HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Sommersemester 2020

Design und Entwicklung eines webbasierten Geoinformationssystems zur Visualisierung und Analyse von Volunteered Geographic Information

Im Auftrag des NorOvA-Forschungsprojektes

Thesis

zur Erlangung des Grades
Bachelor of Science
im Studiengang Wirtschaftsinformatik
an der Fakultät Wirtschaftsinformatik
der Hochschule Furtwangen University

vorgelegt von

Alexander Merz
256548

Referenten:	Prof. Dr. Oliver Taminé
	Prof. Dr. Ulf Schreier
Eingereicht am	31. August 2020

Abstract

Das stetige Wachstum der Weltbevölkerung, die Zunahme von internationalen Verflechtungen durch die Globalisierungen und der daraus resultierende Mehrverkehr und erhöhte CO2-Ausstoß sind Auslöser eines drastischen Umdenkens in Politik, Forschung und Industrie in den letzten Jahrzehnten. Die Sicherstellung einer nachhaltigen Mobilität durch eine gezielte eine Optimierung des öffentlichen Personennahverkehrs ist ein erster Schritt zur Reduktion einer hohen Fahrzeug-pro-Kopf-Quote in Industrieländern und letztlich zur Eindämmung der Umweltverschmutzung durch Personenkraftwägen.

Das regionale Förderprogramm InKoMo 4.0 adressiert Innovationspartnerschaften zwischen Kommunen und Mobilitätswirtschaften in Baden-Württemberg. Das Ziel ist dabei die Umsetzung von vernetzten, digitalen und intelligenten Mobilitätslösungen. Das Forschungsprojekt NorOvA ist eine Partnerschaft der Hochschule Furtwangen und dem Verkehrsbund Schwarzwald-Baar zur Ermittlung von Mobilitätsbedürfnissen von Studenten im ländlichen Raum durch die Analyse von freiwillig erhobenen und zur Verfügung gestellten Mobilitätsdaten, sog. Volunteered Geographic Information (VGI).

Die vorliegende Thesis befasst sich mit der VGI-konformen Datenakquisition über eine hochschuleigene Android-App, der Exploration der Datenbasis und die Vorbereitung der Daten für Analysezwecke. Der Schwerpunkt der Arbeit liegt in dem Design und der Entwicklung eines webbasierten geografischen Informationssystems (Web GIS) für eine Visualisierung von Mobilitätsdaten. Die Ergebnisse der Datenanalyse sind die Entscheidungsgrundlage des VSB, um Optimierungen am ÖPNV vorzunehmen.

Eidesstattliche Erklärung

Ich, Alexander Merz, erkläre hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit selbständig und ohne unzulässige fremde Hilfe angefertigt habe.

Die verwendeten Quellen sind vollständig zitiert.

Furtwangen, den 31. August 2020

gez. Alexander Merz

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
DOM	Document Object Model
ER	Entity Relationship
ERM	Entity Relationship Model
ES5/6	ECMAScript Version 5/6
GIS	Geographic Information System
GPS	Global Positioning System
HFU	Hochschule Furtwangen University
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IEC	International Electrotechnical Commission
ISO	International Organization of Standardization
I/O	Input/Output
JSON	JavaScript Object Notation
JSX	JavaScript XML
MIME	Multipurpose Internet Mail Extension
MobiArch	Mobilitätsdatenarchitektur für innovative Anwendungen
NorOvA	Nutzer orientierte Optimierung verkehrlicher Angebote
NPM	Node Package Manager
OO	Objektorientierung/Objektorientierte
OS	Operating System
PC	Personal Computer
PPGIS	Public Participitory GIS
REST	Representational State Transfer
RPC	Remote Procedure Call
SPA	Single-Page-Applikation
SSL	Secure Socket Layer
SWEBOK	Software Engineering Body of Knowledge
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UML	Unified Modelling Language
URI	Unifrom Resource Indentifier
URL	Uniform Resource Locator
VGI	Volunteered Geographic Information
VSB	Verkehrsbund Schwarzwald-Baar
WWW	World Wide Web
XP	Extreme Programming
ÖPNV	Öffentlicher Personennahverkehr

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Die Motivation dieser Abschlussarbeit liegt darin, Konzepte und Methoden aus den Gebieten der Webentwicklung und Big Data zu vereinen und Forschung hinsichtlich einer webgestützten Analyse von Geodaten zu betreiben. Die Datenanalyse ist seit geraumer Zeit eine Kerndisziplin der Wirtschaftsinformatik.

Der in dem Fachgebiet diskutierte analytische Mehrwert zeitbezogener Geodaten und die Erhebungsstrategien dieser Daten bilden die Ausgangslage. Disziplinen des Software Engineerings werden angewandt, um die im Rahmen dieser Arbeit entworfene Software durch Standardnotationen abzubilden.

Ziel ist es, die Design- und Entwicklungsphase der Software zu veranschaulichen, das Endprodukt vorzustellen und in die Fachliteratur einzuordnen.

1.2 Projektbeschreibung

Die Analysesoftware NorOvA Analytics (vgl. Kapitel 1.2.3) ist Bestandteil eines geplanten digitalen Werkzeugkastens des NorOvA-Forschungsprojektes (vgl. Kapitel 1.2.2) der Hochschule Furtwangen University und reiht sich am Ende einer Vielzahl von Projekten an, ausgehend von einem im Jahr 2018 gestarteten Förderprogramms zur Entwicklung digitaler Mobilitätslösungen.

1.2.1 Förderprogramm MobiArch BW

„Sieben Projekte zur digitalen Mobilität werden im Rahmen der Förderlinie MobiArch BW [...] umgesetzt“ (Ministerium für Verkehr Baden-Württemberg, 2019) heißt es in einer Pressemitteilung vom 19. Juni 2019. Finanziert werden die Projekte durch das Programm Innovationspartnerschaften zwischen Kommunen und Mobilitätswirtschaft 4.0[1]. Das Programm wurde im April 2018 mit dem Ziel gegründet, Partnerschaften zwischen Kommunen und Mobilitätswirtschaften (Unternehmen, Start-Ups, öffentliche Institutionen und Bildungseinrichtungen) zu initiieren und zu fördern. Das Verkehrsministerium von Baden-Württemberg unterstützt sieben ausgewählte Projekte der Förderlinie MobiArch BW (Mobilitätsdatenarchitektur für innovative Anwendungen Baden-Württemberg) über eine Laufzeit von drei Jahren mit einem Budget von insgesamt 1,3 Millionen Euro. In derselben Pressemitteilung betont Verkehrsminister Winfried Herrmann, dass die Projekte den „Open-Data-

[1] vgl. hierzu die Webseite des Förderprogramms: https://inkomo-bw.de/foerderung

Gedanken“ verfolgen, denn „Daten sind eine Grundvoraussetzung für digitale Mobilität“ (Ministerium für Verkehr Baden-Württemberg, 2019). Hinter dieser Aussage steckt die Intention, die akquirierten Daten zu einer gemeinsamen Datenbasis zusammenzuführen, um eine digitale Entscheidungsgrundlage aufzubauen.

1.2.2 Forschungsprojekt NorOvA

Abbildung 1-1: NorOvA-Projektteilnehmer der HFU und des VSB
Quelle: NorOvA

Das Forschungsprojekt „Nutzer orientierte Optimierung verkehrlicher Angebote“, kurz: NorOvA, ist eine auf zwei Jahre angesetzte Partnerschaft der Hochschule Furtwangen (HFU) und dem Verkehrsbund Schwarzwald-Baar (VSB). Die Kooperation verfolgt das Ziel die steigenden Mobilitätsbedürfnisse von Hochschulangehörigen mit weniger Verkehr abzuwickeln, indem die Mobilität bestimmter Zielgruppen erfasst, analysiert und für Verbesserungsvorschläge an das Verkehrsministerium aufbereitet wird. Diese Maßnahmen sollen den öffentlichen Personennahverkehr im ländlichen Raum Baden-Württembergs nachhaltig optimieren, damit Hochschulangehörige zukünftig auf ein eigenes Verkehrsmittel verzichten können.

Zunächst werden Mobilitätsbedürfnisse durch eine hochschuleigene Android-App erfasst. Der App-Nutzer entscheidet, wann und wie lange er getrackt werden möchte. Somit können Hin- und Rückweg zur und von der Hochschule aufgezeichnet werden, die in Kombination mit Ankunfts- und Abfahrtszeiten wertvolle Informationen für das Projekt liefern. Die Hochschule setzt dabei auf das Engagement der Studierenden und versucht mit Gewinnspielen und Preisen, wie die VSB-Studentencard, zur Teilnahme anzuregen. Das Verkehrsministerium validiert die Rückschlüsse der Analyse und zieht ggf. Veränderungen des Verkehrsbetriebes in Betracht.

Die App ist Bestandteil eines digitalen Werkzeugkastens und wurde im Wintersemester 2019 erstmals an der HFU erprobt, wenig später im Playstore angeboten und befindet sich seitdem in ständiger Weiterentwicklung. Im Sommersemester 2020 sind App-Features wie eine Verkehrsmittelerkennung (Modal-Split), ein Wegetagebuch und ein tagesaktuelles Ranking geplant. Ende Juni 2020 erhielt die App ein komplett überarbeitetes Design (siehe Abbildung 1-2).

Abbildung 1-2: Benutzeroberfläche der NorOvA-App
Quelle: NorOvA

1.2.3 Abschlussarbeit NorOvA Analytics

NorOvA Analytics ist die technische Grundlage dieser Arbeit und entstand nach Rücksprache mit NorOvA-Projektleiter Oliver Taminé. Nach bestätigter Förderung vom Verkehrsministerium und Veröffentlichung der Android-App stand das Projektteam vor der Herausforderung den rapid wachsenden Datenbestand einer effektiven Datenanalyse zu unterziehen. Die Software NorOvA Analytics setzt an genau diesem Punkt an und befasst sich mit einer ausgereiften Visualisierung und deckt zahlreiche Analysemöglichkeiten ab. Die Software ist fähig, zurückgelegte Strecken tabellarisch und auf einer interaktiven Karte abzubilden, die Ergebnisse zu sortieren und nach Zeit und Raum zu filtern. Das Hauptaugenmerk liegt dabei auf einer umfangreichen Routenanalyse. Anhand von frei wählbaren Start- und Endpunkten werden übereinstimmende Streckenaufzeichnungen abgebildet und über einen integrierten Kalender in einen zeitlichen Kontext gebracht. NorOvA Analytics kann Stand-Alone oder als eine Komponente in anderen Projekten eingesetzt werden.

1.3 Aufbau der Thesis

Die Thesis ist in insgesamt fünf Abschnitte gegliedert.

Den Einstieg bilden die technischen Grundlagen. Sie sind essenziell, um das Hintergrundwissen für die Folgekapitel zu vermitteln. Das Internet als Anwendungsplattform, die neuartige Kommunikation von verteilten Systemen und das Potenzial eines Web 2.0 zur Kollaboration werden unter den Gesichtspunkten der Webentwicklung näher beleuchtet. Crowdsourcing, Crowdcasting und VGI sind Konzepte der Erhebung von großen Datenmengen. Sie schließen das Grundlagenkapitel ab und vereinfachen den Übergang zur der Datenarchitektur von NorOvA Analytics.

Die Datenarchitektur umfasst die Datenakquise in der Android-App, die Modellierung von Beziehungen durch ein ER-Modell, den relationalen Datenbankentwurf und die Beschreibung von Datenfluss und Schnittstellen. Der Datenfluss wird von der Erhebung bis zur Verwertung in NorOvA Analytics veranschaulicht.

Das Hauptaugenmerk der Abschlussarbeit liegt in der Beschreibung von Design- und Entwicklungsprozess von NorOvA Analytics. Anforderungsanalyse, Design und Entwicklung sind Disziplinen des Software Engineerings und geben einen Einblick in die Umsetzung und das Innenleben der Webanwendung. Ausgangspunkt für die Aufteilung und Ablaufbeschreibung des Programms sind Designprobleme und die davon abgeleiteten Designentscheidungen. Mithilfe von UML wird die Software abgebildet.

Abschließend wird der Funktionsumfang des Endproduktes vorgestellt und in die Fachliteratur eingeordnet.

2 Technische Grundlagen

Das World Wide Web, folglich als Web abgekürzt, ist eine in den 1990er Jahren entstandene Initiative, Informationen durch den gezielten Einsatz von Technologien weltweit zugänglich zu machen (vgl. Berners-Lee, et al., 1992). Web-Mitgründer Tim Berners-Lee entwarf 1992 das W3-Modell. Das Modell stützt sich auf zwei komplementäre Paradigmen: Verlinkungen via Hypertext und indizierte Suchen, auch unter dem Namen Information Retrieval[2] bekannt. Es ist vorgesehen, dass der Nutzer über Hyperlinks Ressourcen bezieht und mithilfe von Texteingaben die Ergebnismenge relevanter Dokumente einschränken kann.

Zur damaligen Zeit barg das Web ungeahnte Potenziale. Im Gegensatz zum Buchdruck müssen Informationen zur Verbreitung nicht kopiert, sondern referenziert werden. Dazu ist es nicht nötig, dass sie als statische Dateien in einem Dateisystemen vorliegen, stattdessen können sie über einen Query-Anhang in der URI[3] dynamisch generiert werden. Dadurch können sich Dokumente automatisch zeitlich ändern: „[Documents] can therefore represent views of databases, or snapshots of changing data.“ (Berners-Lee, et al., 1992).

Das Web ist ein globales Rechnernetzwerk. Damit eine Kommunikation von unabhängigen Systemen stattfinden kann, sind Standards unerlässlich. Auf die Frage, warum ein globales Informationssystem nicht früher eingeführt wurde, schließt Tim Berners-Lee auf einen Mangel an netzwerkbasierten Kommunikationsprotokollen[4].

2.1 Hypertext Transfer Protocol

Das Hypertext Transfer Protocol ist ein auf der Applikationsebene angesiedeltes Netzwerkprotokoll für eine leichtgewichtige und schnelle Datenübertragung zwischen zwei verteilten, hypermedialen Informationssystemen (vgl. Berners-Lee, et al., 1996).

HTTP-Nachrichten werden unterschieden nach Anfrage und Antwort. Anfragen werden vom Client verschickt, Antworten hingegen vom Server. HTTP-Nachrichten sind aus Kopf (Header) und Rumpf (Body) zusammengesetzt. Die erste Zeile einer

[2] Information Retrieval bezeichnet die computergestützte Suche nach für den Sucher relevanten Informationen. Sie sind in einem System so gespeichert, dass sie unter verschiedenen Gesichtspunkten gesucht und gefunden werden können.

[3] URI steht für Unifrom Resource Indentifier und ist eine Zeichenkette, die eine abstrakte oder physische Ressource identifiziert. Für ein besseres Verständnis kann URI einer Linkadresse (URL) gleichsetzt werden.

[4] Ein Protokoll ist eine Vereinbarung, nach der die Datenübertragung stattfindet. Es definiert Syntax und Semantik von Nachrichten. Im Falle eines Netzwerkprotokolls sind die Parteien der Datenübertragung Computer, die über ein Rechnernetz miteinander verbunden sind.

Anfrage umfasst Methode, URI und HTTP-Version. Die erste Zeile einer Antwort beinhaltet Methode und Statusinformationen. HTTP-Methoden sind den klassischen CRUD[5]-Operationen nachempfunden. Statusinformationen bestehen aus Statuscode und -text. Sie geben Auskunft über Erfolg oder Misserfolg der Anfrage. Die HTTP-Header sind Schlüsselwertpaare und beschreiben den Inhalt der Nachricht, z.B. um welchen Content-Type es sich bei der mitgeschickten Ressource handelt. Ein Rumpfinhalt ist bei Anfragen optional und bei Antworten zwingend notwendig. Wird z.B. ein HTML-Dokument angefragt, wird nicht etwa das Dokument über das Netzwerk geschickt, stattdessen befindet es sich im Rumpf der Antwort (vgl. Abbildung 2-1).

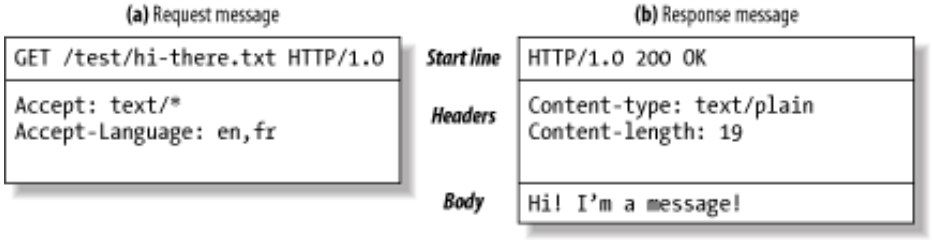

Abbildung 2-1: Aufbau von Anfrage und Antwort nach HTTP
Quelle: Gourley, et al., 2002

Seit der Spezifikation aus dem Jahr 1996 durchlief das Protokoll zwei wesentliche Überarbeitungen. Die Nachfolgeversion HTTP/1.1 adressiert die Problematik eines unnötig redundanten Verbindungsauf- und -abbaus: „Prior to persistent connections, a separate TCP connection was established to fetch each URL [...]" (Fielding, et al., 1999). Das bedeutet, dass HTTP/1.0 für den Versand von einer Nachricht eine TCP-Verbindung auf- und wieder abbaut, was zwangsläufig zu einer hohen Netzlast und schlechten Performanz führte.

HTTP/2.0 ist keine Nachfolgeversion von HTTP/1.1. Die Spezifikation definiert es als eine losgelöste Alternative. Fälschlicherweise werden HTTP/2.0 und HTTPS häufig synonym verwendet. HTTPS ist ein URI-Präfix und ist auf die Spezifikation HTTP over TLS (vgl. Rescorla, et al., 2000) zurückzuführen. Wie der Name schon vermuten lässt, findet die Kommunikation über TLS bzw. SSL statt. Daten werden vor dem Versand über das Netzwerk verschlüsselt. Ursprünglich wurden sie als Klartext transportiert. Ein heutzutage unvorstellbarer Sachverhalt, wenn man bedenkt, wie viele Anwendungen mit sensiblen und personenbezogenen Daten arbeiten.

[5] CRUD ist ein Akronym der vier fundamentalen Operationen zur Manipulation von persistentem Speicher: Create, Read, Update und Delete.

2.2 Representational State Transfer

Representational State Transfer (abgekürzt: REST) ist ein um die Jahrtausendwende abgeleiteter architektonischer Stil zur Umsetzung verteilter, hypermedialer Systeme (vgl. Fielding, 2000). REST ist ein auf dem Client-Server-Modell aufbauendes Kommunikationsparadigma und ist der defacto Standard für die Implementation von Schnittstellen zwischen Client- und Serveranwendungen. Schnittstellen zwischen Anwendungsprogrammen, im Fachjargon API genannt, die auf den REST-Prinzipien aufbauen, werden als REST-APIs bezeichnet.

Im Gegensatz zu RPC[6]-Anwendungen arbeitet REST nicht mit Objekten und netzwerkübergreifenden Funktionsaufrufen, sondern mit der Übertragung von leichten Datenstrukturen, einer Zustandsübertragung über HTTP und einer homogenen Schnittstellendefinition nach der URI-Spezifikation. Der Fokus auf eine normierte Schnittstelle zwischen REST-Komponenten unterscheidet REST von anderen architektonischen Stilen.

Im Folgenden werden die Teilbegriffe von REST und die Bedingungen, die damit einhergehen, anhand von Fieldings vielzitierter Dissertation[7] erläutert.

