette:

- Idee: Abgleich von OSM-Daten mit Daten von DB Netz AG
- Umsetzung: Leaflet (Laut Quellcode)
- Autor: Constantin Müller
- Lizenz: CC-BY
- Showcase-URL: <http://data.deutschebahn.com/showcase/bahnroulette>
- URL: <http://blattspinat.com/index.php/posts/bahnroulette> (aufgerufen und aufgenommen: 16. Mai 2017)

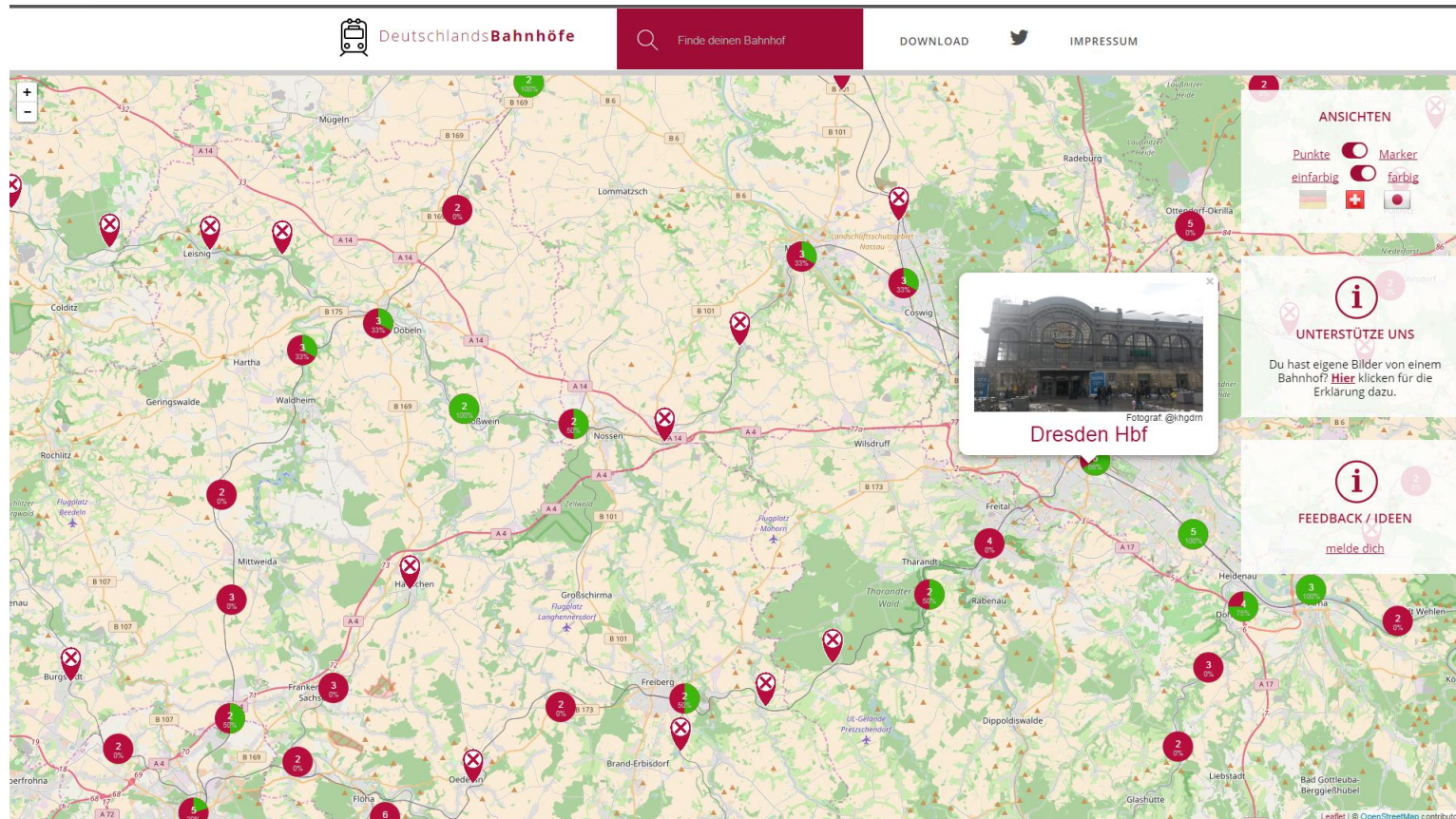
Beispiel 2: Aufzugwächter



Details zu Aufzugwächter:

- Idee: Aktuelle Störungen/Ausfälle von Aufzügen mit Echtzeitdaten von DB Station&Service AG
- Umsetzung: Leaflet (Laut Quellcode)
- Autor: Alexey Valikov (DB Systel GmbH)
- Lizenz: BSD 2-clause "Simplified" License
- Showcase-URL: <http://data.deutschebahn.com/showcase/aufzugswaechter>
- URL: <http://www.aufzugswaechter.org> (aufgerufen und aufgenommen: 16. Mai 2017)

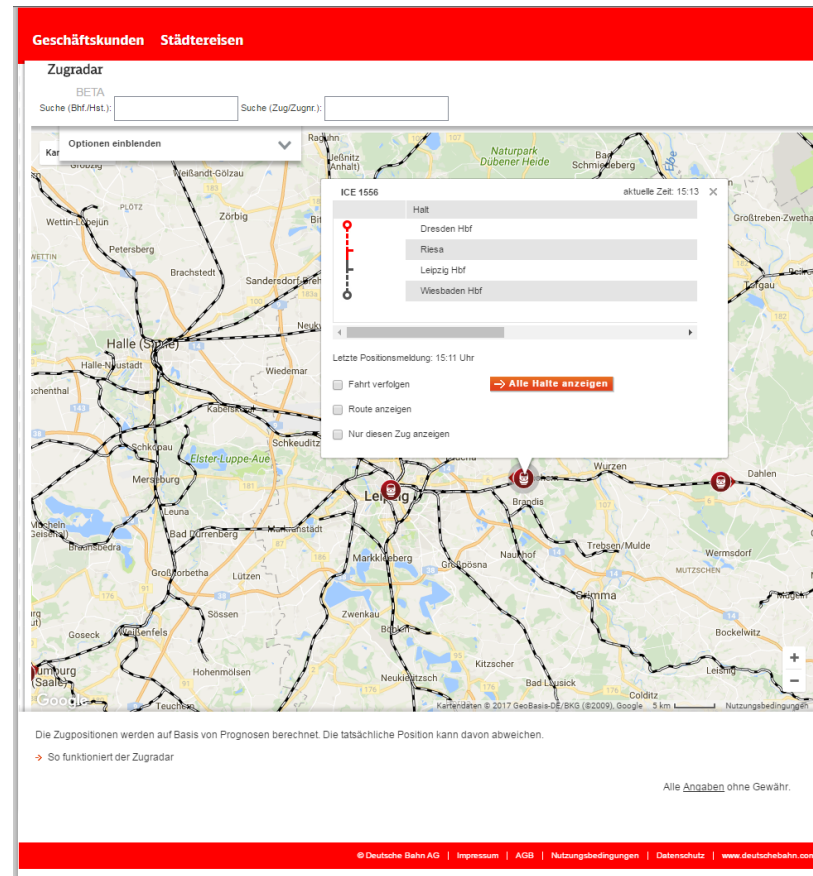
Beispiel 3: Deutschlands Bahnhöfe



Details zu Deutschlands Bahnhöfe:

- Idee: Sammlung von Echtzeitdaten, Fotos und Daten von Bahnhöfen in Deutschland
- Umsetzung: Leaflet
- Autor: Gruppe webgrrls.de
- Lizenz: keine
- Showcase-URL: <http://data.deutschebahn.com/showcase/deutschlands-bahnhofe>
- URL: <http://www.deutschlands-bahnhofe.de> (aufgerufen und aufgenommen: 16. Mai 2017)