REST-Komponenten interagieren mit Ressourcen, indem die Repräsentation der Ressource abgefragt oder manipuliert wird. Eine Repräsentation entspricht einer um Metadaten beigefügten Byte-Sequenz. Ein HTML-Dokument ist ein Beispiel einer Ressource. Streng genommen ist ein HTML-Dokument eine lange Zeichenkette, die im Rumpf der Antwort verschickt wird. Metadaten signalisieren dem Webbrowser, dass die erhaltenen Informationen interpretierbar und visuell darstellbar sind.

Die Kommunikation von REST-Komponenten ist zustandslos, d.h. dass der Server keine Informationen über den anfragenden Client speichert. REST-Nachrichten sind deskriptiv und isoliert. Sie müssen alle Informationen beinhalten, die für eine vollständige Bearbeitung notwendig sind. Client und Server sind verpflichtet allen Nachrichten den aktuellen Zustand beizufügen (State Transfer). Dadurch, dass der Server kein Management in Form einer Session vornimmt, sind REST-APIs schnell in der Bearbeitung und skalierbar.

[6] RPC steht für Remote Procedure Call und ist eine Technik zur Realisierung von Interprozesskommunikationen. Vgl. hierzu die offizielle RFC-Spezifikation: https://tools.ietf.org/html/rfc1050

[7] vgl. Fielding, 2000

2.3 Asynchronous JavaScript and XML

Das Web war ursprünglich als eine digitale Informationsquelle geplant. Schnell erkannten Unternehmen das Potenzial mit einem geringen Aufwand eine große Zahl an Kunden zu erreichen. Es sprach viel dafür das Internet zu nutzen, um Produkte oder Dienstleistungen zu umwerben. Der technische Fortschritt, die Verbreitung von internetfähigen Endgeräten und die kontinuierliche Verbesserung von Webbrowsern veranlasste schließlich die nächste naheliegende Utilisierung des Webs ins Visier zu nehmen: eine Digitalisierung von kundenorientierten Geschäftsprozessen.

Die Zustandslosigkeit von REST steht im Konflikt mit der Funktionsweise von zahlreichen Webanwendungen. E-Commerce-Seiten müssen die Zustände von Kunde und Warenkorb über mehrere Anfragen aufrechterhalten. Instant Messaging wäre nicht möglich ohne Hintergrundsynchronisation. Online-Kartendienste müssen nach jeder Interaktion neue Kartenteile laden und bei Navigationshilfe im Sekundentakt den Standort erfassen, verwerten und dem Nutzer auf Basis dessen Rückmeldung geben. Eine Nutzung dieser Dienste ist nach traditionellen Methoden, bei denen neue Informationen ausschließlich per Seitenaufruf eingeholt werden, undenkbar.

Gegenüber leistungsstarken Desktopanwendungen hat sich der Funktionsumfang, die Antwortzeit und die Benutzerfreundlichkeit von Webanwendungen in den letzten Jahren verbessert. „Desktop applications have a richness and responsiveness that has seemed out of reach on the Web. [...] That gap is closing." (Garrett, 2005) formuliert es Jesse James Garret in einem Essay, in welchem er die Funktionsweise hinter dem Buzzword Asynchronous JavaScript and XML (Ajax) beschreibt. Dabei handelt es sich um keine neue Technologie, sondern vielmehr um ein Zusammenspiel mehrerer eng verzahnter Technologien (vgl. Garrett, 2005):

- Eine Präsentation durch HTML und CSS
- Dynamik und Interaktion durch das Document Object Model (DOM)
- Datenaustausch durch XML
- Asynchrone Datenbeschaffung durch das XMLHttpRequest-Objekt
- JavaScript als Bindeglied der oben genannten Technologien

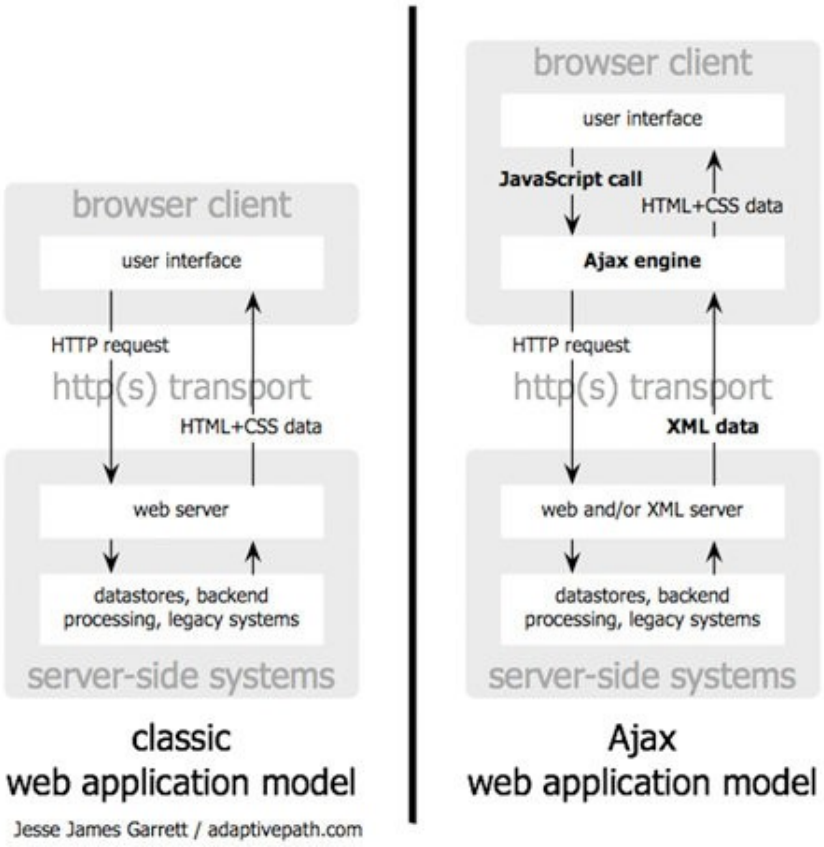

Abbildung 2-2: Vergleich zwischen dem traditionellen Modell und AJAX
Quelle: Garrett, 2005

Eine Möglichkeit eine Benutzerfreundlichkeit wie bei Desktopanwendungen oder nativen Apps zu erzielen besteht darin, das Rendern auf der Clientseite, statt wie ursprünglich dafür vorgesehenen, auf der Serverseite zu realisieren. Angesichts dieser Veränderung hat sich eine neue Rollenverteilung für Client und Server ergeben. Der Client weiß welche Informationen er benötigt und wie er sie darzustellen hat, was automatisch zu einem erhöhten Programmieraufwand im Frontend führt. Der Server kann in den meisten Fällen als ein Thin-Server fungieren, der eine reine Daten-API bereitstellt und kein Session-Management vorzunehmen hat.

Streng genommen widerspricht die Funktionsweise von Ajax den traditionellen Ansätzen einer Client-Server-Kommunikation. Anwendungen bestanden ähnlich wie Webseiten aus mehreren statischen Dokumenten, die über Hyperlinks verbunden sind. Eine zunehmend kritische Erwartungshaltung des Endverbrauchers zwingt Entwickler Webanwendungen so zu konzipieren, dass eine herausragende Benutzerfreundlichkeit mit praktisch keiner Wartezeit entsteht. Wie Farrell und Nezlek im Jahr 2007 feststellten, ist der Faktor Usability zum Hauptaugenmerk der Softwareentwicklung und Auslöser eines hitzigen Konkurrenzkampfes geworden.

2.4 Das kollaborative Web 2.0

Im Februar 2003 schrieb Ex-Microsoft-Advocate Dave Stutz einen inspirierenden offenen Brief an seinen ehemaligen Arbeitgeber kurze Zeit, nachdem er das Unternehmen verließ. Seiner Meinung nach entwickelt Microsoft zwar die beste Client-Software, doch scheitere daran über den Tellerrand hinaus zu sehen und Potenziale zu erkennen: „Windows has yet to move past its PC-centric roots to capture a significant part of the larger network space [...]“ (Stutz, 2003).

Weiterhin rät er Microsoft ab, Software als einen kommerziellen Verbrauchsgegenstand zu sehen und zeigt anhand des direkten Konkurrenten Linux auf, dass Open-Source Produkte sich am Markt durchsetzen können. Der Brief endet mit der Prognose, dass erfolgreiche Software nicht an ein Gerät gebunden sein darf: „Useful software written above the level of the single device will command high margins for a long time to come.“ (Stutz, 2003).

Die Annahme soll sich, wie wir heute wissen, bewahrheiten. Das Gebiet der Webentwicklung profitiert von dieser Entwicklung: JavaScript belegt erneut Platz eins der beliebtesten Programmiersprachen auf Github, dem führenden Anbieter einer Online-Versionsverwaltung von Open-Source-Projekten[8]. Ein weiterer Indikator ist die diesjährige Umfrage der Plattform Stack Overflow: “Unsurprisingly, for the eight year in a row, JavaScript has maintained its stronghold as the most commonly used programming language.”[9]. Es ist vertretbar anzunehmen, dass die von der Open-Source-Bewegung ausgehende Entkommerzialisierung von Software, in Kombination mit einer Distanzierung eines PC-zentrierten Internets die Grundlage einer Web-2.0-Plattform ist.

Nun stellt sich die Frage „Was ist überhaupt das Web 2.0?“. Tim O’Reilly, Gründer des gleichnamigen Verlages, trug mit seinem im September 2005 veröffentlichten Artikel über das Web 2.0 maßgeblich zur Popularität des Schlagwortes bei. Er nennt Stutz‘ Brief als eine der Inspirationsquellen. Bei dem Versuch das Web 2.0 zu definieren entstand die Web 2.0 Meme Map (siehe Abbildung 2-3, S. 11).

[8] vgl. die offizielle Statistik von Github: https://madnight.github.io/githut/#/pushes/2020/1

[9] vgl. die Umfrage auf Stack Overflow: https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-professional-developers

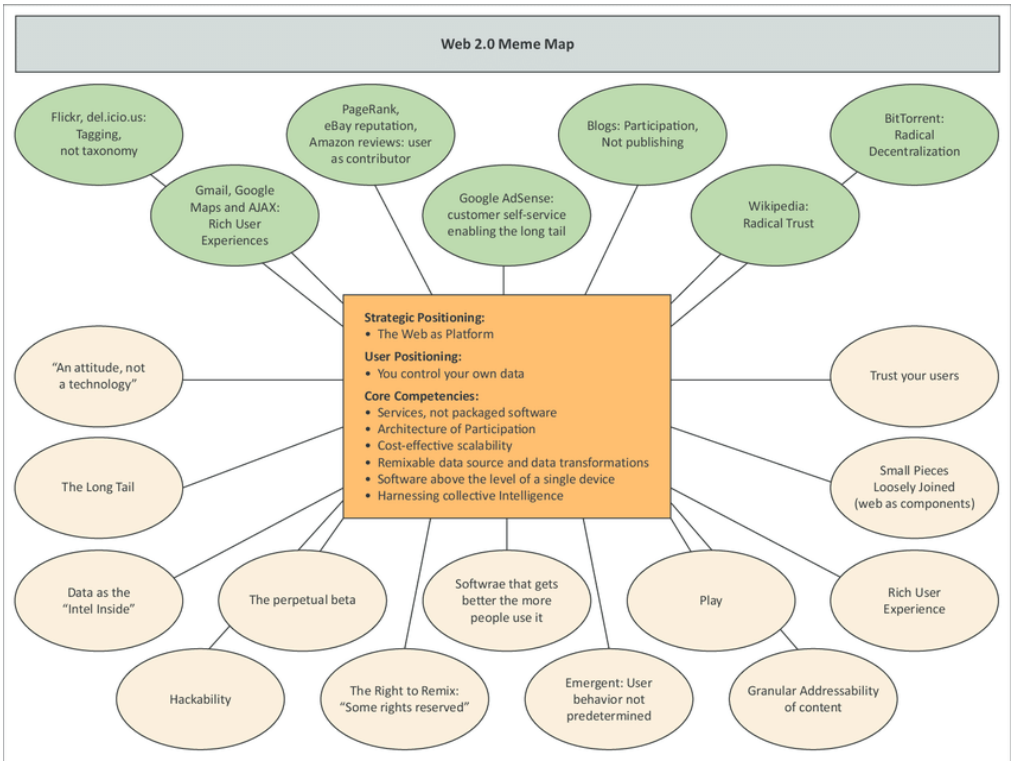

Abbildung 2-3: Web 2.0 Meme Map
Quelle: O'Reilly, 2005

Die Kollaboration der Benutzer („Harnessing collective Intelligence") sei eine der Schlüsselkompetenzen von Web 2.0-Anwendungen. Noch bevor Social Media zu dem Phänomen wurde, das es heute ist, existierten Webseiten, bei denen der Seiteninhalt gänzlich von den Beiträgen der Webseitbesucher abhängig war. Wikipedia, eine von jedem Besucher frei erweiterbare und bearbeitbare Enzyklopädie, ist ein Vorreiter dieses Ansatzes. Die Web-2.0-konformen Charakteristiken von Wikipedia und Social Media-Plattformen können an den Punkten „Trust your users", „Hackability" und „the software gets better the more people use it" festgemacht werden.

O'Reilly betitelt das Unternehmen Google als einen „Träger" des Web 2.0. Diese Bezeichnung ist auf Googles Ursprung als Webapplikation zurückzuführen, welche nie verkauft, sondern als Dienst mit Möglichkeit zur Erweiterung angeboten wurde. Googles massiver Erfolg beruht neben dem berühmten PageRank-Algorithmus auf einem unnachahmlichen Datenbankmanagement: „Database management is a core competency of Web 2.0 companies" (O'Reilly, 2009).

2.5 Crowdsourcing, Crowdcasting und VGI

Typische Web 2.0-Anwendungen wie Social Media-Plattformen oder Wikipedia setzen auf den Intellekt und die Kreativität der Nutzer, um Inhalte zu kreieren. Das Web 2.0 ist nicht auf das Konzept interaktiver Internetplattformen beschränkt, vielmehr ist es mit der Theorie eines daten- statt dokumentenorientierten semantischen Webs gekoppelt. Die Internetplattform eignet sich Daten visuell zu präsentieren, vor

allem mit Hinblick auf die zunehmende Leistungsfähigkeit von Browsern, und ist die Basis einer schnellen Datenübertragung von verteilten Systemen. Das Web hat keine Voraussatzung an die zugrundeliegende Hardware und ist Geräte-agnostisch. Dieser Sachverhalt erlaubt die Verknüpfung von nativen und webbasierten Anwendungen.

Der Begriff Crowdsourcing wurde im Jahr 2005 von Jeff Howe und Mark Robinson geprägt und ist an den in der Wirtschaft bekannten Term Outsourcing, also der Auslagerung von Unternehmensaufgaben an Externe, angelehnt. Im Gegensatz zum Outsourcing nimmt jedoch kein anderes Unternehmen die Aufgaben ab, sondern eine unorganisierte und meist unbezahlte Menge an Privatpersonen. Der technische Fortschritt im Gebiet der Mobilgeräte erlaubt es, dass Privatpersonen mit einem geringen Aufwand persönliche Daten teilen können. Crowdsourcing reduziert die Kosten auf Seiten der Betreiber: „Technological advances [...] are breaking down the cost barriers that once separated amateurs from professionals.“ (Howe, 2006).

Crowdcasting vereint die Gebiete Crowdsourcing und Broadcasting. Das Konzept Crowdsourcing wird durch sog. Push- und Pull-Faktoren erweitert: „[...] the potential crowd being pulled in as a direct result of the push“ (Hudson-Smith, et al., 2009). Ein „Push“ kommt stets von den Verantwortlichen, die die Zielaudienz erreichen, ihr Produkt umwerben und zur Teilnahme motivieren möchten. Altmodische Broadcast-Medien sind Funk und Fernsehen. Heutzutage werden Werbekampagnen meist mit Internetauftritten verbunden. Die Push-Maßnahmen stehen den individuellen Pull-Faktoren der Zielgruppe gegenüber, also der Nutzen, den die Privatpersonen in ihrer Teilnahme sehen.

Die Entwicklung von Informationssystemen, die auf Crowdcasting aufbauen, sind einem hohen Risiko ausgesetzt und stark abhängig von dem Mehrwert, den die Zielgruppe in der Teilnahme sieht (vgl. Hudson-Smith, et al., 2009). In diesem Zusammenhang propagieren Betreiber nicht selten den Einsatz von monetären oder nichtmonetären Belohnungen: „Some crowdcasting systems actually introduce incentives for users to key in their own data by offering rewards” (Hudson-Smith, et al., 2009).

VGI ist ein Akronym für Volunteered Geographic Information und beantwortet einige offene Fragen des Crowdcastings, mitunter welche Informationen aggregiert und ob die Teilnehmer für ihre Beiträge entschädigt werden. „Quellen räumlicher Daten sind neben traditionellen geodätischen Aufnahmen [...] sog. VGI-Daten“ (Sester, 2019). VGI sind Geodaten, die auf freiwilliger Basis von Laien erhoben und zur Verfügung gestellt werden.

Das VGI-Konzept gewann durch das Wachsen des Web 2.0 an Aufwind und hat sich als eine ernstzunehmende Alternative etabliert. Die polarisierende Metapher „Menschen als Sensoren“ von Michael Frank Goodchild (2007) schließt an den in der Forschung erkennbaren Trend eines Zusammenspiels von GPS und dem Web als Datenübertragungsmedium an. Das größte Defizit von VGI gegenüber herkömmlichen Methoden ist eine geringfügig schlechtere Datenqualität (vgl. Goodchild, 2012).

3 Datenarchitektur

Der Entwurf einer Datenarchitektur umfasst die Disziplinen Datenmodellierung, Datenbankentwurf und eine Visualisierung des Daten-Lebenslaufes. Der Lebenslauf verdeutlicht, wie Daten akquiriert und innerhalb von Systemen und Anwendungen fließen und genutzt werden. Die Entwicklung einer Datenarchitektur besteht aus einer fachlichen, logischen und physischen Ebene (vgl. Masak, 2006). Fachlich ist zu prüfen, welche Daten für das geplante Informationssystem relevant sind. Danach gilt es, die Daten in eine logische Beziehung zu bringen und durch ein semantisches Modell auszudrücken. Die Schnittstellenimplementierung entspricht der physischen Ebene und schließt den Entwicklungsprozess der Datenarchitektur ab.

3.1 Datenakquise

Die Datenakquise ist der erste und wichtigste Schritt in der Datenanalyse. Neben der Frage welche Daten zu erfassen sind ist es besonders bei Daten mit räumlichem Bezug wichtig zu klären, wie die Daten zu erfassen sind.

Die Anschaffung von Equipment, wie von hochleistungsfähigen GPS-Trackern, ist eine kostspielige Angelegenheit und schränkt die Zielgruppe auf ein Minimum ein. Nicht zuletzt wegen eines vorgegebenen Kostenplans ist es naheliegend eine Strategie anzuwenden, die kostengünstig ist, einfach umsetzbar und Reichweite verspricht: „Mobile phones [...] have opened up new possibilities for the investigation of human behaviour“ (Ahas, 2011). Die immer besser werdenden Smartphone-Prozessoren und der flächendeckende Ausbau des 5G-Netzwerkes sind gute Gründe, warum es Sinn macht, in eine Datenerhebung über die Anwendungsebene von Mobilgeräten zu investieren. Bereits Mobilfunknetze der dritten Generation wurden in der Fachliteratur als ein geeignetes Rahmenwerk zur Standorterfassung eingestuft (vgl. Ahonen und Eskelinen, 2003).

Das NorOvA-Team hat sich für eine VGI-konforme Datenerhebung entschieden. Die Daten der Zielgruppen werden durch eine selbstentwickelte Android-App aufgezeichnet. Warum sich das Projektteam für eine Android-Entwicklung entschieden hat, lässt sich an den Marktanteilen in Deutschland festmachen. Smartphones mit Android-Betriebssystem sind seit Jahren Marktführer (siehe Abbildung 3-1, S. 14). Wenn von einer ähnlichen Verteilung unter den Hochschulangehörigen ausgeht, stößt man auf eine größere Resonanz, wodurch mehr Privatpersonen mitwirken können und für eine breitere Datenbasis sorgen.

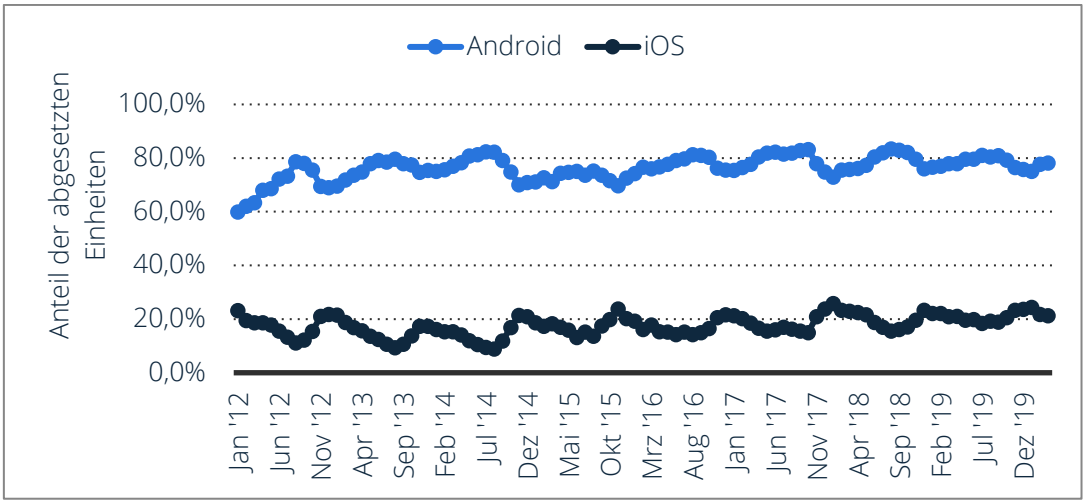

Abbildung 3-1: Vergleich der Marktanteile von Android und iOS in Deutschland (2012-2020)
Quelle: Kantar, zitiert nach statista.com

Hochschulangehörige können die im Playstore erhältliche App herunterladen und nach Belieben eine im Hintergrund laufende Standorterfassung aktivieren. Die App bedient sich nach vorheriger Einwilligung des Endnutzers über die nativen Android-APIs an den GPS- und Sensorinformationen des mobilen Endgerätes.

Konkret heißt das, dass nach Aktivierung einer Aufzeichnung viertelminütig Längengrad, Breitengrad, Geschwindigkeit und Genauigkeit[10] in Kombination mit einem Zeitstempel lokal temporär abgespeichert werden. Das Zeitfenster von 15 Sekunden ist ein Kompromiss zwischen Präzision und Akkufreundlichkeit. Ist zwischenzeitlich die Verbindung unterbrochen, registriert die App weiterhin die Koordinatenpunkte und puffert diese im Arbeitsspeicher. Datensätze mit einer zu geringen Genauigkeit oder einer zu großen räumlichen Distanz zum vorherigen Datensatz werden verworfen. Die Streckenaufzeichnung läuft, bis sie manuell beendet wird, der Benutzer die GPS-Funktion ausschaltet oder die App über einen längeren Zeitraum vom Internet getrennt ist. Liegen die Daten nach Beenden der Aufzeichnung in einem konsistenten Zustand vor, werden sie über das Netzwerk an eine REST-Schnittstelle (vgl. hierzu Kapitel 2.2) eines Java Spring-Boot-Servers[11] geschickt und dort in einer zentralen MySQL-Datenbank persistiert. Der Server akzeptiert die Datenübertragung, da der Nutzer authentifiziert ist.

[10] Mit Genauigkeit ist die geschätzte horizontale Genauigkeit gemeint. Zieht man einen Kreis an dem erfassten Standort mit dem Radius gleich dem numerischen Wert der horizontalen Genauigkeit in Metern, beträgt die Wahrscheinlichkeit ca. 68% Prozent, dass die Koordinate sich in diesem Kreis befindet. Vgl. https://developer.android.com/reference/android/location/Location#getAccuracy()

[11] Spring Boot ist eine Konfiguration des populären Java Spring-Frameworks, das häufig genutzt wird, um Webanwendungen zu programmieren. Vgl. https://spring.io/projects/spring-boot

3.2 Datenmodellierung

Die Datenmodellierung ist die Strukturierung von in Beziehung stehenden Daten und eine Schlüsselkomponente zum effizienten Datenbankentwurf. Die Modellierung sollte sorgfältig durchgeführt werden, weil sie die Grundlage für die Entwicklung von Informationssystemen ist (vgl. Gadatsch, 2019). Ein Datenmodell wird in der Fachliteratur verschieden gedeutet. Das relationale Datenmodell (vgl. Codd, 1970) ist die Grundlage für die Konzeption und Umsetzung relationaler Datenbanken.

Zur Darstellung von relationalen Datenmodellen wird in der Praxis häufig das 1976 vorgestellte Entity-Relationship-Modell (kurz: ER-Modell) eingesetzt. Das ER-Modell vereinigt die Vorteile von drei anderen Modellen und legt den Fokus dabei auf eine einfache Abbildung der realen Welt durch die semantischen Elemente Entität, Beziehung und Attribut (vgl. Chen, 1976). Erfinder Peter Chen definiert eine Entität als eine eindeutige „Sache“, sei es eine Person, ein Unternehmen oder ein Ereignis. Eine Beziehung beschreibt die Assoziation zwischen Entitäten. Attribute sind Informationen über eine Entität oder Beziehung.

Seit der Veröffentlichung der sog. Chen-Notation (siehe Abbildung 3-2) nahm das Repertoire an logischen und grafischen Komponenten durch die Einflüsse verschiedener Wissenschaftler zu. Jean-Raymond Abrial stellte 1974 im Rahmen seines Modellentwurfs die Min-Max-Notation vor, die daraufhin wenig später von Chen adaptiert wurde. Die Min-Max-Notation erlaubt es die Kardinalität einer Beziehung einzuschränken, indem der Minimal- und Maximalwert der an einer Beziehung beteiligten Entität angegeben wird.

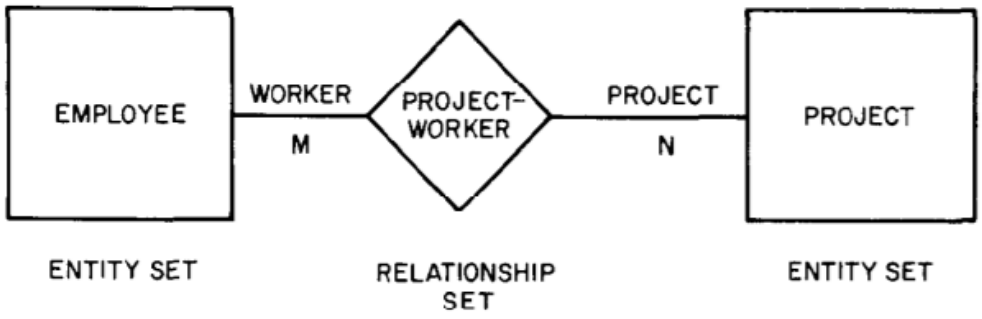

Fig. 10. A simple entity-relationship diagram

Abbildung 3-2: Basisnotation des Entity-Relationship-Modells (Chen-Notation)
Quelle: Chen, 1976

Im vorherigen Kapitel wurde die Herkunft der Datenbasis von der Erfassung der Daten durch die eingebauten Android-Sensoren bis zur Übertragung an die REST-Schnittstelle eines Webservers. Das Datenbankschema umfasst zahlreiche Tabellen, wobei drei Tabellen für NorOvA Analytics relevant sind: `tracking_data`, `route` und `user` (Abbildung 3-3, S. 16).

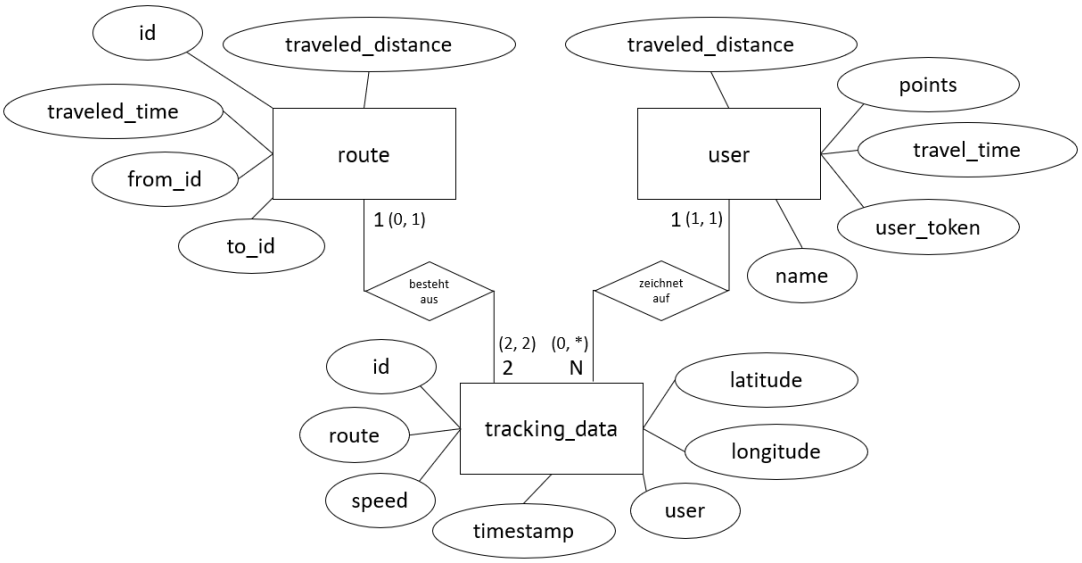

Abbildung 3-3: Auszug des Datenbankschemas als Entity-Relationship-Diagramm

Trackingdaten sind die aufgezeichneten Koordinatenpunkte. Zunächst wird überprüft, ob die Aggregation der Daten bestimmte Kriterien erfüllen:

- eine zurückgelegte Distanz von mindestens 100 Metern aufweisen und
- insgesamt aus mehr als zehn Koordinatenpunkten bestehen.

Erfüllen sie die Bedingungen, wird ein korrespondierender Routen-Datensatz in der Tabelle `route` angelegt und von jedem Trackingdatensatz über das gleichnamige Attributfeld referenziert. Andernfalls enthält es einen Nullwert. Daneben gibt es einen weiteren Fremdschlüssel zu der Tabelle `user`. Die Zuordnung der Trackingdaten zu einem Benutzer ist entscheidend. Eine geplante Logbuch-Funktion in der App und Direktvergleiche zwischen Benutzern werden dadurch ermöglicht.

Die erfassten Standort- und Sensordaten erhalten einen Zeitstempel zur eindeutigen Identifikation innerhalb der Streckenaufzeichnung. Diese Art der Identifikation ist auf Datenbankebene, welche die Daten aller registrierten Benutzer enthält, nicht weiter sinnvoll, da es durchaus sein kann, dass zwei Benutzer gleichzeitig die Aufzeichnung starten und Koordinatenpunkte mit identischem Zeitstempel an den Server schicken. Als Gegenmaßnahme erhält die Tabelle ein ID-Attribut.

Ein Benutzer steht in einer 1-zu-N-Beziehung zu den von ihm erfassten Trackingdaten. Handelt es sich um einen kürzlich registrierten Nutzer, sind ihm logischerweise keine Daten zugeordnet. Für eine Authentifizierung des Nutzers gegenüber der Serveranwendung ist ein User-Token notwendig.

Eine Route referenziert stets die erste (`from_id`) und letzte (`to_id`) Koordinate einer Streckenaufzeichnung, vorausgesetzt sie erfüllen die genannten Bedingungen. Es entsteht eine 1-zu-2-Beziehung.

Routen- und Benutzereinträge kumulieren Strecken- und Zeitwerte. Ersichtlich wird das durch die gemeinsamen Attribute `traveled_time` und `traveled_distance`. Die zurückgelegte Distanz ist die Bewertungsgrundlage von Gewinnspielen.

3.3 Schnittstellen und Datenfluss

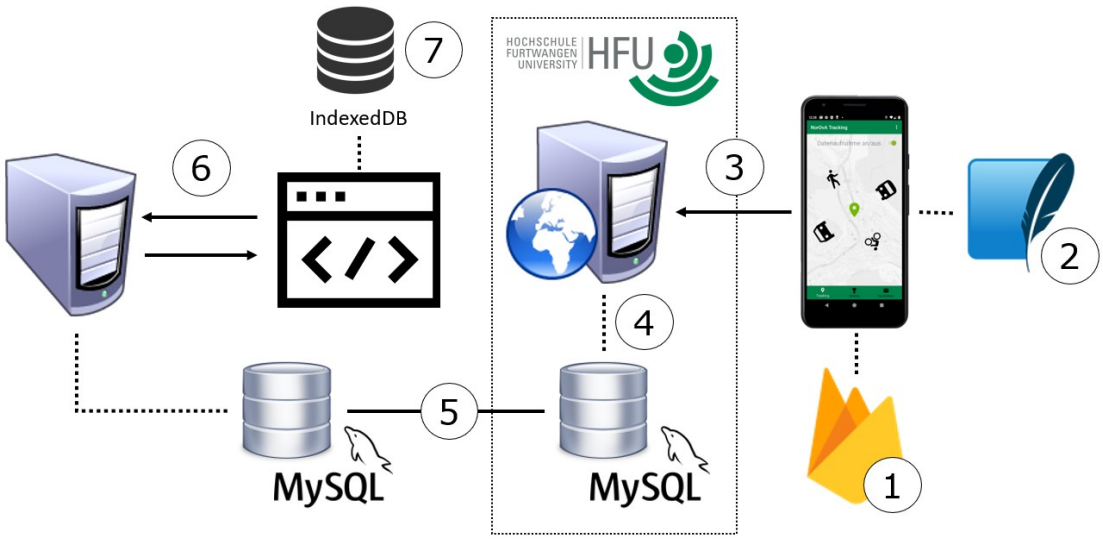

Abbildung 3-4: Schnittstellen und Datenfluss zwischen den Systemen

Anhand von Abbildung 3-4 wird der Datenfluss systemübergreifend erläutert.

(1) Sobald sich ein Student die App aus dem Playstore herunterlädt, wird beim erstmaligen Öffnen das Firebase-SDK initialisiert und ein Benutzer angelegt. Ein Token wird auf dem Endgerät gespeichert. Zeitgleich wird ein Benutzerdatensatz in der Datenbank erstellt, wobei das Token die ID des Datensatzes ist. Vorteilhaft ist, dass der Benutzer keine Registrierung durchlaufen muss. Auf der anderen Seite wird ein neuer Benutzer bei einer Deinstallation und darauffolgenden Neuinstallation angelegt. Dem Benutzer steht es offen, einen Benutzernamen zu wählen oder anonym zu bleiben, wobei er den Benutzernamen „anonymous“ zugewiesen bekommt.

(2) Die App speichert während der Streckenaufzeichnung die Koordinatenpunkte in einer lokalen SQLite-Datenbank temporär ab.

(3) Nach dem Beenden der Aufzeichnung gelangen die Daten an eine Schnittstelle des zentralen Anwendungsservers, welcher sich im Rechenzentrum der Hochschule befindet. Die Daten werden in dem Rumpf der HTTP-POST-Nachricht an den Endpunkt `/user` gesendet.

(4) Die Daten werden auf die in Kapitel 3.2 beschriebenen Bedingung geprüft. Erfüllen sie die Bedingungen schickt der Anwendungsserver die Daten über

den Standard-Port 3306 an die zentrale MySQL-Datenbank, in der sie schließlich persistiert werden.

(5) Wie in Kapitel 1.2.3 bereits erwähnt, ist NorOvA Analytics eine Stand-Alone-Web-App. Um den Datenverkehr der zentralen Datenbank niedrig zu halten, bedient sich NorOvA Analytics an einem lokalen Anwendungsserver mit einer Kopie des Datenbestandes. Die Kopie wird über eine Weboberfläche bezogen.

(6) Der lokale Anwendungsserver impliziert ebenfalls das REST-Paradigma und ist dementsprechend eine REST-API. Daten werden im JSON-Format an den Client übertragen. Das Frontend schickt eine HTTP-GET-Anfrage, um Daten zu beziehen. Die URI definiert Endpunkt und Operationsname. Parameter in der Query schränken die Ergebnismenge ein. Die konkrete Funktionsweise des Servers und die exakten Endpunkte werden in Kapitel 4.5.3 erörtert.

(7) Handelt es sich um Benutzer- oder Routendaten, werden sie in einem Speicher im Frontend namens IndexedDB abgelegt, um ausgewählte Analysen, wie die Benutzeranalyse, zu beschleunigen. Die Trackingdaten werden aufgrund ihres Umfangs[12] nicht in der IndexedDB abgelegt.

Alles in Allem sind fünf Datenspeicher beim Datenfluss von der Erhebung bis hin zur Analyse involviert. Die Übertragung der Koordinatenpunkte und die Bereitstellung der angereicherten Daten des lokalen Anwendungsservers erfolgen über HTTP. Der Datenbankdump wird manuell geladen, wobei es Potenzial zur Automatisierung gibt. Die IndexedDB speichert Daten als JSON-Objekte, die über ihren Index abrufbar sind.

[12] Die Tabelle der Trackingdaten umfasst 8 Millionen Datensätze. Davon kommen zirka 400.000 (Stand: Juni 2019) für Analysezwecke in Frage (route-Attribut ungleich einem Nullwert).

4 Software Design

"At the heart of every well-engineered software system is a good software architecture - a good set of design decisions" (Medvidovic und Taylor, 2010). Das Gebiet des Software Designs erhielt in den 1970er Jahren viel Aufmerksamkeit. Die Prämisse war es, dass das Design eine Aktivität unabhängig von der Implementation ist, diese jedoch positiv beeinflussen kann, wenn sie umfangreich, präzise und normiert durchgeführt wird. Dem Software Design werden verschiedene Notationen, Techniken und Werkzeuge zugeordnet (vgl. Perry und Wolf, 1992). Eine praktische und bewährte Notation ist die Unified Modelling Language (UML).

Software Design ist eine Disziplin des Software Engineerings. Als Leitfaden und Definitionsgrundlage dient der Leitfaden „Guide to the Software Engineering Body of Knowledge" (SWEBOK) der IEEE Computer Society. In dem besagten Dokument ist Design als Prozess zur Gestaltung von Architektur, Komponenten, Schnittstellen und anderen Merkmalen von Systemen und Subsystemen definiert (vgl. Bourque, et al., 2014). Der Entwurf eines Designs ist der zweite Teilschritt des Software Engineerings und schließt direkt an die Anforderungsanalyse an.

Design kann zudem als eine Möglichkeit angesehen werden, ein Problem zu lösen; ein Problem zur der es keine absolute Lösung gibt. Jede Interpretation einer Lösung birgt Vor- und Nachteile und ist an bestimmte Bedingungen geknüpft, wie z.B. die zur Verfügung stehenden Technologien und der Kenntnisstand und die Erfahrungen des Programmierers.