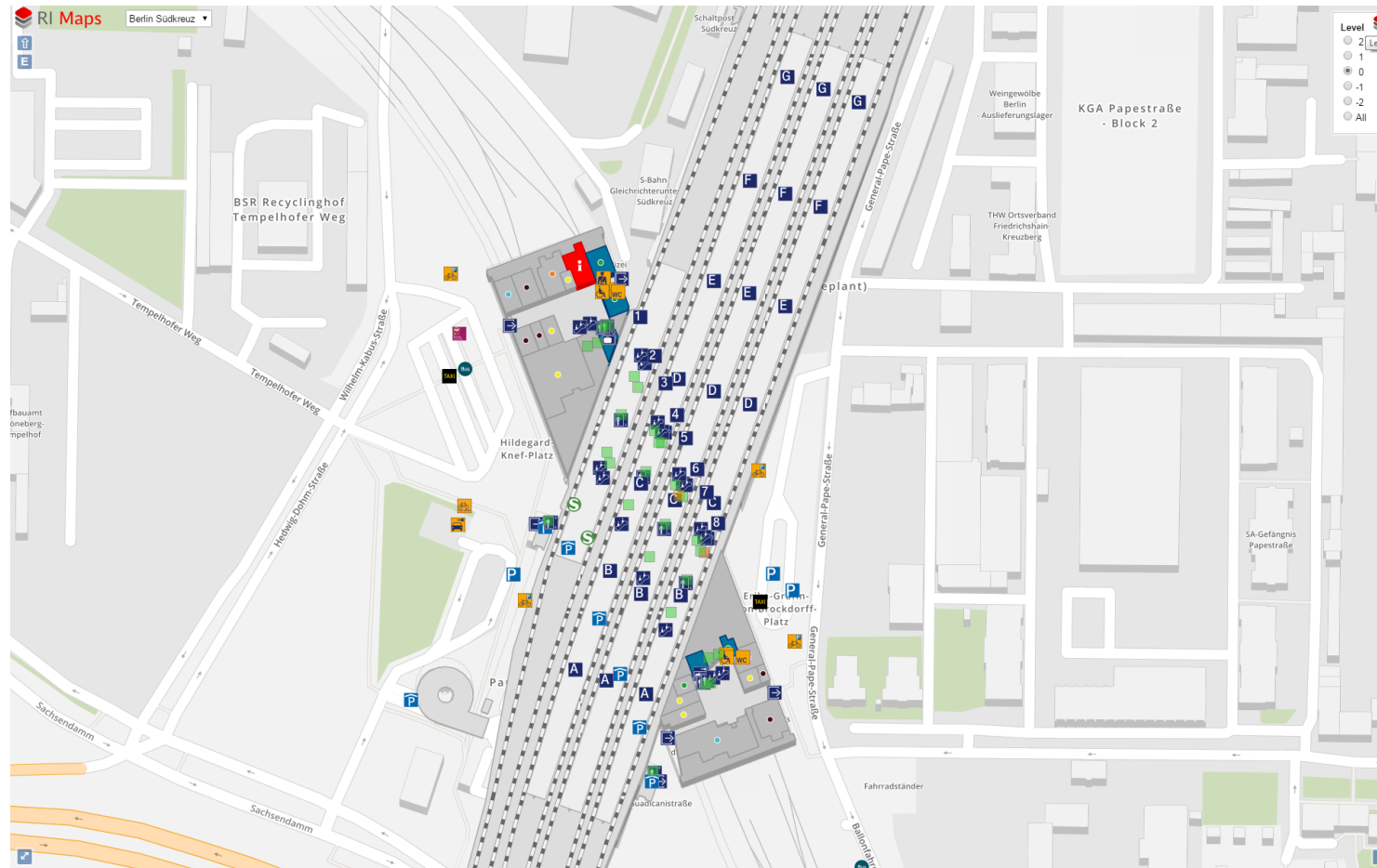
Beispiel 4: DB Zugradar



Details zu DB Zugradar:

- Idee: Anzeige der aktuellen Position von Fern- und Regionverkehrszügen der DB AG in Echtzeit
- Umsetzung: Google Maps API (Laut Quellcode)
- Autor: DB AG (keine genaueren Angaben vorhanden)
- Lizenz: keine
- URL: <http://bahn.de/zugradar> (aufgerufen und aufgenommen: 16. Mai 2017)