Das Resultat des Software-Designs ist eine einheitliche, verständliche und durch Artefakte festgehaltene Architektur, die darstellt, wie Software in Komponenten und Schnittstellen aufgeteilt ist (vgl. Bourque, et al., 2014). Gemäß des internationalen Standards ISO 12207 impliziert die Disziplin des Software Designs zwei Teilschritte: Der Entwurf eines architektonischen Designs, das die Top-Level-Struktur beschreibt und der Entwurf eines detaillierten, feingranularen Designs auf Ebene der kleinsten Softwareeinheiten.

4.1 Ausgangssituation und Anforderungsanalyse

Kapitel 1.2.3 hat einen Einblick in den Funktionsumfang von NorOvA Analytics gegeben. Dieses Kapitel beschreibt den Stand des Forschungsprojektes zu Beginn der Projektarbeit und fungiert als Anforderungsanalyse für die zu entwerfende Software.

Im letztjährigen Wintersemester hat das NorOvA-Team das digitale Werkzeug zur Datenakquise, die Android-App, erfolgreich promotet und in den Alltag von vielen Studierenden integrieren können. Während des Sommersemesters waren neben App-Features, die auf den Endnutzer ausgerichtet sind, projektinterne Vorkehrungen für eine Datenanalyse an der Spitze der Prioritätenliste.

Vor und während dem Beginn der Abschlussarbeit wurde eifrig an einer Modal-Split-Lösung gearbeitet, um den Streckenaufzeichnungen ein Transportmittel zuzuordnen. Das Projektteam saß auf einer stattlichen Menge von Bewegungsdaten, aber hatte keinerlei Möglichkeit diese zu analysieren. Es konnte auch keine Entwicklung an einem Analysewerkzeug begonnen werden, weil der Modal-Split höchste Priorität besaß und die Kapazitäten vom NorOvA-Team ausgelastet waren. Es hätte aus Sicht der Projektteilnehmer praktisch keinen besseren Zeitpunkt für eine Abschlussarbeit mit dem Schwerpunkt Datenvisualisierung und -analyse geben können, welche das Entwicklerteam entlastet und das Projekt finanziell nicht weiter einschränkt.

Zum Beginn standen folgende Anforderungen an die Software fest:

- Die Software muss als eine React-Komponente umgesetzt werden, sodass sie in die bestehende, projektinterne und React-betriebene Weboberfläche integriert werden kann.
- Die Nutzung der Open-Source-Bibliothek Leaflet als Kartendienst wurde aufgrund der jüngsten Kommerzialisierung von Google Maps nahegelegt.
- Die Software soll mehrere Analysemöglichkeiten abdecken, mitunter Zeit- und Raumanalysen. Die ersten Anforderungen waren sehr vage formuliert und wurden im Projektverlauf zunehmend konkretisiert.
- Nach Abschluss der Thesis wird das Analysewerkzeug weiter ausgebaut. Um Robustheit und Dokumentationsfähigkeit der Software zu verbessern, wird client- als auch serverseitig TypeScript eingesetzt.

4.2 Programmiersprache und Bibliotheken

Für die Umsetzung eines komponentenbasierten Designs und der Abbildung einer interaktiven Karte sind die Bibliotheken React und Leaflet eingebunden.

4.2.1 React

React ist eine von Facebook stammende Open-Source JavaScript-Bibliothek und ist, gemessen an den Downloadzahlen über npm seit dem Jahr 2015 (siehe Abbildung 4-1, S. 21), das mit Abstand beliebteste Frontend-Framework[13] zur Umsetzung von Single-Page-Applikationen.

[13] Strenggenommen ist React kein Framework, sondern eine Bibliothek: „A framework calls your code. Your code calls a library.“ (Strazzullo, 2019, S.11). Die Begriffe werden zur Verbesserung des Leseflusses synonym verwendet.

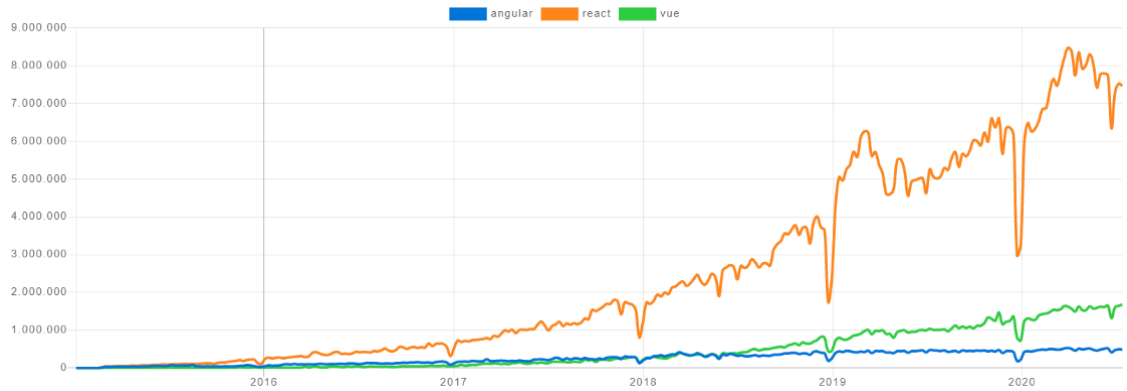

Abbildung 4-1: NPM-Downloadzahlen von Angular, React und Vue seit 2015
Quelle: https://www.npmtrends.com/react-vs-angular-vs-vue

Wie der Name suggeriert, ist eine SPA ein Anwendungsprogramm, das auf einer einzigen Webseite läuft. Zu keinem Zeitpunkt findet eine Navigation auf eine andere Seite statt. Für die Interaktivität und eine asynchrone Datenbeschaffung sind die in Kapitel 2.3 beschriebenen Ajax-Technologien verantwortlich. Frontend-Frameworks bauen auf diesen Technologien auf und bieten ein Rahmenwerk für ein schnelles, sicheres, geprüftes, performantes und komponentenbasiertes Design: "In object-oriented (OO) programming, a key related notion is that of a framework: a partially completed software system that can be extended by appropriately instantiating specific extensions" (Bourque, et al., 2014). Diese Formulierung ist Sinnesgemäß für die von React bereitgestellte Klasse `Component`.

Wie bereits angedeutet, ist React ein „performantes" Rahmenwerk. Diese Aussage rührt daher, dass sich React auf das Konzept eines virtuellen DOMs stützt, um die Benutzeroberfläche partiell zu aktualisieren. Der virtuelle DOM ist eine zwischengespeicherte Repräsentation der Seitenstruktur. Zustandsänderungen aktualisieren den virtuellen DOM, der mit dem echten DOM synchronisiert wird. Dadurch aktualisiert React nur die von einer Veränderung betroffenen Komponenten und nicht das ganze Dokument. Ein nicht zu unterschätzender Vorteil gegenüber Webanwendungen ohne Einsatz eines Frontend-Frameworks. Es ist möglich das partielle Rendern selbst zu programmieren (vgl. Strazzullo, 2019), wobei die Lösung die Performanz von React höchstwahrscheinlich nicht übersteigen wird.

Im Gegensatz zu Angluar und Vue macht React keinen Gebrauch von Two-Way-Data-Binding. Eine Veränderung von Attributwerten haben zwar eine Veränderung der Benutzeroberfläche zur Folge, aber nicht umgekehrt. Dieser Mechanismus wird über eine Veränderung von Zuständen repliziert. Ein sorgfältig durchgeführtes State Management ist eine der Hauptaufgaben in dem Design von React-Applikationen.

Code-Wiederverwendung kann durch die zwei geläufigen Ansätze Vererbung oder Komposition erreicht werden. In der Objektorientierung gibt es eine Technik namens „Composition over inheritance". Auf den ersten Blick erscheint dieser Ansatz widersprüchlich, denn die Vererbung ist das Auszeichnungsmerkmal der objektorientierten

Programmierung. Der Ursprung dieser Technik beruht auf schwerwiegende Nachteile und Einschränkungen der Klassenvererbung. Bei der Komposition wird neue Funktionalität durch das Zusammensetzen von Objekten erzielt (vgl. Gamma, 1993).

Die Komposition sieht vor das Verhalten von Klassen als eigenständige Klassen, sog. Verhaltensklassen, abzubilden, die von Schnittstellen abgeleitet sind. Die React-Dokumentation widmet dem Thema einen ausführlichen Artikel[14] und schließt sich der Empfehlung der Gang of Four[15] an, die Komposition der Vererbung vorzuziehen.

4.2.2 Leaflet

Die Leaflet-Bibliothek betitelt sich auf der offiziellen Webseite als ist die führende Open-Source-Bibliothek für responsive und interaktive Online-Karten. Ein Blick auf die Downloadzahlen der beliebtesten Kartendienste vermittelt ein Kopf-an-Kopf-Rennen zwischen Leaflet und Mapbox (siehe Abbildung 4-2). Mapbox ist mit 3D-Fähigkeit und Augumented Reality weitaus fortgeschrittener als Leaflet, welches einen leichtgewichtigen 2D-Kartendienst bietet. Google Maps ist von dem Vergleich ausgeschlossen, da es nicht über die npm-Plattform verteilt wird. Trotz der jüngsten Kommerzialisierung ist Google Maps nach wie vor unangefochtener Marktführer der Online-Kartendienste[16]. Für eine einfache 2D-Abbildung der Strecken ist die Leaflet-Bibliothek vollkommend ausreichend.

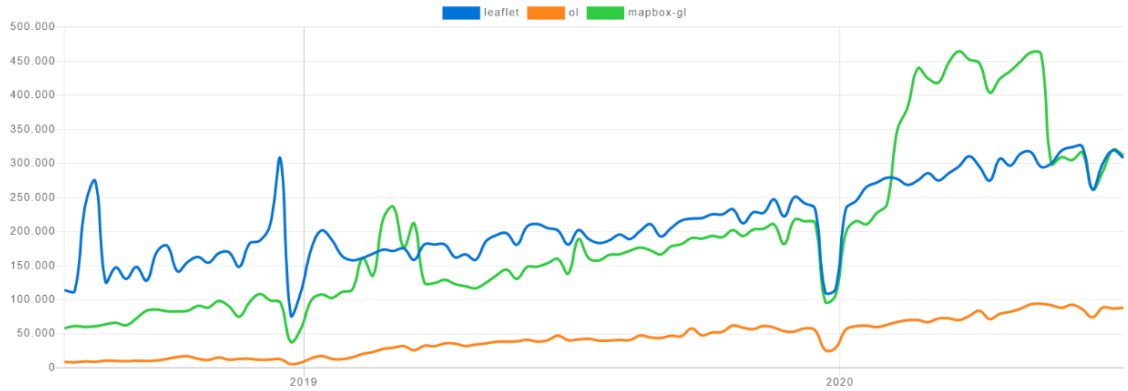

Abbildung 4-2: NPM-Downloadzahlen von Leaflet, OpenLayer und Mapbox
Quelle: https://www.npmtrends.com/leaflet-vs-ol-vs-mapbox-gl

[14] vgl. https://reactjs.org/docs/composition-vs-inheritance.html

[15] Die Gang of Four ist ein Spitzname für die vier Autoren des Buches „Design Patterns. Elements of Reusable Object-Oriented Software. Vgl. Gamma, 1993.

[16] vgl. https://www.similartech.com/compare/google-maps-vs-leaflet

4.2.3 React-Leaflet

Leaflet zeichnet sich durch Einfachheit, Performanz[17] und intuitive Bedienbarkeit aus. Die Mapping-Funktionalitäten sind über exportierte Variablen und Funktionen zugänglich. Die Arbeiten an der Leaflet-Bibliothek begannen in den 2000er Jahren. TypeScript wurde im Oktober 2012 veröffentlicht. Es ist offensichtlich, dass Leaflet auf veraltete Best Practices aufbaut und keine Typisierung oder Abstraktion in Form von Schnittstellen bieten kann. Dieser Sachverhalt erschwert das Einbinden in eine stark typisierte React-Anwendungsumgebung erheblich.

Zur Überbrückung dieser Gegensätze dient eine weitere Bibliothek: React-Leaflet. „React-Leaflet provides an abstraction of Leaflat as React components" heißt es auf der offiziellen Webseite[18]. Die Leaflet-Funktionalitäten sind in React-Komponenten gekapselt. React-Leaflet ist kein Ersatz der originären Leaflet-Bibliothek, sondern eine Erweiterung: „It does not replace Leaflet, only leverages React's lifecycle methods to call relevant Leaflet handlers".

4.2.4 TypeScript

TypeScript hat in den vergangenen Jahren deutlich an Beliebtheit gewonnen. Die wöchentlichen Downloads über npm haben sich seit Juli 2019 verdoppelt[19].

TypeScript bringt einige Vorteile mit sich, unter Anderem

- eine Robustheit und geringere Fehleranfälligkeit durch Typisierung,
- den Einsatz von Interfaces zur zusätzlichen Abstraktion,
- einen konfigurierbaren JavaScript-Output (ES5, ES6, usw.) und
- eine verbesserte Developer Experience und Dokumentationsfähigkeit

Der letzte Aufzählungspunkt war ein entscheidender Faktor für die Adoption von TypeScript in NorOvA Analytics. Das Projektteam plant nach Abschluss der Thesis das Werkzeug weiter auszubauen. Für eine schnelle und frustfreie Einarbeitung in den Programmcode sollte die Software idealerweise schnittstellenorientiert und stark typisiert sein. Ein weiterer, offensichtlicher Vorteil aus Entwicklersicht ist der, dass für den Einsatz einer Typisierung keine neue Sprache gelernt werden muss.

[17] Leaflet ist eine leichtgewichtige Bibliothek: Der JavaScript-Code umfasst lediglich 38 KB.
[18] vgl. https://react-leaflet.js.org/
[19] vgl. https://www.npmjs.com/package/typescript

4.3 Entwicklungsmethodologie

Nach einer Abstimmung bzgl. Anforderungen, Technologien und Bibliotheken war es Aufgabe aller Beteiligten eine für das Projekt geeignete Entwicklungsmethode zu wählen. Sie minimiert das Risiko, dass die Software den Ansprüchen nicht gerecht wird oder der geplante Funktionsumfang nicht mit der Projektlaufzeit vereinbar ist.

Das Wasserfallmodell ist eines der ältesten Modelle der Softwareentwicklung. Es ist ein lineares Modell. Die Entwicklung ist in aufeinanderfolgende Phasen separiert. Ursprünglich waren keine Iterationen vorgesehen (vgl. Abbildung 4-3). Anforderungen und Verbesserungsvorschläge an die Software, die während der Implementation aufkamen, blieben unberücksichtigt. Extreme Programming (XP) ist eine Neuinterpretation des Entwicklungsprozesses: „XP turns the conventional software process sideways.“ (Beck, 2000). Analyse, Design, Implementation und Testen werden im Rahmen von XP gleichzeitig und in Intervallen verübt (vgl. Abbildung 4-3).

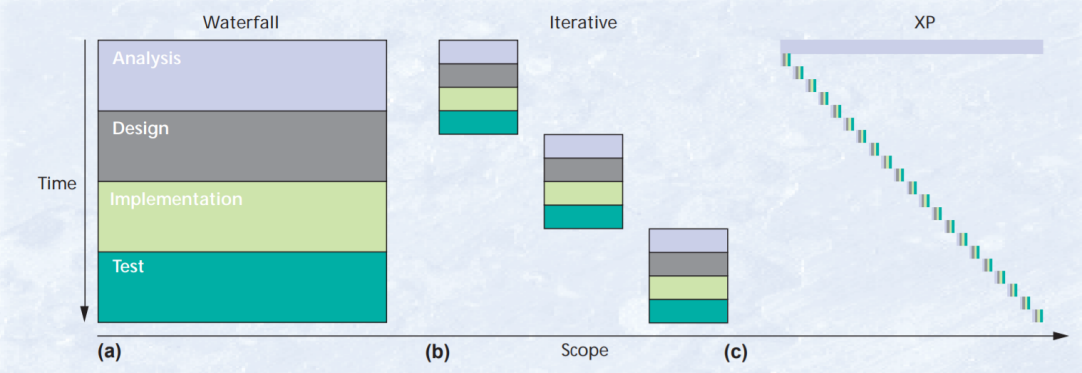

Abbildung 4-3: XP als Ableitung des Wasserfallmodells
Quelle: Beck, 2000

XP stellt das Lösen von Programmieraufgaben in den Vordergrund. Ein formales Vorgehen ist zweitrangig. Durch die Parallelisierung der Phasen ist die Entwicklung wesentlich flexibler. Es können jederzeit Ideen und Änderungswünsche eingebracht werden. Eine engere Zusammenarbeit mit Projektverantwortlichen kommt zustande.

Die Entwicklungsmethodologie von NorOvA Analytics ist eine Kombination von XP-Praktiken und evolutionärem Prototyping. Das Ziel war es, dem Kunden so früh wie möglich eine lauffähige Software zu präsentieren. Die Absprache über den Stand der Entwicklung fand wöchentlich statt, wobei Verbesserungsvorschläge entgegengenommen wurden, um die Software Stück für Stück funktionell auszubauen.

„One of the important goals [...] is to address risk early“ (Conallen, 2003). Die Zyklen von XP bieten einen Rahmen für einen ständigen Austausch mit Projektverantwortlichen, die zu jedem Zeitpunkt steuernd auf das Projekt einwirken können. Damit wird das Risiko einer Unzufriedenheit seitens des Kunden minimiert.

4.4 Designprobleme und -entscheidungen

“A number of key issues must be dealt with when designing software” (Bourque, et al., 2014). Der SWEBOK-Leitfaden adressiert geläufige Designprobleme:

1. Concurreny
2. Control and Handling of Events
3. Data Persistence
4. Distribution of Components

Concurrency: Das Design von Nebenläufigkeit beschäftigt sich mit der Aufteilung der Software in Prozesse, Tasks und Threads.

Control and Handling of Events: Zur Definition des Programmflusses muss sich der Entwickler mit der Organisation von Daten, Datenstrukturen und Algorithmen auseinandersetzen. Die Behandlung von Ereignissen ist in der Programmierung von Benutzeroberflächen ein geläufiger Mechanismus zur Regelung des Kontrollflusses.

Data Persistence: Der Punkt „Data Persistence“ adressiert die Handhabung von langlebigen Daten. Ein wichtiger Faktor bei datengetriebenen Anwendungen wie im Fall von NorOvA Analytics. Kapitel 3 deckt die Lösung dieses Designproblems ab.

Distribution of Components: Die Aufteilung von Programmcode in Komponenten ist so alt wie die Softwaretechnik selbst. Ebenso wie die Aufteilung von Programmcode in Softwarekomponenten ist die Aufteilung von Softwaresysteme auf die zur Verfügung stehende Hardware zu definieren und festzuhalten.

Während der Planungs- und Entwicklungsphase kam es zu Designentscheidungen und inkrementellen Überarbeitungen, welche die Struktur und Funktionsweise maßgeblich beeinflusst haben. Es galt folgende Designprobleme zu lösen:

Allgemeine Designprobleme:

- Aufteilung der Programmlogik auf Client und Server
- Bestimmung von Programmablauf und Ereignisbehandlung

Anwendungsspezifische Probleme (Frontend):

- Entwurf eines sinnvollen State Managements
- Interaktion zwischen Analysekomponenten und Leaflet

4.5 Aufteilung der Programmlogik

“[...] activities of designing Web applications [and therefore distributed systems] are significantly different from designing other systems: partitioning of objects onto the client or server and defining Web page user interfaces” (Conallen, 2003).

4.5.1 Verteiltes System

Unter einem verteilten System versteht man die Verteilung einer Anwendung auf zwei oder mehr räumlich getrennte Rechner. Die unabhängigen Bestandteile ergeben in ihrer Gesamtheit ein vollständiges System. Als Kommunikationsplattform dient das Web. Das Kommunikationsmittel sind Protokolle. Netzwerkprotokolle regulieren den netzwerkübergreifenden Nachrichtenverkehr und, im Falle eines verbindungsorientierten Protokolls wie TCP, den Verbindungsauf- und -abbau. „Die Erhöhung der Verarbeitungsleistung durch Parallelisierung kann ein weiteres Motiv für den Einsatz Verteilter Systeme sein“ (Schill und Springer, 2012). Von Parallelisierung betroffen sind Webserver, die das Bindeglied verteilter Systeme sind.

Einerseits ist zu klären, wie sich die Software auf die Hardware verteilt, anderseits sind Software-Komponenten, ihre Kommunikation und der Einsatz von Middleware zu definieren (vgl. Bourque, et al., 2014).

Je nach Einsatzzweck von NorOvA Analytics verteilt sich die Anwendung unterschiedlich. Während dem Entwicklungsprozess und bei der Einrichtung als autonome Applikation laufen Client- als auch Serveranwendung auf derselben Maschine. Wird NorOvA Analytics als eine Komponente in einem anderen Projekt integriert, müssen die Schnittstellen angepasst werden, damit Daten von dem zentralen Spring-Boot-Server bezogen werden können. Die Kommunikation zwischen Client und Server verläuft in beiden Fällen gleich, nämlich über die Protokolle TCP/IP und HTTP.

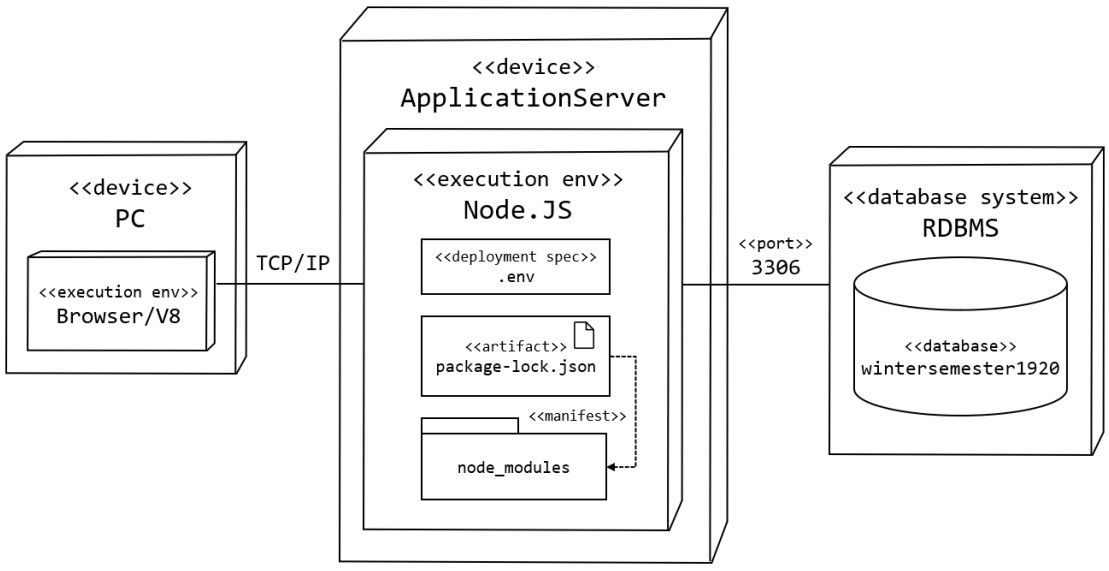

Abbildung 4-4: UML-Verteilungsdiagramm von NorOvA Analytics

NorOvA Analytics verkörpert das traditionelle Drei-Schichten-Modell, bestehend aus einer Daten-, Anwendungs- und Präsentationsschicht. Ein Verteilungsdiagramm (siehe Abbildung 4-4) beschreibt die drei Schichten hardwaretechnisch.

4.5.2 Backend

Nebenläufigkeit oder auch Parallelisierung ist eine Eigenschaft eines Systems, mehrere Befehle augenscheinlich parallel auszuführen. Ein Server muss nebenläufig arbeiten, um mehrere Clients bedienen zu können.

"The debate over whether threads or events are best suited to systems software has been raging for decades." (Dabek, et al., 2002). Das Mooresche Gesetz besagt, dass sich die Anzahl an Transistoren in integrierten Schaltkreisen pro Flächeneinheit etwa alle zwei Jahre verdoppelt. Die Thread-basierte Programmierung nutzt die Leistung von Mehrkernprozessoren. Ein Thread ist ein Teil eines Prozesses, welcher mit der Abarbeitung eines Programms beschäftigt ist. Klassische Webserver nutzen Multithreading, um mehrere Clients zu bedienen. Je mehr Anfragen an den Server gelangen, desto mehr Threads werden erzeugt, was sich negativ auf die Performanz auswirken kann. Entwickler sind verantwortlich Threads zu orchestrieren: "[...] the programmer must carefully protect shared data structures with locks and use condition variables to coordinate the execution of threads" (Dabek, et al., 2002).

Die ereignisgesteuerte Programmierung verzichtet auf die Erzeugung und gezielte Synchronisation von Threads und erzielt Nebenläufigkeit durch Ereignisse, Ereignisbehandlung und Callback-Funktionen, sodass Prozesse niemals blockieren. Ein Nachteil dieser Technik ist, dass der Programmfluss weniger logisch und mehr komplex erscheint: „The most cited drawback of the event-driven model is programming difficulty" (Darek, et al., 2002).

Nicht zuletzt wegen JavaScripts Omnipräsenz in der clientseitigen Webprogrammierung und der Tatsache, dass jeder Computer ohne Vorkehrungen mit mindestens einem JavaScript-Interpreter ausgestattet ist (vgl. Crockford, 2001), ist JavaScript eine der beliebtesten Programmiersprachen weltweit. "It is best known as the language embedded in web browsers but has also been widely adopted for server and embedded applications." (Ecma, 2020). Seit dem Aufkommen von Node.js, der ersten serverseitigen Laufzeitumgebung für JavaScript, ist es möglich Serveranwendungen in JavaScript zu schreiben, was bereits früh antizipiert wurde: „Any application that can be written in JavaScript, will eventually be written in JavaScript" (Atwood, 2007). Es existiert somit kein Sprachbruch zwischen Frontend und Backend.

Die Laufzeitumgebung baut auf Googles V8-Engine auf. V8 wandelt JavaScript mittels Just-In-Time-Kompilierung in schneller ausführbaren Maschinencode um. Das Auszeichnungsmerkmal von Node.js ist die Ereignisschleife: „The event loop is what allows Node.js to perform non-blocking I/O operations - despite the fact that JavaScript is single-threaded - by offloading operations to the system kernel

whenever possible“[20]. Die Standard-Bibliothek stützt sich dabei auf Callback-Funktionen: “Upon each connection, the callback is fired, [...]. This is in contrast to today's more common concurrency model, in which OS threads are employed”[21]. Callback-Funktionen sind Funktionen, die als Argumente anderen Funktionen übergeben werden. Sie definieren was zu tun ist, sobald die übergeordnete Funktion terminiert und die Callback-Funktion aufruft.

Aufgrund des ereignisgesteuerten Kerns sind Node.js-Server äußerst performant und leicht skalierbar. Node.js ist für Anwendungen mit einem großen Datenbestand und vielen Datenbankabfragen geeignet. Ein weiterer Einsatzzweck von Node.js liegt in der Umsetzung von Echtzeitanwendungen. Eine Echtzeitaktualisierung ist für eine Analysesoftware denkbar.

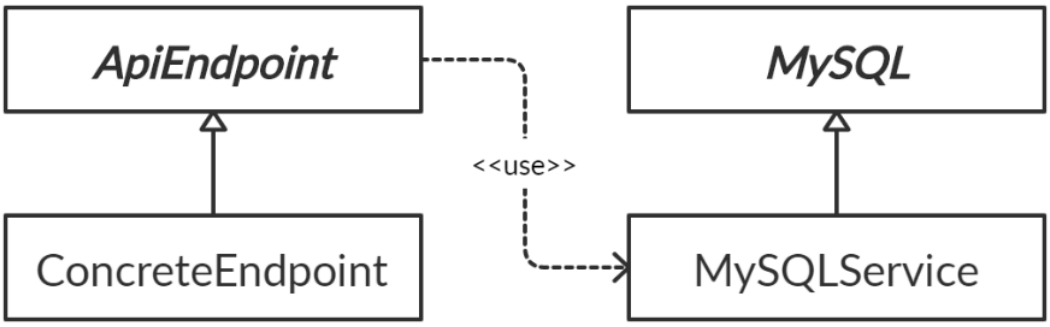

Abbildung 4-5: Vereinfachtes UML-Klassendiagramm des Backends

Wie in Kapitel drei beschrieben, handelt es sich bei der zentralen Datenbank des Spring-Boot-Servers um eine MySQL-Datenbank. Die Ausgangslage der Entwicklung von NorOvA Analytics war eine Kopie des Datenbestandes mit den Daten aus dem letztjährigen Wintersemester. Die Anwendungsschicht kommuniziert über Port 3306 mit dem relationalen Datenbankmanagementsystem (vgl. Abbildung 4-4, S. 26). Ein selbstgeschriebener Dienst nimmt dabei die Position als Vermittler zur Datenbank ein (vgl. Abbildung 4-5). Der Dienst stellt Methoden zur Konkatenation von Queries und Rückgabe der Ergebnismenge bereit. Wie Abbildung 4-5 illustriert, ist dafür ein klassenbasierter Dienst namens `MySQLService` zuständig, welcher Attribute von der abstrakten Oberklasse `MySQL` zum Verbindungsaufbau erbt.

Das Backend fungiert als ein zustandsloser Thin-Server. Der Fokus liegt auf einer reinen Daten-API ohne Session-Management (vgl. Kapitel 2.3). Die Schnittstelle des Servers verkörpert die REST-Paradigmen (vgl. Kapitel 2.2). Der Client sendet

[20] vgl. https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/
[21] vgl. https://nodejs.org/en/about/

initiativ eine GET-Anfrage an die Zugangspunkte der Serveranwendung (sog. *Endpoints*, dt. Endpunkte), die laut REST-Bedingungen einer URI entsprechen. Es ist eine weit verbreitete Konvention den Endpunkten selbstsprechende Namen zu geben. So kann unter dem Endpunkt `/api/users/` mit Benutzern als Ergebnis gerechnet werden. Das erleichtert Entwicklung und Nutzung der API.

Die Endpunkte sind an den Tabellennamen der Datenbank angelehnt. Datensätze der Tabelle `tracking_data` können über die gleichnamige Endpunkt-Klasse bezogen werden. Endpunkte erben von der abstrakten Oberklasse `ApiEndpoint`, welche die statischen Dienste des `MySQLService` nutzen, um Daten zu beziehen. Die konkreten Unterklassen sind unter dem Namen `ConcreteEndpoint` zusammengefasst.

4.5.3 REST-API

Dynamische Bindungen sind ein fundamentales Konzept objektorientierter Programmiersprachen. Eine zur Laufzeit dynamisch aufgebaute Bindung zwischen zwei Objekten erlaubt Abstraktion, Flexibilität und Erweiterbarkeit, da neuer Code von einer alten unveränderten Codebasis aufgerufen werden kann, vorausgesetzt die Objekte erben oder implementieren von einer intakten Oberklasse bzw. Schnittstelle (vgl. Milton und Schmidt, 1994). „Dynamic Dispatch“ baut auf diesem Mechanismus auf, um Methoden dynamisch zur Laufzeit zu binden und aufzurufen: „Typically it uses runtime type information to lookup, or bind to, the proper function.“ (Milton und Schmidt, 1994). „Dynamic Dispatch“ ist von einem URL-Dispatch abzugrenzen. Unter dem URL-Dispatch versteht man die Aufteilung der URL in Teilzeichenketten zur Bestimmung der weiteren Verarbeitung. Gelangt eine Anfrage an das Backend wird die URL in Endpunkt, Methode und Query zerlegt und ist wie folgt aufgebaut:

```
[Serveradresse]:[Port]/api/[Endpunkt]/[Methode]?[Query]
```

Listing 1: Bestandteile einer API-URL

Serveradresse und Port können je nach Server-Konfiguration variieren. Ein Port kann über Umgebungsvariablen festgelegt werden. Das `api`-Präfix kennzeichnet einen Daten-Endpunkt unter der Serveradresse. Der Methodenname aus der URI muss mit einer öffentlichen Methode der Klasse übereinstimmen. Abhängig von dem Endpunkt wird die korrespondierende Klasse importiert und die Methode aufgerufen.

Idealerweise wird der Programmcode auf mehrere Dateien verteilt, um die Software modular, wartbar und austauschbar zu gestalten. Damit die Anwendung wie gewünscht funktioniert, müssen Programmteile importiert werden. Ein dynamischer Import ist ein Sprachmerkmal von JavaScript und ist das Gegenstück zum statischen Import. Ein Import ist statisch, sobald er am Anfang der Datei und vor der Deklaration von Programmlogik stattfindet. Dynamische Importe, auf der anderen Seite, können im Programm eingebettet werden. Das Laden von Programmteilen kann an Bedingungen geknüpft werden, z.B. wenn eine gewisse Kontrollstruktur durchlaufen wird oder ein Attribut einen konkreten Wert hält.

```
const endpoint = new (await import(`.${urlEndpoint}`)).default()
result = await endpoint[method](query)
```

Listing 2: Dynamic Dispatch durch einen dynamischen Import

Das Unkonventionelle an diesem Ansatz ist die Tatsache, dass, damit der Import glückt, die Quelltextdatei eine von `ApiEndpoint` abgeleitete Klasse exportieren und sich in dem Ordner `api` befinden muss. Das Argument der Importanweisung, egal ob statisch oder dynamisch, ist entweder ein absoluter oder relativer Pfad zur Quelltextdatei. Die Konstante `urlEndpoint` hält die Zeichenkette `/api/[Endpunkt]` und wird vom Server als ein Pfad interpretiert. Handelt es sich beim Endpunkt um z.B. `/api/users/get?username=anonymous`, entspricht der relative Pfad `/api/users`. Das Programm versucht den Inhalt der Datei `users` zu importieren.

Methoden können sowohl über die Punkt- als auch Bracket-Notation gebunden (und aufgerufen) werden. Damit der Methodenname durch einen dynamischen Wert bestimmt werden kann, ist der Einsatz der Bracket-Notation zwingend notwendig. Die Punkt-Notation wirft in diesem Zusammenhang eine Fehlermeldung, weil der Compiler einen Syntax-Fehler erkennt. Der Code aus Listing 1 befindet sich in der Datei `index.ts`. Listing 2 zeigt die vorgesehene Verzeichnisstruktur.

```
 /src
|__ index.ts
|__ /api
    |__ users.ts
    |__ routes.ts
    |__ tracking_data.ts
```

Listing 3: Auszug der Server-Verzeichnisstruktur

Das API-Design verspricht einen hohen Grad an Flexibilität. Zur Erstellung von neuen Endpunkten müssen keine konkreten URIs und Funktionen definiert werden. Stattdessen ist es vorgesehen, eine Datei im Verzeichnis `api` anzulegen. `ApiEndpoint` stellt eine triviale `get`-Methode bereit. Es steht dem Entwickler offen, Methoden für komplexere Datenbankabfragen in der Unterklasse zu implementieren. Das verzeichnisorientierte und dynamische Design der REST-API zeichnet das Backend aus.

Die Serveranwendung gehört logischerweise zur Anwendungsschicht. Schwieriger wird es bei der Betrachtung und Einordnung der Clientanwendung. “When building Web applications, important consideration must be given to the use of business logic on the client.” (Conallen, 2003). Das Frontend ist eine SPA und ein Thick-Client. Es kommt zu einem erhöhten Programmieraufwand im Frontend (vgl. Kapitel 2.3). Das darf nicht vernachlässigt werden, wenn es darum geht, die Präsentationsschicht von der Anwendungsschicht abzugrenzen. Bei verteilten Systemen ist es naheliegend, das Backend als die Anwendungs- und das Frontend als Präsentationsschicht zu sehen. Nichts wäre im Fall von NorOvA Analytics ferner von der Wahrheit. Der Großteil der Programmlogik befindet sich im Client. Eine klare Abgrenzung zwischen Präsentation und Logik ist nicht möglich. Ein Manko von React und SPA allgemein.

4.5.4 Frontend

Eine interaktive und intuitive Benutzeroberfläche ist das Herzstück webbasierter Anwendungen und nicht selten Ausgangspunkt des Designs. Eine grobe Skizzierung der Benutzeroberfläche ist ein effektiver erster Schritt und erleichtert das Erkennen von Komponentenstrukturen. Die offizielle React-Dokumentation rät dazu, einen Mock-Up zu entwerfen und Komponentengrenzen zu ziehen[22].

Mock-Ups liefern technisch wertvolle Informationen für Entwickler und sind für den Kunden leicht verständlich (vgl. Mukasa und Kaindl, 2008). Der Zwischenschritt besitzt Parallelen zum Reverse Engineering, indem am Anfang ein theoretisches Endprodukt diskutiert wird. Nach dem Motto „What you see is what you get“ kann der Kunde überprüfen, wie die Entwicklung verläuft und ob die Software planmäßig umgesetzt wird.

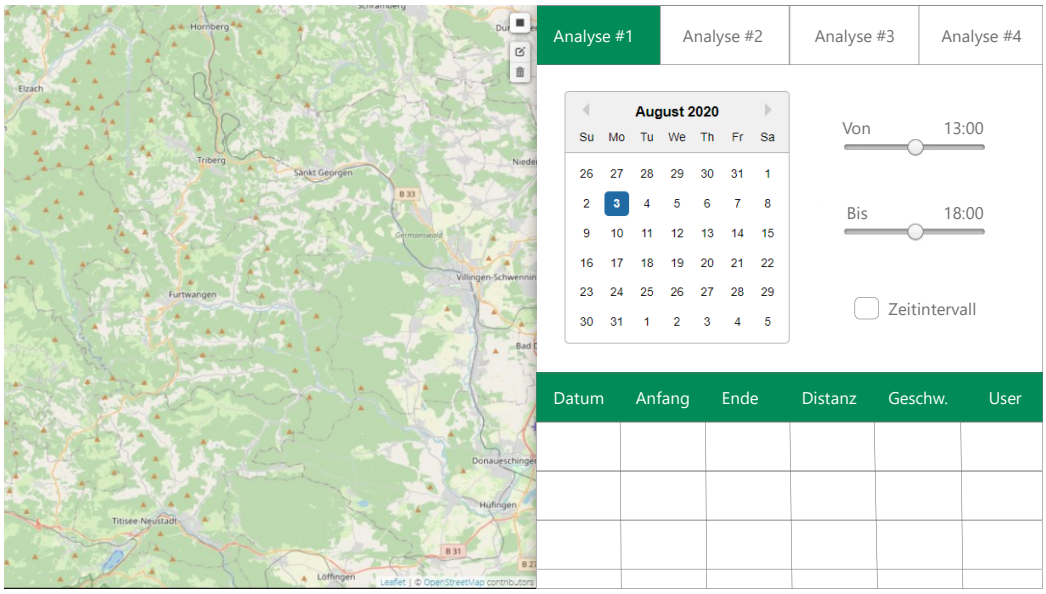

Abbildung 4-6: Mock-Up der Benutzeroberfläche

Abbildung 4-6 zeigt einen frühen Sketch von NorOvA Analytics. Die Umwandlung von gesammelten Ideen und Anforderungen in eine ansprechende und sinnvoll aufgeteilte Benutzeroberfläche ist ein schwieriges Unterfangen. Die interaktive Karte ist zweifelslos Hauptaugenmerk der Anwendung und sollte dementsprechend optisch hervorgehoben werden. Demgegenüber stehen die Analysekomponenten, die Konfigurationsmöglichkeiten bieten und die Ergebnisse tabellarisch anzeigen. Sowohl die Konfiguration als auch die Visualisierung sind wichtige Faktoren einer Datenanalyse. Das Ergebnis ist eine 50/50-Aufteilung des Viewports (vgl. Abbildung 4-6).