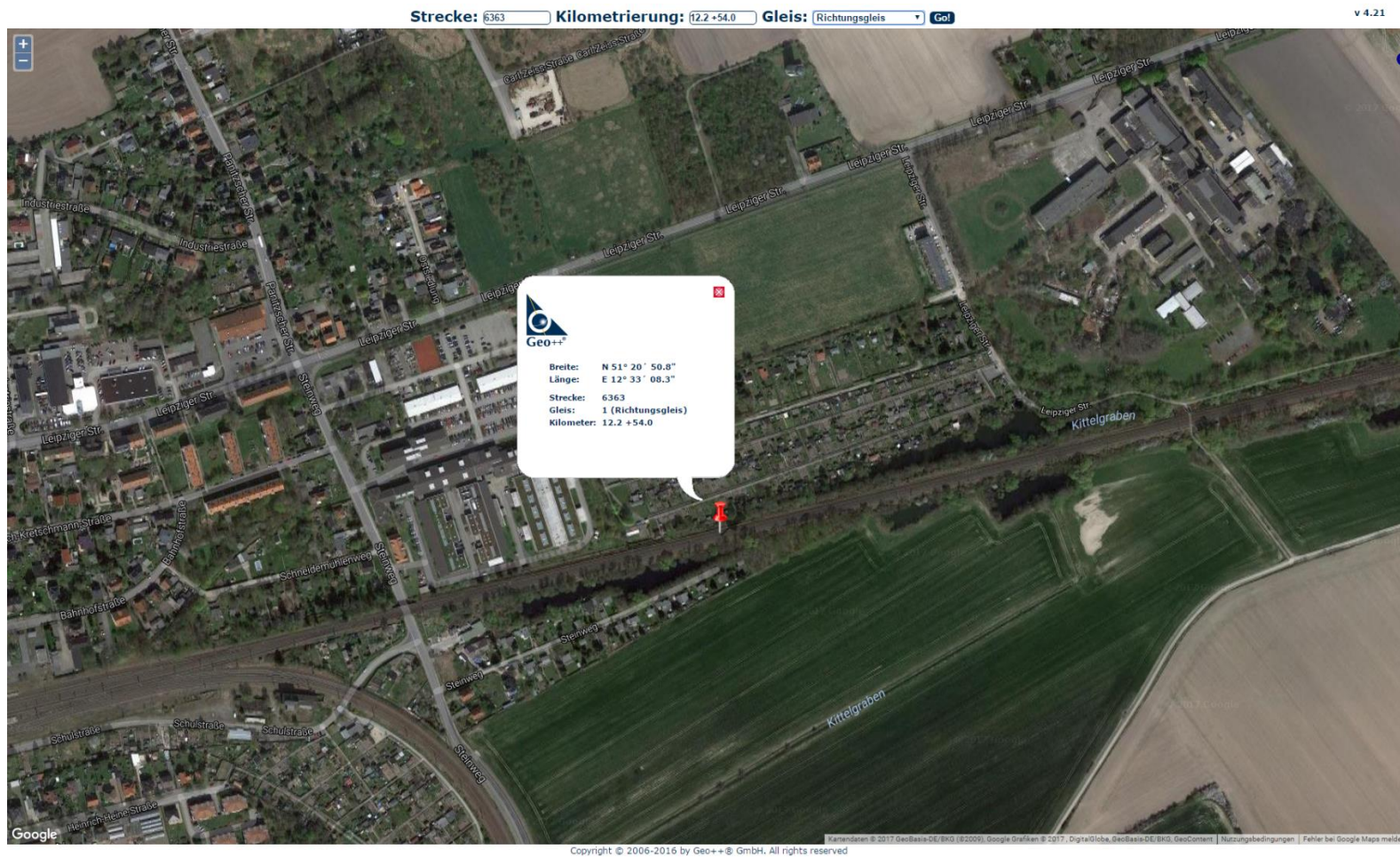
Beispiel 6: RI Maps



Details zu RI Maps:

- Idee: Übersichtspläne mit POIs in einer interaktiven Karte
- Umsetzung: OpenLayers
- Autor: VXR
- Lizenz: keine
- URL: <https://maps.reisendeninfo.aws.db.de/> (aufgerufen und aufgenommen: 17. Mai 2017, nicht öffentlich zugänglich)

Beispiel 5: GNRaiLNav



Details zu GNRaiLNav:

- Idee: Anzeige von Standorten auf Eisenbahnstrecken anhand von Streckennummer und Streckenkilometer
- Umsetzung: OpenLayers, Google Maps?
- Autor: Geo++ GmbH
- Lizenz: keine
- URL: http://db.geopp.de/gnrailnav_servlet/GNOpenLayersV3 (aufgerufen und aufgenommen: 23. Mai 2017)

Anlage 2: Dresden Hauptbahnhof Gleis 19 in OSM exportiert als XML (Stand 25. August 2017)