[22] vgl. https://reactjs.org/docs/thinking-in-react.html

Ein typischer Anwendungsfall ist das Filtern nach Wochentag und Stunden, um Stoßzeiten zu analysieren. Der Mock-Up orientiert sich an diesem Beispiel und zeigt ein denkbares Layout einer Zeitanalyse. Ein Kalender und zwei Schieberegler sind zur Konfiguration vorgesehen. Optional wird über das Setzen eines Hakens bei der Checkbox „Zeitintervall“ ein zweiter Kalender gerendert, der genutzt werden kann, um mehrere aufeinanderfolgende Wochentage zu analysieren. Auf diese Weise kann das Projektteam Wochen, Monate oder gar Semester miteinander vergleichen.

Aus dem Mock-Up werden Komponenten und ihre Schachtelung ersichtlich. Eine vorläufige Modellierung der erkennbaren Komponenten als DOM-Baum ist nützlich, um die Erkenntnisse aus dem Sketch festzuhalten und ein vorläufiges Template zu entwerfen (siehe Abbildung 4-7).

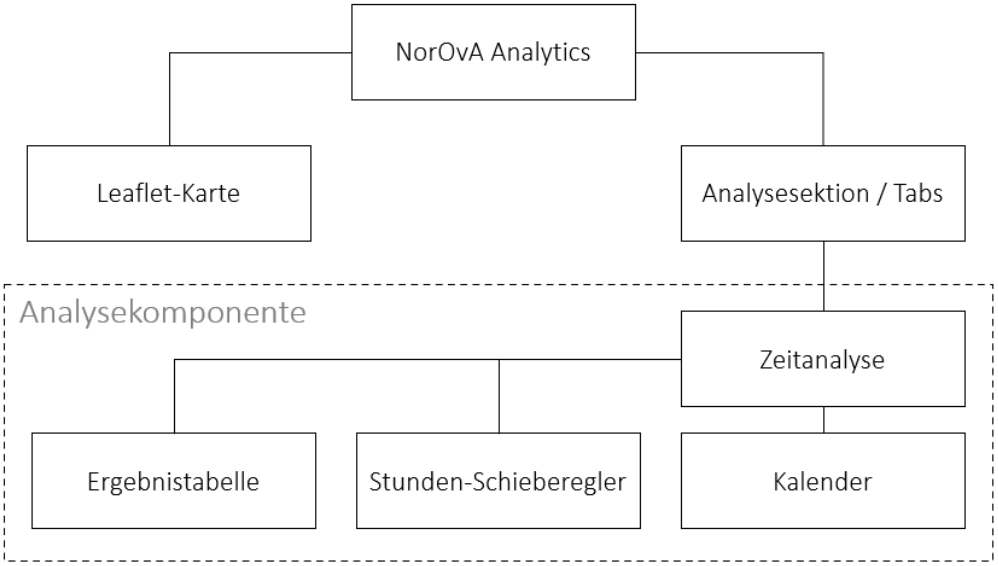

Abbildung 4-7: Komponenten als Baumstruktur

NorOvA Analytics muss aus Gründen, die in Kapitel 4.1 aufgelistet sind, als eine React-Komponente umgesetzt werden und ist die Wurzel des Baumes. In ihr werden die Leaflet-Karte und die Analysereiter gerendert. Analysekomponenten können weitere Komponenten schachteln, wie z.B. die Ergebnistabelle oder Input-Elemente (in diesem Beispiel Kalender und Schieberegler).

Das Resultat einer Aufteilung der Logik wirft weitere Fragen auf: „Wie sieht die inter-komponenten Kommunikation aus?“ und „Wie werden Analysekomponente und Streckenabbildungen aufeinander abgestimmt und synchronisiert?“.

4.6 Programmablauf und State Management

Die Konzeption der inter-komponenten Kommunikation ist eine Schlüsselaufgabe und entscheidet letztendlich über den Programmablauf und das State Management.

JavaScript ist ereignisgesteuert, d.h. der Programmfluss kommt durch Ereignisse und Ereignisbehandlung zustande. Dieses Paradigma hat historische Wurzeln in der frühen Mensch-Maschinen-Kommunikation (vgl. Garian und Shaw, 1993). Ereignissteuerung hat sich als der defacto Standard für die Programmierung von grafischen Benutzeroberflächen durchgesetzt. Elemente können EventListener registrieren.

Ereignisse sind die direkte Schnittstelle zum Benutzer und initiieren Veränderungen der Anwendung. Eine Veränderung der Anwendung ist in React gleichbedeutend mit einer Zustandsveränderung von Komponenten. Wie in Kapitel 4.2.1 adressiert, unterstützt React kein Two-Way-Data-Binding. Eingriffe in die Benutzeroberfläche haben keine direkte Auswirkung auf Attributwerte. Der Zustand muss manuell über `setState` gesetzt werden, woraufhin sich die Komponente und demzufolge ein Teil der Benutzeroberfläche aktualisiert.

Komponenten ähneln Funktionen. Sie akzeptieren einen Input (Props) und liefern JSX-Elemente als Output. Props dienen dazu eine Komponente in ihren initialen Zustand zu setzen. Beispielsweise ist im Fall der Zeitanalyse ein Startdatum denkbar, das im Vorfeld selektiert ist. Props sind das Bindeglied zwischen Eltern- und Kindskomponenten und nicht auf primitive Datentypen beschränkt. Es können auch Objekte und Funktionen übergeben werden.

Neben internen Zuständen von Komponenten ist der anwendungsweite Zustand, sprich die Aggregation aller gegenwärtig aktiven Komponenten und deren Zustände, zu managen. Das kann je nach Größe der Applikation schnell komplex werden.

React stellt zahlreiche Möglichkeiten bereit, den anwendungsweiten Zustand mit so wenig Komplexität wie möglich zu bewältigen. Ein prominentes Beispiel ist die Context-API, die Attribute global zur Verfügung stellt: „Context provides a way to pass data through the component tree without having to pass props down manually at every level“[23]. Diese Alternative bietet sich an, wenn viele oder weit auseinander liegende Komponenten ihren Zustand an identischen Attributen festmachen und eine Übergabe über Props nicht für alle Zwischenkomponenten sinnvoll erscheint.

Ein State Management über die Context-API wurde aus zwei Gründen abgelehnt: einerseits ist die Tiefe des Komponenten-Baumes überschaubar, andererseits darf der Zustand von NorOvA Analytics als migrierte Komponente unter keinen Umständen mit dem Zustand der übergeordneten Applikation kollidieren. Die Wahl fiel auf eine traditionelle Kommunikation über Props und Callback-Funktionen, um den Zustand von NorOvA Analytics zu kontaminieren.

[23] vgl. https://reactjs.org/docs/context.html#when-to-use-context

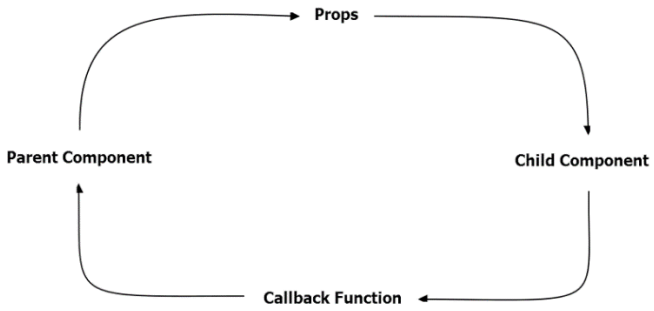

Abbildung 4-8: Komponenten-Kommunikation in React
Quelle: Arnold, 2017, zitiert nach medium.com

Eine Elternkomponente übergibt Props an die Kindskomponente, um sie in ihren initialen Zustand zu versetzen. Funktionen können als Props übergeben und jederzeit in der Kindskomponente über die Membervariable `props` aufgerufen werden. Auf dieser Technik beruht das Emittieren von Ereignissen, um Elternkomponente über Zustandsveränderungen in der Kindskomponente zu unterrichten und ggf. mit dem Kind zu synchronisieren. Eine Synchronisation ist gleichbedeutend mit der Übergabe von veränderten Props, woraufhin das Kind mit einer Zustandsveränderung reagiert.

Für den Programmablauf bedeutet das konkret, dass auf eine Benutzereingabe eine Ereignisbehandlung isoliert in der Komponente oder über Ereignis-Emission in der Elternkomponente folgt. Das Letztere ist bei Input-Elementen sinnvoll, die selbst nichts mit dem Wert anfangen können und ihn an die Elternkomponente emittieren. Ist die Emission bis zu der Analysekomponente vorangeschritten, kommt es zu einer Zustandsveränderung und ggf. zum Datenbeschaffungsprozess.

4.7 Datenbeschaffungsprozess

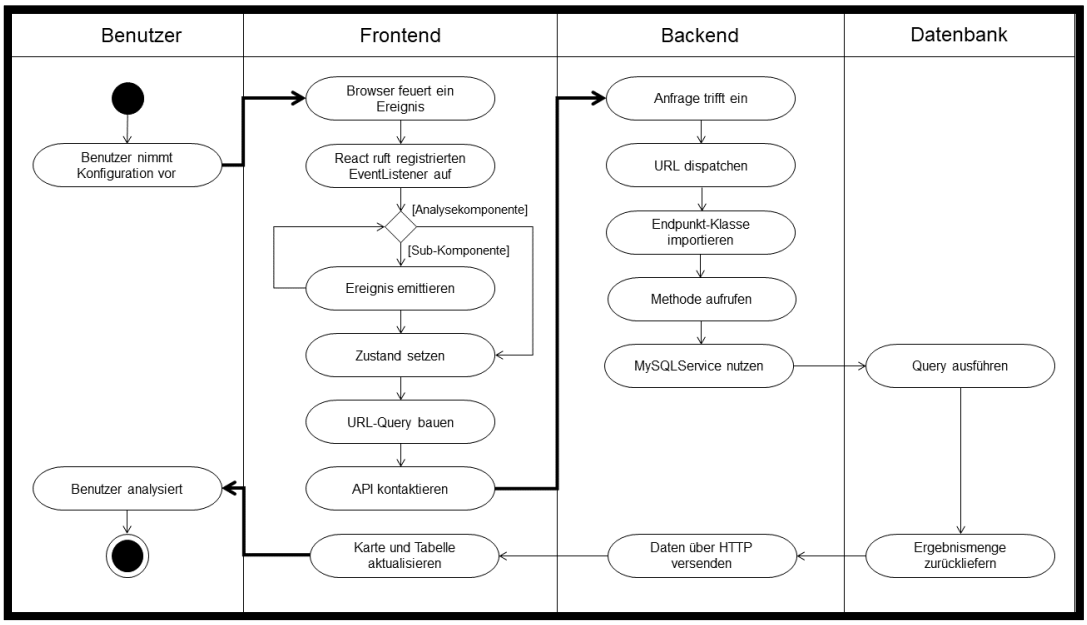

Abbildung 4-9: UML-Aktivitätsdiagramm des Datenbeschaffungsprozesses

Nach erfolgreicher Emission und Zustandsveränderung durchläuft das Programm Kontrollstrukturen, um zu überprüfen, ob die Konfiguration eine Datenbeschaffung zufolge hat oder nicht. Werden neue Daten benötigt, bindet die Analysekomponente alle Konfigurationsoptionen als Query-Parameter an die Serveradresse an.

Das Backend dispatcht die URL und importiert die Endpunkt-Klasse dynamisch (vgl. Kapitel 4.5.3). Abhängig von der kontaktierten Serveradresse ruft das Backend die gewünschte Methode auf und übergibt die Query-Parameter als Argument. Der Dienst `MySQLService` ist die programmatische Schnittstelle zur Datenbank.

Die selektierten Datensätze werden an das Backend zurückgegeben, ehe sie über das Netzwerk und im JSON-Format an das Frontend gelangen.

Um dem Beispiel der Zeitanalyse treu zu bleiben, wird folgendes Analyseszenario betrachtet: Für eine Analyse aller Strecken vom 13. Januar zwischen sechs und zwölf Uhr müsste das Datum in dem Kalender ausgewählt und die Schieberegler angepasst werden. Das Frontend generiert einen Schnappschuss der aktuellen Konfigurationen und sendet eine GET-Nachricht an folgende Serveradresse:

```
/api/tracking_data/getByDate?year=2020&month=1&day=13&startHour=6&endHour=12
```

Listing 4: Beispielhafte Serveradresse zur Datenbeschaffung

Die Serverantwort umfasst Trackingdaten im JSON-Format. Trackingdaten sind angereicherte Koordinatenpunkte (vgl. Kapitel 3.2 und Listing 5).

```
{

    “id”: 14707612,
    “lat”: 47.7442993,
    “lng”: 8.8448868,
    “speed”: 0,
    “time”: “2020-01-13T05:22:06.000Z”,
    „user“: 213106,
    „route“: 15181263

}
```

Listing 5: Trackingdatensatz im JSON-Format

Listing 5 zeigt ein Trackingdatensatz der Serverantwort. Die Datenbankabfrage ist ein SELECT-Statement der Tabelle `tracking_data`. Die Daten werden nahezu[24] unverändert an das Frontend gesendet, was bedeutet, dass die Trackingdaten nicht nach Routenzugehörigkeit gruppiert sind. Um eine Strecke auf der Karte abzubilden

[24] Aus Sicherheitsgründen findet ein Mapping der Attributnamen statt, da die Spaltennamen der Tabelle ansonsten in der Serverantwort ausgelesen werden können.

müssen alle zusammengehörigen Koordinatenpunkte in einem Datenobjekt gekapselt werden. Die Gruppier- und Mapping-Funktionen befinden sich im Frontend.

Nachdem die Programmlogik aufgeteilt, der ereignisgesteuerte Programmablauf beschrieben, das State Management ausgewählt und die Datenbeschaffung modelliert wurde, ist es nun an der Zeit zu klären, wie die erhaltenen Mobilitätsdaten als eine Strecke auf der Karte abgebildet werden.

4.8 Streckenabbildung

Wie in Kapitel 4.2.3 bereits aufgegriffen, lag eine der größten Herausforderungen in der Überbrückung bzw. Harmonisierung der grundverschiedenen Funktionsweisen von React und Leaflet. Die Intention hinter dem Einbetten von React-Leaflet ist es, den Kartengrundriss und die Tile-Schichten[25] unkompliziert als React-Komponenten zu integrieren. Die von der Dokumentation vorgesehene Initialisierung in JavaScript findet in den Lebenslaufmethoden statt. Neben einer interaktiven Kartendarstellung umfasst die originäre Leaflet-Bibliothek zusätzlich Vektor-Ebenen und geometrische Figuren, wie z.B. Linien, Polygone und Kreise.

Das Problem liegt nicht in der Darstellung der Karte, sondern in der Interaktion zwischen Analysekomponente und Karte. Analysekomponente rendern den Konfigurationsbereich, überwachen Veränderungen und regeln die Beschaffung von Daten. Was geschieht, sobald die Daten im Frontend eintreffen? Sollen Analysekomponenten auf die Karte zugreifen? Wenn ja, auf welche Art und Weise?

Zunächst ist die Frage zu beantworten, wie Vektoren gezeichnet werden sollen. Es gibt genau zwei Möglichkeiten, und zwar: direkt oder indirekt. Mit „direkt“ ist gemeint, die bereitgestellten Methoden der Leaflet-Bibliothek zu nutzen. Andernfalls können Vektoren indirekt über die gleichnamigen Komponenten der React-Leaflet-Bibliothek gezeichnet werden. Koordinatenpunkte werden als Props übergeben.

Der Komponenten-Ansatz bietet sich an, wenn im Vorfeld bekannte Routen oder Markierungen abgebildet werden, aber nicht, wenn sich der Karteninhalt dynamisch ändert. Für eine dynamische Abbildung müssten Kontrollstrukturen direkt im JSX der NorOvA Analytics Komponente untergebracht werden. Je mehr unterschiedliche Vektorengruppen eingesetzt werden, desto unübersichtlicher wird die Komponente. Aus diesen Gründen wurde ein Zeichnen über die Leaflet-Methoden bevorzugt.

[25] Eine Tile-Schicht ist ein Bildauszug einer Karte. Tile-Schichten unterscheiden sich in Farbe und Form von Strecken, Gebirge, Gewässer, usw.

Mit Berücksichtigung der Designprinzipien Single Responsibility und Seperation of Concerns zur Aufteilung von Programmlogik und Minimierung von Abhängigkeit, ist eine lose Kopplung wünschenswert. Die Leaflet-Bibliothek definiert die Karte als ein Objekt mit Methoden zur Manipulation. Eine Referenz des Objektes auf Seiten der Analysekomponente entspricht einer engen Kopplung und ist mit einem hohen Grad an Abhängigkeit verbunden. Um eine lose Kopplung zu erreichen ist es sinnvoll eine Zwischenkomponente (Middleware) einzusetzen (siehe Abbildung 4-10).

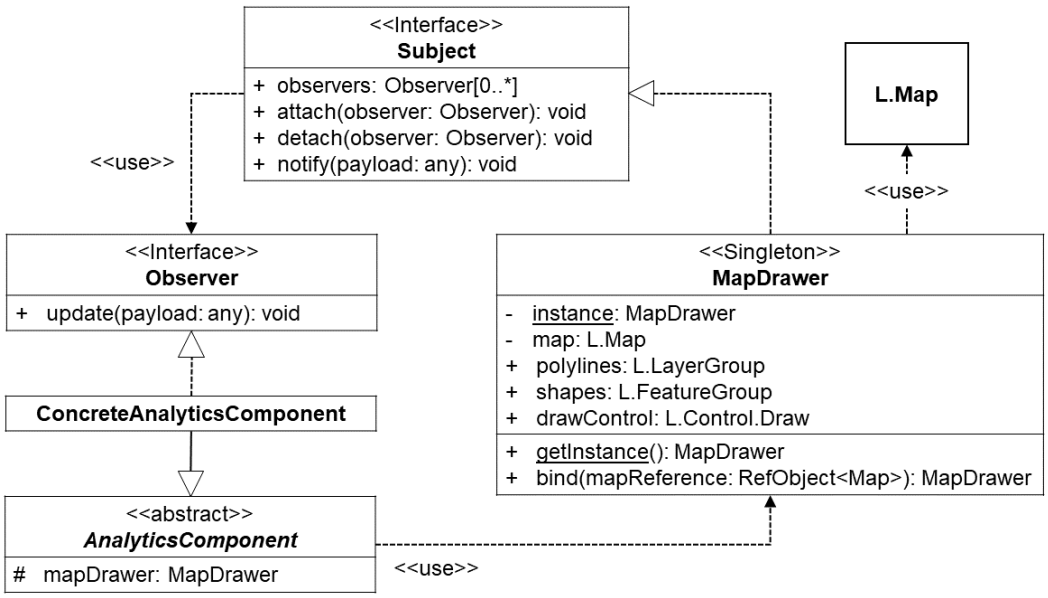

Abbildung 4-10: MapDrawer-Middleware

Zur Umsetzung der Middleware wurden zwei Entwurfsmuster kombiniert: das Singleton- und Observer-Muster. Der Zweck des Singleton-Musters ist das Sicherstellen der Existenz einer einzigen Klasseninstanz (vgl. Gamma, 1993). Dies wird ermöglicht, indem die Sichtbarkeit des Konstruktors auf die Klasse reduziert wird und die Instanziierung und der Abruf des Objektes über eine statische Methode erfolgt. Die Klasseninstanz wird als Variable abgelegt. Die Methode `getInstance` durchläuft eine Kontrollstruktur, die überprüft, ob die Klassenvariable instanziiert ist. Ist das der Fall wird diese zurückgeliefert, andernfalls wird eine Instanz erzeugt und gespeichert. Der Einsatz des Entwurfsmusters ist sinnvoll, da ausschließlich eine Karteninstanz (`L.Map`) existiert und für den Informationsfluss zwischen Analysekomponente und Karte nur ein Objekt allokiert werden muss.

Statt einer Referenz auf die Karte selbst, was eine enge Kopplung zur Folge hat, referenziert die abstrakte Oberklasse `AnalyticsComponent` die Klasse `MapDrawer`, die wiederum `L.Map` referenziert. Diese Referenz muss zunächst gesetzt werden. Es sind folgende Schritte notwendig:

1. Erzeugung eines `RefObject` und Kopplung mit der `Map`-Komponente aus der React-Leaflet-Bibliothek:

```
private mapReference: React.RefObject<Map> = React.createRef()
<Map ref={this.mapReference}>
```

Listing 6: Kartenreferenz

2. `MapDrawer` instanziieren, das `RefObject` übergeben und die Karte binden:

```
MapDrawer.getInstance().bind(this.mapReference)
```

Listing 7: Singleton-Instanziierung und Kartenbindung

Ist die Karte gebunden, rendert die Wurzel-Komponente NorOvA Analytics die Analysekomponenten. Die Konstruktoren von Analysekomponente und Basisklasse werden aufgerufen. Analysekomponenten erben das Attribut `mapDrawer` und können auf der Karte zeichnen. Aus Gründen der Übersichtlichkeit sind die Methoden nicht im Klassendiagramm genannt.

Diese Lösung deckt den Anwendungsfall ab, in dem die Analysekomponente eine Veränderung der Konfiguration registriert, Trackingdaten vom Backend bezieht und die Koordinatenpunkte an die Klasse `MapDrawer` übergibt. In diesem Szenario ist die Analysekomponente initiativ. Es gibt jedoch auch den Anwendungsfall, in dem eine Interaktion mit der Karte eine Datenbeschaffung auslöst. Die Komponente muss in diesem Fall über das Resultat der Interaktion, i. d. R. eine Menge von Koordinaten, unterrichtet werden.

Zur Realisierung dieses Anwendungsfalles ist das Observer-Muster geeignet. Der Zweck des Musters ist die Definition einer 1-zu-N-Abhängigkeit zwischen Objekten, damit im Fall einer Zustandsveränderung des beobachteten Objektes alle Beobachter benachrichtigt und aktualisiert werden (vgl. Gamma, 1993). Die abhängigen Objekte sind die Beobachter. Das beobachtete Objekt wird als Subjekt bezeichnet. Ein Subjekt kann eine Menge von Beobachtern registrieren, um sie bei einer Zustandsveränderung zu kontaktieren. Subjekt und Beobachter implementieren die gleichnamigen Schnittstellen, um die Methodenaufrufe zu realisieren.

Für die wichtigste und aufschlussreichste Analysemöglichkeit, die Routenanalyse, ist es vorgesehen zwei Rechtecke auf der Karte zu zeichnen, um Start- und Endpunkt zu bestimmen. Durch benutzerdefinierte Ereignisse der Leaflet-Bibliothek können die Koordinaten aller Eckpunkte in Erfahrung gebracht werden. Koordinaten gelangen über das Zusammenspiel von `notify` und `update` an die Beobachter. Sie registrieren sich während ihrer Initialisierung beim Subjekt.

```
componentDidMount() {
    this.mapDrawer.attach(this);
}
```

Listing 8: Registrierung von Beobachtern

Die Ereignisbehandlung von Zeichnungen sieht wie folgt aus:

```
this.map.on(L.Draw.Event.CREATED, ({ layer }) => {
      layer.addTo(this.shapes);
      this.notify(layer.getBounds());
});
```

Listing 9: Ereignisbehandlung von Zeichnungen

Die Leaflet-Klasse `L.Map` stellt die Methode `on` bereit, die genutzt werden kann, um auf Leaflet-spezifische Ereignisse zu reagieren. Das erste Argument der Signatur ist die Kategorie des Ereignisses als Zeichenkette. In Listing 9 wird stattdessen eine Konstante genutzt. Das zweite Argument ist, typisch für eine Ereignisbehandlung in JavaScript, eine Callback-Funktion, die ausgeführt wird, sobald das Ereignis eintritt. Das Argument der Callback-Funktion ist die gezeichnete Ebene. Die Zeichnung wird zu einem Sammelobjekt hinzugefügt. Das Subjekt ruft `notify` auf und iteriert über alle registrierten Beobachter und ruft die Methode `update` auf, um die Längen- und Breitengrade der Eckpunkte zu übergeben (vgl. Listing 10).

```
private notify(payload: any) {
    for (const observer of this.observers) {
          observer.update(payload);
    }
}
```

Listing 10: Benachrichtigung von Beobachtern

Der Beobachter erhält die Koordinaten vom Subjekt (vgl. Listing 11) und startet die Datenbeschaffung, sobald zwei Rechtecke gezeichnet wurden.

```
async update(payload: LatLngBounds) {
   if (this.state.isFetchingData) return;
   if (!this.origin || (this.origin && this.destination)) {
        this.origin = payload;
        this.destination = undefined;
        return;
   }
   if (this.origin && !this.destination) {
        this.destination = payload;
        if (this.props.onRouteEnd) {
              this.props.onRouteEnd([this.origin, this.destination]);
        } else {
              this.setState(state => ({
                    ...state,
                    isFetchingData: true,
                    route: [this.origin, this.destination],
              }));
        }
   }
}
```

Listing 11: Update-Methode auf Seiten der Routenanalyse

5 Endprodukt

Die Benutzeroberfläche des Endproduktes (vgl. Abbildung 5-1) orientiert sich an dem skizzierten Mock-Up aus Kapitel 4.5.4. Karte und Analysesektion nehmen die Hälfte des Bildschirmes ein. Die Analysekomponenten werden gerendert, sobald der Benutzer den entsprechenden Reiter auswählt. Das Endprodukt deckt insgesamt vier Analysemöglichkeiten und eine Kombination zweier Analysekomponenten ab.

5.1 Benutzeranalyse

Die Benutzeranalyse erlaubt es, die Streckenaufzeichnungen von registrierten und anonymen Benutzern einzusehen. Sie ist die vorausgewählte Komponente, sobald die Anwendung die initiale Datenbeschaffung abgeschlossen hat. Ehe der Benutzer aktiv analysieren kann, werden Benutzer- und Routendaten vom Backend geladen und in der IndexedDB als JSON-Objekte abgelegt, die über ihren Index abrufbar sind. Dies hat zur Folge, dass beim Wechsel auf den Reiter Benutzer die Daten nicht über das Netzwerk bezogen werden müssen, sondern aus der clientseitigen IndexedDB geladen werden können. Die Zeitersparnis beträgt etwa hundert Millisekunden.

Die korrespondierenden Trackingdaten werden nach Wahl der Strecke aus dem Dropdown-Menü über die Route-ID von der API geladen. An das Frontend gelangen Koordinatenpunkte, die als eine Polyline gezeichnet werden (siehe Abbildung 5-1).

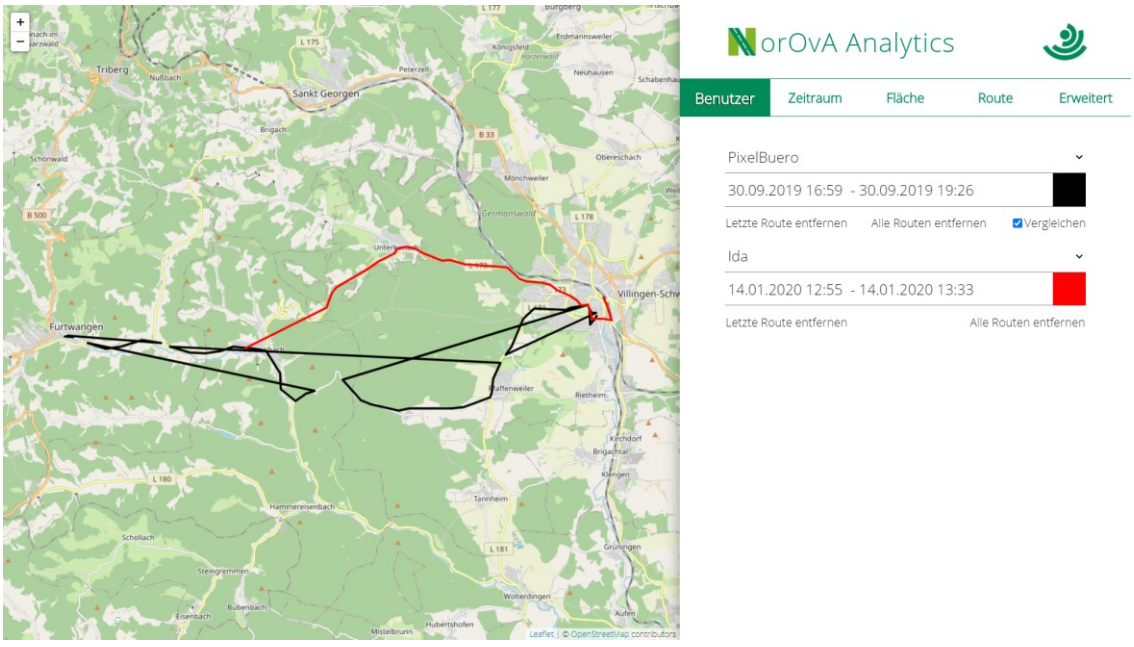

Abbildung 5-1: Streckenvergleich zwischen Benutzern

Durch das Setzen eines Hakens bei Vergleichen wird das Menü erneut gerendert, wodurch die Strecken von zwei Benutzern miteinander verglichen werden können. Zur Wiederverwendung von Programmcode ist dieses Problem mit Rekursion gelöst:

die Benutzeranalyse rendert in ihrem JSX abhängig vom Zustand der Checkbox eine weitere Benutzeranalyse, die durch den gezielten Einsatz von Props in den korrekten Zustand versetzt wird. Dazu zählt z.B. eine zufällig generierte Farbe, statt der Farbe Schwarz zur klaren Differenzierung der Streckenaufzeichnungen.

Wie in Kapitel 3.2 bereits angedeutet, ermöglicht diese Analysekomponente einen Vergleich zwischen Benutzern, um Leistungen bei Gewinnspielen nachzuweisen oder die Streckenaufzeichnungen der aktivsten Benutzer einzusehen.

5.2 Zeitraumanalyse

Die Zeitraumanalyse integriert eine Bibliothek namens React-Datepicker für eine einfache Umsetzung der Kalenderkomponente. Die beiden Schieberegler sind in einer eigenständigen Komponente gekapselt. Der Kalender und die Schieberegler emittieren Wertveränderungen an die übergeordnete Analysekomponente, woraufhin diese eine Zustandsveränderung eingeht und ggf. einen Datenbeschaffungsprozess startet.

Durch das Setzen eines Hakens bei Periode wird wie bei der Benutzeranalyse die Komponente rekursiv gerendert, wobei der erste Kalender das Startdatum und der zweite Kalender das Enddatum bestimmt. Sowohl die Kalender als auch die Regler sind aufeinander abgestimmt, sodass keine unlogischen Zeitspannen auswählbar sind.

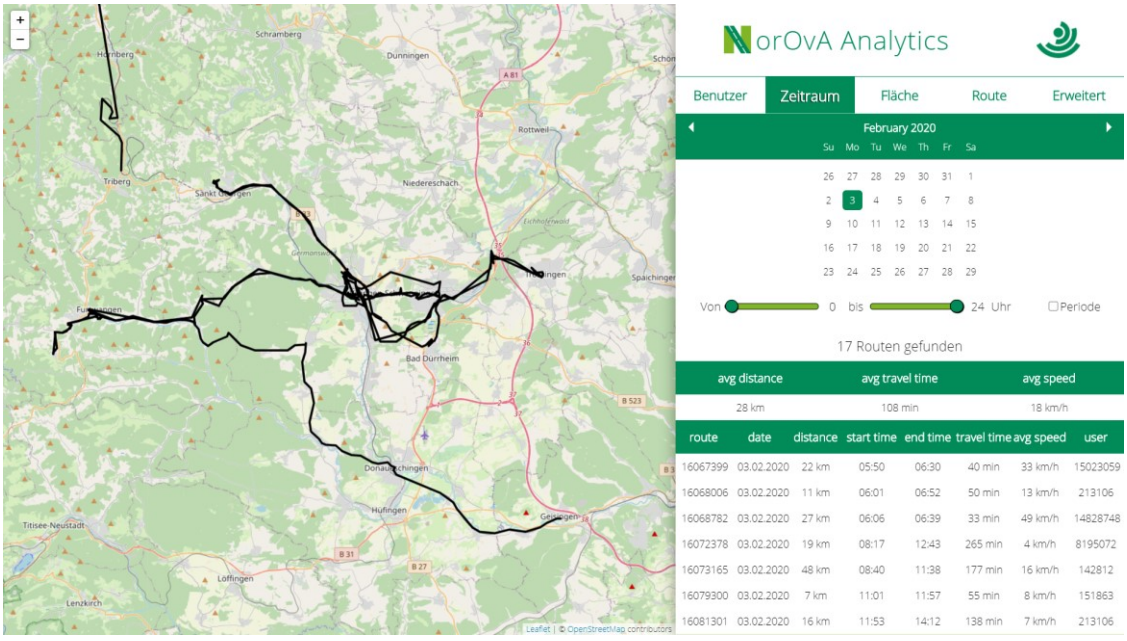

Abbildung 5-2: Analyse von Zeitpunkt und -spanne

Die Zeitraumanalyse visualisiert die Ergebnismenge auf der Karte und bietet eine Tabelle zur schnellen Einsicht in die Streckendaten. Das Datum, Start- und Endzeit der Aufzeichnung, Fahrtzeit und Geschwindigkeit sind wertvolle Informationen, die aus der Tabelle entnommen und sogar auf- und absteigend sortiert werden können. Für einen zusätzlichen Informationsgewinn werden Durchschnittswerte präsentiert.

Die Zeitraumanalyse ermöglicht es Tage, Wochen, Monate oder gar Semester zu vergleichen. Daraus können Rückschlüsse gewonnen werden, ob und inwieweit sich die Mobilität von Studierenden abhängig vom Wochentag ändert, was wiederum auf die Nutzung und Fahrplaneffizienz vom ÖPNV schließen lässt. Für die Hochschule interessant ist die Analyse von Präsenz im Rahmen eines Semesters, ob ein genereller Trend vernehmbar ist und ob es Unterschiede zwischen Pendlern, Studenten vor Ort und Nutzern des ÖPNVs gibt.

5.3 Flächen- und Routenanalyse

Die Flächen- und Routenanalyse repräsentieren den konträren Anwendungsfall, in dem die Analysekomponente keine Möglichkeit zur Konfiguration bietet, sondern die Datenbeschaffung abhängig von einer Interaktion mit der Karte stattfindet. Der Anwendungsfall wurde in Kapitel 4.8 mit Hinblick auf das Observer-Muster vertieft. Die Bibliothek Leaflet-Draw erweitert Leaflet um eine Symbolleiste zur Zeichnung von Vektorebenen und geometrischen Figuren. Die Leiste wird bei der Initialisierung von MapDrawer erzeugt und kann ein- und ausgeblendet werden. Sie befindet sich im rechten oberen Bildrand der Karte (siehe Abbildung 5-3).

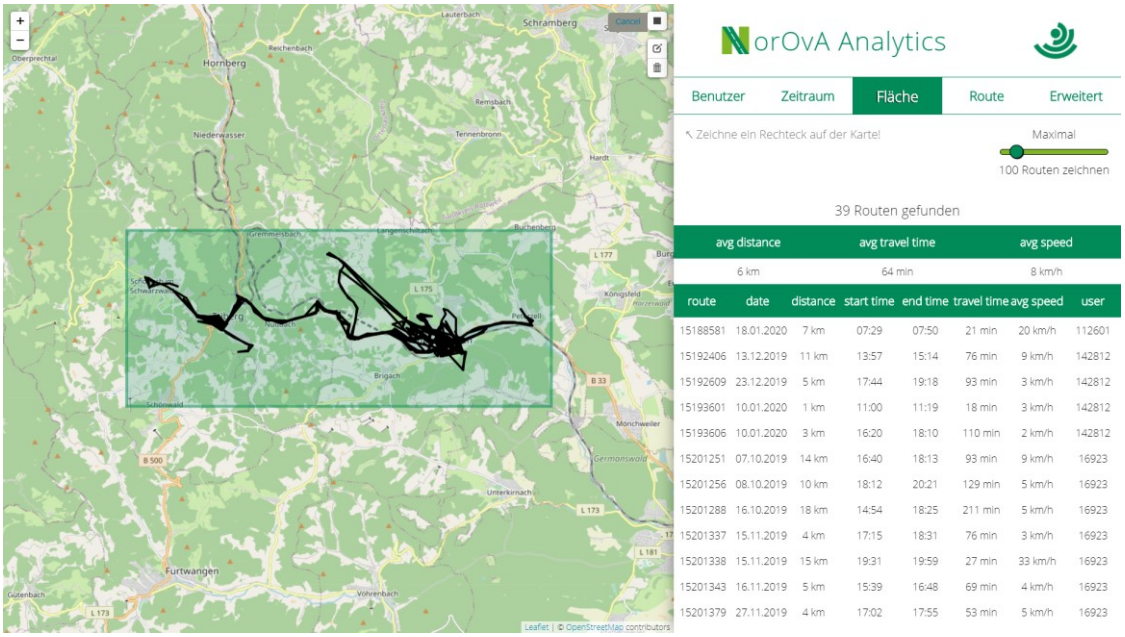

Abbildung 5-3: Räumliche Eingrenzung von Strecken

Bei beiden Analysekomponenten ist das Zeichnen von Rechtecken der Initiator der Datenbeschaffung. Die Flächenanalyse ermöglicht es Streckenaufzeichnungen auf eine Stadt oder ein Gebiet einzuschränken. Abbildung 5-3 zeigt eine Einschränkung auf die Städte Triberg und Sankt Georgen.

Durch die Flächenanalyse ist das Projektteam imstande Städte und Gebiete, wie den gesamten Schwarzwald-Baar-Kreis, zu analysieren und den städtischen ÖPNV einem Vergleich zu unterziehen. Außerdem können so die drei Hochschul-Standorte

Furtwangen, Tuttlingen und Villingen-Schwenningen separat auf z.B. Studenten vor Ort und deren Weg zur Hochschule analysiert werden.

Die Analyse von Routen stützt sich auf das Zeichnen von zwei Rechtecken. Die Rechtecke bestimmen das Längen- und Breitengradintervall von der ersten respektive letzten Koordinate der Streckenaufzeichnung (siehe Abbildung 5-4).