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.6.0 (28297 thorn-
01.openstreetmap.org)" copyright="OpenStreetMap and contributors"
attribution="http://www.openstreetmap.org/copyright"
license="http://opendatacommons.org/licenses/odbl/1-0/">
  <way id="31152599" visible="true" version="26"
changeset="49755317" timestamp="2017-06-22T20:53:12Z" user="Nakaner-
repair" uid="2149259">
    <nd ref="346633864"/>
    <nd ref="2375515445"/>
    <nd ref="3811423417"/>
    <nd ref="1417195470"/>
    <nd ref="3732032691"/>
    <nd ref="2375515458"/>
    <tag k="electrified" v="contact_line"/>
    <tag k="frequency" v="16.7"/>
    <tag k="gauge" v="1435"/>
    <tag k="highspeed" v="no"/>
    <tag k="layer" v="1"/>
    <tag k="level" v="1"/>
    <tag k="maxspeed" v="100"/>
    <tag k="operator" v="DB Netz AG"/>
    <tag k="passenger_lines" v="2"/>
    <tag k="railway" v="rail"/>
    <tag k="railway:etcs" v="no"/>
    <tag k="railway:gnt" v="no"/>
    <tag k="railway:lzb" v="no"/>
    <tag k="railway:pzb" v="yes"/>
    <tag k="railway:radio" v="gsm-r"/>
    <tag k="railway:tilting" v="no"/>
    <tag k="railway:track_ref" v="293"/>
    <tag k="railway:traffic_mode" v="passenger"/>
    <tag k="ref" v="6239"/>
    <tag k="source:maxspeed" v="Esig B Zs3 &quot;10&quot;"/>
    <tag k="usage" v="main"/>
    <tag k="voltage" v="15000"/>
  </way>
</osm>
```

Quelle: <https://www.openstreetmap.org/api/0.6/way/31152599> [abgerufen 25. August 2017 10:14 MESZ]

Anlage 3: Dresden Hauptbahnhof Gleis 19 in OSM konvertiert zu GeoJSON (Stand: 25. August 2017)

```
{
  "type": "Feature",
  "properties": {
    "electrified": "contact_line",
    "frequency": "16.7",
    "gauge": "1435",
    "highspeed": "no",
    "layer": "1",
    "level": "1",
    "maxspeed": "100",
    "operator": "DB Netz AG",
    "passenger_lines": "2",
    "railway": "rail",
    "railway:etcs": "no",
    "railway:gnt": "no",
    "railway:lzb": "no",
    "railway:pzb": "yes",
    "railway:radio": "gsm-r",
    "railway:tilting": "no",
    "railway:track_ref": "293",
    "railway:traffic_mode": "passenger",
    "ref": "6239",
    "source:maxspeed": "Esig B Zs3 \"10\"",
    "usage": "main",
    "voltage": "15000"
  },
  "geometry": {
    "type": "LineString",
    "coordinates": [
      [
        13.7335188,
        51.0403574
      ],
      [
        13.7332365,
        51.0404491
      ],
      [
        13.7321195,
        51.0408143
      ],
      [
        13.7319157,
        51.0408809
      ],
      [
        13.7316426,
        51.0409695
      ],
      [
        13.7303496,
        51.0413892
      ]
    ]
  ]
}
```

Quelle: <https://www.openstreetmap.org/api/0.6/way/31152599> [abgerufen 25. August 2017 10:14 MESZ]

Anlage 4: Verweise auf den Quellcode und Demonstrationsexemplar dieser Arbeit

Der Quellcode der Software, sowie die Struktur der und Inhalte der Datenbanken sind auf dem elektronischen Datenträger zu dieser Arbeit zu finden. Der Stand dieser Daten gleicht dem Stand zum Abgabetermin.

Der **Quellcode** der Software ist frei unter der MIT-Lizenz veröffentlicht. Der Quellcode kann unter der URL

<https://github.com/wlanowski/wlanowski.github.io>

abgerufen werden.

Die **Dokumentation** dieser Arbeit befindet sich unter der folgenden URL:

<https://github.com/wlanowski/t2000doku>

Eine **Demonstrationsinstallation** mit fiktiven Projektdaten ist unter der URL

<http://maps.wlanowski.de>

auffindbar. Die Zugangsdaten für einen Benutzer mit eingeschränkten Rechten lauten:

Benutzername: max.muster

Passwort: qay

Das Kontrollzentrum mit erweiterten Rechten kann mit den Zugangsdaten

Benutzername: john.nitzsche

Passwort: qay

eingesehen werden.