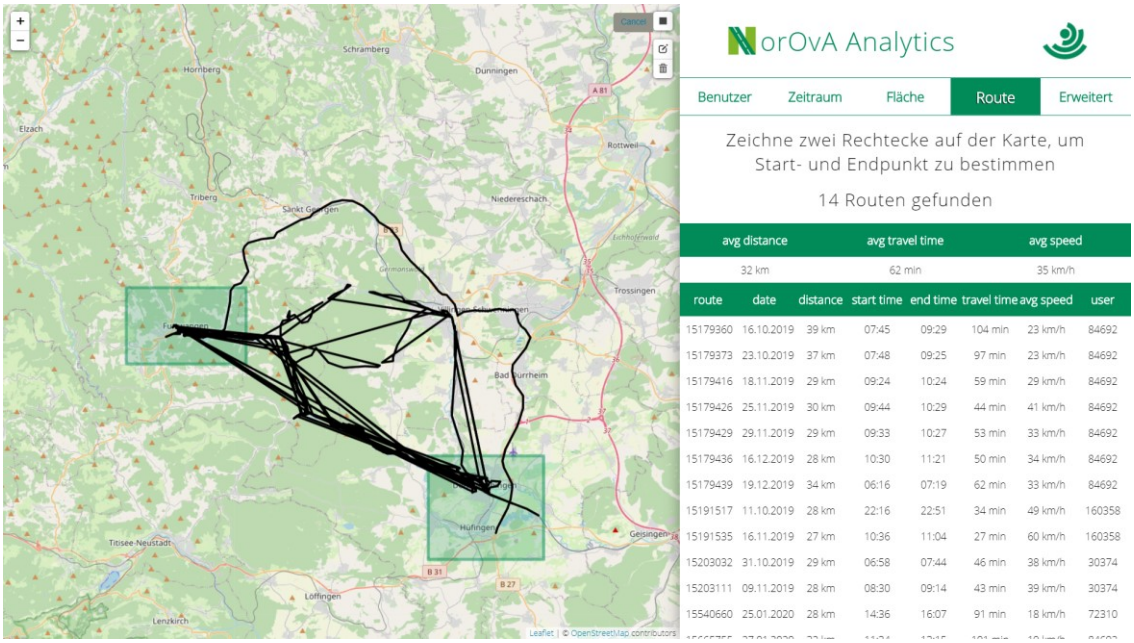

Abbildung 5-4: Routenanalyse anhand von Start- und Endpunkt

Die Zeichnungen münden in einer komplexen Datenbankabfrage. Da es äußerst schwer (bzw. unmöglich) ist, Trackingdaten zeitgleich nach Routenzugehörigkeit zu gruppieren, nach Zeitstempel zu sortieren und den ersten und letzten Datensatz der Sortierung darauf zu überprüfen, ob er in dem gegebenen Intervall liegt, wurde die Datenbankabfrage in drei Teile gegliedert. Die ersten beiden Abfragen selektieren alle Routen (Route-ID), deren Start- und Endpunkt die Bedingungen erfüllen. Die darauffolgende Abfrage verknüpft OR-Bedingungen, um Trackingdaten zu erhalten, deren Route-Attribut mit einer Route-ID übereinstimmt. Eine Lösung mit deutlich schlechter Performanz, die mit Sicherheit optimiert werden kann.

Mithilfe der Routenanalyse kann schnell eingesehen werden, woher die meisten Hochschul-Pendler kommen und inwiefern die Strecken sich unterscheiden. Folglich kann die meistgenutzte Verbindung abgeleitet und vertieft analysiert werden.

Ebenso sind Potenziale für Fahrgemeinschaften ersichtlich. Gegenwärtig arbeitet das Projektteam an einer Lösung, in der Ähnlichkeiten von Strecken erkannt werden und die App-Nutzer über die Möglichkeit einer Fahrgemeinschaft informiert. Die Routenanalyse von NorOvA Analytics ist ein erster Schritt in diese Richtung und kann programmatisch erweitert werden, um die Benutzernamen der Ergebnismenge zu erhalten und infolgedessen eine Push-Nachricht an die App zu senden.

5.4 Erweiterte Analyse

Die erweiterte Analyse ist eine Kombination von Routen- und Zeitraumanalyse. Anwendungszweck ist die Analyse von Wochentagen und Ableitung von Stoßzeiten. Die Komponente integriert einen Wochentag-Picker, die zwei Stunden-Schieberegler der Zeitraumanalyse und die Symbolleiste zum Zeichnen von Rechtecken. Die erste Datenbeschaffung wird durch das Zeichnen von Rechtecken ausgelöst. Sind die Start- und Endpunkte definiert, sind Konfigurationen bzgl. Wochentag und Zeitspanne für Veränderungen der Visualisierung verantwortlich. Die erweiterte Analyse verkörpert somit beide Anwendungsfälle (vgl. Kapitel 4.8) und nutzt Ereignis-Emission und das Observer-Muster zur Bestimmung des internen Zustandes.

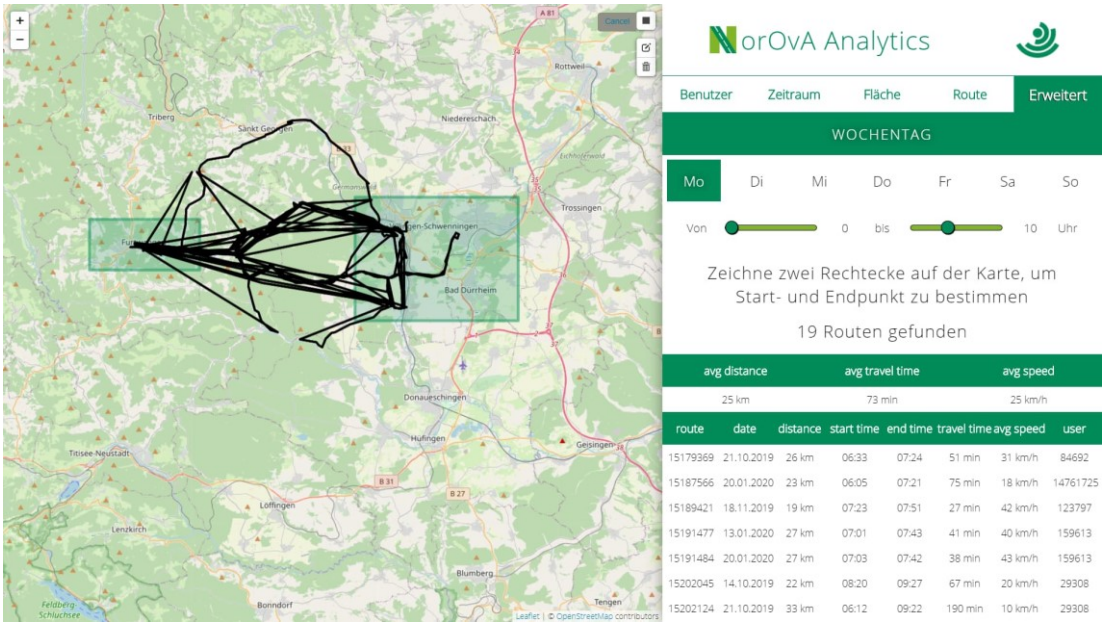

Abbildung 5-5: Analyse von Wochentagen und Stoßzeiten

6 Einordnung in die Literatur

In diesem Kapitel werden die VGI-Datenerhebung und webbasierte Analyse von Geodaten mit den Erkenntnissen aus der Forschung verglichen. Es wird untersucht, ob und inwieweit die Forschungsgebiete eParticipation, Web 2.0, Crowdsourcing und VGI harmonieren und sich für die Entscheidungsfindung von Organisationen eignet. Abschließend wird die literarische Entwicklung von GIS und Web GIS untersucht.

Regierungen sind wie Unternehmen dem ständigen Druck ausgesetzt, ihre Dienste und Prozesse im Zuge der Digitalisierung zu modernisieren und über die Internetplattform zugänglich zu machen. Ein digitaler Austausch zwischen Bevölkerung und Regierung ist für beide Seiten vorteilhaft. Die Regierung ist auf die Beteiligung und Meinung von Privatpersonen angewiesen, um Entscheidungen abzuleiten. Die Beteiligung über das Internet wird in der Forschung unter dem Ausdruck eParticipation untersucht: „E-participation involves the extension and transformation of participation in societal democratic and consultative processes mediated by information and communication technologies [...], primarily the Internet" (Sabeo, et al., 2008).

Das Internet wie wir es kennen ist dieser Aufgabe gewachsen, kann aber gezielt optimiert werden: „The Internet is often taken for granted rather than explored" (Sabeo, et al., 2008). Das Konzept Web 2.0 kann in diesem Forschungsgebiet effektiv eingesetzt werden. Transparenz, Effizienz und Effektivität können verbessert werden (vgl. Johnson und Sieber, 2013). Die Erhebung von geografischen Daten auf freiwilliger Basis (VGI) ist ein neuer Ansatz gegenüber der eher klassischen Citizen Science. Die Betrachtung von Menschen als Datenquelle ist eine polarisierende Vorstellung, der datenschutzrechtlich und sicherheitstechnisch kritisch gegenübergestanden wird, aber auf der anderen Seite neue Erkenntnisse liefern kann (vgl. Goodchild, 2007).

Mobile Endgeräte, allem voran das Hauptmedium zum Surfen im Internet: das Smartphone, sind heutzutage allgegenwärtig und nicht mehr aus dem Alltag wegzudenken. Smartphones sind GPS-fähig und eignen sich für diesen Anwendungsfall: „In mobile crowdsourcing mobile devices are used for datacollection tasks delegated to a larger number of people" (Fuchs-Kittowski, Faust, 2014). Smartphones bieten eine Plattform, um eine größere Zahl an Personen zu erreichen. Darüber hinaus besitzen sie Multisensorfunktionen, d.h. Sie können neben Standortdaten auch Daten bzgl. des Klimas und der Fortbewegung liefern (vgl. Chatzimilioudis, et al., 2012).

Die Verbindung eines Web 2.0, der Erhebung von VGI über Smartphones und ein digitaler Informationsfluss zwischen Bevölkerung und Regierung kann für viele praktische Projekte angewandt werden. Im Fall des NorOvA-Projektes werden diese Konzepte kombiniert, um die Nutzung des ÖPNVs zu analysieren, sodass Menschen auf das Auto verzichten können: „Governments world wide are seeking to tackle the ever-rising levels of car use amongst their populations" (Keynon und Lyons, 2003).

Entscheidungen, die den öffentlichen Verkehr betreffen, sind unter zahlreichen Gesichtspunkten zu treffen (vgl. Giuffrida, et al., 2019). Die Reiselust und -qualtität von Bürgern darf nicht eingeschränkt werden, das Interesse von staatlichen Transportunternehmen ist zu bewahren und, wie so oft, ist das Budget ausschlaggebend.

NorOvA Analytics kann als ein Geographic Information System (GIS) eingestuft werden. GIS sind Anwendungen, die Geodaten visualisieren: „GIS software has enabled users to view spatial data in proper format“ (Aleshkeih, et al., 2002). Sie sind seit Jahren in der Forschung präsent. Die Abwandlungen von klassischen GIS zu webbasierten GIS ist ein seit langem vernehmbarer Trend. Giuffrida, et al., haben 2019 Forschungen betrieben, die sich stark mit dem Anwendungsgebiet von NorOvA Analytics überschneiden: „In this study, we provide an overview [...] of the use of VGI and PPGIS in transport studies“. PPGIS steht für Public Participitory GIS und beschreibt ein System, bei dem nicht-privilegierte Gruppen aktiv mitwirken können.

Die Forschung hinsichtlich des Einsatzes von VGI und PPGIS zur Analyse des öffentlichen Verkehrs ist eine junge Entwicklung und wird vermehrt seit Ende der 2010er Jahre betrieben: „[...] it can be said that a continuous and growing scientific production on the issues related to PPGIS in the field of transport can be played starting from 2010.“ (Giuffrida, et al., 2019). Giuffrida, et al, haben 81 inhaltlich übereinstimmende wissenschaftliche Dokumente in dem Zeitraum von 1997 bis 2019 untersucht, um Schwerpunkte zu analysieren. 78 Prozent der Artikel befassen sich demnach mit „Passenger Transportation“. Sieben Prozent betreffen Infrastruktur.

Die Hälfte aller Artikel orientieren sich an der Forschung von Goodchild (2007) und stimmen mit dem Vorgehen von NorOvA überein: „More than the half of the articles (54%) include the participation of citizens as sensors: before the beginning of the study, participants gave their consent to the digital tracking of their position, speed, direction of travel and time information through the GPS and Accelerometer of a mobile device” (Giuffrida, et al., 2019). Die Statistik zeigt, wie intensiv VGI und PPGIS in den letzten Jahren kombiniert wurden, um die Entscheidungsfindung von Organisationen bzgl. dem Lösen von verkehrlichen Problemen zu unterstützten.

Literaturverzeichnis

ABRIAL, Jean-Raymond, et al. *Data semantics*. Université scientifique et médicale, 1974.

AHAS, Rein. Mobile positioning. *Mobile methods*, 2011, S. 183-199.

AHONEN, Suvi; **ESKELINEN**, Pekka. Mobile terminal location for UMTS. *IEEE Aerospace and Electronic Systems Magazine*, 2003, 18. Jg., Nr. 2, S. 23-27.

ALESHEIKH, Ali Asghar; **HELALI**, Hussein; **BEHROZ**, H. A. Web GIS: technologies and its applications. In: *Symposium on geospatial theory, processing and applications*. 2002.

ARNDOLD, Jason. Callback Functions in React. medium.com. 2017.

BECK, Kent. Extreme Programming Explained, chapter Glossary. The XP Series. 2000.

BERNERS-LEE, Tim, et al. World-Wide Web: the information universe. *Internet Research*, 1992.

BERNERS-LEE, Tim; **FIELDING**, Roy; **FRYSTYK**, Henrik. Hypertext transfer protocol--HTTP/1.0. 1996.

BOURQUE, Pierre, et al. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.

CHATZIMILIOUDIS, Georgios, et al. Crowdsourcing with smartphones. *IEEE Internet Computing*, 2012, 16. Jg., Nr. 5, S. 36-44.

CHEN, Peter Pin-Shan. The entity-relationship model—toward a unified view of data. *ACM transactions on database systems (TODS)*, 1976, 1. Jg., Nr. 1, S. 9-36.

CODD, E. A Relational Model of Data for Large Shared Data Banks. *Communication of the ACM*, 1970, Volume 13, Number 6, S. 377-387.

CONALLEN, Jim. *Building Web applications with UML*. Addison-Wesley Professional, 2003.

CROCKFORD, Douglas. A Survey of the JavaScript Programming Language. crockford.com. 2002.

DABEK, Frank, et al. Event-driven programming for robust software. In: *Proceedings of the 10th workshop on ACM SIGOPS European workshop*. 2002. S. 186-189.

ECMA, E. ECMAScript language specification. ecma-international.org. 2020.

FARRELL, Jason; **NEZLEK**, George S. Rich internet applications the next stage of application development. In: *2007 29th International Conference on Information Technology Interfaces*. IEEE, 2007. S. 413-418.

FIELDING, Roy, et al. Hypertext transfer protocol–HTTP/1.1. 1999.

FIELDING, Roy.; **TAYLOR**, Richard. *Architectural styles and the design of network-based software architectures*. Irvine: University of California, Irvine, 2000.

FUCHS-KITTOWSKI, Frank; **FAUST**, Daniel. Architecture of mobile crowdsourcing systems. In: *CYTED-RITOS International Workshop on Groupware*. Springer, Cham, 2014. S. 121-136.

GADATSCH, Andreas. Einführung in die Datenmodellierung. In: *Datenmodellierung*. Springer Vieweg, Wiesbaden, 2019. S. 1-7.

GAMMA, Erich. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

GARLAN, David; **SHAW**, Mary. An introduction to software architecture. In: *Advances in software engineering and knowledge engineering*. 1993. S. 1-39.

GARRETT, Jesse James, et al. Ajax: A new approach to web applications. 2005.

GIUFFRIDA, Nadia, et al. Mapping with stakeholders: An overview of public participatory GIS and VGI in transport decision-making. *ISPRS International Journal of Geo-Information*, 2019, 8. Jg., Nr. 4, S. 198.

GOODCHILD, Michael F. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 2007, 69. Jg., Nr. 4, S. 211-221.

GOODCHILD, Michael F.; **LI**, Linna. Assuring the quality of volunteered geographic information. *Spatial statistics*, 2012, 1. Jg., S. 110-120.

GOURLEY, David, et al. *HTTP: the definitive guide*. " O'Reilly Media, Inc.", 2002.

HOWE, Jeff. The rise of crowdsourcing. *Wired magazine*, 2006, 14. Jg., Nr. 6, S. 1-4.

HUDSON-SMITH, Andrew, et al. Mapping for the masses: Accessing Web 2.0 through crowdsourcing. *Social science computer review*, 2009, 27. Jg., Nr. 4, S. 524-538.

JOHNSON, Peter A.; **SIEBER**, Renee E. Situating the adoption of VGI by government. In: *Crowdsourcing geographic knowledge*. Springer, Dordrecht, 2013. S. 65-81.

KENYON, Susan; **LYONS**, Glenn. The value of integrated multimodal traveller information and its potential contribution to modal change. *Transportation research part F: Traffic psychology and behaviour*, 2003, 6. Jg., Nr. 1, S. 1-21.

MASAK, Dieter. *Moderne Enterprise Architekturen*. Springer-Verlag, 2006.

MEDVIDOVIC, Nenad; **TAYLOR**, Richard N. Software architecture: foundations, theory, and practice. In: *2010 ACM/IEEE 32nd International Conference on Software Engineering*. IEEE, 2010. S. 471-472.

MILTON, Scott, et al. Dynamic dispatch in object-oriented languages. 1994.

MINISTERIUM FÜR VERKEHR BADEN-WÜRTTEMBERG. Mit neuen Projekten zu digitaler Mobilität. vm.baden-wuerttemberg.de. 2019.

MUKASA, Kizito Ssamula; **KAINDL**, Hermann. An integration of requirements and user interface specifications. In: *2008 16th IEEE International Requirements Engineering Conference*. IEEE, 2008. S. 327-328.

O'REILLY, Tim. *What is web 2.0*. " O'Reilly Media, Inc.", 2009.

PERRY, Dewayne E.; **WOLF**, Alexander L. Foundations for the study of software architecture. *ACM SIGSOFT Software engineering notes*, 1992, 17. Jg., Nr. 4, S. 40-52.

RESCORLA, Eric, et al. Http over tls. 2000.

SÆBØ, Øystein; **ROSE**, Jeremy; **FLAK**, Leif Skiftenes. The shape of eParticipation: Characterizing an emerging research area. *Government information quarterly*, 2008, 25. Jg., Nr. 3, S. 400-428.

SCHILL, Alexander; **SPRINGER**, Thomas. *Verteilte Systeme: Grundlagen und Basistechnologien*. Springer-Verlag, 2012.

SESTER, Monika (Hg.). *Geoinformatik: Handbuch der Geodäsie, herausgegeben von Willi Freeden und Reiner Rummel*. Springer-Verlag, 2019.

SINGH, Raghu. International standard ISO/IEC 12207 software life cycle processes. *Software Process Improvement and Practice*, 1996, 2. Jg., Nr. 1, S. 35-50.

STUTZ, Dave. Advice to Microsoft regarding commodity software. synthesists.net. 2003.

STRAZZULLO, Francesco. Let's Talk About Frameworks. In: *Frameworkless Front-End Development*. Apress, Berkeley, CA, 2019. S. 1-21.